Evolutionary Techniques for Permutation Based Problems

A thesis submitted during 2022 to the University of Hyderabad in partial fulfillment of the award of a **Ph.D. degree** in School of Computer and Information Sciences

by

Gaurav Srivastava

School of Computer and Information Sciences
University of Hyderabad
P.O. Central University, Gachibowli
Hyderabad – 500 046
Telangana, India

January 2022

CERTIFICATE

This is to certify that the thesis entitled **"Evolutionary Techniques for Permutation Based Problems"** submitted by **Gaurav Srivastava** bearing **Reg. No. 15MCPC05** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under my supervision and guidance.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

The student has the following publications before submission of the thesis for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of his research:

1. **Gaurav Srivastava**, Alok Singh. "Boosting an evolution strategy with a preprocessing step: application to group scheduling problem in directional sensor network". *Applied Intelligence, **48**: 4760-4774, 2018, Springer (ISSN: 0924-669X)*. Work reported in this paper appears in **Chapter 3**.
2. **Gaurav Srivastava**, Venkatesh Pandiri and Alok Singh. "An evolution strategy based approach for cover scheduling problem in wireless sensor networks". *International Journal of Machine Learning and Cybernetics, **11**: 1981-2006, 2020, Springer (ISSN: 1868-8071)*. Work reported in this paper appears in **Chapter 2**.
3. **Gaurav Srivastava**, Alok Singh and Rammohan Mallipeddi. "A hybrid discrete differential evolution approach for the single machine total stepwise tardiness problem with release dates". *Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC-2021), 2021, IEEE*. Work reported in this paper appears in **Chapter 4**.

4. **Gaurav Srivastava**, Alok Singh and Rammohan Mallipeddi. "NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows". *Expert Systems with Applications,* ***176****: 114779, 2021, Elsevier (ISSN: 0957-4174).* Work reported in this paper appears in **Chapter 7**.

and has made the presentations in the following conferences:

1. 2021 IEEE Congress on Evolutionary Computation (CEC-2021), June 28 - July 1, 2021, Kraków, Poland.

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D.:

Course Code	Name	Credits	Pass/Fail
CS 801	Data Structures and Algorithms	4	Pass
CS 802	Operating Systems and Programming	4	Pass
AI 810	Metaheuristic Techniques	4	Pass
AI 852	Learning & Reasoning	4	Pass

(Prof. Alok Singh)
Supervisor
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

(Prof. Chakravarthy Bhagvati)
Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

DECLARATION

I, **Gaurav Srivastava**, hereby declare that this thesis entitled **"Evolutionary Techniques for Permutation Based Problems"** submitted by me under the guidance and supervision of **Prof. Alok Singh** is a bonafide research work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Library is enclosed.

Date :

Name: Gaurav Srivastava

Signature of the Student:

Reg. No.: 15MCPC05

Signature of the Supervisor:

Abstract

During the last several decades, the research field of combinatorial optimization has attracted many researchers across various scientific fields owing to their practical importance in day to day life. The advancement of technology and accelerated computer evolution makes large-scale computation practical. Consequently, many industries have started employing the state-of-the-art techniques available in the literature of combinatorial optimization for efficiently solving their problems. These problems include allocation of resources and more effective planning, scheduling, manufacturing, transportation and distribution. Permutation based combinatorial optimization problems are a specific category of combinatorial optimization problems, where the problem possess permutation characteristic. Many real world problems like routing, scheduling, networking, timetabling have permutation aspect. Since many practical applications can be modeled as a permutation based problem, these problems have huge practical importance. Apart from practical applications, these problems pose a serious challenge from theoretical perspective also. Any improvement that can be made while addressing a permutation based problem will provide a scope for improvement for several other related permutation based problems. Motivated by these facts, in this thesis, we have focused on solving some recent $\mathcal{NP}$-hard permutation based combinatorial optimization problems using three evolutionary techniques, viz. genetic algorithm (GA), evolution strategy (ES) and discrete differential evolution (DDE).

Six $\mathcal{NP}$-hard permutation based problems have been addressed in this thesis. These six problems are as follows: cover scheduling problem in wireless sensor networks, total rotation minimization problem in directional sensor networks, single machine total stepwise tardiness problem with release dates, rescue unit allocation and scheduling problem, quality of service vehicle routing problem with time windows and multiobjective vehicle routing problem with time windows.

These six problems not only have several practical applications in different fields such as wireless sensor networks, transportation, logistics, planning & scheduling, disaster management, but are also challenging from theoretical perspective. To address these problems, we have developed evolution strategies based approaches for first two problems and a hybrid discrete differential evolution approach for third problem. Fourth and fifth problems are solved by grouping genetic algorithm based approaches, whereas the last problem being multiobjective in nature uses nondominated sorting genetic algorithm II (NSGA-II) based approach. Appropriate problem-specific knowledge has been incorporated in all our approaches. Computational results demonstrate the effectiveness of our approaches. The proposed approaches can be easily adapted for other related permutation based problems. The insight gained from this thesis can be useful to solve many other combinatorial optimization problems.

To my parents,

Shri Deep Chandra Srivastava and Smt. Suman Lata Srivastava

my dear wife and lovely daughter,

Anvita and Vaanya

without their endless love, support and encouragement, this would not have been possible.

Acknowledgements

Pursuing the PhD has been a truly life-changing journey for me. This journey would not have been possible without the support of many people and I would like to express my deep appreciation to all of them.

First and foremost, I express my sincere gratitude towards my supervisor **Prof. Alok Singh** for his endless support and guidance throughout this PhD. His immense knowledge and insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. I am thankful to him for introducing me to the field of combinatorial optimization using evolutionary techniques which is a perfect mix of my research interests: optimization and natural computing. I am obliged to him for his availability, patience and continuous encouragement due to which I successfully crossed many hurdles throughout this course. I feel fortunate to have him as my PhD supervisor and words alone are not enough to express my gratitude towards him.

Next, I would like to thank my doctoral review committee (DRC) members, **Dr. Anupama Potluri** and **Prof. C. Raghavendra Rao** for their invaluable insights and feedback about my research work. Their hard questions during the review process drove me to analyse my research work more deeply, resulting in new ideas to further enhance the quality of my work.

During my PhD, I was very fortunate to get the opportunity to visit and learn from **Dr. Rammohan Mallipeddi** at Kyungpook National University, South Korea. I will always remain indebted to him.

I take this opportunity to thank the Dean of the School **Prof. Chakravarthy Bhagvati** for providing all the necessary facilities to pursue my research work. I would also like to thank other faculty members and staff of the school for their support. I am thankful for the unstinting support that I received from the research infrastructure and the effervescent ambiance of the University. Due credit to the

University for building a research oriented School of Computer & Information Sciences (SCIS), a library rich in a wide range of research books & articles, and most importantly a healthy campus atmosphere.

I am indebted to my senior lab mates **Dr. Sachchida Nand Chaurasia** and **Dr. Venkatesh Pandiri**, who had helped me in almost every phase of my PhD. I found them always available, whenever I needed them. In addition, I would like to acknowledge my other senior lab mates **Dr. Shyam Sundar, Dr. B. Jayalakshmi** and **Dr. Abobakr Khalil Alshamiri** for their valuable suggestions and support. A special thanks to **Rajesh** for helping me with MATLAB codes. I am thankful to my fellow lab mates (**Edukondalu, Mallikarjuna, Preeti, Kasi, Sebanti and Danish**) for stimulating discussions, providing me company for tea/snacks and for all the fun we had together. I am thankful to all my PhD colleagues in SCIS for their support and encouragement. I would like to thank my fellow researchers (**Trinadh, Vikas, Sameera** and **Fitri**) at Kyungpook National University, South Korea for bearing with me and having the patience to answer all my silly questions either related to the research or the new unknown place.

I would like to make a special mention about my dear friend and labmate **Mallikarjuna** whom we lost due to COVID. I shared a close bond with him and miss his pleasant smile and presence in the lab.

I am grateful to a number of researchers in my field who furnished me with indispensable research data and clarified my doubts regarding the definitions and their proposed approaches, especially to **Mr. Airam Expósito** and **Mr. Victor Cunha**.

A major part of my research work was supported by Council of Scientific & Industrial Research (CSIR), Government of India through Senior Research Fellowship. I am grateful to this organization for the financial support.

Finally, I could not have achieved anything in life without the unconditional and constant support of my family. I would like to dedicate this thesis to my parents **Shri Deep Chandra Srivastava** and **Smt. Suman Lata Srivastava** who always encouraged me to achieve new heights. I would like to thank a very special person, my wife, **Anvita** for her continued and unfailing love, support and understanding

during my pursuit of PhD. I am also thankful to my brother **Vaibhav** and sister-in-law **Anveshika** for their affection, patience and encouragement.

Gaurav Srivastava

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Introduction

A combinatorial optimization problem (*COP*) is discrete and finite in nature, and seeks an optimal solution from its large but finite set of feasible solutions. It includes both minimization and maximization problems. Optimality is decided with respect to an objective function that is specific to the problem and that assigns a numerical value to each feasible solution according to its composition. The combinatorial optimization problems are divided into three categories on the basis of their characteristics. The three categories are namely, permutation problems, subset selection problems and grouping problems. The permutation problems seek an arrangement of a given set of items into an optimal order subject to some constraints. In permutation problems, any solution is represented as a permutation. The traveling salesman problem and single machine scheduling problem are well-known examples of permutation problems. The subset selection problems intend to find an optimal subset of objects from a pool of objects under some constraints. The knapsack problem and minimum spanning tree problem are two common problems which fall under the category of subset selection problems. On the other hand, the objective of a grouping problem is to optimally divide a given set of items into various groups under certain constraints. The graph coloring problem and clustering are two famous examples of grouping problems. However, some combinatorial optimization problems have characteristics of more than one category of problems, i.e., these three categories are not disjoint. For example, vehicle routing problem has characteristics of permutation and grouping both. Likewise, shortest path problem on graphs has the characteristics of subset selection and permutation both. By a permutation based problem, we mean any combinatorial optimization problem possessing the permutation characteristic. In addition, it can have other characteristics too. Presence of multiple characteristics in a problem usually makes it harder.

1. INTRODUCTION

Many real-world problems like routing, scheduling, networking, timetabling have permutation aspect. Over the last five decades, researchers across different fields have studied numerous permutation based optimization problems, e.g. traveling salesman problem (TSP), flowshop scheduling problem (FSSP), quadratic assignment problem (QAP), linear ordering problem (LOP) and many more. Despite the fact that all of these problems have permutation characteristic, the interpretation of the permutation with respect to the objective of the problem is different in different problems. To show the difference in semantic of permutation from the perspective of the objective of the problem, we present the brief description of these problems in the subsequent paragraphs.

The traveling salesman problem (TSP) is one of the earliest known permutation based combinatorial optimization problems. Given a set of cities along with distances between each pair of cities, the TSP seeks an optimal permutation of cities that results in shortest possible route. Karl Menger in 1932 proposed "das boten problem" [1] which translates as the "messenger problem" in English. Literature considers this one as the first published work on TSP [2, 3, 4].

FSSP was introduced by Johnson in 1954 in [5]. It consists of *n* jobs to be processed on *m* machines. Each job requires *m* operations, thus need to be processed by each of the *m* machines. A job i can be processed on j^{th} machine if and only if its $(j-1)^{th}$ operation has finished on $(j-1)^{th}$ machine and the j^{th} machine is available. The processing time of job i on j^{th} machine is p_{ij}. The objective of FSSP is to find an optimal permutation of jobs for each machine that minimizes the makespan, i.e., the completion time of the last job. This problem has a total $(n!)^m$ possible schedules (solutions). A simplified version of FSSP known as permutation FSSP considers the same order (sequence) in which these jobs are processed on each machine [6]. In the permutation FSSP, the number of possible schedules (solutions) reduces to $n!$.

QAP was introduced by Koopmans et al. [7] in 1957 as a mathematical model that deals with the assignment of a set of facilities to a set of locations. It is based on the theory that the cost incurred due to some economic activity at one location may depends on the facilities at other locations [8]. The problem is to assign *n* facilities to *n* sites such that each pair of locations have a specific distance and each pair of facilities have a particular amount of flow or weight. Each facility must be assigned to exactly one location. The cost function is associated with the distance between locations and flow between the facilities. Two $n \times n$ matrices provide the flow ($W = w_{pq}$) between facilities *p* and *q*, and distance ($D = d_{rs}$) between locations *r* and *s*. For *n* facilities, the solution of QAP is represented as a permutation ρ=$\{\rho_1, \rho_2, \ldots, \rho_n\}$ where the

facility ρ_i is assigned to the i^{th} location. The objective function that need to be minimized, is defined as [8, 9]:

$$f(\rho) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} d_{\rho_i \rho_j}$$

The TSP can be formulated as a special case of QAP where the distance matrix of TSP can be considered as the distance matrix of QAP and the flow matrix (W) being equal to the adjacency matrix of a cycle on n vertices/cities of TSP [8].

The linear ordering problem (LOP) has received the attention of researchers since its introduction by Chenery et al. in 1958 [10]. Given a $n \times n$ matrix of numerical values $M = \{x_{ij}\}$, the LOP seeks a (simultaneous) permutation ρ of rows (and columns) that maximizes the sum of numerical values above the principal diagonal of matrix M. Formally, the objective function is defined as :

$$f(\rho) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{\rho_i \rho_j}$$

where ρ_i represents the index of row (and column) placed at i^{th} place in permutation ρ.

Even though all of the above problems possess permutation characteristics, they can not be effectively solved by a single approach. It is necessary to exploit the semantic of permutation while solving such problems. Thus, the approaches whose design is based on the characteristics of the problem along with the objective can be more effective than the corresponding generic approaches employed for the problem. It is pertinent to mention that, we have neither provided any further discussion on these aforementioned problems nor solved any of them in this thesis. The sole purpose of discussing these problems is to highlight the differences in aspects of permutation for different permutation based *COPs*.

With the emergence of new technology and the advancement of human civilization, new permutation based problems continue to emerge. Despite the common aspect of permutation, these problems may have significant differences due to the objective function, additional constraints or the occurrence of other characteristics (grouping and/or subset selection) along with permutation in them. To address such problems in a more suitable way, it is necessary to find specific requirements by performing a deep analysis of the problem characteristics and then develop problem-specific approaches for them.

Though some permutation based problems such as shortest path problem can be optimally solved in polynomial time, most permutation based problems are $\mathcal{NP}$-hard. In fact, this is true not only for permutation based problems, but for all *COPs*, i.e., though some *COPs* such as shortest path problem, minimum spanning tree problem can be optimally solved in polynomial time, most *COPs* are $\mathcal{NP}$-hard. For $\mathcal{NP}$-hard optimization problems, no polynomial time exact algorithm is known so far and even the optimality of a given solution can not be verified in polynomial time. Therefore, the applicability of exact methods is limited to small size instances only. To solve large instances of such problems in a reasonable amount of time, one has to consider other approaches which can find high quality solutions which need not be optimal. *Approximate methods* [11] is a hypernym used for all such approaches. The development of approximate methods to address various *COPs* is a well-known area of research which is highly active for the last five decades.

1.1 Approximate methods

Approximate methods are the collection of approaches that provide good quality solutions in a reasonable amount of time. The obtained solutions need not be optimal, but usually they are quite near to optimal solutions in terms of quality. Usually, these methods have polynomial runtime complexity and yields solutions in a reasonable amount of time. Due to the discrete nature of *COPs*, the search (solution) space of $\mathcal{NP}$-hard problems is rugged, in general. The chances of many locally optimal solutions are also possible and some of these locally optimal solutions can be very near to optimal solution. Even though the locally optimal solution can be near to optimal solution in terms of objective value, but these solutions can be quite far in solution space. Approximate methods can be further divided into following four categories

- *Heuristics* are intuitive approach designed by considering the structure of the problem. In general, heuristic generates a good quality solution in a very small amount of time. However, it does not provide any guarantee about the quality of the generated solution. If the heuristic is designed by considering the characteristics of the problem appropriately, then the quality of the generated solution is usually good.

- *Metaheuristics* are the high-level framework consisting of a general-purpose heuristic that can be used for any optimization problem. They follow an iterative search process to explore promising regions of the search space [11]. Metaheuristics are problem-agnostic

and can be applied to a wide variety of problems with suitable modifications. They are stochastic algorithms and generally non-deterministic in nature. Usually, the problem needs to be represented in a specific form as per the suitability of the metaheuristic framework. The genetic algorithms [12, 13], ant colony optimization [11, 14], artificial bee colony algorithm [15], variable neighborhood search [16, 17], differential evolution [18, 19], evolution strategy [20, 21] are some of the well-known metaheuristics.

- *Approximation algorithms* include those methods which provide solutions with proven solution quality. The obtained solutions are guaranteed to satisfy the approximation ratio upper bound value. Thus, these methods provide a solution whose quality is always within a certain factor of the quality of the optimal solution.

- The fourth category consists of those approaches that are based on premature termination of exact approaches. For example, a mixed-integer linear programming (MILP) solver for a problem can be prematurely stopped by allowing it to execute for a certain amount of time, and then output the best solution found at the end if at least one feasible solution is found in this fixed time interval.

The literature contains theoretical results of many combinatorial optimization problems which reveal the fact that designing good polynomial-time approximation algorithms for all kinds of problems is not possible. It is also observed that premature termination of exact approaches turns out to be unsuccessful in finding even one feasible solution in many cases. On the other hand, metaheuristic approaches are found to be highly successful for addressing such problems. Hybrid approaches where some problem-specific heuristics are used in metaheuristic framework is a common practice to address such $\mathcal{NP}$-hard problems. In this thesis, we have considered three metaheuristic approaches, viz. genetic algorithm (GA), evolution strategy (ES) and discrete differential evolution (DDE) to address six $\mathcal{NP}$-hard permutation based problems. All the three metaheuristics belong to the broad class of evolutionary algorithms. Each approach that we have developed as a part of this thesis is hybridized with appropriate problem-specific heuristics. For first three problems considered in this thesis, which have only permutation characteristic, we have used NEH heuristic and concept of opposition based solution generation to generate initial solutions. The results demonstrate that our approach of initial solution generation is highly effective. It can be used for any problem having only permutation characteristic. The subsequent sections provide an overview of the evolutionary algorithms, NEH heuristic and the concept of opposition based solution used in this thesis.

1.2 Evolutionary algorithms

Evolutionary algorithms (EAs) is a hypernym for population based metaheuristic approaches derived from the models of biological evolution. These are stochastic optimization algorithms and use stochastic mechanism for selection, variation operators such as crossover and mutation. They imitate biological evolution in several ways such as, environmental changes cause natural selection, consequencing in an increase in the fitness of existing population. Each EA maintains a population of solutions which is also known as candidate solutions, chromosomes, individuals etc. The population is evolved towards better region of search space, using an iterative process. During each iteration (also known as generation) new solutions are created from existing solutions by employing the variation operators and solutions for next generation is selected from among newly created solutions and existing solutions by following a "survival of fittest strategy". This iterative process continues until the defined termination criteria is met.

Evolutionary algorithms incepted during 1960s with the idea that the approaches inspired from natural evolution can be used for addressing engineering applications. EAs can be broadly divided into three main research streams, viz. evolutionary programming, genetic algorithm and evolution strategies. These three approaches were developed independently and were contemporary. In USA, Fogel [22] presented evolutionary programming, whereas genetic algorithm was presented by Holland [13]. In Germany, Rechenberg presented evolution strategies whose development is continued parallel to development of GA. Later in 90's, other EAs such as differential evolution (DE), estimation of distribution algorithms (EDAs) etc. were also introduced and applied successfully on various problems. All of these follow the same core process in which a population of solutions evolve using genetic operators. Despite the framework similarity, all of these algorithms differ in numerous aspects from each other. Each of them uses their own set of genetic operators based on the principle of biological evolution. Over the year, EAs have gained huge popularity and literature contains various applications which address numerous optimization problems. Due to their ability to solve complex optimization problems, many variants of EAs have been developed consequencing in sheer volume of research articles on them. For a comprehensive survey on these algorithms, reader can refer to articles [23, 24, 25, 26].

1.3 Overview of genetic algorithm

Genetic algorithm (GA) is one of the most widely used global optimization method, which simulates the idea of natural selection and natural genetics. It is also one of the earliest proposed EA which still very popular for addressing optimization problems. It was introduced in 1960s by John H. Holland [13] and was originally proposed for for simulating the biological evolution of adaptive natural systems. Later, due to their ability to exhibit biological evolution and their adaptive characteristic, it was used for addressing complex optimization problems. In [13], Holland presented Schema Theorem which provide a strong theoretical explanation of genetic algorithm and discusses the mechanism related to working of GA. Schema Theorem states that the schemata having fitness more than average fitness will be increased in number with successive generations. It works like an analysis tool for GA and accounts which schema has more chances to survive in further process of GA. Using Schema Theorem, Holland also shows the behavior of implicit parallelism in GA. GAs have been successfully used for solving various kinds of complex optimization problems. It is an evergreen area of research in the field of optimization, and thus many new variants of GA get published in literature, each year. The success of GA is mainly because of its simple structure, applicability on diverse set of problems and robustness in finding high quality solutions due to its characteristic of better exploration of search space [26, 27, 28].

GA uses terminology borrowed from natural genetics and evolution theory. Thus, it would be convenient to briefly explain these terms in the context of GA before providing the detailed discussion. The Table 1.1 presents the explanation of the regularly used terms in GA parlance.

GA begins with a population of individuals known as chromosomes. These individuals are usually generated in a random manner, but we can also use problem-specific heuristics to generate these individual solutions. Fitness function is used to evaluate the fitness of each individual, which provides a ranking among the individuals about how much better one is with respect to others. Usually, the fitness function is same as the objective function for the problem but it can be different. After assigning fitness to each solution, GA follows an iterative process. In each generation, some individuals (solutions) are selected based on a selection mechanism and considered as parents. The selection mechanism usually prefers the more fit individuals to make a pool of parents. Such mechanism is supported by the fact that better parents have higher chance of producing further improved offspring. The solutions in parent set undergoes reproduction process with the help of genetic operators such as crossover and

Table 1.1: Regularly used terms in genetic algorithm parlance

Term	Explanation
Phenotype	A candidate solution for the problem under consideration
Genotype or Chromosome	A representation of candidate solution in an encoded form which is convenient for applying genetic algorithm
Gene	A smallest entity in chromosome and a chromosome is composed of several genes
Alleles	A set of possible values that can be assigned to a gene
Population	A collection of chromosomes that undergoes evolution
Generation	A single pass from the present population to the next one
Fitness	A measure of suitability of a chromosome as per the criteria GA is optimizing
Evaluation	The decoding process of a genotype into its corresponding phenotype and the determination of its fitness
Phenotype Space	The space containing all possible candidate solutions to the problem under consideration
Genotype Space	The space containing all possible genotypes of the problem under consideration

mutation. The crossover or recombination operator combines two or more parents to produce one or more offspring. Consequently, the offspring generated through crossover inherits traits from all the parents. Usually, only two individuals are recombined in the crossover to produce one or two offspring. The crossover occurs only with certain probability known as crossover rate. A crossover rate of 0.8 means around 80% of offspring will be generated through crossover. When no crossover is used, an offspring is produced by making an exact copy of the parent. Mutation is applied on the newly produced offspring (irrespective of whether crossover is used or not) in which offspring solution undergoes some random changes. Like crossover, mutation also occurs with certain probability known as mutation rate. Mutation rate decides how many random changes a solution undergoes. Crossover and mutation operators are usually applied in a sequential manner in which crossover is followed by mutation. Based on the problem, we can apply these operators in a mutually exclusive manner also. As crossover and mutation occur with certain probabilities, there is a possibility that generated offspring is an exact copy of the parent. This process continues till desired number of offspring are generated. These offspring and existing population members compete for inclusion into the population for the next generation as per population evolution (population replacement) policy. Once the population for the next generation is determined, the next generation begins. In aforementioned manner, the

Algorithm 1: Pseudo-code of basic GA

Input: GA parameters such as p (size of population), p_c (crossover rate), p_m (mutation rate), q (offspring produced in each generation)
Output: Best solution obtained through GA

```
P ← φ;
for (i=1 to p) do
    S_i ← Generate_Solution();
    S_i.fitness ← Evaluate fitness of Solution(S_i);
    P ← P ∪ S_i;
while (the termination criteria remains unsatisfied) do
    P' ← φ;
    for (i=1 to q) do
        parents ← Selection(P);
        // Select parents as per selection mechanism
        offspring ← crossover(parents);
        // Apply crossover as per p_c
        offspring ← mutation(offspring);
        // Apply mutation as per p_m
        offspring.fitness ← Evaluate fitness of offspring;
        P' ← P' ∪ offspring;
    P ← evolution_policy(P,P');
return best;
```

population is evolved with the help of these genetic operators.

Algorithm 1 presents the pseudo-code of GA and different components of GA are explained in the upcoming subsections. The version of GA described in this section is generic and independent from the characteristics of the problem under consideration. Whereas, in next two sections we have provided an overview of GA for permutation and grouping problems.

1.3.1 Representation of individuals

GA processes population of chromosomes, where each chromosome represents a candidate solution. In GA, in what form we represent a solution is very crucial, as it influences the manipulation of the chromosomes to generate new chromosomes. In most commonly used terminology of GA, the phenotype form represents the actual solution of the problem which is encoded into genotype form. The GA manipulates the genotype with the help of genetic operators. The genotype representation of individuals should be done in most natural way so that the distribution of solution in genotype space is analogous to distribution in phenotype space. For success of GA, such representation scheme should be selected which can represent all

candidate solutions with either no redundancy or with minimum redundancy. A representation scheme is redundant if more than one genotype represent the same phenotype, i.e., two or more chromosomes represent the same solution. Redundancy in representation is undesirable, as it will result in larger genotype space for corresponding phenotype space. Since, GA works in genotype space, so redundancy will make GA to explore a larger search space. Thus, it can result in poor performance of GA [29]. The success of GA for a problem highly depends on the representation scheme. The selection of suitable representation scheme for an optimization problem can be done on the basis of experience and expertise.

The traditional GA uses binary representation (also known as bit string representation) to encode the solutions. In binary representation, the genotype is represented as an array of *n* binary variables. Binary representation is highly suitable for subset selection problems such as Knapsack problem. Here, *n* binary variables represent the number of objects in the problem and a value of 1 at i^{th} position represents the presence of i^{th} item in solution representing a subset. For example, a solution represented as [1 0 1 1 0] for Knapsack problem with five objects, indicates the presence of first, third and fourth object in subset. This [1 0 1 1 0] represents the genotype whereas [1 3 4] is phenotype which reflects the actual solution of Knapsack problem. Binary representation may not be appropriate for other categories of problems. Like TSP, which is a permutation based problem and a solution represents the permutation of cities, binary representation is not suitable. For such problems, integer representation is more suitable. For example, consider a solution for TSP with five cities given in integer representation as [3 1 5 2 4]. Although, this solution can also be encoded in binary representation as [011 001 101 010 100], but such representation will result in more computation for GA and thus, it is not as suitable as integer representation, for problems like TSP. Various other representations are also available in literature such as real-valued representation, random-key encoding etc., which can be used as per the characteristics of the problem in hand.

1.3.2 Selection methods

GA uses selection method to select a set of solutions that take part in breeding and generate new superior quality solutions by exploring more promising regions of search space. A selection method intends to improve the quality of solutions in the population by providing more fit solutions with higher probability for breeding. Different selection methods vary in terms of selection pressure and degree of randomness used in selection of parent set. Thus, it acts as a controlling entity that provide the balance between exploration and exploitation. The choice of

selection method influences the success of GA. Literature contains various selection methods [30]. Fitness proportionate selection [13], ranking selection [31], binary tournament selection [32] are some of the well-known selection methods.

1.3.2.1 Fitness proportionate selection

Fitness proportionate selection scheme was used in traditional GA [13]. In this scheme, selection of a potential individual is governed by the probability which is proportional to the fitness of that individual with respect to other candidate solutions in population. Formally, the probability ρ_i of selection of an individual i is determined as

$$\rho_i = \frac{f_i}{\sum_{k=1}^{n} f_k}$$

where f_i is the fitness of the individual i and population has total n individuals. Following this scheme, a more fit individual has a higher chance of selection of multiple times as a parent. The selection of individual is done by roulette wheel method or any other sampling method. In roulette wheel method, the probability value determined by the fitness of each individual is assigned as roulette portion. Thereafter, the roulette is drawn same number of times as the number of individuals required. For example, consider a population containing four individuals and their selection probabilities are 0.4, 0.2, 0.1 and 0.3, respectively. These four intervals are determined as [0, 0.4], (0.4, 0.6], (0.6, 0.7] and (0.7, 1.0]. Now, a random number from [0, 1] is generated and the individual under whose sub-interval the obtained random number lies is selected to be parent and added into the mating pool. This process is reiterated as many times as the number of individuals required.

The fitness proportionate selection method suffers from two major drawbacks. First, in the initial phase of GA, the fitness gap between solutions is high in population, so more fit members take over entire population after some generation leading to premature convergence of GA. Premature convergence occurs when the individuals in population stucks to a state where genetic algorithm is unable to generate offspring better than individuals in population. Second, the entire population have similar fitness after some generations. Thus, in such scenario all individuals have similar probability of selection making this selection scheme ineffective.

1.3.2.2 Ranking selection

Ranking selection method [31] was successful to overcome the aforementioned drawbacks of fitness proportionate selection. This selection scheme, assigns ranks to individuals based on

their relative fitnesses instead of considering absolute fitnesses. It first sorts the individuals as per their fitnesses and then assigns rank to them. In this scheme, sum of ranks is calculated and then probability ρ_i of selection of an individual i is determined as

$$\rho_i = \frac{rank_i}{\sum_{k=1}^{n} rank_k}$$

1.3.2.3 Tournament selection

Tournament selection scheme uses small sample for selection instead of considering entire population. A set of *k* individuals are randomly selected as the sample from the population and the solution with best fitness is selected for reproduction, either deterministically or by following a probabilistic approach. Many rounds of tournament are conducted to get the desired number of individuals as parent. When *k*=2, this scheme is called as binary tournament selection. Binary tournament selection shows similar similar selection pressure as ranking selection with being computationally more efficient [32].

In probabilistic binary tournament selection, two individuals are selected uniformly at random from the population and their fitness is compared. The individual with better fitness is selected with some probability, otherwise the inferior one is selected, as parent.

1.3.3 Crossover

Crossover is the mechanism which combines the genetic informations of two (or more) parents to produce offspring. It is based on the idea that combination of good quality parents may produce even better quality child solution. However, it is possible that after crossover the resulting offspring may sometimes be worse than parent, but multiple application of crossover will produce some better quality offspring. The design of the crossover operator should be done in such a manner that it should be able to recombine the information relevant to the problem under consideration while recombining the two parents. Further, if the two parents undergoing crossover are nearly same then the offspring must also be similar to the parents. This is called similarity requirement. The success of GA is highly depends on the design of crossover which, in turn, depends on the structure and characteristics of the problem under consideration along with the representation scheme used by GA. Some well-known crossover operators used in GA are described below.

1.3.3.1 1-point crossover

1-point crossover was used in traditional GA proposed by Holland [13]. It is commonly used with binary and integer representation schemes. This crossover first select one crossing point by choosing a random position from [0, n-1], where n is length of string. Thereafter, the portions of strings after that cross point are swapped between the two parents resulting in two offspring. Figure 1.1 provides an illustration of this crossover.

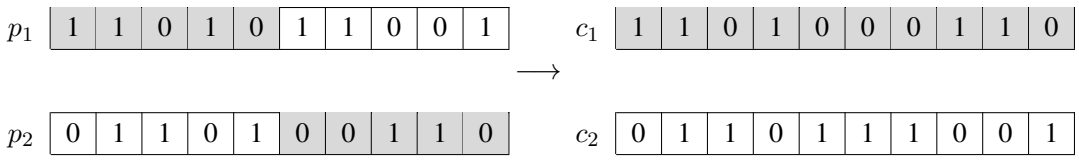


Figure 1.1: 1-point crossover

The N-point crossover is a generalized version of 1-point crossover which considers N crossover points. These N crossover points divide the offspring into $N + 1$ segments. Every alternate segment is swapped with other parent and two offspring are produced.

1.3.3.2 Uniform crossover

In uniform crossover [33], for each position in the offspring, individual decision is made regarding the value at that position. A random value from the interval [0, 1] is generated for each position and if this value is greater than the user defined value p, then the value at the position under consideration in offspring is copied from the corresponding position of first parent, otherwise second parent value is copied. The second child is generated by swapping the role of two parents. Figure 1.2 provides an illustration of uniform crossover for p value taken to be equal to 0.5.

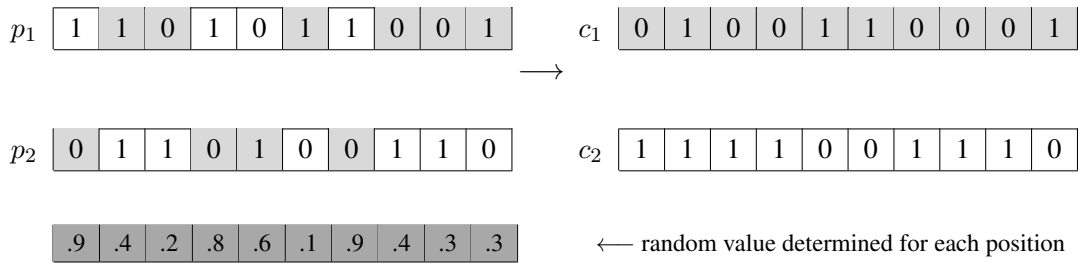


Figure 1.2: Uniform crossover

1.3.4 Mutation

Mutation operator intends to introduce diversity in the population. It mitigates the chances of premature convergence by preventing the solutions in population from getting too similar. Usually, this variation operator possesses explorative characteristics and responsible for exploration of new regions in search space. Like crossover operator, it also depends on the representation scheme used by GA. Literature contains numerous mutation operators designed for various representation schemes.

1.3.4.1 Bitwise mutation

Bitwise mutation is suitable for binary representation scheme. Each bit is considered individually and inverted with small probability. Figure 1.3 demonstrates bitwise mutation.

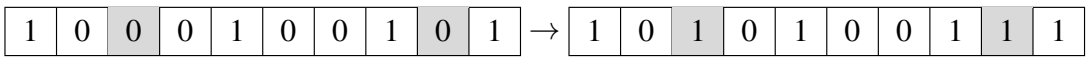


Figure 1.3: Bitwise mutation

1.3.4.2 Random reset mutation

It is an extended version of bitwise mutation suitable for integer representation scheme. Here also, each bit is considered individually and a new value taken from the permissible values of gene is assigned with small probability. Figure 1.4 provides a demo of random reset mutation where a gene can have permissible value from the set {1, 2, 3, 4, 5}.

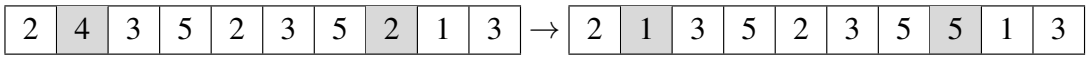


Figure 1.4: Random reset mutation

1.3.5 Population evolution models

Population evolution model governs the selection of individuals for next generation. Traditional GA follows generational model, whereas another popular model is steady-state model [34].

1.3.5.1 Generational model

In generational model, one generation produces number of individuals equal to size of the population. These individuals are used as the new population for next generation. This model may result in worse new population in comparison to current one. Thus, some GAs use an elitist strategy in which best solution or few best solutions from current population are passed to new population unaltered.

1.3.5.2 Steady-state model

Steady-state model generates only few solutions (usually just one individual) in each generation. These newly generated solutions replace same number of solutions in current population by following some replacement strategy. The replacement strategy takes the decision regarding the selection of those individuals which are going to be removed in order to create room for newly generated solutions. One such strategy is to remove the worst quality solutions from the population. Another replacement strategy considers the diversity factor and removes the solution that is most similar to the new one. Another feature of steady-state model is that duplicate solutions are forbidden in this model. The uniqueness of new solution with respect to current population members is always verified before including the new solution into population. The new solution is discarded in case it is same as an existing population member. Hence, multiple copies of same solution can never co-exist in the population. This helps in avoiding premature convergence that can occur due to same solution taking over entire population.

1.4 Genetic algorithm for permutation problems

Permutation problems intend to find the arrangement of a given set of objects into an optimal order. These problems can be classified into two types. In the first type of problems known as as order-based problems, the relative order among objects in important, whereas another category known as adjacency-based permutation problems, focus on the adjacency between objects. Several job scheduling problems fall into the class of first type of problems, on the other hand traveling salesman problem comes under second type [35]. Representation of individuals as a linear order of objects (permutation) is highly suitable for both types of permutation problems. In this representation, each position holds an unique object and each object must be present in the solution. As we discussed earlier that the genetic operators highly depend on the representation of individuals, thus, the crossover and mutation operators should be designed in such a manner so that the permutation property of individuals is always preserved.

1.4.1 Crossover operators for permutation problems

Literature contains various crossover operators which have been designed by considering the characteristics of permutation problems. The overview of some well-known crossover operators are provided below.

1.4.1.1 Partially matched crossover (PMX)

PMX was proposed by Goldberg and Lingle [36] for the traveling salesmen problem. However, it is suitable for any permutation problem and used in many applications. First, it selects two positions uniformly at random, and divide both parents into three parts. Then, the alleles between the selected positions (inner part) of one parent is swapped with the other and the repeated alleles in outer part are updated as per the mapping rules defined by the alleles in inner part. Figure 1.5 illustrates PMX crossover, where two positions, viz. 2 & 4 are selected randomly. The inner part of one parent is swapped with other parent. Afterwards, the repeated alleles are updated by using mapping rules defined by alleles in inner part, i.e., $\{3 \rightarrow 3, 1 \rightarrow 7, 6 \rightarrow 4\}$.

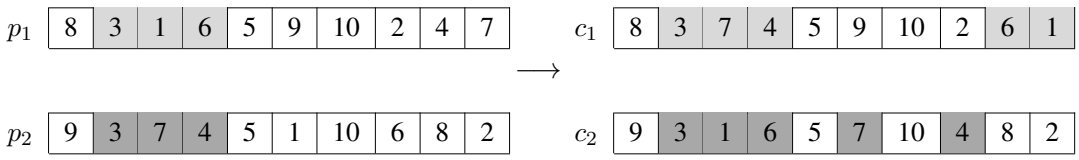


Figure 1.5: PMX Crossover

1.4.1.2 Order crossover

The order crossover operator was introduced by Davis [34] and well suited for circular permutation problems. Two positions (lets say r_1 and r_2) are selected uniformly at random. The portion of string between the selected positions from parent one is drop down at same positions in first offspring. The alleles copied from first parent in the offspring are removed from second parent. Starting from the next position after r_2 the remaining alleles of second parent are copied by assuming chromosome as circular. Another offspring is generated by altering the role of two parents and following the same approach. Figure 1.6 illustrates the order crossover by selecting two positions $r_1 = 3$ and $r_2 = 7$.

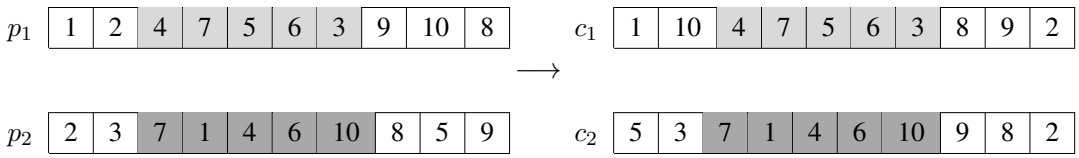


Figure 1.6: Order crossover

1.4.1.3 Uniform order based crossover

Uniform order based (UOB) crossover [34] is suitable for order-based permutation problems. Unlike order crossover, UOB considers each gene individually. It copies alleles at each position in first parent to the corresponding position in offspring with certain probability p. Thereafter, the remaining vacant positions in offspring are filled with the unselected alleles in second parent in the same sequence. The second offspring is generated by reversing the role of parents. Figure 1.7 demonstrates UOB crossover with p value equal to 0.5.

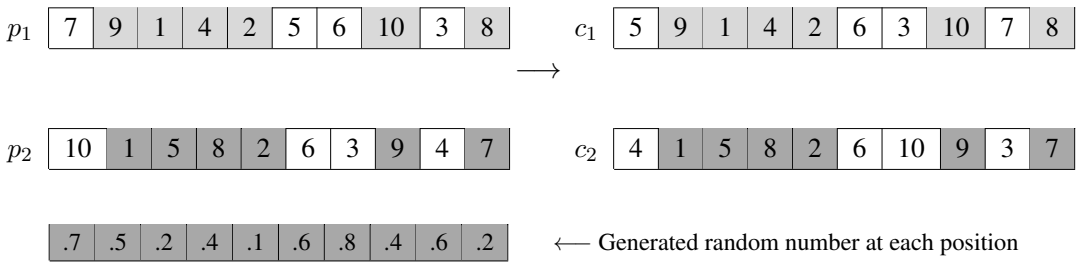


Figure 1.7: Uniform order based crossover

Cycle crossover [37] and Edge recombination crossover [38] are other important crossover operators for permutation based problems.

1.4.2 Mutation

Unlike simple genetic algorithm, where each gene is considered independently for mutation, in permutation representation, genes can not be considered independently. Mutation operator for traditional genetic algorithm assigns random values at some positions of chromosome without bothering about the values at other positions. Such a strategy is not suitable for permutation based problems, as each gene has a unique value in a chromosome. Thus, the mutation operator for permutation problems needs to be designed in such a manner that preserves the permutation property. In the literature, various mutation operators, suitable for permutation problems have been proposed. Swap mutation, insert mutation and scramble mutation are some commonly used mutation operators for order-based problems, whereas inversion mutation works well on adjacency-based permutation problems.

1.4.2.1 Swap mutation

Swap mutation first selects two random positions in chromosome and then the alleles at the selected positions are inter-changed. Figure 1.8 illustrates swap mutation, where allele at position 2 is interchanged with allele at position 6.

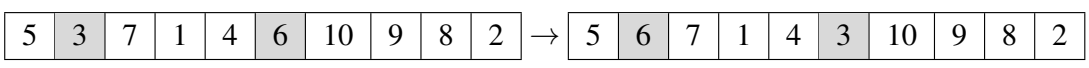


Figure 1.8: Swap mutation

1.4.2.2 Insert mutation

Insert mutation selects two random positions (p_1 & p_2) in chromosome. The allele at position p_2 is placed adjacent to the allele at position p_1 by moving all the alleles after p_1 one position towards the end. Figure 1.9 reveals the working of insert mutation, where $p_1 = 3$ and $p_2 = 8$.

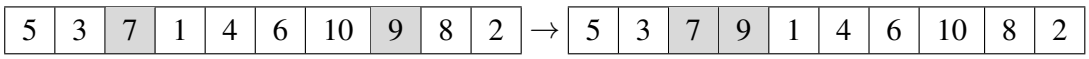


Figure 1.9: Insert mutation

1.4.2.3 Scramble mutation

In scramble mutation a substring of chromosome is selected by picking two positions randomly and the alleles present in selected portion are scrambled. For illustration, in Figure 1.10 the alleles present at position 4 to 7 are scrambled.

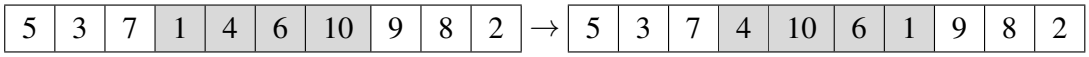


Figure 1.10: Scramble mutation

1.4.2.4 Inversion mutation

Inversion mutation performs well on adjacency-based permutation problems. In this mutation, a portion of chromosome is selected by randomly selecting two positions and the order of alleles in selected portion is reversed. Figure 1.11 depicts the inversion mutation.

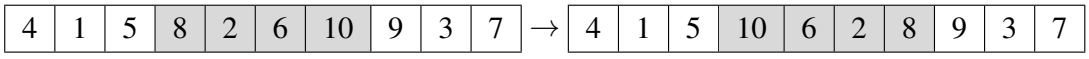


Figure 1.11: Inversion mutation

1.5 Grouping genetic algorithm (GGA)

Grouping problems also require suitable solution representation and variation operators due to their special grouping structure. Consider a solution representation that uses one gene per

object as used in traditional GA. For example: chromosome "ACBCDA" represents first object in group A, second in group C, third in group B, fourth in group C, fifth in group D and sixth in group A. One can infer that a given solution with g groups can be represented by $g!$ different chromosomes, each corresponding to one assignment of the names {A,B, ... } to groups [39]. Thus, using traditional GA solution representation schemes to address grouping problems can result in high degree of redundancy, which will result in poor performance of GA. In order to make GA suitable for grouping problems, Falkenauer [40, 41] presented grouping genetic algorithm. Grouping genetic algorithm (GGA) is an extended version of traditional GA where the solution representation and variation operators are suitably modified by considering the structure of grouping problems.

1.5.1 Representations in grouping genetic algorithm

For grouping problems, Falkenauer [40] proposed a solution representation scheme which consists of two parts. The former part provides the grouping of objects, whereas the latter one contains the list of groups. The idea behind including the groups into chromosomes is to make genetic operators use this information and perform a better exploration of search space. The given below example represents a chromosome for a grouping problem having six objects:

ACBCDACBDABD:ABCD

The first part provides the grouping of objects, i.e., the first object is in group A, second object is in group C, third in group B and so on. The second part provides a list of all groups {A,B,C,D}. In this representation, the chromosomes may have varying length as many problems such as bin packing problem, the number of groups are not static since the objective is to minimize number of bins. Literature also contains plenty of grouping problems where number of groups are fixed and the requirement is to divide the objects into different groups as per some objective. The advantage of this scheme is genetic operators can use the information presented in second part (group labels). But, such representation may induce redundancy, because group labels can be interchanged in group part without affecting the solution represented, e.g. ACBCDACBDABD:ABCD also represent the same solution as ACBCDACBDABD:DBCA, where the former part remains the same whereas the group part is altered. This ordering of groups in group part affects the performance of genetic algorithm and to reduce the effect of this ordering, an inversion operator was suggested. This scheme also suffers from another type

of redundancy when we change the labelling of groups, e.g., ACBCDACBDABD:ABCD and CADABCADBCDB:CDBA represent the same solution where objects 1, 6 and 10 belong to one group, objects 2, 4 and 7 to another group, objects 3, 8 and 11 to yet another group and objects 5, 9 and 12 to yet another group.

Other representation which can be used for grouping problems is to represent each chromosome as a set of groups [42] and each group contains a set of objects. For example, solution "ACBCDACBDABD:ABCD" can be represented as { {1, 6, 10}, {3, 8, 11}, {2, 4, 7}, {5, 9, 12} } in this scheme. This scheme does not label groups, and hence, does not suffer from the problems of redundancy like the scheme described in the previous paragraph. Another advantage of this scheme is that the genetic operators can be so designed that their results depend only on composition of groups, and not on relative ordering among groups. Hence, there is no need of inversion operator with this scheme.

1.5.2 Crossover

Different grouping problems may differ in constraints and the objective function to optimize. Thus, Falkenauer [40] presented a general pattern of crossover operator which can be used for grouping problems. Two crossover sites in each parent are chosen randomly. The two crossover sites delimit subset of the groups from each parent. The subset of group from first parent is inserted at the first crossover site of second parent. The redundant objects due to insertion of new groups are removed from the groups originally belonging to second parent. As the next step, suitable modifications, based on the objective function and constraints, are made in resulting offspring. By reversing the role of parents, another offspring can be generated in the same manner.

Crossover operator suitable for the second representation begins with an empty offspring and follows an iterative process to produce complete offspring [42]. In each iteration, one parent is chosen (deterministically or randomly) and then a group in this parent is selected. The selection of group can be done in a random manner or by following some problem/objective-specific knowledge. The selected group is added in offspring. The objects of selected group are removed from both parents. The same process is continued until some termination criteria satisfies or one of the parent becomes empty. In case some objects are left unassigned in offspring, some heuristic method can be used to add these objects in offspring.

1.5.3 Mutation

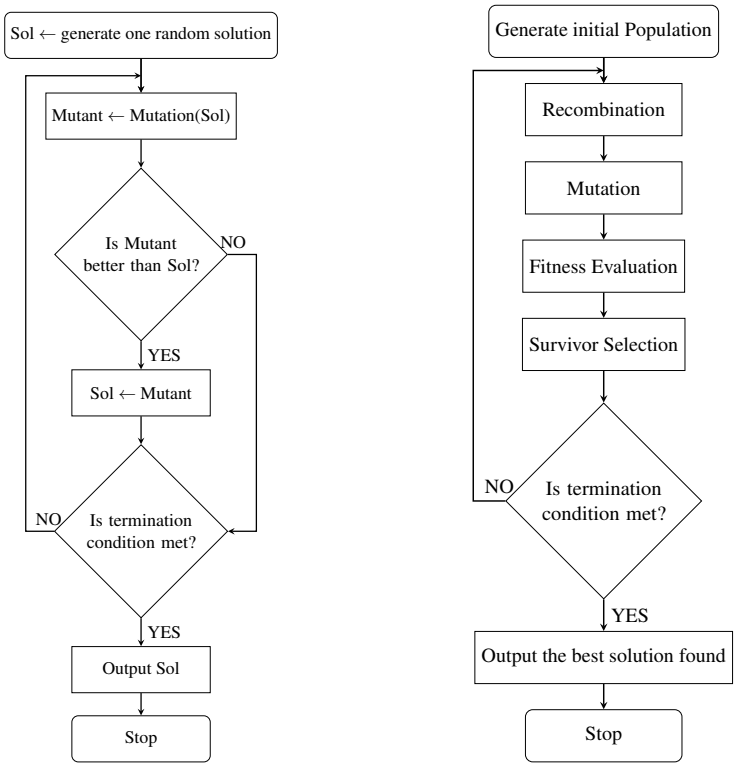
Similar to crossover operator, the design of mutation operator for a grouping problems highly depends on the particular problem. Following general strategies can be used for designing a mutation operator for grouping problems [40]:

- Delete some randomly selected group(s) from solution and then reassign the objects of the deleted group(s) by following some heuristic approach.
- Make a new group by selecting few objects (randomly or by following some specific approach) from different groups and deleting these selected objects from their respective original groups. If some groups become empty due to this process then delete these groups also from the solution.
- Delete some objects (randomly or by following some specific approach) from their respective groups and then reinserts these objects into solution by following some problem-specific heuristic.

1.6 Overview of evolution strategies

Evolution strategies fall into class of stochastic optimization algorithms based on Darwinian biological evolution. It mimics organic evolution motivated by nature such as, a change in environmental surroundings causes natural selection, resulting in an emergence of new population with higher fitness. Evolution strategies were introduced by Rechenberg [20] during the early 1960s and further developments were carried out by Schwefel [21]. Evolution strategies consider whole population as parents and process simultaneously the whole population, and the mutation operator is utilized as primary operator [43] to produce offspring. These properties make them unique and different from other evolutionary approaches, as most of the evolution based approaches select some solutions randomly from the population as parents, and in general, use recombination as primary operator. Furthermore, they possess a special feature known as self-adaptivity, where some parameters of the operators are varied as per fitness of the existing population during the run. This feature imparts them the ability to control the operators, so that a compromise can be reached between exploration and exploitation depending on the need. Now, many variants of evolution strategies exist in the literature.

The original version of evolution strategy proposed by Rechenberg [20] was *(1+1)-ES*, also known as *Two-membered ES* and later, the concept of population was introduced in *Multi-membered ES*. The *(1+1)-ES* has resemblance with hill climbing technique. It utilizes a simple *mutation selection scheme*, and maintains a single parent solution instead of a population of solutions. The parent is mutated to produce one offspring and the parent is replaced by the offspring whenever offspring has better fitness than the parent. The original version of ES has constant mutation parameter and also known as *basic two-membered ES* in the literature. Figure 1.12 (a) presents the flowchart of the *basic two-membered ES* approach where *Mutation()* is a function that performs the mutation. The self adaptability characteristic is added in *basic two-membered ES* by Schwefel [44] and this variant of two-membered ES got massive popularity due to the unique feature of self adaptability. In the self adaptable variant of two-membered ES, the mutation parameter has been governed by the fitness of the descendants produced in the past few generations [45, 46].



(a) Two-membered ES (b) Multi-membered ES

Figure 1.12: Flowcharts of two-membered and multi-membered evolution strategies

Multi-membered ES utilizes a population of μ individuals to generate λ offspring and employs recombination operator in addition to mutation operator. The concept of multi-parents population bestows the flexibility of imitating sexual reproduction by making use of recombination operator. First, for every descendant, a temporary parent is built by means of recombination of all parents, i.e., a temporary parent vector $Y = [y_1, y_2, \ldots y_m]$ is created where the i^{th} component of Y is chosen randomly from the i^{th} components of all μ parent vectors. Then λ descendants are produced by utilizing mutation operator on each temporary parent[47].The next step is the determination of survivors for next iteration. Figure 1.12 (b) presents the flowchart of the *multi-membered evolution strategies* approach. There are two variants of the survivor selection strategy, namely $(\mu + \lambda) - ES$ and $(\mu, \lambda) - ES$. In the $(\mu + \lambda)$ scheme, the fittest μ members are chosen from combined pool of parents and offspring, in a deterministic manner. Evidently, it is an elitist selection. The second approach viz. (μ, λ) scheme of ES, the fittest μ members are selected only from the λ offspring. This scheme rejects the complete pool of existing parents and provides opportunity to the offspring only to participate in survivor selection scheme. Apparently, in this approach λ must be greater than or equal to μ. It is to be noted that *(1+1)-ES* is a particular case of $(\mu + \lambda) - ES$ with $\mu = 1$ and $\lambda = 1$.

Evolution strategies were originally proposed to solve optimization problems in continuous domain and the initial variants of ES were also proposed for continuous domain only. However, [48] proposed some suitable changes in the mutation and recombination operators of the ES used in continuous domain, so that it can also be applied to the optimization problems in discrete domain. In continuous optimization problem, all constituents of the parent can be altered to small extent and these alteration in constituents of parent induces a small but significant mutation. It is analogous to natural biological evolution, since natural evolution also exhibits more frequent small mutations and rarely large ones. However, for discrete optimization problems, altering all constituents of the parents would not exhibit small mutation. This is supported by the fact that, in optimization problems of discrete domain, the constituents can be assigned with the values belonging to their corresponding discrete set and two adjacent discrete values may not have a small difference between them. Due to this, it is recommended in [48] to alter only some of the constituents of the parent to achieve small mutation in discrete optimization problem. Some modifications in recombination operator of ES, to make it applicable for problems in discrete domain, is also presented in [48]. Some recent works involving evolution strategies can be found in [49, 50, 51, 52, 53].

1.7 Overview of differential evolution

Differential evolution (DE) is a population based stochastic search method proposed by Storn and Price [18]. The traditional DE follows the floating-point encoding scheme and it was originally proposed to address continuous optimization problems. Initial population can be generated in a random manner if it is an unknown problem. In case of availability of some preliminary solutions of the problem, initial solution can be created by adding normally distributed random deviations to the available preliminary solutions. DE uses similar computational steps as used by any other standard evolutionary algorithm. However, it differs from other EAs in the generation of new solutions. It follows an entire different approach in which a new perturbed solution is generated by adding the weighted difference of two randomly selected distinct population members with the third one. It can be mathematically presented as:

$$S_P \ = \ S_A + w(S_B - S_C)$$

Here, S_A, S_B, S_C are three randomly chosen distinct population member and w is a user defined weighting factor. DE has gained vast popularity due to its ease of implementation, generic structure and robustness. Consequently, literature contains many variants of basic DE which differ in the strategies such as number of solutions involved in perturbation, type of crossover, mutation adopted as per the type of problem. Reader can refer to [54] for a detailed survey on differential evolution.

Evidently traditional DE can not be applied on discrete optimization problem. Tasgetiren et al. proposed a novel discrete differential evolution (DDE) in [19, 55], where solutions are based on discrete values and this version of differential evolution is having compatibility with all types of discrete combinatorial optimization problems. In DDE approach, each solution of the population is considered one-by-one. The solution under consideration is called the target solution. The mutant solution is generated by perturbing the best solution or any other solution (random or target solution) in the population [56]. The crossover operator is applied on the mutant and the target solution with some probability and a trial solution is generated. Subsequently, a selection mechanism is used to evaluate the fitness value of both the competing solutions, i.e., the target and the trial solutions in order to find out which solution will be going to survive for the next generation. Usually, if the trial solution is better than the target solution then the target solution is replaced by the trial solution. The pseudo-code of the discrete differential

evolution is provided in Algorithm 2, where p is the population size and *Initial_Solution()* is a function that generates an initial solution.

In general, most of the other evolutionary techniques select both parents from the population, whereas in DDE one parent is selected from the population and the other parent is obtained by perturbing another solution from the population. Usually, the best solution of the population or a random solution of the population is perturbed to obtain the second parent. As a result, one parent used in recombination operator is a diverse solution most of the times. This provides two advantages. First, it provides a better exploration of search space, and thus, helps in avoiding premature convergence. Second, it is observed that most of the crossover operators yield offspring similar to parents, if both parents share high similarity. Thus perturbing a solution from the population and then recombining with other solution can be considered a better strategy. Due to its salient features, literature contains numerous application of DDE on permutation based problems. The articles in [55, 57, 58, 59, 60, 61, 62] report some of the applications of DDE for permutation based problems.

Algorithm 2: Pseudo-code of DDE

Input: DDE parameters and an instance of problem under investigation
Output: Best solution obtained through DDE

```
for (i=1 to p) do
    S_i ← Initial_Solution();
best ← best solution among S_1, S_2, ... S_i, ..., S_p;
while (the termination criteria remains unsatisfied) do
    for (i=1 to p) do
        M ← Mutant(S);
        // S can be the best solution or a random solution in
           population
        T_i ← Crossover(M,S_i);
        // S_i is the target solution & T_i is the trial solution
        if (T_i is better than S_i) then
            S_i ← T_i;
            if (S_i is better than best) then
                best ← S_i;
return best;
```

1.8 Overview of NEH heuristic

The NEH heuristic was proposed by Nawaz et al. in [63], and the literature reveals its effectiveness in terms of addressing various permutation flowshop problems. In general, the objective of flow shop sequencing problem is to find a permutation of jobs that minimizes the makespan, i.e., the completion time of the last job. Following steps summarize the NEH heuristic assuming that i jobs are being processed on j machines by keeping the same relative order.

1. Sort i jobs as per the decreasing sums of their processing times on j (all) machines. Consider that the sorted permutation of jobs is π. The job with more total processing time is given preference over a job with less total processing time, and that is the idea behind sorting the jobs [63]. This step is skipped when there is no clear criteria to determine preference among jobs or when there is a need to find more than one solution. Instead, the next two steps use a randomly generated sequence of jobs.

2. Choose the jobs at position one and two from π and determine the best sequence of these two jobs by evaluating the fitness of two possible permutations of these. Keep the best sequence as partial solution for the next step.

3. This phase iteratively build the complete solution by picking k^{th} job, $k = (3, \ldots, i)$ and placing in k possible locations in the partial solution, resulting in k partial solutions. All of the k partial solutions fitness are evaluated and the best one among them is kept as partial solution for next iteration. This phase is continued until a complete solution is obtained.

1.9 Overview of opposition based solution generation

Opposition based optimization was proposed in [64] for the faster convergence of evolutionary algorithms. Literature reveals that, the idea of considering opposite solutions facilitates an extensive search space as compared with purely random solutions and hence, a more diverse initial population can be obtained. The concept of opposition based solution generation in [64] was proposed for the continuous optimization problems. Let us assume that $S = (y_1, y_2, \ldots, y_N)$ is a solution in an N-dimensional space where $y_1, y_2, \ldots, y_N \in \Re$, where each dimension j is bounded by a lower limit l_j and an upper limit u_j, i.e., $y_j \in [l_j, u_j]\ \forall j\ \in \{1, 2, \ldots, N\}$, then

the computation of opposite solution $S' = (y'_1, y'_2, \ldots, y'_N)$ corresponding to solution S can be done in the following manner:

$$y_j \;=\; l_j + u_j - y_j \;\; \forall j \in \{1, 2, \ldots, N\} \tag{1.1}$$

Since its introduction, the idea of opposite solutions has been used in a number of studies, e.g. [65, 66, 67, 68, 69].

For a permutation based combinatorial optimization problem, the opposite solution can not be computed in the manner described above for optimization problems in continuous domain. For the permutation problems discussed in the thesis, to find the corresponding opposite solution a permutation needs to be carefully reordered. To compute the opposite solution for a permutation, the element at the j^{th} position in the permutation, where $j \in \{1, 3, 5, \ldots, \lfloor \frac{m}{2} \rfloor\}$, is exchanged with the element at $(m - j + 1)^{\text{th}}$ position in the permutation. Figure 1.13 illustrates the computation of an opposite solution in the context of problems discussed in this thesis.

A given solution	1	2	3	4	5	6	7	8	9	10
Corresponding opposite solution	10	2	8	4	6	5	7	3	9	1

Figure 1.13: Illustrating the concept of opposite solution in case of permutation based COP

1.10 Scope of the thesis

This thesis is focused on solving some $\mathcal{NP}$-hard permutation based combinatorial optimization problems using three evolutionary techniques, viz. genetic algorithm (GA), evolution strategy (ES) and discrete differential evolution (DDE). We have addressed six $\mathcal{NP}$-hard permutation based problems which are recent problems introduced during the last decade. Out of these six problems, first three problems are pure permutation problems, and the last three problems have aspects of permutation as well as grouping. First four problems have single objective to optimize, second last problem has two independent objectives and the last problem is a multiobjective optimization problem. Table 1.2 presents the categorization of these problems based on the type of objective of the problem and its COP characteristic.

This thesis is divided into eight chapters. The thesis begins with this introductory chapter which is followed by six chapters devoted to six problems considered, and, then a concluding chapter at the end. In the following, we summarize the content of each of these chapters.

Table 1.2: Categorization of the problems, their type of the objective and combinatorial optimization characteristics

Problem name	Objective type	Combinatorial optimization characteristics
Cover scheduling problem in wireless sensor networks (WSN-CSP)	Single objective	Permutation
Total rotation minimization problem in directional sensor networks (TRMP)	Single objective	
Single machine total stepwise tardiness problem with release dates (SMTSTP-R)	Single objective	
Rescue unit allocation and scheduling problem (RUASP)	Single objective	Permutation & grouping
Quality of service vehicle routing problem with time windows (QSVRPTW)	Two independent objectives	
Multiobjective vehicle routing problem with time windows (MOVRPTW)	Multiobjective	

Chapter 2 addresses the cover scheduling problem in wireless sensor networks (WSN-CSP). Wireless sensor networks (WSNs) have a wide range of applications such as warzone surveillance, air pollution monitoring, fire monitoring in forests, ecological and geological monitoring in deep sea. The use of WSNs for data gathering in remote or hostile environments is quite common nowadays. The accurate placement of sensors in hostile environments is not possible due to the risks/costs involved, and hence, they are usually placed in an ad hoc or random manner. Random deployment may not be able to cover all the targets, and hence, to cope up with this, usually more number of sensors are deployed as compared to the actual requirement. This excess deployment also facilitates more resilience to faults as some targets are redundantly monitored by more than one sensor. Actually, each sensor is driven by a battery, with a fixed amount of stored energy in it. Replacing the drained out batteries with a new one are not possible in remote or hostile environments and, hence it results in an upper limit on the total lifetime of a sensor specified by the energy storage capacity of its associated battery. Therefore, in the aforementioned situation, the primary goal in the design of WSNs is to maximize the network lifetime by using the existing resources in an efficient manner. To deal with the problem of lifetime maximization in WSNs, most of the existing approaches make use of the excessively deployed sensors, by dividing the set of sensors into various non-disjoint subsets, also known as *covers* or *groups* in the literature, in such a manner that sensors belonging to each cover are able to monitor all the targets on their own. The work duration of each cover is also determined in such a manner that the sum total of work durations of all the covers is as large as possible and no sensor monitors the targets beyond the operating life dictated by its battery. These covers

are then used for their predetermined work durations in a sequential and mutually exclusive manner. At any instance of time, the sensors of currently active cover is only used to monitor the targets and remaining sensors which are not in currently active cover are considered to be in an inoperative state where they consume negligible energy. This cover based approach for WSNs yields a better network lifetime.

The WSN-CSP problem arises in those sensing environments which permit the coverage breach, i.e., at any instant of time, all the targets need not be monitored. The coverage breach may occur owing to either technical restrictions such as bandwidth limitations or intentionally. This problem seeks a schedule of covers which minimizes the longest continuous duration of time for which no sensor in the network is able to monitor a target. In this chapter, we have proposed a *two-membered evolution strategy*, also referred to as *(1+1)-ES* in the literature, for WSN-CSP. We have used *NEH heuristic* and the concept of opposition based solution generation, to generate a superior quality initial solution. Unlike other evolutionary algorithms which employs both recombination as well as mutation, the *(1+1)-ES* utilizes mutation alone. In each iteration, the mutant solution is obtained by applying the mutation operator on the present solution, and the more fit solution between the present and the mutant becomes the present solution for the next iteration. In our proposed approach, we have presented three mutation operators for maintaining an adequate balance between exploration and exploitation. The first and third mutation operators have fixed as well as low mutation step size and can be seen as the operators performing exploitation, where as second mutation operator has variable mutation step size and due to this it has been used for exploration as well as exploitation by appropriately selecting the step size. For our approach, we have extended the basic *(1+1)-ES* by adding a reshuffle procedure to escape from a locally optimal solution. If there is no improvement in the fitness of the solution, for a certain number of consecutive iterations, then the reshuffle procedure is invoked to get a new solution which replaces the present solution. To evaluate the performance of our proposed approach, we have used the same set of standard benchmark instances as used by the state-of-the-art approaches for WSN-CSP in literature. Computational results show the effectiveness of our approach.

Chapter 3 is concerned with the total rotation minimization problem (TRMP) pertaining to directional sensor networks. Unlike omnidirectional sensors, a directional sensor at any specific instant of time can observe targets in an angular sector only (also known as its working direction at that instant). The directional sensors primarily include infra-red sensors, ultrasonic sensors and video sensors. Inability of directional sensors to monitor all surrounding targets at any

instant of time does not preclude the possibility of monitoring all surrounding targets at different instants of time. To facilitate this, each directional sensor is equipped with a device that can rotate it, thereby changing its working direction. With such an arrangement, working direction of a directional sensor can be adjusted as per the application's requirements. As discussed previously, the groups or covers are obtained by dividing the set of sensors into non-disjoint subsets. Each group contains a set of sensors which are capable to monitor all the targets in their vicinity. In case of the directional sensor networks, each group not only contains list of its constituent sensors, but also the working direction for each one of them. As the groups are activated one-by-one, sensors may need to rotate so as to face their working directions dictated by the currently active group. It is to be noted that only those sensors need to be rotated which are present in the currently active group and which do not face the working direction dictated by this group. However, rotation of a directional sensor from one direction to another consumes energy, and hence, the order in which groups are scheduled does matter in directional sensor networks. This chapter is concerned with how to schedule a given set of groups so that the energy consumption due to rotations of sensors is minimized.

To address TRMP, we have again proposed a *(1+1)-ES* which share some features with the approach proposed in previous chapter. Our *(1+1)-ES* approach has incorporated a pre-processing step, which is used to boost the performance of the evolution strategy (ES). In addition, a reshuffle procedure is also used to diversify the search process and escape from local optima. In the pre-processing step, we have applied NEH heuristic on some randomly generated solutions and their corresponding opposite solutions, and the best solution obtained is passed as input to *(1+1)-ES*. The proposed *(1+1)-ES* employs a mutation operator that is based on a destruction and reconstruction strategy where a solution is partially destroyed and then reconstructed following a problem-specific greedy strategy. The performance of the proposed *(1+1)-ES* with and without the pre-processing step has been compared with the state-of-the-art approach available in the literature on the same set of instances as used by the state-of-the-art approach. Computational results demonstrated that our *(1+1)-ES* based approach both with and without pre-processing step is better than the state-of-the-art approach available in the literature.

Chapter 4 presents a novel hybrid discrete differential evolution based approach for a single machine scheduling problem where each job has a release date and the tardiness cost of the job increases stepwise with respect to its various due dates. In the literature, this problem is termed as the single machine total stepwise tardiness problem with release dates (SMTSTP-R). A job incurs tardiness if the job is completed after its due date. The classical scheduling problem

based on tardiness criterion considers only a single due date for each job. The tardiness cost of the job is usually a linear function of its completion time. In contrast, in stepwise tardiness scheduling problem, each job has various due dates and the tardiness cost increases in stepwise manner with respect to various due dates. Stepwise tardiness cost is natural in many real-world applications mostly pertaining to transportation. Usually, the transportation services follow a time schedule, and hence, the transportation service is not always available. If a job is finished after the current transport service departs, then the finished job has to compulsorily wait till the availability of the next transport service. As a result, such a job gets delivered to the end customer at the same time no matter where it exactly finishes after the current service departs and before the availability of the next service. Hence, such a job has the same tardiness cost no matter what is its exact completion time.

To address SMTSTP-R, we have presented a hybrid approach combining discrete differential evolution with a series of local searches. We have used several heuristics and the concept of opposition based solution generation to generate initial population. The perturbation procedure is used to evade from local optima. A specific uniqueness procedure is designed to check the uniqueness of the solution. The local search is used only if the generated solution is with in certain range of the current best solution. The computational results demonstrate the superiority of the proposed DDE approach over the state-of-the-art approaches for SMTSTP-R.

Chapter 5 is devoted to rescue unit allocation and scheduling problem (RUASP) with fuzzy processing times. It is a generalization of unrelated parallel machine scheduling problem with sequence and machine dependent setup times. The problem consists of *m* rescue units and *n* incidents, such that $m \leq n$. Each incident must be processed by exactly one rescue unit. Each rescue unit has different capability and can cater to some specific incident(s) only. Different rescue units take different travel times to travel from one incident to another. It considers fuzzy processing times which are both incident-specific and unit-specific. Each incident *i* has a severity factor represented by w_i. The objective is to find the schedule of incidents for each rescue unit so that the sum of all weighted completion time ($\sum_{i=1}^{n} w_i c_i$) is minimized. RUASP focuses on the efficient allocation and scheduling of rescue units in case of emergency response such as natural disasters. It is related to both routing as well as scheduling problems.

To address RUASP, we have proposed a steady-state grouping genetic algorithm based approach, which uses several constructive heuristics to generate initial population. The purpose of using these heuristics is to find better quality initial solutions to start with. We have designed crossover and mutation operators keeping in mind the characteristics of RUASP. Computational

results on the benchmark instances demonstrate the superiority of our approaches over the state-of-the-art approach in terms of solution quality as well as in terms of running time.

Chapter 6 addresses the vehicle routing problem with time windows (VRPTW) designed for objectives seeking the quality of service to the customers. VRPTW is a generalization of vehicle routing problem (VRP) where a customer can only be served with in its time window. It is one of the most extensively researched variant of VRP due to its resemblance with various real-world applications. The VRPTW variant addressed in this chapter considered the objectives which aimed at measuring the quality of service to the customers. This problem in the literature is known as Quality of Service Vehicle Routing Problem with Time Windows (QSVRPTW). The problem addresses two objectives: the first objective minimizes the total amount of time customers have to wait in their time windows and the other objective minimizes the average of relative time that customers have to wait with in their time window to get the service.

To address QSVRPTW, we have presented a steady-state grouping genetic algorithm based approach, which uses greedy genetic operators and, several constructive heuristics for initial population generation. The proposed heuristics can be classified into three categories. This classification is based on the quality of solutions generated by the particular heuristic. Three heuristics are designed by aiming at better quality initial solutions which aids in faster convergence of the proposed approach. Fourth heuristic generates a random feasible solution and the last heuristic is used to find a feasible solution in case all other heuristics fail to find a feasible solution. We have also proposed two lower bounds for each objective. The proposed approach provides better results in very less computation time as compared to the state-of-the-art approach.

The penultimate chapter of the thesis, viz. Chapter 7 is concerned with a multiobjective optimization problem. It is a general multiobjective VRPTW (MOVRPTW) with five objectives along with MOVRPTW benchmark instances that are derived from real-world data. The five objectives are number of vehicles, total traveled distance by all vehicles, longest travel time among all routes, total waiting time due to early arrivals and total delay time due to late arrivals respectively. These objectives are conflicting in nature, i.e., the optimization of one objective may lead to deterioration in the values of other objectives. Also, the relative importance of different objectives may vary from one domain to another or from one scenario to another. Therefore, instead of finding a single solution for this VRPTW, it is desirable to present a set of solutions having trade-offs among different objectives.

In this work, we have used the nondominated sorting genetic algorithm II (NSGA-II) framework with objective-wise crossover and mutation operators. We have developed greedy variation operators which are not only problem-specific, but also objective-specific, i.e., our crossover and mutation operators are designed as per the characteristics of MOVRPTW and the characteristics of each objective. Hence, we have a dedicated crossover operator and a dedicated mutation operator for each objective, thereby leading to five different crossover operators and five different mutation operators. To assess the superiority of our approach, we have used three performance metrics, viz. inverted generational distance (IGD), hypervolume (HV), coverage metric (C-metric). The computational results demonstrate that the proposed approach is better than the state-of-the-art approach for MOVRPTW in terms of IGD and HV. Performance of the proposed approach, particularly in terms of C-metric, can be improved further at the expense of increased execution times if the objective-specific local searches like those used in literature are incorporated in our approach.

Chapter 8 concludes the thesis by summarizing the contributions of the thesis. In addition, some guidelines and directions for future research are also provided.

Chapter 2

Cover Scheduling Problem in Wireless Sensor Networks

2.1 Introduction

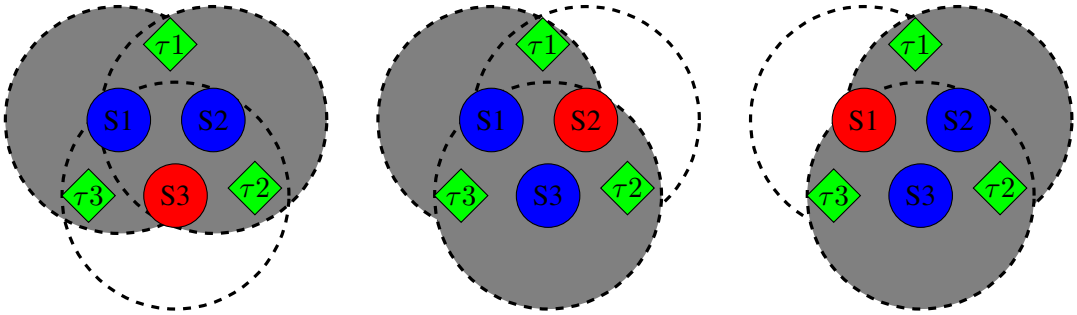
In this chapter, we have addressed the cover scheduling problem in wireless sensor networks (WSN-CSP). Wireless sensor networks (WSNs) have a wide range of applications such as war-zone surveillance, air pollution monitoring, fire monitoring in forests, ecological and geological monitoring in deep sea. The use of WSNs for data gathering in remote or hostile environments is quite common nowadays. The accurate placement of sensors in hostile environments is not possible due to the risks/costs involved, and hence, they are usually placed in an ad hoc or random manner. Random deployment may not be able to cover all the targets, and hence, to cope up with this, usually more number of sensors are deployed as compared to the actual requirement. This excess deployment also facilitates more resilience to faults as some targets are redundantly monitored by more than one sensor. Actually, each sensor is driven by a battery, with a fixed amount of stored energy in it. Replacing the drained out batteries with new ones are not possible in remote or hostile environments and, hence, there is an upper limit on the lifetime of each sensor specified by the energy storage capacity of its associated battery. Therefore, in the aforementioned situation, the primary goal in the design of WSNs is to maximize the network lifetime by using the existing resources in an efficient manner. To deal with the problem of lifetime maximization in WSNs, most of the existing approaches make use of the excessively deployed sensors, by dividing the set of sensors into various non-disjoint subsets, also known as *covers* or *groups* in the literature, in such a manner that sensors belonging to each cover are able

to monitor all the targets on their own. The work duration of each cover is also determined in such a manner that the sum total of work durations of all the covers is as large as possible and no sensor monitors the targets beyond the operating life dictated by its battery. These covers are then used for their predetermined work durations in a sequential and mutually exclusive manner. At any instance of time, the sensors of the currently active cover are only used to monitor the targets and remaining sensors which are not in currently active cover are considered to be in an inoperative state where they consume negligible energy. This cover based approach for WSNs yields a better network lifetime because of the following factors:

- The consumption of energy by an active sensor is much more as compared to the energy consumed by an inactive sensor [70, 71].
- It has been shown that if a sensor is used in a frequently oscillating manner between on and off states, then the sensor's battery lasts for longer duration. Actually, a battery which is used in many short bursts with significant intermediate inactive times may have twice the lifetime in contrast to the situation where the same battery is used in a continuous manner [72].

The problem of maximizing the lifetime of WSNs has gained a lot of attention during the last decade, consequencing in plentiful literature on determining the set of covers and their corresponding work durations. In most applications of WSNs, the sequence of activation of covers is of no importance. This is substantiated by the fact that all targets are fully covered by each cover, and hence, these covers can occupy any position in the schedule. Hence, the term 'scheduling' by default means the generation of covers and determining their corresponding activation times [73], and the actual sequence of covers remains unspecified in most of the literature pertaining to WSNs. For the sake of illustration, consider an example of WSN with three sensors S1, S2 and S3 monitoring three targets $\tau 1$, $\tau 2$ and $\tau 3$ such that S1 can monitor $\tau 1$ and $\tau 3$, S2 can monitor $\tau 1$ and $\tau 2$ and S3 can monitor $\tau 2$ and $\tau 3$. Hence, each target is redundantly monitored by more than one sensor. Also assume each sensor has a operating life of 1 time unit. If all the sensors are used simultaneously to monitor the three targets then we will get a network lifetime of 1 time unit only. However, dividing the set of these three sensors into three covers, viz. {S1, S2}, {S1, S3} and {S2, S3} and activating each cover for 0.5 time units will provide a network lifetime of 1.5 time units. Figure 2.1 illustrates the activation of these covers where Figure 2.1(a), Figure 2.1(b) and Figure 2.1(c) show the activation of covers {S1, S2}, {S1, S3} and {S2, S3} respectively. In this figure, targets are shown by green

squares, active sensors by blue circles and inactive sensors by red circles. Sensing range of each sensor is shown by a dotted circle centred at that sensor. The dotted circle corresponding to a sensor is grey if it is in active state. It is to be noted that covers can be activated in any order, and sequencing of Figure 2.1(a), Figure 2.1(b) and Figure 2.1(c) is only for definiteness in illustration.



(a) Sensors S1 and S2 are in active state for 0.5 time units

(b) Sensors S1 and S3 are in active state for 0.5 time units

(c) Sensors S2 and S3 are in active state for 0.5 time units

Figure 2.1: Illustrating activation of covers in a WSN

However, this scenario changes drastically when considering bandwidth constraints in WSNs, i.e., when the maximum number of sensors which can send the data to central server simultaneously, is restricted to ω. In such a situation, the full coverage of targets cannot be guaranteed due to the restriction on the size of cover by the value of ω. If a target is not covered by any of the sensor present in the currently active cover, then the target is said to be *breached*. The bandwidth constraints in the WSNs lead to two engaging problems based on the level of compromise between two competing factors, viz. network lifetime and coverage breach. The objective of the first problem (termed as MNLB in literature) is to maximize the network lifetime with the bandwidth constraints bounded by an upper limit on the breach rate, whereas the latter problem (termed as MCBB in literature) has the objective to minimize the total breach time during the entire life under the bandwidth constraints subjected to a lower limit on network lifetime [74, 75]. The breach rate is defined as the average breach time per target as a fraction of total lifetime. The value 0 of breach rate specifies that all targets are fully monitored during the complete lifetime of network, and the value 1 of breach rate specifies that none of the target is monitored during the complete lifetime. In general, a breach rate of $\nu \in (0, 1)$ can indicate any scenario between the following two extreme scenarios [76]:

1. All targets except ν fraction of the total targets are always monitored by some sensor in the network.

2. Each target remain uncovered exactly for ν fraction of the network lifetime.

A column generation based approach [76] can provide an efficient solution to MCBB and MNLB problems. However, such an approach only produces the covers along with their respective work durations, and the actual schedule of these covers again remain undetermined. MCBB and MNLB problems are examples of those class of problems where technical constraints lead to coverage breach. There are few cases where coverage breach is permitted purposely to enhance the network lifetime. For instance, [77] presented a lifetime maximization problem in WSNs, where some fractions of targets remain unmonitored in every cover, so that an increase in the network lifetime can be achieved.

However, the actual sequence in which covers are scheduled becomes important when coverage breach is permitted. It is due to fact that, different schedules of covers may have different targets breached for different continuous duration of time, and hence, it is preferable to have a schedule in which none of the target has continuous breach for a long timespan. This chapter deals with the problem of scheduling a given set of covers in such a manner that the longest continuous target breach is minimized. In the literature, this problem is known by the name *cover scheduling problem in wireless sensor networks (WSN-CSP)*.

Rossi et al. [78] introduced the *WSN-CSP* and proved its $\mathcal{NP}$-hardness. In [78], *WSN-CSP* is addressed using a greedy heuristic (GH) and a genetic algorithm (GA) augmented with a local search. A lower bound is also presented which is used to assess the number of optimal solutions achieved by each proposed approach. GA performed much better in comparison to GH. Gopinadh et al. [79] developed two hybrid swarm intelligence approaches utilizing artificial bee colony algorithm (ABC) and invasive weed optimization algorithm (IWO) for *WSN-CSP*. These ABC and IWO based approaches obtained better quality solutions in comparison to the approaches of [78], and, ABC based approach performed better than IWO based approach.

In this chapter, we present a *two-membered evolution strategy* (*(1+1)-ES*) to address *WSN-CSP* problem. In our proposed approach, we have used *NEH heuristic* [63] and concept of opposition based solution generation, to generate a good quality initial solution. Initially a random solution is generated and its corresponding opposite solution is determined. On both solutions, we have employed NEH heuristic and the fitness of two new resulting solutions is evaluated. The better one between them is considered as initial solution for the proposed

approach. The proposed approach has used three mutation operators in a mutually exclusive manner. Our mutation operator uses ruin and recreate strategy, where the original solution is partially ruined and then again recreated by using a problem-specific greedy approach. Furthermore, a reshuffle procedure is used to circumvent the solution from stucking in a local optima. We have compared the performance of our approach with the approaches available in the literature, viz. those presented in [78, 79]. The computational results clearly demonstrate the superiority of our approach.

The remaining part of this chapter is structured as follows. Section 2.2 presents the formal definition and an illustrative example of the *WSN-CSP*. Section 2.3 discusses the related works. The proposed *(1+1)-ES* approach for the *WSN-CSP* is described in Section 2.4. The computational results are presented in Section 2.5. Finally, Section 2.6 presents some concluding remarks.

2.2 Problem definition

This section introduces the notational conventions and provides a formal definition of *WSN-CSP*. The input to *WSN-CSP* consists of a set of covers, their corresponding work durations and the set of targets monitored by each cover. The objective of *WSN-CSP* is to find a schedule of covers which minimizes the longest continuous duration of time for which a target is breached. Clearly, a schedule is a linear permutation of covers where the positions of the covers specify the order in which covers are activated.

Formally, given a set D of n targets, viz. $D = \{D_1, D_2, \ldots, D_n\}$, a set G consisting of m covers, viz. $G = \{G_1, G_2, \ldots, G_m\}$ along with their respective work durations $\{t_1, t_2, \ldots, t_m\}$ and the sets $P_i = \big\{D_j : (D_j \in D) \wedge (G_i \text{ covers } D_j)\big\}$, $\forall i \in \{1, 2, \ldots, m\}$. Let $Q_j = \big\{G_i : (G_i \in G) \wedge (G_i \text{ covers } D_j)\big\}$, $\forall j \in \{1, 2, \ldots, n\}$. Hence, P_i is the set of targets covered by cover G_i, and, Q_j is the set of covers which cover target D_j. The sets Q_j, $\forall j \in \{1, 2, \ldots, n\}$ can be computed from sets P_i, $\forall i \in \{1, 2, \ldots, m\}$ given as input. A target D_j is breached for duration t_i whenever the currently active cover G_i does not belong to Q_j. Given a schedule of covers $\pi = G_{\pi_1} G_{\pi_2} \ldots G_{\pi_m}$, the target D_j can be breached for a continuous duration longer than the work duration of a single cover, if two or more consecutive covers in the schedule π do not belong to Q_j. Let T_j, $\forall j \in \{1, 2, \ldots, n\}$ be the longest continuous

duration of time for which target D_j is not covered by any cover in the schedule, i.e.,

$$T_j = \max \sum_{\{\pi_k : G_{\pi_k} \notin Q_j, \forall 1 \leq k_1 \leq k \leq k_2 \leq m\}} t_{\pi_k}, \quad \forall j \in \{1, 2, \ldots, n\} \tag{2.1}$$

where k_1 and k_2 provide the starting and ending positions in the schedule such that all covers between these two positions in the schedule do not cover D_j. Let Δ_{max} be the longest continuous duration of time for which a target is breached in a schedule, i.e.,

$$\Delta_{max} = \max(T_1, T_2, \ldots, T_n) \tag{2.2}$$

The *WSN-CSP* seeks a schedule of covers that minimizes Δ_{max}.

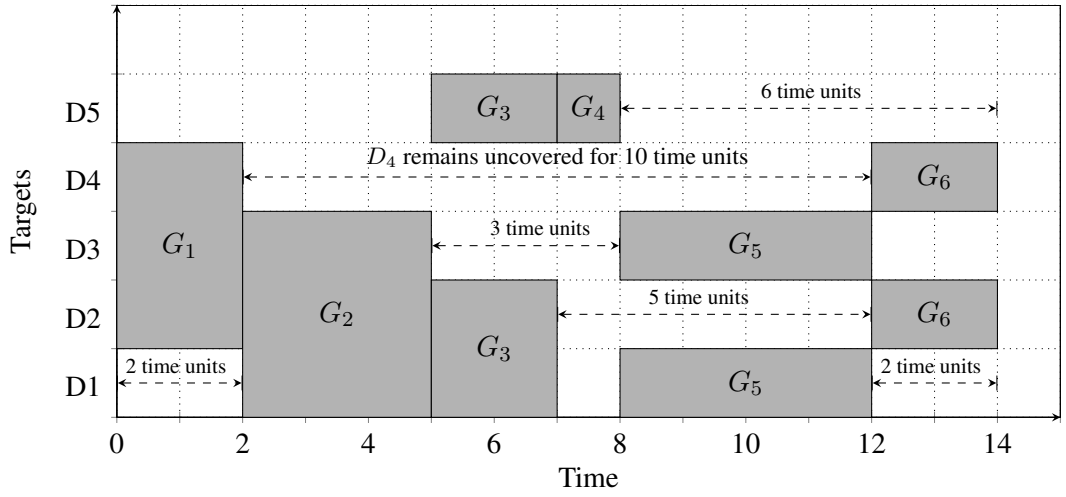


Figure 2.2: Illustrating WSN-CSP assuming covers are scheduled as per their natural order, i.e., $G_1, G_2, G_3, G_4, G_5, G_6$

To illustrate *WSN-CSP*, let us consider an example consisting of five targets and six covers, i.e., n = 5 and m = 6 such that $D = \{D_1, D_2, D_3, D_4, D_5\}$, $G = \{G_1, G_2, G_3, G_4, G_5, G_6\}$ with their respective working durations $\{2, 3, 2, 1, 4, 2\}$ and $P_1 = \{D_2, D_3, D_4\}$, $P_2 = \{D_1, D_2, D_3\}$, $P_3 = \{D_1, D_2, D_5\}$, $P_4 = \{D_5\}$, $P_5 = \{D_1, D_3\}$ and $P_6 = \{D_2, D_4\}$. On the basis of given information, $Q_1 = \{G_2, G_3, G_5\}$, $Q_2 = \{G_1, G_2, G_3, G_6\}$, $Q_3 = \{G_1, G_2, G_5\}$, $Q_4 = \{G_1, G_6\}$, $Q_5 = \{G_3, G_4\}$. Figure 2.2 illustrates a solution of *WSN-CSP*, where covers are scheduled in their natural order, viz. $G_1, G_2, G_3, G_4, G_5, G_6$. From this figure, it can be clearly seen that T_1 = 2 (k_1 = 1, k_2 = 1), T_2 = 5 (k_1 = 4, k_2 = 5),T_3 = 3 (k_1 = 3, k_2 = 4), T_4=10 (k_1 = 2, k_2 = 5), T_5 = 6 (k_1 = 5, k_2 = 6), and, hence Δ_{max}, i.e., the longest continuous

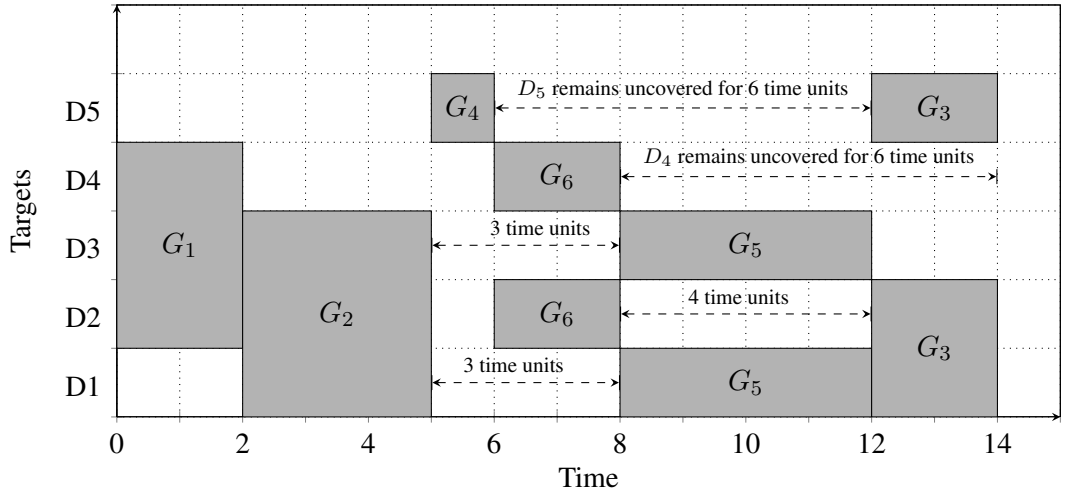


Figure 2.3: Illustrating WSN-CSP assuming covers are scheduled in the order $G_1, G_2, G_4, G_6, G_5, G_3$

breach is 10 time units for target D_4. Figure 2.3 illustrates the scheduling of covers in the order $G_1, G_2, G_4, G_6, G_5, G_3$ that leads to an optimal solution which has the maximum breach of 6 time units for targets D_4 and D_5. In this example, T_1 = 3 (k_1 = 3, k_2 = 4), T_2 = 4 (k_1 = 5, k_2 = 5),T_3 = 3 (k_1 = 3, k_2 = 4), T_4 = 6 (k_1 = 5, k_2 = 6), T_5 = 6 (k_1 = 4, k_2 = 5).

2.3 Related work

This section provides the details of previously proposed approaches used for addressing *WSN-CSP*.

As already mentioned in Section 2.1, *WSN-CSP* was introduced and proved $\mathcal{NP}$-hard by Rossi et al. in [78] where mixed integer linear programming (MILP) formulation and a lower bound for *WSN-CSP* were also presented. However, this MILP formulation proved too difficult to use in practice as it could not be solved even for the smallest instance considered. The lower bound is based on two approaches. First approach is based on the fact that Δ_{max} is lower bounded by the maximum value from all the durations of covers which does not cover all the targets. In other words, if a cover does not provide coverage to all the targets then definitely during the working duration of this cover some targets will be breached. For example, in Figure 2.2 and Figure 2.3, cover G_5 has maximum working duration among all covers which is *4* time units and it does not cover targets $\{D_2, D_4, D_5\}$. Hence *4* is the minimum value of Δ_{max} in the example presented in Section 2.2 as per the first approach. The second approach

is based on alternative argument. For some specific target D_j, the total time for which the target is not covered can be calculated by subtracting the total duration of the covers monitoring this target from the network lifetime. This total time for which the target is not covered can be split into $|Q_j| + 1$ time intervals, by scattering the $|Q_j|$ covers covering target D_j in an equally distributed separation. For example, the target D_4 is monitored by covers G_1 and G_6 and their total working duration is *4*. The total duration of all the covers is *14*. Thus, D_4 remains uncovered for total *10* time units and by positioning appropriately the covers G_1 and G_6, the minimum value of Δ_{max} for target D_4 is $10/3 \approx 3.33$. If we calculate this value for all targets, then the maximum value from these will provide the lower bound for Δ_{max}. The lower bound is taken as maximum value from the two lower bounds found by these two approaches.

Rossi et al. [78] proposed a greedy heuristic (GH) and a genetic algorithm (referred to as CSGA) approach to solve *WSN-CSP*. GH determines an ordering among the covers by processing the targets in decreasing order of their potential to cause longest breach. The covers are scheduled as per their determined order. The genetic algorithm uses probabilistic binary tournament selection to select two parents for crossover. The cycle crossover operator (CX) is used to produce offspring, and with small probability swap based mutation is applied on the offspring generated by the crossover operator. The proposed CSGA relies on steady state population replacement model instead of the commonly used generational model. The best solution obtained through genetic algorithm is improved further by an exhaustive local search. Computation results demonstrated the effectiveness of CSGA in solving *WSN-CSP* as it performed much better than GH.

Gopinadh et al. [79] presented two hybrid swarm intelligence based approaches for *WSN-CSP*. The first approach was based on artificial bee colony algorithm (referred to as CSABC), whereas the latter approach was based on invasive weed optimization algorithm (referred to as CSIWO). In CSABC approach, 3-point swap (3PS) method was adopted to generate a neighboring solution, and the concept of scout bees was not used at all as it deteriorated the final solution obtained in almost all cases. The CSIWO approach also uses 3PS method to produce new solutions. The 3PS method selects three positions (say p, q and r) randomly in a solution. The cover at position p is swapped with cover at position q and then the new cover at position p is swapped with cover at position r. The solution generated by 3PS method is at distance 2 from the original solution, i.e., new solution differs from the original solution by 2 in terms of number of swaps. The CSABC and CSIWO were augmented with a local search to further improve the obtained solutions. The local search procedure finds the target involved in longest

breach and identify the middle position by using beginning and ending positions of the covers involved in longest breach. Then the covers which can provide coverage to the target involved in longest breach are tried one-by-one for placement at the calculated middle position. The cover which reduces the longest breach by maximum amount is inserted at the middle position. To reduce the computational effort, the search process is restricted to finding 5 covers reducing the longest breach and choosing the best one among these five. CSABC and CSIWO based approaches obtained better quality solutions in comparison to approaches of [78], and, CSABC based approach performed better than CSIWO based approach.

2.4 Two-membered evolution strategy approach for *WSN-CSP*

The proposed two-membered evolution strategy approach for the *WSN-CSP* is an extended version of the basic two-membered evolution strategy where a reshuffle procedure is added to escape from a locally optimal solution. Our proposed approach will be referred as *ES-CSP* hereafter. The process begins with the initial solution generation by employing NEH and opposition based solution generation. The *ES-CSP* is a mutation based approach. Hence, in each iteration, the mutant solution is obtained by applying the mutation operator on the present solution and the mutant solution replaces the present solution for the next iteration in case the mutant solution is as good as the present solution. We are replacing the present solution even if the mutant solution has the same fitness as the present solution due to the existence of large number of different solutions with same fitness for *WSN-CSP*. However, if there is no improvement in the fitness of the solution, for a fixed number of consecutive iterations, then the reshuffle procedure is invoked to get a new solution which replaces the present solution. The entire process is repeated again and again as long as termination condition remains unsatisfied. In the proposed approach, we have used three mutation operators for maintaining an adequate balance between exploration and exploitation. The first and third mutation operators have fixed as well as low mutation step size, hence they can be seen as the operators performing exploitation, whereas second mutation operator has variable mutation step size and due to this it can be used for exploration as well as exploitation by appropriately selecting the step size. It is to be noted that if mutation parameter value is large, then the exploration of search space will be more, but at the same time it has a higher chance of missing nearby better quality solutions. On the other hand, mutation with smaller parameter value facilitates exploitation rather than

exploration [80]. The main components of our proposed approach are discussed in the following subsections.

2.4.1 Solution encoding and fitness

We have used the same encoding approach as described in [79]. Here, a linear permutation of covers represents a solution, i.e., if there are total m covers, then a solution represents the permutation of m covers such that the positions of the covers in the solution govern the order in which covers are activated.

We have used the objective function itself as the fitness function. It is pertinent to mention that *WSN-CSP* being a minimization problem, a lesser value of the objective function signifies a more fit solution.

2.4.2 Initial solution generation

Initially, a random solution is generated and its corresponding opposite solution (refer to Section 1.9) is determined. On both solutions, we have employed NEH heuristic and the fitness of two new resulting solutions is evaluated. The more fit between the two is used as initial solution for *ES-CSP*. Algorithm 3 provides the pseudo-code for initial solution generation.

It is observed in many combinatorial optimization problems, that the superior quality local minima have more inclination towards global minima, in comparison with lower quality local minima [81]. In general, evolutionary computation paradigm employs randomly generated initial population. However, the use of problem-specific greedy heuristic can provide initial population of superior quality [82]. The use of NEH heuristic is inspired by these observations. The description of NEH presented in Section 1.8 is in terms of flow shop scheduling problem. With reference to *WSN-CSP*, $\pi = G_{\pi_1} G_{\pi_2} \ldots G_{\pi_m}$ is a permutation of m covers, i.e., π represents a schedule of covers for *WSN-CSP*. For *WSN-CSP*, first step is skipped as there is no clear criteria to determine the relative preference of covers, so π is a random permutation of m covers initially. Steps 2 and 3 are applied in the same manner as described in Section 1.8. It is pertinent to mention that, if there are only two covers to schedule then any order (that is, $G_{\pi_1} G_{\pi_2}$ or $G_{\pi_2} G_{\pi_1}$) will result in same value of breach. Hence, in step 2 of NEH we always choose the order $G_{\pi_1} G_{\pi_2}$.

Algorithm 3: Pseudo-code of initial solution generation

Input: Set of n targets, set of m covers & their respective work durations and set of targets covered by each cover
Output: Δ_{SOL}

function Initial_Sol_Generation()
begin
    $\Delta_{SOL} \leftarrow$ generate a solution randomly;
    $\Delta_{OPPSOL} \leftarrow$ opposite solution of Δ_{SOL};
// Generating a random solution and its corresponding opposite solution
    $\Delta_{SOL} \leftarrow$ NEH (Δ_{SOL});
    $\Delta_{OPPSOL} \leftarrow$ NEH(Δ_{OPPSOL});
// Applying NEH on both solutions
    if *(Δ_{OPPSOL} is better than Δ_{SOL})* **then**
        $\Delta_{SOL} \leftarrow \Delta_{OPPSOL}$;
    return Δ_{SOL} ;

2.4.3 Mutation operators

In the proposed approach, we have used three mutation operators, viz. MUT_1, MUT_2 and MUT_3 in a mutually exclusive manner. The MUT_1, MUT_2 are used with probabilities 0.5 and 0.45. With remaining probability 0.05, we have used MUT_3 only when the ratio *R* of number of covers not covering the target involved in longest breach to the total number of covers is greater than *0.85*, otherwise MUT_2 is used in place of MUT_3. In other words, if the target involved in longest breach is covered by a small number of covers then only MUT_3 is used. For the sake of illustration, consider the example in Figure 2.2, the target involved in longest breach is D_4, which is covered by covers G_1 & G_6 and not covered by G_2, G_3, G_4 & G_5. For the given scenario, the value of *R* is $\frac{4}{6}$, i.e., 0.67 which is not greater than 0.85, and hence, third mutation operator will not be used in this scenario. In each iteration, one among these three mutation operators partially ruins the solution, and then recreates the solution following a greedy approach. Ruin-and-recreate strategies have been used in the literature in combination with several different metaheuristic frameworks to provide solutions to a wide range of problems, e.g. [55, 83, 84, 85, 86, 87, 88].

The working principle of all three mutation operators is same. It selects the υ covers (υ_1, υ_2, υ_3 respectively for the three mutation operators) and remove them, from the solution. Each removed cover is reinserted at its best possible position in the solution, following the same

order of their removal. For each removed cover, the best position is found by using a greedy based construction strategy. The removed cover is tried for all the remaining $(m - \upsilon + k)$ positions in the partial solution, consequencing in $(m - \upsilon + k)$ sequences, where m is total number of covers and $k \in \{1, 2, \ldots \upsilon\}$ represents the cover selected for reinsertion. Out of these tentative sequences, the one with minimum objective function cost is selected as partial solution for next iteration and the same steps are continued until all the removed covers are reinserted into the solution. The three mutation operators only differ in the selection of covers to be removed from the solution. First mutation operator (MUT_1), selects υ_1 covers randomly from the entire solution. On the other hand, second and third mutation operators (viz. MUT_2 and MUT_3 respectively), select randomly υ_2 and υ_3 covers respectively from a subset of the solution. The subset for MUT_2 is restricted to covers involved in the longest breach, whereas subset for MUT_3 is restricted to covers which are covering the target involved in longest breach. It is pertinent to mention that MUT_3 is used only if number of covers monitoring the target involved in longest continuous breach is very small as we have observed that on a few instances without the use of third mutation operator, we get worse results no matter with what probabilities we use other two mutation operators. It is to be noted that there can be more than one target involved in longest continuous breach, and, when this happens the second and third mutation operators process appropriate covers corresponding to lowest numbered target involved in longest continuous breach. The algorithms 4, 5 and 6 respectively provide the pseudo-code for MUT_1, MUT_2 and MUT_3.

Algorithm 4: Pseudo-code of first mutation (MUT_1)

Input: A solution Δ_{SOL}
Output: Mutated Solution Δ_{SOL}
// Δ_{add} stores the covers removed from Δ_{SOL} during mutation

function $MUT_1(\Delta_{SOL})$
begin
  for $(i := 1 \; to \; \upsilon_1)$ **do**
    Remove one cover G_Δ from Δ_{SOL} randomly without altering relative order of remaining covers;
    $\Delta_{add}(i) \leftarrow G_\Delta$;
  for $(i := 1 \; to \; \upsilon_1)$ **do**
    $\Delta_{SOL} \leftarrow$ Best permutation retrieved by inserting cover $\Delta_{add}(i)$ in all possible positions of Δ_{SOL};
  return Δ_{SOL} ;

Algorithm 5: Pseudo-code of second mutation (MUT_2)

Input: A solution Δ_{SOL} & the target D_j involved in longest breach in Δ_{SOL}.
Output: Mutated Solution Δ_{SOL}

function $MUT_2(\Delta_{SOL})$
begin
  // Δ_{add} stores the covers removed from Δ_{SOL} during mutation
  $\Delta_{pos} \leftarrow$ sequence of covers responsible for longest target breach in Δ_{SOL};
  $\upsilon_2 \leftarrow \lfloor \frac{|\Delta_{pos}|}{2} \rfloor$;
  for $(i := 1 \; to \; \upsilon_2)$ **do**
    Select one cover G_Δ from Δ_{pos} randomly and remove this G_Δ from Δ_{SOL} & Δ_{pos} without altering relative order of remaining covers;
    $\Delta_{add}(i) \leftarrow G_\Delta$;
  for $(i := 1 \; to \; \upsilon_2)$ **do**
    $\Delta_{SOL} \leftarrow$ Best permutation retrieved by inserting cover $\Delta_{add}(i)$ in all possible positions of Δ_{SOL};
  return Δ_{SOL} ;

Algorithm 6: Pseudo-code of third mutation (MUT_3)

Input: A solution Δ_{SOL} & the target D_j involved in longest breach in Δ_{SOL}.
Output: Mutated Solution Δ_{SOL}

function $MUT_3(\Delta_{SOL})$
begin
  // Δ_{add} stores the covers removed from Δ_{SOL} during mutation
  $\Psi_j \leftarrow Q_j$;
  $\upsilon_3 \leftarrow \min(4, |Q_j|)$;
  for $(i := 1 \; to \; \upsilon_3)$ **do**
    Select one cover G_Δ from Ψ_j randomly, and remove this G_Δ from Δ_{SOL} without altering relative order of remaining covers;
    $\Delta_{add}(i) \leftarrow G_\Delta$;
    $\Psi_j \leftarrow \Psi_j - \{G_\Delta\}$;
  for $(i := 1 \; to \; \upsilon_3)$ **do**
    $\Delta_{SOL} \leftarrow$ Best permutation retrieved by inserting cover $\Delta_{add}(i)$ in all possible positions of Δ_{SOL};
  return Δ_{SOL} ;

2.4.4 Reshuffle procedure

Reshuffle procedure is used to diversify the search when there is no improvement in the fitness of the best solution for κ consecutive iterations. It is used to circumvent the solution from stucking in a local optima. In this procedure, we have used two different approaches one after

the other. First, we have applied the Reshuffle$_1$ procedure on the present solution. If the solution obtained by this procedure is worst than the fitness of the initial solution supplied by Algorithm 3, then only the Reshuffle$_2$ procedure is used. The Reshuffle$_1$ procedure is same as the second mutation operator (MUT_2), but the parameter υ_2 here is set to a value different from the one used in MUT_2. In Reshuffle$_2$ procedure, we have applied Algorithm 3 with one difference. Instead of generating a solution randomly, the present solution is assigned to Δ_{SOL}.

Algorithm 7 provides the pseudo-code of our *ES-CSP* approach, where the function *Longest_Breach(Δ_{SOL})* is used to find target D_j involved in longest breach in solution Δ_{SOL}. If two or more targets are involved in longest continuous breach in solution Δ_{SOL}, then this function returns the lowest numbered such target. $MUT_1(\Delta_{SOL})$, $MUT_2(\Delta_{SOL}, D_j)$ and $MUT_3(\Delta_{SOL}, D_j)$ are three functions used to perform mutation (Section 2.4.3) and the function *Reshuffle(Δ_{SOL}, D_j)* implements the reshuffle procedure (Section 2.4.4).

2.5 Computational results

Our *ES-CSP* approach is implemented in C language and executed on a Linux based 3.10 GHz Intel Core i5 processor based system with 4GB of RAM. To evaluate the performance of *ES-CSP*, we have used the same set of instances as used in [78, 79]. These test instances are the outcome of addressing MNLB problem (Maximizing network lifetime under bandwidth constraints) in WSNs via the column generation approach of [76]. Each of these test instances consists of covers & their respective work durations, and the set of targets monitored by each cover. For these test instances, the number of sensors s belongs to the set $\{50, 100, 150, 200\}$ and the number of targets n is taken to be equal to $0.6s$. It is assumed that the sensors and targets are distributed in a random manner in an area with dimensions 500×500. The sensing range (S_R) of the sensors is taken to be 150, i.e., a target D is monitored by a sensor S, if the distance between target D and sensor S is less than or equal to S_R. The bandwidth constraint $\omega \in \{5, 10, s\}$ and breach rate $\nu \in \{0.1, 0.2\}$. 30 instances were generated for each combination of values of n, ω and ν, thereby yielding 720 instances in total for MNLB problem. These 720 instances of MNLB problem when solved through the column generation approach of [76] yielded the corresponding 720 instances for *WSN-CSP*. The *ES-CSP* approach is executed once on each of these test instances.

The termination criteria for ES-CSP approach is set as 1200 iterations. The parameter υ_1 in first mutation operator is set to 4, parameter υ_2 in second mutation operator is set to $\lfloor \frac{|\Delta_{pos}|}{2} \rfloor$,

Algorithm 7: Pseudo-code of *ES-CSP*

Input: A set $D = \{D_1, D_2, \ldots, D_n\}$ of n targets, a set G consisting of m covers $\{G_1, G_2, \ldots, G_m\}$ along with their respective work durations $\{t_1, t_2, \ldots, t_m\}$ and the sets $P_i = \{D_j : (D_j \in D) \wedge (G_i \text{ covers } D_j)\}, \forall i \in \{1, 2, \ldots, m\}$.

Output: Δ_{BEST}, the best solution obtained through *ES-CSP*

```
// u01 is a uniform variate between [0,1]
```

function ES-CSP()
begin
  Compute $Q_j = \{G_i : (G_i \in G) \wedge (G_i \text{ covers } D_j)\}, \forall j \in \{1, 2, \ldots, n\}$;
  $\Delta_{SOL} \leftarrow$ Solution supplied by initial solution generation procedure;
  $\Delta_{BEST} \leftarrow \Delta_{SOL}$;
  $No\_Improvement \leftarrow 0$;
  while *(termination condition remains unsatisfied)* **do**
    $D_j \leftarrow$ Longest_Breach(Δ_{SOL});
    $R = (m - |Q_j|)/m$;
    if *(*u_{01} is greater than equal to 0.5*)* **then**
      $Mutant \leftarrow MUT_1$ (Δ_{SOL});
    else
      if *(*u_{01} is less than 0.9*)* **then**
        $Mutant \leftarrow MUT_2$ (Δ_{SOL}, D_j);
      else
        if *(*R is greater than 0.85*)* **then**
          $Mutant \leftarrow MUT_3$ (Δ_{SOL}, D_j);
        else
          $Mutant \leftarrow MUT_2$ (Δ_{SOL}, D_j);
    if *(*$Mutant$ is as good as Δ_{SOL}*)* **then**
      $\Delta_{SOL} \leftarrow Mutant$;
      if *(*$Mutant$ is better than Δ_{SOL}*)* **then**
        $No\_Improvement \leftarrow 0$;
    $No\_Improvement \leftarrow No\_Improvement + 1$;
    if *(*Δ_{SOL} is better than Δ_{BEST}*)* **then**
      $\Delta_{BEST} \leftarrow \Delta_{SOL}$;
    if *(*$No\_Improvement = \kappa$*)* **then**
      $\Delta_{SOL} \leftarrow$ Reshuffle(Δ_{SOL}, D_j);
      $No\_Improvement \leftarrow 0$;
  return Δ_{BEST};

where $|\Delta_{pos}|$ is number of covers involved in longest breach and the parameter υ_3 in third mutation operator is set to $\min(4, |Q_j|)$, where $|Q_j|$ is the number of covers which provide coverage to the target D_j involved in longest breach in current solution. The parameter υ_2

in Reshuffle$_1$ is set to $\min(4, |\Delta_{pos}|)$ (Please refer to Section 2.4.4). The value κ in reshuffle function is fixed to 50 for first 600 iterations and then it is reset to 100, i.e., if present solution doesn't improve for κ successive iterations, then reshuffle function is used. The various parameter values used in the proposed approach are selected empirically after executing our approach multiple times. Although these parameter values achieve good results on most of the instances, but they cannot be regarded as optimal parameter values for all the instances.

The performance of *ES-CSP* has been compared with the state-of-the-art approaches available in the literature. We have used the same criteria to measure the performance as used in [78, 79] and report the results in Table 2.1 & Table 2.2. The Table 2.1 provides the comparison of *ES-CSP* with *CSGA* approach of [78] and with *CSABC* & *CSIWO* approaches of [79]. The performance of the greedy heuristic (*GH*) proposed in [78] was found to be much worse than *CSGA*, *CSABC* & *CSIWO* approaches [78, 79], and hence, we have not included the greedy heuristic in this comparison. In this table, first column represents the instance parameters, viz. number of sensors (s), number of targets (n), the bandwidth constraint (ω) and the average number of covers ($\bar{m}$) in *60* instances with same value of s, n and ω. In Table 2.1, each row reports the results of various approaches on 30 instances with $\nu = 0.1$ and 30 instances with $\nu = 0.2$. The first eight columns after instance parameters provide the results for the instances with breach rate $\nu = 0.1$ and last eight columns provide the results for the instances with breach rate $\nu = 0.2$. The results of each approach for *WSN-CSP* is represented by two columns. The first column describes the average percentage deviation value over the lower bound (LB) and second column represents the number of proven optimal solutions found by that approach. The percentage deviation of a solution A of an instance over the corresponding lower bound L is defined as $100 \times \frac{A-L}{L}$. The reported average percentage deviation is average of percentage deviations of 30 instances. A solution yielded by an approach is guaranteed to be optimal if the objective function value of solution is same as its corresponding lower bound value. The value of lower bound for each instances is presented in [78]. The last two rows provide the summarized results on 360 instances for each value of breach rate (ν). The lower bound proposed in [78] may not be tight for large value of ν. The results in Table 2.1 clearly show the superiority of *ES-CSP* approach over the state-of-the-art approaches for *WSN-CSP*. Our approach performed better than other approaches in terms of both parameters, viz. average percentage deviation and number of instances solved optimally on most of the instances. The *CSGA*, *CSIWO*, *CSABC* find the 391 ($\approx 54\%$), 426 ($\approx 59\%$), 442 ($\approx 61\%$) optimal values respectively out of 720 instances, whereas the *ES-CSP* is able to find the 473 ($\approx 66\%$) optimal values out of total 720 instances.

Table 2.1: Comparison of ES-CSP with CSGA, CSABC and CSIWO on two parameters, viz. average percentage deviation from the lower bound and the count of optimally solved instances

	$\nu = 0.1$							$\nu = 0.2$						
Instance parameters	%CSGA	#CSGA	%CSABC	#CSABC	%CSIWO	#CSIWO	%ES-CSP	#ES-CSP	%CSGA	#CSGA	%CSABC	#CSABC	%CSIWO	#CSIWO	%ES-CSP	#ES-CSP
s = 50, n = 30, ω = 5, $\bar{m}$ = 41	1.81	23	1.48	22	0.58	22	0.72	22	3.53	12	1.88	17	1.06	19	1.31	17
s = 50, n = 30, ω = 10, $\bar{m}$ = 47	1.25	22	0.26	23	0.21	23	0.73	23	2.66	14	0.97	18	0.61	18	0.78	18
s = 50, n = 30, ω = 50, $\bar{m}$ = 47	0.19	24	0.17	24	0.10	25	0.29	23	3.50	14	1.05	17	0.90	17	1.19	17
s = 100, n = 60, ω = 5, $\bar{m}$ = 90	0.24	28	0.12	28	0.09	28	0.19	28	9.20	8	3.25	17	3.21	17	**2.09**	**18**
s = 100, n = 60, ω = 10, $\bar{m}$ = 96	0.70	26	0.34	28	0.28	28	0.29	28	7.79	10	2.83	12	2.50	15	**1.36**	**18**
s = 100, n = 60, ω = 100, $\bar{m}$ = 96	0.00	30	0.0	30	0.0	30	0.00	30	6.78	10	2.57	16	2.98	13	**1.55**	**19**
s = 150, n = 90, ω = 5, $\bar{m}$ = 137	0.49	26	0.32	28	0.24	27	**0.22**	28	12.25	3	5.62	3	8.65	3	**3.36**	**8**
s = 150, n = 90, ω = 10, $\bar{m}$ = 140	1.02	26	0.22	27	0.34	26	**0.16**	**28**	18.93	2	8.22	7	10.07	2	**3.88**	**10**
s = 150, n = 90, ω = 150, $\bar{m}$ = 142	0.65	27	0.58	27	0.43	27	**0.31**	27	11.40	4	6.60	8	8.35	6	**3.55**	**12**
s = 200, n = 120, ω = 5, $\bar{m}$ = 182	1.07	26	0.78	26	1.02	24	**0.38**	**28**	22.23	0	14.66	2	18.35	0	**8.94**	**2**
s = 200, n = 120, ω = 10, $\bar{m}$ = 188	0.03	27	0.15	27	0.10	28	**0.01**	**29**	16.21	3	11.30	4	16.34	1	**4.98**	**7**
s = 200, n = 120, ω = 200, $\bar{m}$ = 189	1.30	24	0.56	28	0.80	26	**0.14**	28	17.57	2	9.63	3	14.92	1	**5.17**	**5**
Total proven optimal solutions		309		318		314		**322**		82		124		112		**151**
Average % deviation to LB	0.73%		0.42%		0.35%		**0.28**%		11.00%		5.71%		7.33%		**3.18%**	

Table 2.2: Comparison of ES-CSP with CSGA, CSABC and CSIWO in terms of count of the instances on which ES-CSP achieved better (<), equal (=) and worse (>) solutions

Instance parameters	$\nu = 0.1$									$\nu = 0.2$								
	CSGA			CSABC			CSIWO			CSGA			CSABC			CSIWO		
	<	=	>	<	=	>	<	=	>	<	=	>	<	=	>	<	=	>
s = 50, n = 30, ω = 5, $\bar{m}$ = 41	6	22	2	1	26	3	0	27	3	13	14	3	6	19	5	1	21	8
s = 50, n = 30, ω = 10, $\bar{m}$ = 47	3	22	5	0	23	7	0	23	7	10	15	5	1	20	9	1	21	8
s = 50, n = 30, ω = 50, $\bar{m}$ = 47	2	24	4	2	24	4	1	24	5	9	18	3	3	20	7	2	19	9
s = 100, n = 60, ω = 5, $\bar{m}$ = 90	2	28	0	1	28	1	0	28	2	22	8	0	8	17	5	8	17	5
s = 100, n = 60, ω = 10, $\bar{m}$ = 96	4	26	0	1	28	1	1	28	1	19	10	1	14	14	2	10	17	3
s = 100, n = 60, ω = 100, $\bar{m}$ = 96	0	30	0	0	30	0	0	30	0	19	11	0	9	17	4	12	15	3
s = 150, n = 90, ω = 5, $\bar{m}$ = 137	4	26	0	2	28	0	1	27	2	27	1	2	23	5	2	24	4	2
s = 150, n = 90, ω = 10, $\bar{m}$ = 140	4	26	0	2	28	0	4	26	0	27	3	0	22	6	2	26	3	1
s = 150, n = 90, ω = 150, $\bar{m}$ = 142	3	27	0	3	27	0	3	27	0	25	4	1	21	8	1	23	6	1
s = 200, n = 120, ω = 5, $\bar{m}$ = 182	4	26	0	4	26	0	6	24	0	29	1	0	25	1	4	29	1	0
s = 200, n = 120, ω = 10, $\bar{m}$ = 188	3	27	0	2	28	0	2	28	0	27	2	1	24	5	1	29	1	0
s = 200, n = 120, ω = 200, $\bar{m}$ = 189	6	24	0	2	28	0	4	26	0	28	2	0	25	4	1	29	1	0
Total	41	308	11	20	324	16	22	318	20	255	89	16	181	136	43	194	126	40

It is evident from Table 2.1 that the performance of *CSABC* approach is close to that of *ES-CSP* for instances with $\nu = 0.1$, however the gap in performance widens for instances with $\nu = 0.2$. It is pertinent to mention that the hardness of the problem increases with increase in ν due to increased possibility of the presence of longest breach at multiple places in the schedule with increase in ν.

Table 2.2 provides the comparison of *ES-CSP* with *CSGA*, *CSIWO* and *CSABC* in terms of number of instances on which the solution obtained by *ES-CSP* is better (<), equal (=) or worse (>), on each set of 30 instances. The solutions obtained by *ES-CSP* and the *CSABC* approach are very close for instances with $\nu = 0.1$, however the solutions obtained by *ES-CSP* for instances with $\nu = 0.2$, are substantially better than those generated by existing approaches. The *ES-CSP* is able to find *181* better solutions, *136* equal solutions and *43* worse solutions, out of *360* solutions with $\nu = 0.2$, in comparison with *CSABC* approach, which is the previous best approach for *WSN-CSP* in the literature. At the end, the summarized results are also reported. Table 2.2 again shows the superiority of *ES-CSP* over state-of-the-art approaches. Except for groups of small instances, our approach always found as good as or better solutions in comparison to other approaches on each instance group of 30 instances. Further, difference in performance between our approach and other approaches is more on instances with $\nu = 0.2$ than on instances with $\nu = 0.1$.

2.6 Conclusions

We proposed a *two-membered evolution strategy* based approach to address the cover scheduling problem in wireless sensor networks (*WSN-CSP*). It is an $\mathcal{NP}$-hard problem which arises in wireless sensor networks when coverage breach is permitted owing to either technical restrictions or intentionally. Under such circumstances, the actual sequence in which covers are scheduled becomes important, as different sequences lead to different values for longest continuous duration of target breach. We have devised three mutation operators by exploiting the problem characteristics and the requirement of objective. To generate initial solution, we have used greedy heuristic and concept of opposite solutions. Our initial solution generation approach provides a better quality initial solution due to which proposed approach is able to find superior quality final solution. The results obtained by our approach are compared with the results of existing approaches for *WSN-CSP*. The computational results demonstrate that our proposed approach outperforms the state-of-the-art approaches.

Chapter 3

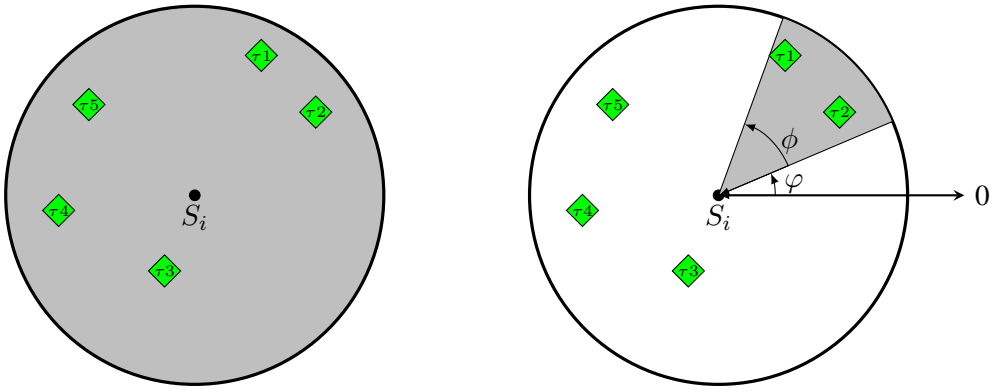
Total Rotation Minimization Problem in Directional Sensor Networks

3.1 Introduction

This chapter is devoted to the cover scheduling problem pertaining to directional sensor networks. Wireless sensor networks (WSNs) comprising only those sensors which can sense in all directions around them are called omnidirectional WSNs, and such sensors are called omnidirectional sensors. The WSN problem addressed in the last chapter has comprised of omnidirectional sensors. Most wireless sensor networks use omnidirectional sensors. Thus omnidirectional WSNs are commonly referred to as wireless sensor networks. In most applications pertaining to omnidirectional WSNs, the order in which covers are scheduled does not matter, except for a few cases discussed in the last chapter.

Unlike omnidirectional sensors, a directional sensor at any specific instant of time can observe targets in an angular sector only (also known as its working direction at that instant). Figure 3.1 illustrates the difference between an omnidirectional sensor and a directional sensor where a sensor is shown by a black dot and its sensing range by a circle centered at that sensor. Sensor S_i shown in Figure 3.1(a) is an omnidirectional sensor, and hence, it can monitor all targets around it that are within its sensing range. That is why entire circle is shown in grey color in Figure 3.1(a). On the other hand, sensor S_i shown in Figure 3.1(b) is a directional sensor with sensing angle ϕ and working direction φ. Hence, it can monitor targets only in the angular sector that begins at an angle φ and have central angle ϕ. This angular sector is shown in grey color in Figure 3.1(b). Hence, it can monitor only two targets, viz. $\tau 1$ and $\tau 2$.

The directional sensors primarily include infra-red sensors [89], ultrasonic sensors [90] and video sensors [91]. Incapability of directional sensors to monitor all surrounding targets at any instant of time does not preclude the possibility of monitoring all surrounding targets at different instants of time. To facilitate this, each directional sensor is equipped with a device that can rotate it, thereby changing its working direction. With such an arrangement, working direction of a directional sensor can be adjusted as per the application's requirements [92]. A detailed survey on directional sensor networks and their applications can be found in [92].



(a) Omnidirectional sensor (b) Directional sensor

Figure 3.1: Omnidirectionl sensor vs directional sensor

The lifetime maximization problem discussed in Section 2.1 of last chapter becomes more complicated in case of directional sensor networks as each cover not only contains list of its constituent sensors, but also the working direction for each one of them [93, 94]. As the covers are activated one-by-one, sensors may need to rotate so as to face their working directions dictated by the currently active cover. It is to be noted that only those sensors need to be rotated which are present in the currently active cover and which do not face the working direction dictated by this cover. However, rotation of a directional sensor from one direction to another consumes energy, and hence, the sequence in which covers are scheduled does matter in directional sensor networks [94]. This chapter is concerned with how to schedule a given set of covers so that the energy consumption due to rotations of sensors is minimized. Such sets of covers originate as a result of not only solving different variants of lifetime maximization problem in directional sensor networks, but also by solving some other problems in directional sensor networks such as those maximizing quality of service subject to certain lower bound on lifetime [93, 94, 95, 96, 97, 98, 99]. As the energy consumed in a rotation of a sensor is directly

proportional to the angular magnitude of the rotation, the problem transforms to scheduling the covers in such a manner so that the sum total of all sensors' rotations is minimized. This problem is an $\mathcal{NP}$-hard problem and termed as the *Total Rotation Minimization Problem (TRMP)* in the literature [100].

TRMP was introduced by Singh and Rossi in [100], where a lower bound and three heuristic approaches for this problem were also presented. The three approaches comprise a nearest neighbor heuristic approach, a genetic algorithm approach and an approach combining the genetic algorithm with an exhaustive swap-based local search. The exhaustive swap-based local search in the combined approach is applied only on the best solution returned by the genetic algorithm. The genetic algorithm utilizes swap based mutation and uniform order based crossover which do not make use of any problem-specific knowledge. Though an exhaustive swap based local search is applied on the best solution returned by the genetic algorithm, it can not compensate for the absence of exploitation of problem-specific knowledge by the genetic algorithm. These facts motivated us to explore more about this problem.

TRMP is essentially a permutation problem that seeks the optimal schedule of covers which has the minimum total transitional cost due to rotation of sensors. The *WSN-CSP* problem addressed in previous chapter, the covers involved in longest breach and the covers which can monitor the target involved in longest breach can be found. By exploiting this characteristic of the problem, we have proposed three mutation operators in evolution strategy approach for *WSN-CSP*. On the other hand, in TRMP, the transitional cost in moving from one cover to another can depend on several preceding covers. Thus, *TRMP* is essentially a more tough problem to solve in comparison to *WSN-CSP*.

To address *TRMP*, we proposed an evolutionary approach based on a *two-membered evolution strategy* (*(1+1)-ES*). The proposed approach for *TRMP* is an extension of *two-membered evolution strategy* approach for *WSN-CSP*. It is incorporated with a pre-processing step, which is used to boost the performance of the evolution strategy (ES). In the pre-processing step, we have applied NEH heuristic on some randomly generated solutions and their corresponding opposite solutions and the best solution obtained is passed as input to *(1+1)-ES*. It is to be noted that in ES approach for *WSN-CSP*, we have applied NEH on only one randomly generated solution and its corresponding opposite solution. On the other hand, NEH is applied on more than one solution in the pre-processing step used here. Use of more than one solution in pre-processing step provides a better exploration of search space, and thus, yields a superior quality initial solution in comparison to the ES approach of *WSN-CSP*. We have also used a reshuffle procedure to

diversify the search process and escape from local optima. Our *(1+1)-ES* employs a mutation operator that is based on a destruction and construction strategy where a solution is partially destroyed and then reconstructed following a problem-specific greedy strategy. The performance of the proposed *(1+1)-ES* with and without the pre-processing step has been compared with the best approach available in the literature, viz. the combined approach of [100] on the same set of instances as used in [100]. Computational results demonstrate that the proposed *(1+1)-ES* based approach both with and without pre-processing step is better than the previously best approach available in the literature. The superior performance of our approach can be attributed to effective exploitation of problem-specific knowledge by the mutation operator.

The remainder of this chapter is organized as follows: A formal definition of the total rotation minimization problem is presented in Section 3.2. Section 3.3 describes our evolution strategy based approach for the TRMP. Computational results and their analysis are presented in Section 3.4. Section 3.5 provides the concluding remarks by listing the contributions made in this chapter.

3.2 Problem definition

Consider a directional sensor network consisting of m directional sensors , and there are n covers to schedule. Each of these n covers consists of a subset of these m sensors along with the working direction for each sensor present in the cover. It is to be noted that the working direction of a sensor can be different in different covers, if the sensor belongs to more than one cover. To represent a cover, we have used a representation scheme which can be considered as an extension of binary representation scheme for sets. For all $i \in \{1, \ldots, n\}$, cover g_i can be represented as a real vector of dimension m in such a way that $g_{i,j}$ is the working direction of sensor j, if sensor j belongs to the cover. The working direction can be specified in radians or in degrees and the permissible values are in the range of $[0, 360)$ for degrees and $[0, 2\pi)$ for radians. Any value beyond this permissible range for $g_{i,j}$ indicates sensor j is not present in the cover. For the sake of illustration, we have represented all working directions in degrees in the range of $[0, 360)$. The direction $g_{i,j}$ is set to 400 degrees for all sensors j which are not present in g_i. As $400 \notin [0, 360)$, so it correctly encodes the absence of sensor j in cover g_i without causing any ambiguity. We will use the term direction, orientation and angular position interchangeably throughout this chapter.

3. TOTAL ROTATION MINIMIZATION PROBLEM IN DIRECTIONAL SENSOR NETWORKS

Further, like [100], initial angular positions of all the sensors are assumed to be given. These initial positions are represented by the special cover g_0, such that $g_{0,j}$ is the initial angular position of sensor j, which is in the range of $[0, 360)\ \forall\ j \in \{1, \ldots, m\}$. Obviously, cover g_0 will always be scheduled at position 0, as it provides the initial angular positions of all the sensors.

The angular positions of all the sensors at any specific instant of time constitute the state of the directional sensor network at that instant. An m-column real vector is used to model a state S_a, such that $S_{a,j}$ is the angular position of sensor j for all $j \in \{1, \ldots, m\}$. Clearly, initial state S_0 of the network is same as the cover g_0, i.e., S_0=g_0 and the state will change as the covers get scheduled one after the other. As the network has a total of n covers to schedule, there are $n + 1$ different states in total including the initial state S_0. These states are denoted as $S_0, S_1, \ldots, S_n$. A state $S_i, \forall i \in \{0, 1, 2, \ldots, n\}$ provides the angular positions of all the sensors after activating the cover occupying the i^{th} position in the schedule. The value of $S_{i,j}$ will always lie in $[0, 360)$. If the sensor j is present in the cover occupying the i^{th} position in the schedule, then $S_{i,j}$ will be same as the angular position of sensor j in the cover occupying the i^{th} position. However, if the sensor j is not present in the cover occupying the i^{th} position, then $S_{i,j}$ will be the angular position corresponding to the cover which contains sensor j and which is scheduled at the position nearest to the position i and less than the position i. It should be noted that the special cover g_0 which is scheduled in position 0 contains all the sensors, i.e., initially all sensors are facing some specific direction.

As a sensor can rotate in either clockwise or anti-clockwise manner, the angular distance between two directions φ_1 and φ_2 in the range [0,360) is computed as follows :

$$d(\varphi_1, \varphi_2) = \begin{cases} |\varphi_1 - \varphi_2| & \text{if } |\varphi_1 - \varphi_2| \leq 180, \\ 360 - |\varphi_1 - \varphi_2| & \text{otherwise.} \end{cases} \tag{3.1}$$

Let $D(S_a, g_i)$ denotes the total amount of rotation required by all the sensors belonging to the cover g_i to face their working directions from a state S_a. Actually, it is the transitional cost required to turn S_a into a new state $S_a^{'}$, such that all the sensors in $S_a^{'}$ which belong to cover g_i have the same orientation as their working direction in g_i and remaining sensors which do not belong to cover g_i remain at the same angular positions as in S_a. More formally,

$$D(S_a, g_i) = \sum_{j \in \{\ell : \ell \in \{1,\ldots,m\} \wedge g_{i,\ell} < 360\}} d(S_{a,j}, g_{i,j}) \tag{3.2}$$

For the sake of illustration, let us assume that S_a and g_i have following compositions:

$$S_a = \begin{bmatrix} 185 \\ 90 \\ 345 \\ 270 \\ 15 \end{bmatrix}, \qquad g_i = \begin{bmatrix} 205 \\ 400 \\ 400 \\ 255 \\ 340 \end{bmatrix}$$

The transitional cost $D(S_a, g_i)$ and the new state $S_a^{'}$ are as follows:

$$D(S_a, g_i) = 20 + 15 + 35, \qquad S_a^{'} = \begin{bmatrix} 205 \\ 90 \\ 345 \\ 255 \\ 340 \end{bmatrix}$$

Therefore, the total rotation minimization problem (TRMP) is to find a permutation of n covers $g_1, g_2, \ldots, g_n$, that minimizes the total transitional cost. More formally, let ξ represents the set of all permutations of these n covers. The TRMP seeks a permutation $\rho = g_{\rho_1} g_{\rho_2} \ldots g_{\rho_n}$ in ξ that minimizes

$$\sum_{i=1}^{n} D(S_{i-1}, g_{\rho_i}) \tag{3.3}$$

Where S_i represents the i^{th} state (i.e. S_i is the m-column vector whose elements are the angular positions of all the sensors after activating the covers $g_{\rho_1}, g_{\rho_2}, \ldots, g_{\rho_i}$ from initial state S_0=g_0).

To illustrate *TRMP*, let us consider an example where network is composed of five sensors and there are three covers g_1,g_2 and g_3 to schedule. Initial state S_0 represents the initial angular positions of all the sensors, which is supplied in the form of cover g_0 as already discussed (S_0=g_0). Let us assume that g_0, g_1,g_2 and g_3 have the following contents

$$S_0 = g_0 = \begin{bmatrix} 110 \\ 75 \\ 180 \\ 0 \\ 345 \end{bmatrix}, \quad g_1 = \begin{bmatrix} 95 \\ 75 \\ 305 \\ 400 \\ 400 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 400 \\ 108 \\ 400 \\ 65 \\ 400 \end{bmatrix}, \quad g_3 = \begin{bmatrix} 75 \\ 65 \\ 336 \\ 400 \\ 10 \end{bmatrix}$$

Table 3.1 and Table 3.2 show the scheduling of these three covers g_1, g_2 and g_3 in two different orders. Table 3.1 shows the scheduling of covers as per their natural order, i.e., g_1, g_2, g_3, which yields total transitional cost of 357 (140+98+119) for the TRMP. Table 3.2

shows the scheduling of covers in order g_1, g_3, g_2, which leads to total transitional cost of 334 (140+86+108).

Table 3.1: Illustrating the TRMP assuming the covers are scheduled in their natural order, i.e., g_1, g_2, g_3

Transition	Angular magnitude	Direction[1]	Total rotation	New state
From S_0=g_0= $\begin{bmatrix}110\\75\\180\\0\\345\end{bmatrix}$ to serve g_1= $\begin{bmatrix}95\\75\\305\\400\\400\end{bmatrix}$	$\begin{bmatrix}15\\0\\125\\0\\0\end{bmatrix}$	$\begin{bmatrix}-\\*\\+\\*\\*\end{bmatrix}$	140	S_1= $\begin{bmatrix}95\\75\\305\\0\\345\end{bmatrix}$
From S_1= $\begin{bmatrix}95\\75\\305\\0\\345\end{bmatrix}$ to serve g_2= $\begin{bmatrix}400\\108\\400\\65\\400\end{bmatrix}$	$\begin{bmatrix}0\\33\\0\\65\\0\end{bmatrix}$	$\begin{bmatrix}*\\+\\*\\+\\*\end{bmatrix}$	98	S_2= $\begin{bmatrix}95\\108\\305\\65\\345\end{bmatrix}$
From S_2= $\begin{bmatrix}95\\108\\305\\65\\345\end{bmatrix}$ to serve g_3= $\begin{bmatrix}75\\65\\336\\400\\10\end{bmatrix}$	$\begin{bmatrix}20\\43\\31\\0\\25\end{bmatrix}$	$\begin{bmatrix}-\\-\\+\\*\\+\end{bmatrix}$	119	S_3= $\begin{bmatrix}75\\65\\336\\65\\10\end{bmatrix}$

[1] '$+$', '$-$' and '$*$' indicate rotation in anti-clockwise direction, clockwise direction and no rotation respectively

3.3 Two-membered evolution strategy based approach for TRMP

Inspired by the success of evolution strategy based approach in addressing *WSN-CSP*, we have proposed a two-membered evolution strategy approach which incorporates a pre-processing step. As already explained in previous chapter, the two-membered evolution strategy approach follows a mutation-selection scheme. The process begins with an initial solution that is supplied by a pre-processing step and then mutation is performed on this solution. The better of parent and offspring, is selected as parent for next iteration. If the solution fails to improve over certain number of iterations then reshuffle procedure is invoked and the current solution is replaced with a solution provided by the reshuffle procedure. The entire process is repeated until the termination criteria is satisfied. In our approach, only mutation operator is used for exploitation as well as for exploration. Mutation with small step size is used for exploitation, where as for exploration, we have used mutation with large step size. Our reshuffle procedure makes use

Table 3.2: Illustrating the TRMP assuming the covers are scheduled in the order g_1, g_3, g_2

Transition	Angular magnitude	Direction[1]	Total rotation	New state
From S_0=g_0= $\begin{bmatrix}110\\75\\180\\0\\345\end{bmatrix}$ to serve g_1= $\begin{bmatrix}95\\75\\305\\400\\400\end{bmatrix}$	$\begin{bmatrix}15\\0\\125\\0\\0\end{bmatrix}$	$\begin{bmatrix}-\\*\\+\\*\\*\end{bmatrix}$	140	S_1= $\begin{bmatrix}95\\75\\305\\0\\345\end{bmatrix}$
From S_1= $\begin{bmatrix}95\\75\\305\\0\\345\end{bmatrix}$ to serve g_3= $\begin{bmatrix}75\\65\\336\\400\\10\end{bmatrix}$	$\begin{bmatrix}20\\10\\31\\0\\25\end{bmatrix}$	$\begin{bmatrix}-\\-\\+\\*\\+\end{bmatrix}$	86	S_2= $\begin{bmatrix}75\\65\\336\\0\\10\end{bmatrix}$
From S_2= $\begin{bmatrix}75\\65\\336\\0\\10\end{bmatrix}$ to serve g_2= $\begin{bmatrix}400\\108\\400\\65\\400\end{bmatrix}$	$\begin{bmatrix}0\\43\\0\\65\\0\end{bmatrix}$	$\begin{bmatrix}*\\+\\*\\+\\*\end{bmatrix}$	108	S_3= $\begin{bmatrix}75\\108\\336\\65\\10\end{bmatrix}$

[1] '$+$', '$-$' and '$*$' indicate rotation in anti-clockwise direction, clockwise direction and no rotation respectively

of mutation operator with large step size to provide a new solution. The details are given in section 3.3.4. It is pertinent to mention that mutation operator with large step size leads to a wider exploration of the search space, but has high probability of missing good solutions. On the other hand, a low mutation rate relates more to exploitation rather than exploration [80]. Hereafter, this approach will be referred to as *ES-TRMP*. The salient features of our *ES-TRMP* approach are described in the following subsections.

3.3.1 Solution encoding and fitness

We have used the same encoding scheme as used in [100]. In this scheme, each solution is a linear permutation of order n (i.e., total number of covers). The positioning of covers in a permutation sequence dictate the order in which covers need to be scheduled. More precisely, a value of k at the i^{th} position in the permutation specifies that the cover k is scheduled i^{th} after preceding $(i-1)$ covers in the permutation.

The objective function (Equation (3.3)) itself is the fitness function, i.e., the fitness of a solution is the total transitional cost in rotation of sensors due to the activation of covers in the schedule. Since *TRMP* is a minimization problem, a solution having a lower value of the fitness

function is regarded as more fit than a solution having a higher value.

3.3.2 Pre-processing step

The introduction of a pre-processing step is motivated by the observation that for a large number of problems in combinatorial optimization domain, the higher quality local minima tend to have more closeness to the global minima, as compared to inferior quality local minima [81]. Mostly, evolutionary algorithms utilize randomly generated initial solutions. However, a constructive heuristic combined with a local search may produce better initial solutions [82]. This is the idea behind pre-processing step. This pre-processing step serves as an initial exploration phase for finding a better initial solution, as compared to randomly generated initial solution. The initial solution generated by the pre-processing step is then exploited by our ES based approach. In the pre-processing step, we have applied NEH heuristic on some random solutions. The pre-processing step begins with χ randomly generated initial solutions (i.e., these solutions are some random permutations of covers). For each of these solutions, their corresponding opposite solution (see Section 1.9) is also generated. The *NEH* heuristic is applied on each solution and on its corresponding opposite solution as well (Total 2χ solutions). The best solution among all these solutions is passed as input to *ES-TRMP*. The pseudo-code of pre-processing step is given in Algorithm 8, where *NEH(S)* is a function that applies the NEH heuristic on a solution S. The NEH algorithm presented in Section 1.8 is modified to make it suitable for TRMP. Here, $\pi = g_{\pi_1} g_{\pi_2} \dots g_{\pi_n}$ is a permutation of n covers, i.e., a schedule of covers for TRMP. Step 1 is not used in case of TRMP as NEH is applied on multiple solutions, so π is a random permutation of n covers initially. Further, among the alternative sequences of covers, the sequence that minimizes the total transitional cost associated with it, is selected. The total transitional cost associated with a sequence $\kappa = g_{\kappa_1} g_{\kappa_2} \dots g_{\kappa_k}$ with k covers is $\sum_{i=1}^{k} D(S_{i-1}, g_{\kappa_i})$. The last two steps are applied in the same manner as described in Section 1.8.

3.3.3 Mutation operator

The mutation operator is based on a destruction and reconstruction strategy, where a solution is partially destroyed and then heuristically reconstructed. Our mutation procedure takes a solution as input and then randomly removes κ covers from this solution. Then, it reinserts the removed covers one-by-one in the solution in the order in which they are removed from the solution. A removed cover is tried for insertion in all possible $(n - \kappa + i)$ positions in partial solution,

Algorithm 8: Pseudo-code of pre-processing step

Input: Number of sensors m, number of covers n and covers $g_0, g_1, g_2, \ldots, g_n$, number of solutions χ
Output: $BEST_{SOL}$, the best solution obtained in pre-processing step
// Fitness of $BEST_{SOL}$ is initialized to a value larger than maximum possible fitness value

function Pre-processing(χ)
begin
  for *($i := 1$ to χ)* **do**
    Generate a solution SOL_i randomly;
    $OPP\_SOL_i \leftarrow$ opposite solution of SOL_i;
    $SOL_i \leftarrow$ NEH(SOL_i);
    $OPP\_SOL_i \leftarrow$ NEH(OPP_SOL_i);
    if *(OPP_SOL_i is better than SOL_i)* **then**
      $SOL_i \leftarrow OPP\_SOL_i$;
    if *(SOL_i is better than $BEST_{SOL}$)* **then**
      $BEST_{SOL} \leftarrow SOL_i$;
  return $BEST_{SOL}$;

where n is the total number of covers in the original solution and the cover in consideration for insertion was i^{th} cover to be removed. Out of $(n - \kappa + i)$ resulting sequences, the sequence which yields minimum total rotation is retained as new partial solution and again next cover is tried for insertion. This process is repeated till all the removed covers are inserted back in the solution. The pseudo-code of our mutation operator is given in Algorithm 9.

Algorithm 9: The pseudo-code of mutation

Input: A solution G_{sol}
Output: Mutated Solution G_{sol}
// G_{add} stores the covers removed from G_{sol} during mutation

function Mutation(G_{sol})
begin
  for *($i := 1$ to κ)* **do**
    Remove one cover g from G_{sol} randomly without disturbing relative ordering of remaining covers;
    $G_{add}(i) \leftarrow g$;
  for *($i := 1$ to κ)* **do**
    $G_{sol} \leftarrow$ Best permutation obtained by inserting cover $G_{add}(i)$ in all possible positions of G_{sol};
  return G_{sol} ;

3.3.4 Reshuffle procedure

The idea behind reshuffle is to diversify the search in case the *ES-TRMP* fails to find a solution better than the best solution so far after η consecutive iterations. To implement this step, same mutation operator as described in previous section is used, but with a large step size δ which is greater than κ. In another perspective, this process can also be seen as the process which aids in escaping from the local optima.

Algorithm 10 provides the pseudo-code of our *ES-TRMP* approach where *Mutation(S)* and *Reshuffle(S)* are two functions which perform mutation (Section 3.3.3) and reshuffling (Section 3.3.4) on a solution S.

Algorithm 10: Pseudo-code of *ES-TRMP*

Input: Number of sensors m, number of covers n and covers $g_0, g_1, g_2, \ldots, g_n$
Output: $BEST_{SOL}$, the best solution obtained through *ES-TRMP*

$BEST_{SOL} \leftarrow$ Solution supplied by pre-processing step;
function ES-TRMP($BEST_{SOL}$)
begin
  $No\_Improvement \leftarrow 0$;
  while *(termination condition remains unsatisfied)* **do**
    $Mutant \leftarrow$ Mutation($BEST_{SOL}$);
    if *(*$Mutant$ is better than $BEST_{SOL}$*)* **then**
      $BEST_{SOL} \leftarrow Mutant$;
      $No\_Improvement \leftarrow 0$;
    else
      $No\_Improvement \leftarrow No\_Improvement + 1$;
    if *(*$NO\_Improvement \geq \eta$*)* **then**
      $BEST_{SOL} \leftarrow$ Reshuffle($BEST_{SOL}$);
      $No\_Improvement \leftarrow 0$;
  return $BEST_{SOL}$;

3.4 Computational results

The proposed evolution strategy approach is implemented in C and executed on Intel Core i5 system with 4GB RAM running under Ubuntu 16.04 at 3.10 GHz. The termination criteria for *ES-TRMP* is set as 4,000 iterations. The mutation step size κ is set to 4, if $4 \leq \frac{n}{2}$, otherwise κ is set to $\max(\lfloor \frac{n}{2} \rfloor, 1)$. The parameter δ in *Reshuffle procedure* is set to 40 for all instances with $n \geq 200$, to 20 for instances with $100 < n < 200$, to 10 for instances with $50 < n \leq 100$ and

$\lfloor \frac{n}{7} \rfloor$ for instances with $35 \leq n \leq 50$, otherwise, it is set to $\min(4, n)$. The value of η is set as 100, i.e., if best solution fails to improve over 100 consecutive iterations, then reshuffle procedure is invoked. The parameter χ in pre-processing step is set to 20 (if $n \geq 4$), otherwise, $\chi = n$. This parameter defines the number of randomly generated initial solutions in pre-processing step. All these parameter values are determined empirically. Although these parameter values fetch good results on most of the instances, these values in no way can be regarded as optimal values for all the instances. To ascertain the benefits of pre-processing step in *ES-TRMP*, we have implemented another version of *ES-TRMP* where the evolution strategy begins with a randomly generated initial solution instead of the solution supplied by the pre-processing step. This version of *ES-TRMP* is referred to as *ES-TRMP$_{NO\text{-}PP}$*. Except for the origin of initial solution, *ES-TRMP$_{NO\text{-}PP}$* is same as *ES-TRMP* in all other aspects.

The performances of *ES-TRMP* and *ES-TRMP$_{NO\text{-}PP}$* have been tested on same 60 instances as used in [100]. These instances were generated as a result of solving the lifetime maximization problem in wireless directional sensor networks through the column generation based approach presented in [94]. These instances have number of sensors $m \in \{50, 100, 200, 400\}$ and the number of targets T as $0.6m$. Sensors and targets are assumed to be randomly placed in a 500×500 area and the sensing range R_S is assumed to be fixed at 150. Furthermore, the sensing angle $\phi \in \{\frac{2\pi}{3}, \frac{\pi}{2}, \frac{\pi}{3}\}$ (in radians). The listing of sensing angle is done in descending order due to the fact that the difficulty of problem increases with the decrease in sensing angle provided all other parameters remain the same. As already mentioned in Section 3.2, initially, all the sensors are assumed to have different random orientations as specified by cover g_0. Hence, to activate the very first cover in the schedule, also requires the rotation of sensors, in order to bring them to their respective working directions as dictated by the first cover in the schedule. Our two approaches have been executed 20 independent times on each instance like the genetic algorithm based approaches of [100]. We have compared our approaches against the *GA-TRMP+LS$_B$* approach of [100] which is the best approach known so far for the TRMP. As the code for *GA-TRMP+LS$_B$* was available, we have re-executed it in same computational environment as used for executing our approaches to ensure fairness in execution time comparison. As a consequence of this, the execution times reported for *GA-TRMP+LS$_B$* here are lower than those reported in [100].

The results for instances with $\phi = \frac{2\pi}{3}$, $\phi = \frac{\pi}{2}$ and $\phi = \frac{\pi}{3}$ are reported in tables 3.3, 3.4 and 3.5, respectively. Each row in these tables specifies the instance name (Instance), the number of covers in it (n), the best solution (Best), the average solution quality (Average), the standard

deviation of solution values (SDev) and the average execution time (Time) of *GA-TRMP+LS$_B$*, *ES-TRMP$_{NO\text{-}PP}$* and *ES-TRMP* respectively. Results of *ES-TRMP$_{NO\text{-}PP}$* and *ES-TRMP* appear in boldface in these tables whenever these two approaches performed better than *GA-TRMP+LS$_B$*. These three tables demonstrate that *ES-TRMP* outperforms *GA-TRMP+LS$_B$* in terms of best as well as average solution quality. In no case, the best or average solution obtained by *GA-TRMP+LS$_B$* is better than *ES-TRMP*. Even the best and average solution quality obtained by *ES-TRMP$_{NO\text{-}PP}$* is better than *GA-TRMP+LS$_B$* in most of the cases. This clearly shows the benefit of an ES based approach.

However, both *ES-TRMP* and *ES-TRMP$_{NO\text{-}PP}$* approaches need more time than *GA-TRMP+LS$_B$*. This can be attributed to the high computational cost associated with heuristic reconstruction in our mutation operator.

As far as comparison between *ES-TRMP$_{NO\text{-}PP}$* and *ES-TRMP* is concerned, the former is quite competitive in terms of solution quality in comparison to latter approach on small instances. However, the benefit of the pre-processing step becomes clearly evident on larger instances as *ES-TRMP* consistently obtained much better results. Hence, using the pre-processing step boost the performance of evolution strategy. It is also evident from tables 3.3, 3.4 and 3.5 that, in most of the cases, the standard deviation of solution values obtained through *ES-TRMP* is quite less compared to *GA-TRMP+LS$_B$* approach, which shows the robustness of *ES-TRMP* in comparison to latter approach.

By comparing the instances of the same size in the result, it can be inferred that total rotation cost usually, increases with the decrease in sensing angle. It is also supported by the fact that, decrease in sensing angle requires more rotations, as fewer and fewer targets are monitored per sector.

Table 3.3: Results on instances with $\phi = \frac{2\pi}{3}$. All angles are in radians and times in seconds

		GA-TRMP+LS$_B$				*ES-TRMP$_{NO\text{-}PP}$*				*ES-TRMP*			
Instance	n	Best	Average	SDev	Time	Best	Average	SDev	Time	Best	Average	SDev	Time
dirn050m030r150s03i00.dat	18	97.458	97.673	0.515	0.09	97.458	**97.458**	**0.000**	0.11	97.458	**97.458**	**0.000**	0.11
dirn050m030r150s03i01.dat	1	17.568	17.568	0.000	0.00	17.568	17.568	0.000	0.00	17.568	17.568	0.000	0.00
dirn050m030r150s03i02.dat	43	186.170	196.280	4.719	0.33	**180.652**	**188.049**	**3.922**	0.77	**180.652**	**184.358**	**4.035**	0.79
dirn050m030r150s03i03.dat	36	121.909	124.264	1.190	0.21	**118.411**	**121.660**	2.076	0.47	**118.411**	**119.703**	1.570	0.50
dirn050m030r150s03i04.dat	9	55.809	55.809	0.000	0.06	55.809	55.809	0.000	0.03	55.809	55.809	0.000	0.04
dirn100m060r150s03i00.dat	13	97.196	97.196	0.000	0.06	97.196	97.196	0.000	0.07	97.196	97.196	0.000	0.07
dirn100m060r150s03i01.dat	25	139.847	141.665	1.234	0.18	139.847	**140.103**	**1.114**	0.26	139.847	**139.847**	**0.000**	0.26
dirn100m060r150s03i02.dat	66	263.823	273.787	3.677	0.58	**258.982**	**265.787**	**3.222**	2.19	**257.327**	**262.681**	**2.560**	2.29
dirn100m060r150s03i03.dat	94	310.662	318.893	5.392	1.23	**304.953**	**312.509**	**4.837**	4.57	**299.090**	**305.439**	**3.641**	4.94
dirn100m060r150s03i04.dat	52	257.284	260.286	2.066	0.44	**252.226**	**255.791**	2.629	1.28	**252.226**	**254.172**	**1.697**	1.37
dirn200m120r150s03i00.dat	78	689.308	700.161	5.801	1.16	**683.702**	**695.350**	6.256	3.89	**675.942**	**686.807**	6.904	4.24
dirn200m120r150s03i01.dat	145	1049.919	1073.337	9.390	4.07	**1027.952**	**1049.805**	13.265	14.52	**1009.637**	**1032.028**	10.209	16.16
dirn200m120r150s03i02.dat	198	1257.145	1279.519	12.337	7.94	**1231.329**	**1260.552**	16.938	28.65	**1195.975**	**1228.606**	12.949	33.89
dirn200m120r150s03i03.dat	200	773.406	783.566	5.820	5.86	**750.518**	**765.035**	6.561	23.47	**736.138**	**749.745**	6.312	28.40
dirn200m120r150s03i04.dat	173	1407.900	1444.971	22.036	6.12	**1395.108**	**1429.699**	**16.353**	22.37	**1383.140**	**1403.215**	**12.627**	26.08
dirn400m240r150s03i00.dat	400	2627.489	2689.661	29.373	40.27	**2587.161**	**2665.180**	**27.958**	149.92	**2517.797**	**2569.235**	**21.989**	201.06
dirn400m240r150s03i01.dat	400	2819.605	2863.694	35.636	44.42	**2786.131**	**2832.127**	**23.870**	149.27	**2723.947**	**2769.008**	**23.112**	200.18
dirn400m240r150s03i02.dat	400	2814.502	2853.335	19.086	41.31	**2786.423**	**2827.213**	20.737	150.82	**2725.782**	**2751.189**	**15.890**	202.84
dirn400m240r150s03i03.dat	400	2416.154	2459.213	26.345	37.89	**2403.842**	**2434.657**	**19.243**	141.38	**2335.526**	**2368.682**	**13.872**	188.72
dirn400m240r150s03i04.dat	400	2628.564	2662.078	20.709	40.94	**2602.309**	**2648.839**	22.441	146.48	**2514.807**	**2561.407**	21.315	196.41

Table 3.4: Results on instances with $\phi = \frac{\pi}{2}$. All angles are in radians and times in seconds

		GA-TRMP+LS$_B$				*ES-TRMP$_{NO-PP}$*				*ES-TRMP*			
Instance	n	Best	Average	SDev	Time	Best	Average	SDev	Time	Best	Average	SDev	Time
dirn050m030r150s04i00.dat	28	139.353	141.337	1.475	0.20	**137.860**	**139.038**	**1.139**	0.32	**137.860**	**139.142**	**1.037**	0.31
dirn050m030r150s04i01.dat	1	19.385	19.385	0.000	0.00	19.385	19.385	0.000	0.00	19.385	19.385	0.000	0.00
dirn050m030r150s04i02.dat	50	172.395	176.994	2.652	0.38	**168.965**	**174.642**	3.420	1.09	**168.965**	**171.629**	**2.467**	1.17
dirn050m030r150s04i03.dat	41	112.241	116.045	1.808	0.20	112.241	**114.895**	2.297	0.67	**112.214**	**112.537**	**0.455**	0.70
dirn050m030r150s04i04.dat	12	84.757	84.757	0.000	0.07	84.757	84.757	0.000	0.05	84.757	84.757	0.000	0.06
dirn100m060r150s04i00.dat	16	131.273	131.273	0.000	0.12	131.273	131.273	0.000	0.13	131.273	131.273	0.000	0.13
dirn100m060r150s04i01.dat	22	165.860	166.479	0.550	0.15	**165.653**	**166.009**	0.664	0.24	**165.653**	**165.895**	**0.274**	0.24
dirn100m060r150s04i02.dat	99	376.482	392.614	8.386	1.44	**375.281**	**390.050**	9.026	5.63	**362.812**	**373.829**	**6.861**	6.21
dirn100m060r150s04i03.dat	98	297.998	307.673	4.887	1.29	**285.788**	**301.038**	6.260	5.50	**283.260**	**291.792**	**4.679**	5.90
dirn100m060r150s04i04.dat	70	372.795	382.555	5.155	0.76	**360.587**	**374.340**	6.594	2.79	**360.912**	**369.460**	**4.776**	2.94
dirn200m120r150s04i00.dat	111	1029.881	1046.334	11.014	2.29	**1007.901**	**1025.352**	13.799	9.33	**1002.560**	**1013.943**	**10.675**	10.59
dirn200m120r150s04i01.dat	200	1425.422	1462.110	18.977	8.29	**1391.979**	**1430.595**	19.509	34.67	**1355.393**	**1397.124**	**16.591**	40.21
dirn200m120r150s04i02.dat	200	1177.928	1214.437	18.650	7.69	**1142.236**	**1176.130**	19.923	31.08	**1120.645**	**1141.890**	**12.115**	36.84
dirn200m120r150s04i03.dat	200	930.877	969.822	15.260	6.90	933.874	**956.084**	**13.638**	28.91	**908.436**	**923.886**	**10.720**	32.97
dirn200m120r150s04i04.dat	200	1303.641	1327.508	16.469	8.00	**1264.303**	**1299.743**	18.042	33.98	**1228.187**	**1262.107**	**13.435**	38.84
dirn400m240r150s04i00.dat	400	3357.996	3426.820	30.294	52.20	**3316.630**	**3385.840**	37.723	179.50	**3227.201**	**3267.053**	**23.352**	240.67
dirn400m240r150s04i01.dat	400	2942.209	2965.109	16.242	50.12	**2909.570**	**2964.235**	35.809	171.50	**2759.442**	**2819.904**	22.964	230.54
dirn400m240r150s04i02.dat	400	2999.651	3069.807	33.434	50.23	**2990.430**	**3063.782**	**32.363**	174.48	**2871.753**	**2912.871**	**22.576**	234.78
dirn400m240r150s04i03.dat	400	2804.076	2882.443	39.721	46.15	2809.775	**2870.482**	**23.303**	168.22	**2697.187**	**2730.558**	**23.175**	226.80
dirn400m240r150s04i04.dat	400	3044.738	3103.734	32.030	50.15	3057.188	**3096.111**	**24.818**	176.20	**2946.706**	**2979.172**	**19.165**	231.53

Table 3.5: Results on instances with $\phi = \frac{\pi}{3}$. All angles are in radians and times in seconds

		GA-TRMP+LS$_B$				*ES-TRMP$_{NO\text{-}PP}$*				*ES-TRMP*			
Instance	n	Best	Average	SDev	Time	Best	Average	SDev	Time	Best	Average	SDev	Time
dirn050m030r150s06i00.dat	33	229.443	236.381	4.263	0.29	**226.752**	**235.448**	4.279	0.51	**224.883**	**232.005**	**4.033**	0.52
dirn050m030r150s06i01.dat	1	18.779	18.779	0.000	0.00	18.779	18.779	0.000	0.00	18.779	18.779	0.000	0.00
dirn050m030r150s06i02.dat	50	187.388	201.862	7.043	0.45	**185.266**	**194.806**	**6.482**	1.28	**183.805**	**191.055**	**3.960**	1.36
dirn050m030r150s06i03.dat	46	189.434	192.518	3.371	0.40	**188.710**	**192.427**	**2.130**	1.01	**187.859**	**191.170**	**2.452**	1.06
dirn050m030r150s06i04.dat	22	164.635	168.105	3.025	0.19	164.635	**167.453**	**2.078**	0.23	164.635	**166.834**	**1.483**	0.24
dirn100m060r150s06i00.dat	27	217.800	222.630	2.741	0.26	**216.813**	**222.217**	3.147	0.44	**216.813**	**218.459**	**1.974**	0.44
dirn100m060r150s06i01.dat	42	316.527	321.402	3.358	0.37	**316.001**	325.361	6.719	1.14	**314.159**	**318.451**	**2.772**	1.19
dirn100m060r150s06i02.dat	91	230.107	241.023	4.715	1.11	232.873	**239.008**	**4.552**	5.41	**224.476**	**233.732**	**4.683**	5.74
dirn100m060r150s06i03.dat	100	400.539	410.085	6.578	1.78	**388.226**	**402.916**	8.273	6.82	**378.016**	**392.536**	**6.498**	7.39
dirn100m060r150s06i04.dat	96	321.215	335.292	6.988	1.47	322.598	337.063	8.234	6.14	**318.364**	**327.380**	**5.822**	6.79
dirn200m120r150s06i00.dat	198	2259.806	2311.846	36.095	10.18	**2225.826**	**2289.834**	**28.523**	44.32	**2198.577**	**2224.518**	**13.914**	52.21
dirn200m120r150s06i01.dat	200	1373.459	1406.716	22.624	9.62	**1362.396**	**1392.319**	**18.082**	40.97	**1320.061**	**1340.500**	**15.225**	48.02
dirn200m120r150s06i02.dat	200	1440.551	1469.665	15.556	10.20	**1404.821**	**1437.314**	16.418	42.56	**1365.951**	**1393.491**	**11.476**	48.97
dirn200m120r150s06i03.dat	200	1238.041	1275.921	24.790	9.15	**1204.040**	**1235.393**	**17.553**	39.66	**1164.405**	**1198.937**	**14.234**	45.37
dirn200m120r150s06i04.dat	200	1337.527	1369.349	15.376	9.52	**1307.320**	**1341.691**	21.995	42.89	**1252.114**	**1288.418**	17.367	50.06
dirn400m240r150s06i00.dat	400	4494.563	4594.853	53.549	66.67	**4491.468**	**4574.797**	**41.705**	233.14	**4349.337**	**4373.350**	**17.225**	309.74
dirn400m240r150s06i01.dat	400	3932.107	4015.341	44.376	63.07	**3919.475**	4019.937	56.061	225.73	**3741.209**	**3801.517**	**33.507**	298.41
dirn400m240r150s06i02.dat	400	4041.051	4146.541	45.990	65.19	**4008.661**	**4134.456**	51.547	231.32	**3870.738**	**3920.388**	**23.876**	307.67
dirn400m240r150s06i03.dat	400	3803.490	3858.936	42.979	62.98	**3774.866**	3865.514	**42.854**	221.19	**3595.846**	**3654.062**	**24.313**	292.42
dirn400m240r150s06i04.dat	400	3809.627	3891.973	32.931	61.57	3809.685	**3884.249**	39.258	224.04	**3612.896**	**3670.571**	**26.651**	297.57

3.5 Conclusions

In this chapter, we proposed a simple two-membered evolution strategy based approach *ES-TRMP* for total rotation minimization problem (TRMP). Our *ES-TRMP* uses a pre-processing step that is used to boost the performance of evolution strategy. The comparison of *ES-TRMP* with the state-of-the-art approach available in the literature is done and the computational results demonstrate the effectiveness of our proposed approach. *ES-TRMP* provides better solutions in comparison with *GA-TRMP+*LS_B on all the instances. Despite the fact that *ES-TRMP*$_{NO\text{-}PP}$ approach begins with a random initial solution, it performs better than state-of-the-art approach on most of the instances. The proposed *ES-TRMP* approach consumes more time in comparison with *GA-TRMP+*LS_B approach, due to the use of heuristic based mutation operator in our proposed approach.

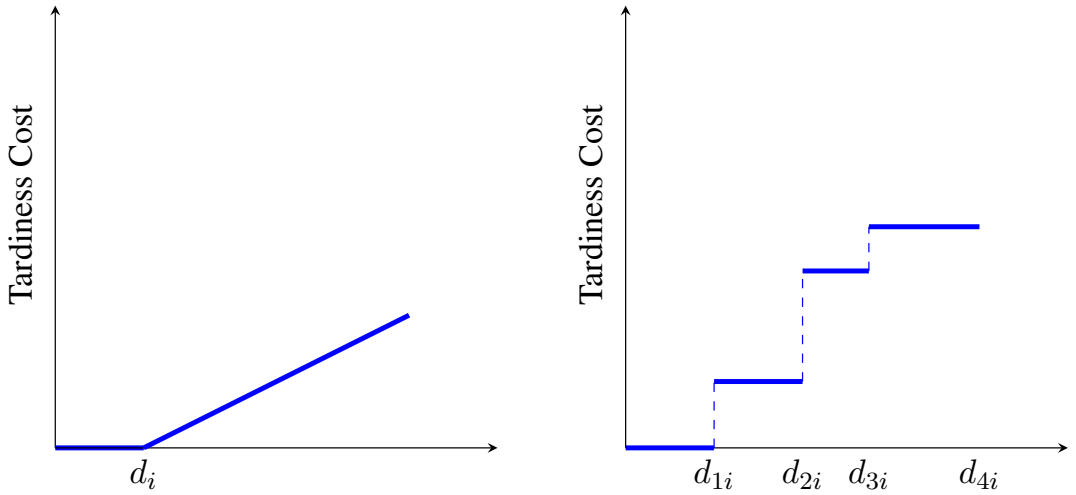
Chapter 4

Single Machine Total Stepwise Tardiness Problem with Release Dates

4.1 Introduction

The classical scheduling problem has been researched extensively over the past several decades. Total tardiness cost is the most widely used performance metric for the classical scheduling problem. A job *i* incurs tardiness if the job is completed after its due date. More formally, tardiness (T_i) of a job i is computed as T_i= max(0, C_i - d_i), where C_i is the completion time of the job i and d_i is its due date. The tardiness cost per unit of tardiness t_i for each job i is also specified, and the tardiness cost for each job i is usually taken as t_iT_i. So the classical scheduling problem based on tardiness criterion considers a single due date (d_i) for each job i, and the tardiness cost is usually a linear function of the completion time. In contrast, in stepwise tardiness scheduling problem, each job has various due dates $\{d_{1i}, d_{2i}, d_{3i}, \dots\}$ and the tardiness cost increases in stepwise manner with respect to various due dates. Evidently, the tardiness cost here is a piecewise constant function of the completion time. Figure 4.1 illustrates the difference between traditional tardiness cost and the stepwise tardiness cost [101, 102].

Stepwise tardiness cost is natural in many real-world applications [101, 103, 104] mostly pertaining to transportation. Usually, the transportation services follow a time schedule, and hence, the transportation service is not always available to serve. If a job is finished after the current transport service departs, then the finished job has to compulsorily wait till the availability of the next transport service. As a result, such a job gets delivered to the end customer at the same time no matter where it exactly finishes after the current service departs



(a) Traditional tardiness cost (b) Stepwise tardiness cost

Figure 4.1: Traditional and stepwise tardiness

and before the availability of the next service. Hence, such a job has the same tardiness cost no matter what is its exact completion time. Likewise, when modes of shipping of jobs (rail, road, air etc.) vary according to completion times of jobs with respect to their various due dates, the tardiness costs of the jobs grow in a stepwise manner. Stepwise tardiness cost also arises in manufacturing sector when customer orders are processed in batches where each batch consists of several customer orders each with its own due date. In this case, tardiness cost of a batch is the sum total of the tardiness costs of its constituent orders which get finished after their due dates. Obviously, tardiness cost of a batch can increase in a stepwise manner only as it depends on number of tardy orders in that batch and tardiness costs associated with each one of them.

In this chapter, a single machine scheduling problem is considered where jobs incur stepwise tardiness costs and each job i has a release date r_i of its own, i.e., a job i is available for processing not before its release date r_i. In the literature, this problem is referred to as the single machine total stepwise tardiness problem with release dates (*SMTSTP-R*). The *SMTSTP-R*, which is formally defined in the next section, was introduced by Tseng et al. in [101]. They have generated two sets of instances for *SMTSTP-R*, and proposed two heuristics, viz. *M-Moore* (modified Moore) and *improved NEH*. The *M-Moore* and *improved NEH* were derived from *Moore's* algorithm [105] and *NEH* heuristic [63] respectively. The various priorities based upon

slack time are considered as the tie-breaking rule in *improved NEH* heuristic. Tseng et al. also proposed a *EM* (electromagnetism like mechanism) metaheuristic [106] based approach. These are the initial approaches to address *SMTSTP-R*. However, all these approaches had assumed release dates to be zero for all jobs, i.e., all jobs are available for processing at time zero. Thus the experimental results reported in [101] on two sets of instances are for the *SMTSTP-R* with all release dates set to 0, i.e., $r_i = 0 \, \forall_i \in \{1, 2, \ldots, n\}$, where n is the total number of jobs. This version is referred to as the single machine total stepwise tardiness problem (*SMTSTP*) in the literature.

To address the problem version with release dates (*SMTSTP-R*), Chaurasia et al. [102] have proposed two hybrid metaheuristic approaches. The first approach is a steady-state genetic algorithm (GA) and the other is an artificial bee colony algorithm (ABC). They have used a series of local searches to further improve the quality of solutions obtained by metaheuristic approaches. These approaches are referred to as *HGA* and *HABC* in [102]. The initial population generation in both these approaches is done by creating solutions in a random manner. Since the results reported by Tseng et al. in [101] is for the *SMTSTP*, hence the comparison of approaches proposed in [101] with *HGA* and *HABC* approaches of [102] has been done for *SMTSTP* only in [102]. For this comparison, same instances as used in Tseng et al. [101] have been used, and both *HGA* and *HABC* performed better than earlier approaches. Chaurasia et al. [102] also report the computational results of *HGA* and *HABC* for *SMTSTP-R* on two sets of instances generated by Tseng et al. [101]. Among these two metaheuristic approaches, *HABC* performed better than *HGA* for both *SMTSTP* and *SMTSTP-R*. So these two metaheuristic approaches were the first metaheuristic approaches for *SMTSTP-R*.

To address *SMTSTP-R*, we have proposed a hybrid approach combining discrete differential evolution with a series of local searches. The local search approaches used in our approach are taken from [102], due to their superior performance. We have compared the performance of our approach with *HGA* and *HABC* approaches of [102], as they are the only existing approaches for *SMTSTP-R*. Computational results clearly show our hybrid approach to be better in comparison to these two approaches.

The remaining part of this chapter is organized as follows. Section 4.2 describes the basic notations and formulation of the *SMTSTP-R*. Our hybrid discrete differential evolution approach for addressing *SMTSTP-R* is described in Section 4.3. Section 4.4 reports the results of empirical evaluation of our hybrid approach and compares these results with those of state-of-the-art approaches in literature. At the end, Section 4.5 outlines some concluding remarks.

4.2 Problem definition

This section introduces the notational conventions used in this chapter and present the formal definition of *SMTSTP-R*. Our notational conventions are similar to those used in [101, 102]. The input to *SMTSTP-R* consists of *n* independent jobs along with their processing times, release dates and various due dates. Formally, given a set of *n* jobs, viz. J_i= $\{J_1, J_2, \ldots, J_n\}$. Each job J_i has to be processed on a single machine without preemption and requires a processing time p_i. The machine is capable of processing only one job at a time, and hence, the jobs have to be processed sequentially. Each job J_i has a release date r_i and the machine can not start processing the job before its release date. Each job has *m* due dates $d_{1i}, d_{2i}, \ldots, d_{mi}$ such that $d_{1i} < d_{2i} < \cdots < d_{mi}$. Any job incurs a tardiness cost if its completion time C_i exceeds its first due date (d_{1i}). The tardiness cost of a tardy job J_i is determined by a function $\overline{f}_i(C_i)$, which is defined below:

$$\overline{f}_i(C_i) = \begin{cases} 0 & \textit{if } C_i \leq d_{1i} \\ w_{1i} & \textit{if } d_{1i} < C_i \leq d_{2i} \\ w_{2i} & \textit{if } d_{2i} < C_i \leq d_{3i} \\ \vdots & \vdots \\ w_{mi} & \textit{if } d_{mi} < C_i \end{cases} \tag{4.1}$$

where w_{ki} is the tardiness cost of job J_i, if completion of job occurs after its k^{th} due date d_{ki} and on or before $(k+1)^{th}$ due date. The objective of *SMTSTP-R* is to find a schedule (η) of jobs that minimizes

$$f(\eta) \; = \; \sum_{i=1}^{n} \overline{f}_i(C_i) \tag{4.2}$$

Therefore, the goal of *SMTSTP-R* is to obtain a schedule of *n* jobs that incurs the minimum value for the sum of the tardiness costs of all the jobs. *SMTSTP-R* is $\mathcal{NP}$-hard [107], and using the three field notation [108], it can be denoted as $1||r_i||\sum_{i=1}^{n} \bar{f}_i(C_i)$.

4.3 Hybrid discrete differential evolution approach for SMTSTP-R

To address *SMTSTP-R*, we have proposed a discrete differential evolution approach hybridized with local search. Hereafter, our approach will be referred as *HDDE*. Our approach begins with initial population generation which uses two heuristics namely, *M-Moore* (modified Moore)

& *NEH* heuristics, and the concept of opposition based solution generation. The motivation behind utilizing these strategies is to find better quality initial solutions, which helps in faster convergence of the proposed approach. Similar methods of initial population generation are also used by the proposed approaches for *WSN-CSP* and *TRMP* presented in previous chapters, and the effectiveness of such methods are already demonstrated there. In each iteration of HDDE, a mutant solution is generated by employing the mutation operator on a randomly selected solution from the population. Thereafter, crossover operator is applied with probability p_c, on mutant and target solution which yields a trial solution. If the mutant is better that current best then crossover is not used and in this case, mutant is considered as trial solution. Afterwards, a series of local searches is employed on the trial solution. These local searches are employed only if the difference of objective value of trial solution from the best solution found so far, is less than ($\mu\times$ objective value of best solution).

If the trial solution after all local searches have been applied on it is unique and better than the target solution, then it replaces the target solution in the population. Here a solution is considered as unique either if it has unique objective function value in current population or if it has same objective function value as any other existing solution in population then it should not be identical before the limiting point with any solution having the same objective value. The limiting point is that position in a schedule where the completion time of the corresponding job has exceeded the maximum due date value among all the jobs. It should be noted that after this limiting point any permutation of remaining jobs incurs the same tardiness cost owing to the fact that the jobs at positions after the limiting point will incur their respective maximum tardiness costs irrespective of their positions in the schedule. Also if the trial solution has the objective value which is same as at least 10 other solutions in the population then this trial solution is is discarded immediately without any further checking. This is done because when so many solutions have the same objective value then it is highly likely that trial solution is either identical or almost identical to some other population members and discarding the trial solution can maintain the diversity.

If any target solution does not improve for consecutive λ iterations then a perturbation procedure is used which takes this solution as input and produce a perturbed solution from it. The local search procedure is applied on this perturbed solution and then resulting solution replaces the target solution. This procedure prevents a solution in the population from permanently getting trapped in a local optima. This entire process repeats until the stopping criteria is satisfied. Ensuring the uniqueness of the population, discarding a trial solution if it has the

same fitness as at least 10 other population members and replacing a solution in the population which has not improved over fixed number of iterations are the unique features of our approach. The first two features help in maintaining sufficient diversity in the population, whereas the last feature helps in escaping from the locally optimal solutions. Algorithm 11 presents the pseudo-code of proposed HDDE approach where *Num_Pop()* is a function that returns the number of solutions in the population having the same objective value as trial solution. The subsequent subsections describe the other important features of our approach.

4.3.1 Encoding of solutions and their fitness

We have used the same solution encoding scheme as presented in [102]. In this scheme, a solution is represented as a linear permutation of jobs such that the position of jobs in the permutation dictates the sequence in which jobs are scheduled for processing on the machine.

Equation (4.2) defines the objective function which is also used to evaluate the fitness of a solution, i.e., the sum total of tardiness cost of all jobs is used as the fitness of a solution. As *SMTSTP-R* is a minimization problem, a solution with lesser objective function value is considered to be more fit than a solution with higher objective function value.

4.3.2 Initial solution generation

For initial population generation, we have used two sub-processes. First solution is obtained through *M-Moore* heuristic [101]. The remaining solutions are generated in an iterative manner by utilizing the concept of opposition based solution generation in combination with concept of *NEH* heuristic, in the same manner as discussed in previous chapters. In each iteration, first a random permutation of jobs is determined and the opposite solution of this random permutation is calculated. Subsequently, *NEH* heuristic is applied on both the solutions. The better solution in objective space is added to the population. The process iterates for (p-1) times, where p is the population size. In each iteration, the uniqueness of the generated solution is ensured by comparing the solution with the existing solutions in the population. If the newly generated solution is not identical with any solution, then only it is added in population.

***M-Moore* heuristic:** *M-Moore* heuristic presented in [101] was derived from *Moore* algorithm [105]. We have used the same approach to generate first solution in the population.

Following steps describes the *M-Moore* heuristic by assuming that all jobs are ready for processing at beginning, i.e, ignoring the release date of the jobs.

Step 1: Assume $\chi = \{J_1, J_2, \ldots, J_n\}$ and two initially empty solutions σ and τ.

Step 2: The jobs in χ are sorted in a non-decreasing order as per their k^{th} due date, where ($k \in \{1, 2, \ldots, m\}$) and the resulting sequence of the sorted jobs is π.

Step 3: The i^{th} job where $i \in (1, \ldots, n)$ is selected from π and inserted in partial solution σ. The completion time of the newly inserted (i^{th}) job is calculated. If the completion time exceeds the k^{th} due date of this job then the job having longest processing time in the partial solution σ is removed and inserted into τ and new longest processing time again recalculated in σ. This step iterates n times and afterwards, the jobs in partial solution τ is copied to solution χ. The steps 2 and 3 are repeated again.

Step 4: The above two steps are reiterated again and again upto *m* times, where *m* is the number of due dates.

Step 5: At end of this procedure, partial solution σ contain all the jobs which satisfies some due date d_k, where $k \in \{1, \ldots m\}$ and τ contains the remaining jobs. The complete solution S is obtained by appending jobs in τ to σ.

4.3.3 Crossover operator

Our crossover operator is a slightly modified version of uniform order based (UOB) crossover [34]. It takes two parents $P1$ & $P2$ as input, where $P1$ is the solution obtained through mutation as explained already and $P2$ is always the target solution. Our crossover operator begins by creating an empty child solution. Thereafter, instead of copying jobs at each position in first parent to the corresponding position in child with certain probability like UOB crossover, our crossover operator selects θ positions randomly in first parent and copies the jobs at these selected positions in child in the same positions as they were in first parent. The remaining positions in child is filled with the unselected jobs in second parent in the same sequence. For illustration, assume the two parents viz. $P1$={8, 4, 7, 2, 5, 9, 3, 1, 6} and $P2$ ={9, 2, 7, 4, 5, 6, 3, 8, 1}. Suppose the θ is equal to 4 and child inherits positions {2, 3, 6, 8} from $P1$ i.e., the partially created child after this step will be {_, 4, 7, _, _, 9, _, 1, _}. The remaining positions in child will be filled with the unselected jobs in $P2$ in the same sequence i.e, unselected jobs

Algorithm 11: Pseudo-code of HDDE approach for *SMTSTP-R*

Input: HDDE parameters and a *SMTSTP-R* instance
Output: Best solution obtained by HDDE

```
for (i=1 to p) do
    S_i ← Initial_Solution();
Best_Sol ← best solution among S_1, S_2, ... S_i, ..., S_p;
while (termination criteria remains unsatisfied) do
    for (i=1 to p) do
        S_rand ← Select one solution from population randomly;
        Mutant=mutation(S_rand);
        if (Mutant is better than Best_Sol) then
            Best_Sol ← Mutant ;
            Trial_Sol ← Mutant;
            Crossover_flag ← 0;
        else
            Crossover_flag ← 1;
        if (Crossover_flag = 1) then
            if (U_01 < p_c) then
                // U_01 is a uniform variate between [0,1].
                Trial_Sol ← crossover(mutant,S_i) ;     // S_i is the i^th population
                member.
        if (f(Trial_Sol) < f(Best_Sol)×(1+μ)) then
            // f(Trial_Sol) & f(Best_Sol) are the objective value of
               Trial_Sol and Best_Sol respectively.
            Trial_Sol ← Local_Search(Trial_Sol) ;
        if ((Trial_Sol is unique) and (Num_Pop(S_1, S_2, ..., S_p, Trial_Sol) < 10)) then
            if (Trial_Sol is better than S_i) then
                S_i ← Trial_Sol ;
                if (S_i is better than Best_Sol) then
                    Best_Sol ← S_i ;
return Best_Sol;
```

present in $P2$ are {2, 5, 6, 3, 8}. The complete child after the crossover step will be {2, 4, 7, 5, 6, 9, 3, 1, 8}. The parameter θ used here is determined empirically.

4.3.3.1 Mutation operator

The mutation operator employed in this work consists of two swap operations. The three positions, say *x*, *y* and *z*, in the original solution, are selected randomly. The first swap operation interchanges the jobs at positions *x* and *y*, and the next swap operation interchanges the new job

at position x with the job at position z. Algorithm 12 represents the pseudo-code of mutation operator.

Algorithm 12: Pseudo-code for mutation

Input: One solution S
Output: The mutant of S

Mutation_Procedure(S)
begin
    k_1=choose a random number between 1 to n ;
    k_2=choose a random number between 1 to n ;
    k_3=choose a random number between 1 to n ;
    // n is the number of jobs
    Swap(S_{k_1},S_{k_2}) ;
    // S_{k_1} and S_{k_2} are the jobs at positions k_1 and k_2 in solution S
    Swap(S_{k_1},S_{k_3}) ;
    return S;

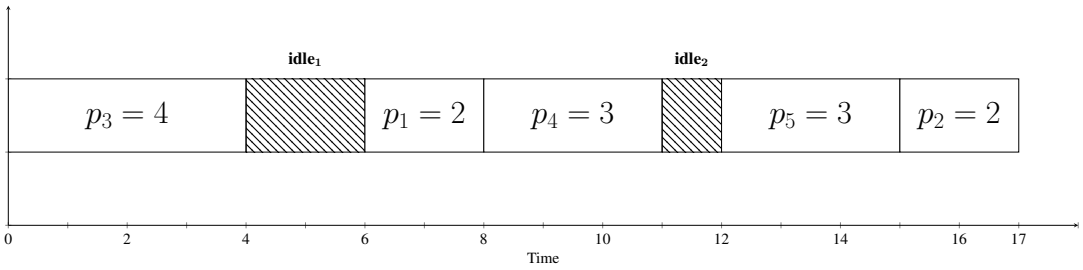
4.3.4 Local search

A series of local searches is used to improve the quality of solutions obtained by crossover/mutation in the proposed approach. We have used four local search procedures which were presented in [102]. The description of these are provided below:

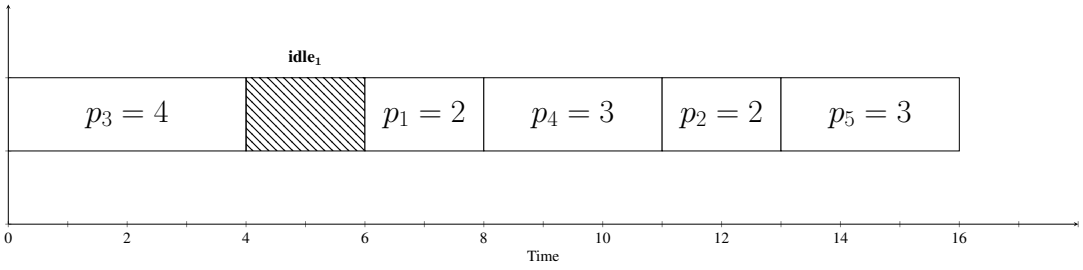
It is possible that a job may have completed prior to the release date of succeeding job in the schedule. This incurs an idle time for machine. Also, this can occur at several places in a schedule. The first local search aims at eliminating these idle times so as to improve the objective function value. This local search considers each idle slot once in the order in which they occur in the schedule. Each job in a schedule is checked one-by-one and the job, say J_k, causing idle time is determined. All the jobs succeeding J_k in the schedule can be checked one-by-one to find out whether inserting any one of them in the position of J_k after moving J_k and other affected jobs one position towards the end eliminates the idle time and improves the objective function value. If J_k is the first job in the schedule, then there may not be any job with release date zero, so instead of searching for a job which can eliminate this idle time altogether, we search for a job which can decrease this idle time and improves the objective function value. The first such job say J_l is inserted by moving the affected jobs one position towards the end, and the next idle slot in the schedule is determined and considered for elimination. If there is

no such job in the schedule, then this idle slot is left unchanged and the next idle slot in the schedule is determined and considered for elimination.

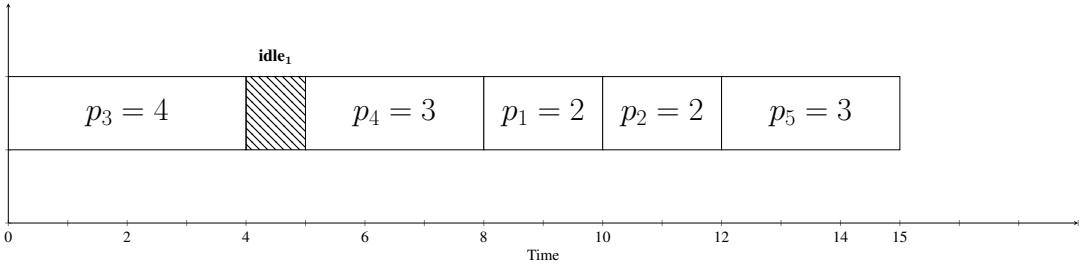
The second local search tries to minimize the idle time instead of eliminating it altogether. If insertion of a job J_l in place of job under consideration J_k incurs the same idle time, but reduces the objective function value then this insertion is also allowed in second local search.



(a) **A schedule of five jobs** $\{J_3, J_1, J_4, J_5, J_2\}$



(b) **After first local search the schedule of jobs** $\{J_3, J_1, J_4, J_2, J_5\}$



(c) **After second local search the schedule of jobs** $\{J_3, J_4, J_1, J_2, J_5\}$

Figure 4.2: Illustrating first and second local searches

To illustrate these two local searches, let us consider an example with five jobs, viz. $\{J_1, J_2, J_3, J_4, J_5\}$. Assume the release dates and processing times of these jobs are $\{6, 9, 0, 5, 12\}$ and $\{2, 2, 4, 3, 3\}$, respectively. Figure 4.2 illustrates the effect of first and second local search procedures on a schedule of these jobs, viz. $\{J_3, J_1, J_4, J_5, J_2\}$. The completion time of J_3 is 4 and the next job that is scheduled after J_3 is J_1. Since, the release date of J_1 is 6, thus it

results in an idle time (idle_1) from time unit 4 to 6. Similarly, the given schedule has another idle time (idle_2) from 11 to 12 because of later release of J_5 and earlier completion of J_4. First local search tries to eliminate idle_1, but unable to eliminate this because of unavailability of any job (other than J_3) with release date less than equal to 4 time unit. The second idle time (idle_2) is eliminated by inserting J_2 in place of J_5. The second local search minimized the first idle time (idle_1) by inserting J_4 in place of J_1 and moving J_1 and other affected jobs one position towards the end. After applying first and second local search procedures on the original schedule, the resulting schedule will be $\{J_3, J_4, J_1, J_2, J_5\}$.

Adjacent pairwise interchange (API) local search is used iteratively in our third local search. During each iteration, API local search considers each position except the last position one-by-one starting at the first position in the schedule and swaps the job at the position, say i, under consideration with the job at next position $(i + 1)$ in the schedule whenever doing so either decreases the objective function value or leaves the objective function value unchanged but the completion time of the job at position $(i + 1)$ after the swap is less than the same before the swap. API local search is iteratively applied till there is no change in objective function value and completion time of any job during an iteration.

The fourth local search is used only when the solution provided to it as input is better than the best solution found so far. The reason for this policy is higher computational cost associated with this local search. Like the previous local search, this local search is also used iteratively and during each iteration considers each position except the last position one-by-one starting at the first position in the schedule. This local search considers the jobs at positions after the position under consideration one-by-one for insertion at position under consideration by shifting the affected jobs one position towards the end. The first job that improves the objective function value is inserted and the search continues with the first job which is yet to be considered for insertion at the position under consideration. When all jobs that are at positions after the position under consideration in the schedule have been tried, the next position in the schedule becomes the position under consideration. This local search stops when there is no improvement in objective function value during an iteration.

4.3.4.1 Perturbation procedure

The perturbation procedure is employed to prevent a solution from permanently getting trapped in a local optima. A target solution is said to be improved if it is replaced by a new better trial solution. The perturbation procedure is used on a target solution, if it does not improve for

consecutive λ generations. Perturbation procedure takes such a solution as input and produce a perturbed solution from it by removing ρ jobs randomly from the solution and then reinserting these ρ jobs one-by-one in the order in which these jobs are removed. To reinsert a job, all possible insertion positions are tried and job is inserted at the best position. λ and ρ are two parameters for the perturbation procedure whose values to be determined empirically. Algorithm 13 provides the pseudo-code of the perturbation procedure. Perturbed solution is subjected to series of local searches and then it replaces the corresponding target solution in the population. This perturbation procedure is not used for the instances having number of jobs equal to 500. This procedure works well on instances with number of jobs less than equal to 200, but on larger instances with number of jobs equal to 500, it does not perform well. The larger instances require more iterations to converge, hence chances of getting trapped in local optima within λ consecutive generations is very less. This could be the possible reason behind the poor performance of perturbation procedure on larger instances.

Algorithm 13: Pseudo-code for Perturbation

Input: A solution S
Output: The perturbed solution S'

Perturb_Procedure(S)
begin
  // S_r stores the jobs removed from S
  for *i=1 to* ρ **do**
    Remove one job J_j from S randomly without disturbing relative ordering of remaining jobs;
    $S_r(i) \leftarrow J_j$;
  $S' \leftarrow S$;
  for *i=1 to* ρ **do**
    $S' \leftarrow$ best permutation obtained by inserting job $S_r(i)$ in all possible positions of S';
  return S';

4.4 Computational results

The proposed *HDDE* approach has been implemented in C language. All computational experiments are carried out on an Intel Core i5-7500 processor based system with 8GB of RAM running at 3.40GHz under Ubuntu 18.04. The performance of our approach is tested on the same instances as used in [101] and [102]. The test instances generated in [101]

consists of two sets, viz. Set I & Set II. Set I consists of instances which contains the jobs $n \in \{10, 30, 50, 100, 200, 500\}$ and number of due dates $m \in \{2, 3, 4, 9\}$. For each combination of n and m, 90 instances were generated in [101]. The approaches are not compared for instances with n = 10, as these instances are too small to assess the relative performance of different approaches. Hence, total 1800 instances of Set 1 are used to evaluate the performance of proposed approach with state-of-the-art approaches. Tseng et al. had not considered the instances with n = 500 to evaluate their approaches in [101], and, hence these are also not considered in [102]. Since the code of [102] is available with us, so we have again re-executed the *HABC* and *HGA* in the same computational environment as available for our approach for all instances including n = 500. Hence, the execution times reported for these two approaches here in this chapter are less than those reported in [102]. Set II consists of instances with number of jobs $n \in \{50, 100, 150, 200\}$ and number of due dates $m \in \{3, 4, 9\}$, and, for each combination of n and m, 80 instances were generated. Therefore, Set II consists of a total 960 instances.

The values for various parameters used in the proposed *HDDE* approach are determined empirically after making numerous trial runs. These parameters are divided into two classes on the basis of size of the instance. The former class (C1) consists of all instances with $n \leq 200$ and the latter (C2) is for all instances with n = 500. For C1, the population size p is set to 100, crossover probability p_c = 1. The parameter θ in crossover is set to $0.5 \times n$. The μ value for applying local search on trial solution is set to 0.6. The parameter ρ in perturbation procedure is equal to 5 for $n < 100$ and 15 otherwise. Analogously in C2, p = 200, p_c = 0.6, $\theta = 0.2 \times n$ and μ value is set to 0.4. The perturbation procedure is not used on instances with n =500, hence ρ is not defined in C2.

If any target solution takes part in crossover for λ consecutive times and it does not get replaced by a new trial solution then the perturbation procedure is used on this target solution. The parameter λ here is set to 1000. The *HDDE* approach terminates if the best solution does not improve for 2000 successive generations or the best solution does not show the improvement for 1000 successive generations in first 1500 iterations. It is pertinent to mention that one generation contains p iterations, where p is the size of population. Our approach has been executed for 10 independent trials on each instance like the *HGA* and *HABC* approaches of [102].

Table 4.1: Results of HGA, HABC and HDDE on Set I instances

n × m	HGA			HABC			HDDE		
	APD		ART	APD		ART	APD		ART
	BEST	AVG	(secs)	BEST	AVG	(secs)	BEST	AVG	(secs)
30 × 2	0.10	0.46	0.22	0.04	0.25	0.07	0.01	0.03	0.32
30 × 3	0.07	0.28	0.23	0.02	0.11	0.10	0.00	0.03	0.41
30 × 4	0.07	0.26	0.24	0.03	0.12	0.11	0.00	0.01	0.46
30 × 9	0.01	0.07	0.28	0.01	0.03	0.17	0.00	0.00	0.69
50 × 2	0.53	1.01	0.27	0.19	0.49	0.15	0.03	0.03	0.97
50 × 3	0.38	0.77	0.29	0.10	0.23	0.18	0.05	0.02	1.10
50 × 4	0.17	0.57	0.33	0.11	0.25	0.21	0.01	0.00	1.20
50 × 9	0.14	0.31	0.46	0.04	0.08	0.33	0.01	0.01	1.80
100 × 2	0.96	1.43	1.16	0.60	0.73	0.73	0.07	0.06	11.54
100 × 3	0.67	1.03	1.26	0.33	0.37	0.83	0.03	0.02	6.79
100 × 4	0.53	0.87	1.39	0.17	0.25	0.98	0.04	0.02	6.10
100 × 9	0.34	0.60	2.11	0.08	0.11	1.49	0.03	0.03	8.02
200 × 2	0.99	1.21	7.29	0.60	0.44	5.13	0.21	0.58	52.57
200 × 3	0.85	1.05	7.89	0.39	0.31	6.11	0.06	0.11	29.47
200 × 4	0.69	0.88	7.96	0.26	0.20	6.43	0.04	0.04	26.01
200 × 9	0.47	0.65	10.79	0.12	0.09	10.35	0.04	0.03	36.04
500 × 2	3.70	4.57	94.44	4.19	4.16	53.61	0.00	0.01	184.11
500 × 3	2.46	3.43	97.43	2.52	2.57	61.14	0.02	0.04	269.26
500 × 4	1.95	2.79	104.39	1.80	1.90	69.10	0.05	0.07	315.83
500 × 9	0.91	1.44	132.13	0.38	0.63	125.05	0.17	0.21	625.68
Overall	0.80	1.18	23.53	0.60	0.67	17.11	0.04	0.07	78.92

4.4.1 Comparison of our approach with previously proposed approaches

We have used the same criteria as used in [101, 102] to show the superiority of *HDDE* over *HGA* and *HABC*. Table 4.1 and Table 4.2 present the comparison of approaches in terms of average percentage deviation (APD). APD is calculated for best as well as average solution quality obtained over 10 runs. APD is calculated by considering the solutions generated by three approaches viz. *HDDE, HABC, HGA*. Assume the best solution over 10 runs by an approach is S_A and the overall best solution by considering all approaches is S_{Best}. The APD is determined as

$$APD \ = \ 100 \times \frac{(S_A - S_{Best})}{S_{Best}} \tag{4.3}$$

In case of average solution quality, S_A is considered as average solution quality over 10 runs by an approach (A) and S_{Best} is the overall best average value obtained by one of the three approaches.

Table 4.2: Results of HGA, HABC and HDDE on Set II instances

n × m	HGA			HABC			HDDE		
	APD		ART	APD		ART	APD		ART
	BEST	AVG	(secs)	BEST	AVG	(secs)	BEST	AVG	(secs)
50 × 3	0.02	0.07	0.26	0.01	0.03	0.12	0.00	0.00	0.64
50 × 4	0.01	0.07	0.27	0.00	0.03	0.13	0.00	0.00	0.71
50 × 9	0.03	0.10	0.33	0.00	0.04	0.22	0.00	0.00	1.19
100 × 3	0.01	0.06	0.69	0.01	0.04	0.48	0.00	0.00	2.46
100 × 4	0.02	0.08	0.82	0.01	0.04	0.52	0.00	0.00	2.66
100 × 9	0.02	0.07	1.16	0.01	0.02	0.78	0.01	0.00	3.86
150 × 3	0.03	0.06	2.37	0.01	0.03	1.98	0.00	0.00	6.77
150 × 4	0.03	0.05	2.65	0.01	0.02	2.25	0.00	0.00	7.89
150 × 9	0.03	0.07	3.54	0.01	0.02	3.01	0.00	0.00	10.92
200 × 3	0.01	0.04	5.98	0.00	0.02	5.68	0.00	0.00	14.26
200 × 4	0.02	0.04	6.51	0.01	0.01	5.91	0.00	0.00	14.46
200 × 9	0.02	0.05	8.84	0.01	0.01	8.16	0.00	0.00	21.83
Overall	0.02	0.06	2.79	0.01	0.03	2.44	0.00	0.00	7.30

Table 4.3: Comparison of HDDE with HGA and HABC on Set I instances in terms of number of the instances on which HDDE yielded better (<), equal (=) and worse (>) solutions

n × m	HDDE </ = / >HGA						HDDE </ = / >HABC					
	Best			Avg			Best			Avg		
	<	=	>	<	=	>	<	=	>	<	=	>
30 × 2	12	77	1	46	43	1	6	82	2	35	50	5
30 × 3	9	81	0	33	57	0	5	85	0	24	65	1
30 × 4	10	80	0	53	37	0	4	85	1	29	57	4
30 × 9	3	87	0	27	63	0	1	89	0	19	70	1
50 × 2	41	48	1	80	9	1	23	65	2	64	17	9
50 × 3	35	55	0	78	11	1	12	67	11	63	15	12
50 × 4	32	58	0	84	6	0	22	63	5	66	20	4
50 × 9	24	63	3	77	12	1	11	74	5	61	18	11
100 × 2	81	5	4	89	0	1	67	14	9	83	0	7
100 × 3	77	8	5	90	0	0	57	24	9	79	0	11
100 × 4	75	13	2	90	0	0	49	29	12	82	0	8
100 × 9	72	15	3	89	1	0	37	31	22	65	3	22
200 × 2	77	1	12	81	0	9	66	1	23	73	1	16
200 × 3	86	0	4	88	0	2	66	0	24	75	0	15
200 × 4	86	0	4	90	0	0	69	0	21	64	0	26
200 × 9	87	0	3	90	0	0	59	1	30	61	0	29
500 × 2	90	0	0	90	0	0	89	0	1	89	0	1
500 × 3	90	0	0	90	0	0	78	0	12	73	0	17
500 × 4	88	0	2	88	0	2	67	0	23	59	0	31
500 × 9	80	0	10	80	0	10	27	0	63	30	0	60
Overall	1155	591	54	1533	239	28	815	710	275	1194	316	290

Table 4.4: Comparison of HDDE with HGA and HABC on Set II instances in terms of number of the instances on which HDDE yielded better (<), equal (=) and worse (>) solutions

n × m	HDDE </ = / >HGA						HDDE </ = / >HABC					
	Best			Avg			Best			Avg		
	<	=	>	<	=	>	<	=	>	<	=	>
50 × 3	4	76	0	38	42	0	2	78	0	26	54	0
50 × 4	5	75	0	39	41	0	0	80	0	30	49	1
50 × 9	13	67	0	57	23	0	3	77	0	40	36	4
100 × 3	7	73	0	42	38	0	10	70	0	36	41	3
100 × 4	13	67	0	52	28	0	8	72	0	44	33	3
100 × 9	22	55	3	71	9	0	10	65	5	57	16	7
150 × 3	17	63	0	50	30	0	13	65	2	47	32	1
150 × 4	15	64	1	50	30	0	13	67	0	43	36	1
150 × 9	30	48	2	62	17	1	15	59	6	50	20	10
200 × 3	14	66	0	48	32	0	8	72	0	47	31	2
200 × 4	19	61	0	44	35	1	12	65	3	43	34	3
200 × 9	34	45	1	68	10	2	16	57	7	56	10	14
Overall	193	760	7	621	335	4	110	827	23	519	392	49

In Table 4.1 and Table 4.2, each row reports the average APD value on a group of instances. Each group in Table 4.1 and Table 4.2 contains 90 and 80 instances respectively, as discussed in the first paragraph of this section. The ART is the average running time in seconds taken by an approach considering all the runs on all the instances in the group. The last row (overall) provides the summarized results over all instance groups. Table 4.3 and Table 4.4 provide the comparison of *HDDE* with *HABC* and *HGA* in terms of number of instances on which the solution obtained by *HDDE* is better ($<$), equal ($=$) and worse ($>$) on each group of instances. This comparison is done for best and average solution values both. These four tables clearly show the superiority of *HDDE* over *HABC* and *HGA* in terms of solution quality and the performance gap widens in general as the instance size increases.

Table 4.1 and Table 4.3 show that the performance of *HDDE* approach is much better for instances with n=500, hence the proposed approach is more suitable in comparison to *HABC* and *HGA* for solving large size instances. However, the execution time of *HDDE* is higher than the two previous approaches. This can be attributed to high computational cost associated with the use of heuristics to generate initial population and the use of perturbation procedure in our approach.

4.4.2 Statistical significance of *HDDE* approach

To check whether the superior performance of *HDDE* approach over the state-of-the-art approaches namely, *HABC* and *HGA* is statistically significant, we have applied two tailed Wilcoxon signed rank test [109]. For this test, the significance level (α) is set to 0.01 and the critical value (z_c) is 2.58. This is a pairwise test conducted by considering each instance's normalized best as well as average objective value, obtained by the three approaches. This test takes two related samples and ranks the absolute differences in ascending order. Table 4.5 and Table 4.6 present the results of this test for Set I for best solution quality and average solution quality over 10 runs respectively. Likewise Table 4.7 and Table 4.8 present the results of this test for Set II. In these tables, the second column NWT represents those instances where differing objective values are yielded by the two compared approaches, and $total$ represent the total number of instances. The R^+ is the total sum of ranks of the instances where performance of *HDDE* approach is superior than the existing approach used in comparison (given in first column in the table). Similarly, the R^- is the total sum of ranks for the instances where performance of *HDDE* approach is worse than the existing approach used in comparison. In Wilcoxon signed rank test, the difference between performances of two approaches is significant if $|Z| > |Z_{Cri}|$. These tables clearly demonstrate that the superior performance of *HDDE* in terms of best as well as average objective values both is statistically significant in comparison with state-of-the-art approaches for *SMTSTP-R*.

Table 4.5: Results of Wilcoxon signed rank test on Set I instances for best solution quality

	HDDE					
	$NWT/Total$	R^+	R^-	$\|Z\|$	Z_{Cri}	Significant
HABC	1090/1800	478141	116454	17.396	2.580	Yes
HGA	1209/1800	712627	18818	28.569	2.580	Yes

Table 4.6: Results of Wilcoxon signed rank test on Set I instances for average solution quality

	HDDE					
	$NWT/Total$	R^+	R^-	$\|Z\|$	Z_{Cri}	Significant
HABC	1484/1800	936062.5	165807.5	23.325	2.580	Yes
HGA	1561/1800	1212866.0	6275.0	33.869	2.580	Yes

Table 4.7: Results of Wilcoxon signed rank test on Set II instances for best solution quality

	HDDE					
	$NWT/Total$	R^+	R^-	$\|Z\|$	Z_{Cri}	Significant
HABC	133/960	7282.5	1628.5	6.349	2.580	Yes
HGA	200/960	19532.5	567.5	11.570	2.580	Yes

Table 4.8: Results of Wilcoxon signed rank test on Set II instances for average solution quality

	HDDE					
	$NWT/Total$	R^+	R^-	$\|Z\|$	Z_{Cri}	Significant
HABC	568/960	153743.5	7852.5	18.642	2.580	Yes
HGA	625/960	195401.0	224.0	21.610	2.580	Yes

4.5 Conclusions

In this chapter, we presented a discrete differential evolution based approach with local search for single machine total stepwise tardiness problem with release dates (*SMTSTP-R*). It is an $\mathcal{NP}$-hard problem and it has resemblance with various real life problems particularly in transportation domain. Our approach makes use of two constructive heuristics and the concept of opposition based solution generation to create the initial population of solutions. A perturbation procedure is used to evade from the local optima. The computational results on the benchmark instances reveal the superiority of our proposed approach over the state-of-the-art approaches in terms of solution quality.

Chapter 5

Rescue Unit Allocation and Scheduling Problem

5.1 Introduction

This chapter addresses the rescue unit allocation and scheduling problem (*RUASP*) with fuzzy processing times. The goal of *RUASP* is to efficiently assign and schedule the rescue units to process the incidents in the event of a natural disaster. This problem can be considered as a variant of the unrelated parallel-machine scheduling problem with sequence and machine-dependent setup times. This problem have characteristics of permutation and grouping both.

A disaster is any event that results in tremendous damage, destruction or harm to humans. Disasters can be geophysical (such as earthquakes, landslides), hydrological (floods, tsunamis) or biological (such as the outbreak of an epidemic). Natural disaster involves all those natural processes of the earth which creates an adverse effect on humans. Each year, it causes the loss of millions of human life and a huge loss of assets. In [110], the authors highlighted the difference between natural hazards and natural disasters. A natural hazard is defined as a naturally occurring phenomenon that leads to loss of human life, destruction of properties or environmental degradation. A natural disaster is the consequence of a natural hazard if the available resources are unable to control the adverse effects of the hazard. In the event of any natural disaster, the first task which needs to be performed is to initiate the immediate response activities. The response activities include the coordination and management of resources to implement disaster response plans. The well-planned usage of available resources such as rescue units reduces the adverse impact of the disaster. Thus, efficient assignment and scheduling of

rescue units in the affected region is considered as a key issue in the response phase [111, 112].

This chapter is concerned with how to assign and schedule a given set of rescue units to process the given set of incidents, such that the total weighted completion time of incidents is minimized. The weights of incidents represent the severity levels of the incidents. This problem is termed as rescue unit allocation and scheduling problem (*RUASP*) in the literature. Wex et al. [113] introduced the *RUASP* and proved its $\mathcal{NP}$-hardness. To address *RUASP*, they proposed several heuristics and *GRASP* metaheuristic. The *GRASP* performed better only for few instances, whereas the performance of heuristics (particularly Schedule7 heuristic) is better for most of the instances.

In case of a natural disaster, the exact information such as the processing times of incidents are not known in advance. Thus, considering fuzzy processing times to represent uncertain information is more appropriate while modeling such problems. Cunha et al. [114] extended the work of Wex et al. [113] by adding fuzzy processing times in the original *RUASP* model. To address the new version of *RUASP* with fuzzy processing times, they presented the Biased Random Key Genetic Algorithm (BRKGA) approach and compared the performance of *BRKGA* with the best constructive heuristic (viz. Schedule7) proposed in [113].

Being a recently introduced problem, *RUASP* is a relatively under-explored problem as only two metaheuristic approaches exist in literature. Hence, there is a lot of scope to explore different metaheuristics for this problem. To address *RUASP*, we have presented a steady-state grouping genetic algorithm approach. The crossover and mutation have been devised by keeping the characteristics of the problem and the objective in mind. We have used a mixed strategy of greedy and random heuristics to generate the initial solution. This combination provides a balance of greediness and randomness and yields a set of solutions with superior quality along with diversity. We have implemented two versions of our approach. The first version addresses the *RUASP* with fuzzy processing times presented in [114]. While the second version seeks the solution for the original version of *RUASP* and is used to compare the approaches presented in [113]. The comparison of experimental results demonstrates that our proposed approach provides better solutions, and also consumes very short execution times in comparison with both the existing approaches for *RUASP*. We have conducted a robustness test and demonstrated that our approach is more robust than the existing approach.

The subsequent sections of this chapter is structured as follows: Section 5.2 presents the problem definition of *RUASP*, whereas Section 5.3 provides an overview of related work, identifies the research gap and overview of related fuzzy theory. In Section 5.4, we have

proposed a steady-state grouping genetic algorithm approach for *RUASP*. The experimental results and their analysis are presented in Section 5.5. Section 5.6 presents the brief summary of the main findings of this study and outlines some concluding remarks.

5.2 Problem definition

This section presents the problem formulation of *RUASP* along with notational conventions used in this chapter. We have used a similar formulation and notational conventions as presented in [113]. The *RUASP* consists of a set R of *m* rescue units viz. $R = \{R_1, R_2, \ldots, R_m\}$ and a set I of *n* incidents, viz. $I = \{i_1, i_2, \ldots, i_n\}$. The incidents need to be processed and each incident must be processed by exactly one rescue unit. A rescue unit cannot process more than one incident at a time. Interruption of the ongoing processing of an incident is not allowed. The number of rescue units is less than or equal to the number of incidents, i.e., $m \leq n$. Such a restriction depicts a realistic scenario in natural disasters. Each incident $i \in \{1, \ldots, n\}$ is associated with an incident severity factor (ω_i), which represents the level of destruction of the incident i. The processing time of an incident i depends on the assigned rescue unit k and it is represented as p_i^k. Any rescue unit begins from the depot (starting point) to process the very first incident and after processing the last incident, eventually returns to the depot. Two fictitious incidents 0 and $n + 1$ have been added here to represent the depot. These fictitious incidents do not require any processing time i.e., $p_0^k = p_{n+1}^k = 0$. Since different incidents have different severity factors, thus all rescue units cannot process all the incidents. A rescue unit that is competent in processing the specific incident can only be assigned to that incident. If a rescue unit k is sufficient to handle an incident i, then Cap_{ki} is represented as 1, otherwise 0. Thus, each incident can be processed by only a subset of available rescue units. The travel time between the depot and an incident or between two incidents varies as per the rescue unit processing the incidents and also depends on the sequence of incidents. The travel time of a rescue unit k to travel from incident i to another incident j is presented as t_{ij}^k. This travel time is also known as sequence and unit dependent setup times in literature. Two decision variables viz., α_{ij}^k and β_{ij}^k have been used in the problem formulation. The α_{ij}^k is 1, if the rescue unit k processes the incident i just before incident j, otherwise it is 0. Similarly, β_{ij}^k is set as 1, if the rescue unit k processes the incident i (at any time) before incident j, or else assigned as 0. Here, $i, j \in$ {0,1,. . . ,n} and $k \in$ {1,2,. . . ,m}. The completion time (C_i) of an incident i is the ending time at which some rescue unit finishes the processing of the incident. The objective of *RUASP*

is to find the schedules and assignment of *m* rescue units to incidents that minimizes the total weighted completion time of incidents:

$$\min \sum_{j=1}^{n} \omega_j C_j \tag{5.1}$$

where C_j is the completion time of an incident j. In [113], the authors have presented the binary quadratic formulation of the model. Based on the formulation, the mathematical model of *RUASP* can be written as:

$$\min_{\alpha_{ij}^k, \beta_{ij}^k} \sum_{j=1}^{n} \left(\omega_j \sum_{i=0}^{n} \sum_{k=1}^{m} \left[p_i^k \beta_{ij}^k + \left(p_j^k + t_{ij}^k \right) \alpha_{ij}^k + \beta_{ij}^k \left(\sum_{l=0}^{n} \alpha_{li}^k t_{li}^k \right) \right] \right) \tag{5.2}$$

subjected to the constraints:

$$\sum_{i=0}^{n} \sum_{k=1}^{m} \alpha_{ij}^k = 1, \qquad j = 1, \ldots, n, \tag{C1}$$

$$\sum_{j=1}^{n+1} \sum_{k=1}^{m} \alpha_{ij}^k = 1, \qquad i = 1, \ldots, n, \tag{C2}$$

$$\sum_{j=1}^{n+1} \alpha_{0j}^k = 1, \qquad k = 1, \ldots, m, \tag{C3}$$

$$\sum_{i=0}^{n} \alpha_{i(n+1)}^k = 1, \qquad k = 1, \ldots, m, \tag{C4}$$

$$\beta_{il}^k + \beta_{lj}^k - 1 \leq \beta_{ij}^k, \qquad i = 0, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m; \quad l = 1, \ldots, n, \tag{C5}$$

$$\sum_{i=0}^{n} \alpha_{il}^k = \sum_{j=1}^{n+1} \alpha_{lj}^k, \qquad l = 1, \ldots, n; \quad k = 1, \ldots, m, \tag{C6}$$

$$\alpha_{ij}^k \leq \beta_{ij}^k, \qquad i = 0, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C7}$$

$$\beta_{ii}^k, \qquad i = 0, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C8}$$

$$\beta_{ij}^{k} \leq cap_{ki}, \quad i = 1, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C9}$$

$$\sum_{l=1}^{n+1} \alpha_{il}^{k} \geq \beta_{ij}^{k}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C10}$$

$$\sum_{l=0}^{n} \alpha_{lj}^{k} \geq \beta_{ij}^{k}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C11}$$

$$\alpha_{ij}^{k}, \beta_{ij}^{k} \in \{0, 1\}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n+1; \quad k = 1, \ldots, m, \tag{C12}$$

The first constraint (C1) guarantees that only one incident is processed before any other incident i. Similarly, constraint C2 assures that exactly one incident is processed after any incident i. C3 makes sure that each rescue unit begins from the depot. Furthermore, constraint C4 ensures the return of the rescue unit to the depot after processing the last incident. The C5 constraint forces the transitivity property in the sequence of incidents, whereas C6 ensures that each incident i must have some incident, including the depot, as immediate successor and predecessor. The equivalence relationship between an immediate predecessor and a general predecessor is established by constraint C7. Condition C8 restricts the reflexive characteristic of any incident as a predecessor. Constraint C9 prohibits the assignment of an incapable rescue unit to an incident. The constraints C10 and C11 ensure that the decision variable β_{ij}^{k} is assigned as 0 in case the incident i is not handled before j by rescue unit k. The final condition (C12) represents the model as the binary program.

RUASP can be considered as a problem related to both routing and scheduling. It can be modeled as modification of multiple Traveling Salesman Problem (mTSP) as well as the parallel-machine scheduling problem with unrelated machines, non-batch sequence-dependent setup times and a weighted sum of completion times as the objective function [113]. To illustrate *RUASP*, let us consider an example consisting of four rescue units and ten incidents, i.e., $m = 4$ and $n = 10$ such that $R = \{R_1, R_2, R_3, R_4\}$ and $I = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}\}$ with their respective severity factor {2,5,4,2,5,3,3,5,1,4}. The incident i_0 represents the depot. Table 5.1 represents the processing time matrix, where the rows correspond to rescue units and

the columns are presented as incidents. Each cell provides the processing time of the specific incident by the corresponding rescue unit. For example first cell contains the value 3, which shows that rescue unit R_1 consumes 3 seconds to process the first incident i_1. Processing time presented as the negative value (-1) depicts the incapability of the corresponding rescue unit to process that incident.

Table 5.1: Processing time matrix for the illustrative example.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
R_1	3	3	-1	3	6	3	-1	4	-1	6
R_2	5	-1	4	-1	-1	3	2	-1	5	9
R_3	-1	4	-1	7	4	-1	3	-1	-1	5
R_4	-1	-1	2	-1	4	-1	-1	7	6	8

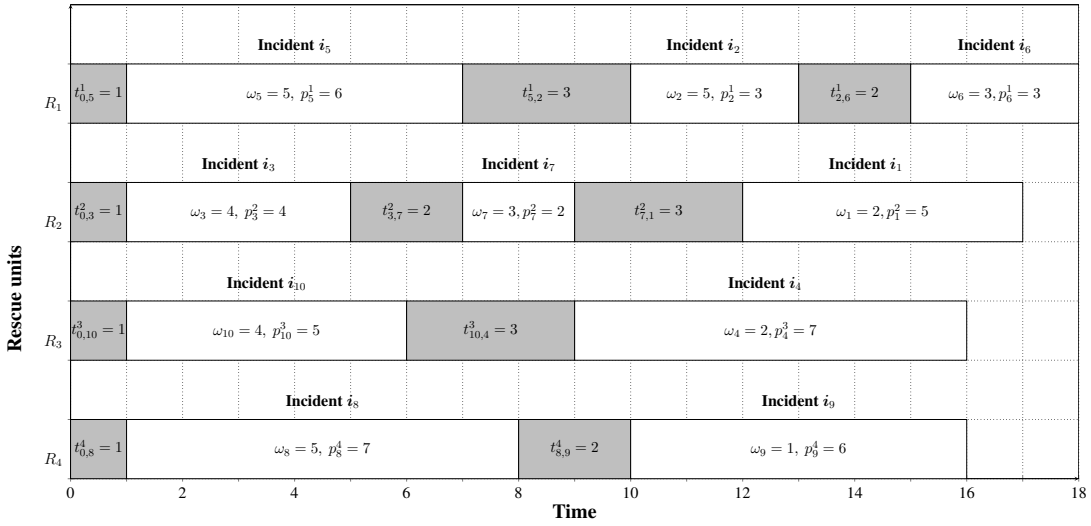
For each rescue unit, the travel time from the depot to the location of the first incident is set as 1 unit. The travel time between two incidents i_x and i_y is considered as 2, if $x < y$, otherwise assigned as 3. The consideration of only three kinds of values as travel time is done only for ease of understanding. More formally, the travel time is assumed as:

$$t_{i_x,i_y} = \begin{cases} 1, & \text{if x=0, i.e., } i_x \text{ is depot,} \\ 2, & \text{if x<y,} \\ 3, & otherwise. \end{cases}$$

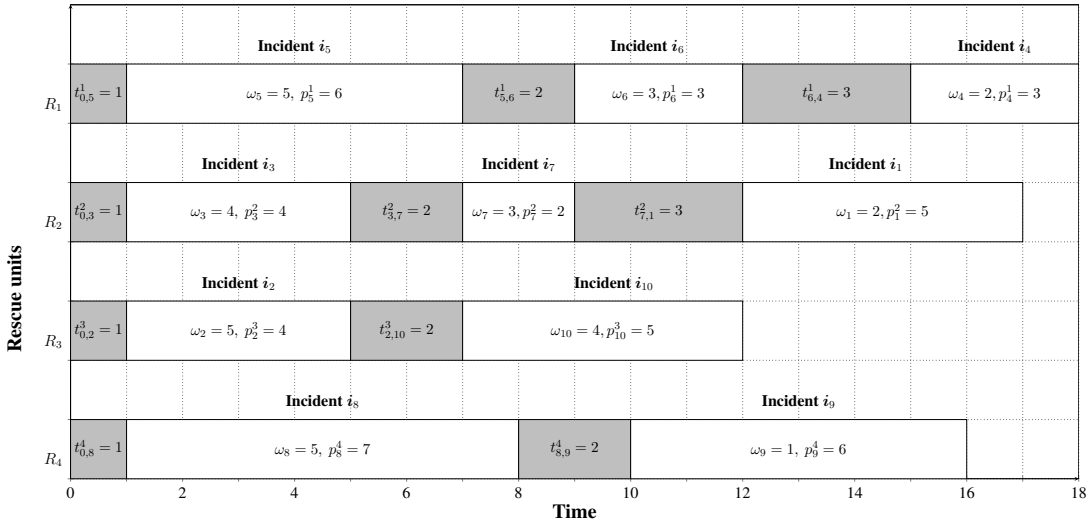
Figure 5.1 presents two solutions A and B of the illustrative example. The assignment and scheduling of incidents in the solution A is done as $R_1 = \{i_5, i_2, i_6\}$, $R_2 = \{i_3, i_7, i_1\}$, $R_3 = \{i_{10}, i_4\}$ and $R_4 = \{i_8, i_9\}$. The total weighted completion time of incidents ($\omega_i C_i$) in the solution A is 347. While, the solution B has the assignment and scheduling as $R_1 = \{i_5, i_6, i_4\}$, $R_2 = \{i_3, i_7, i_1\}$, $R_3 = \{i_2, i_{10}\}$ and $R_4 = \{i_8, i_9\}$, which incurs 311 as the total weighted completion time of incidents. Thus, solution B presents the better assignment and scheduling of incidents, as compared to solution A. Both the solutions differ in the assignment and scheduling of rescue units R_1 and R_3 only, while R_2 and R_4 in both solutions contain the same incidents in the same sequence.

5.3 Related work and existing approaches for *RUASP*

In this section, we first present the related work and then review the previously proposed approaches including the state-of-the-art approach for *RUASP* presented in [114]. We also



(a) **Solution A**



(b) **Solution B**

Figure 5.1: Two solutions of the illustrative example

discuss the research gap and elaborate the motivation for our proposed approach. In addition, we present a brief overview of related fuzzy theory used in *RUASP* presented in [114].

5.3.1 Related work

The activities in disaster management are classified into three phases [115, 116]. The first phase is the preparation phase that accounts for the time before the disaster. This phase involves

the preparation of plans, training the personnel, early forecast and organization of essential services. The second phase is the response phase. It deals with the activities required to perform immediately after the occurrence of the disaster. It primarily includes the coordination and management of resources to implement disaster response plans to mitigate the effects of the disaster. This phase is very crucial as any delay in response may result in more life/property loss. The last phase is the recovery phase which includes the restoration of the affected area back to some degree of normalcy [117, 118, 119, 120, 121, 122]. Some literature considers the three phases as a life cycle [123]. The decision making process used during response phase follows various methodology such as, approaches from applied statistics and probability theory [124, 125, 126, 127, 128], usage of optimization models [129], approaches based on computational intelligence [130, 131] and so on. Some models use empirical investigation based on past decision-making conclusions [132], whereas the research presented in [124, 133] considered the centralized decision-making principles. Literature on disaster management has established the fact that the proper assignments and schedules result in the minimization of loss. The authors proposed a mathematical programming model in [134], which considers the centralized coordination for the assignments and scheduling of distributed rescue units. At this time it was observed that all of the existing models used in emergency operations centers assume the fixed-length time to process all incidents by any rescue unit and does not account for the fact that the incidents may have a different severity level. Wex et al. suggested mathematical models in [135, 136], which address this weakness in the existing models for decision support process in disastrous events.

5.3.2 Analysis of existing methods for RUASP

As already mentioned in Section 5.1, the Rescue Unit Allocation and Scheduling Problem (*RUASP*) was introduced by Wex et al. [113]. They gathered the information related to the earthquake and tsunami that occurred in Japan in 2011 from the German Federal Agency for Technical Relief (*THW*) and developed the model related to the decision support provided by the emergency operation centres during the response phase in case of natural disaster. The proposed model incorporates the fact that the incidents may have different severity levels and thus addressed the shortcomings in the existing models of that time. They have also generated the test instances based on the actual data collected from the *THW*. These instances share high similarities to the real-world scenario, for example, a disastrous event may affect an entire region and all incidents in that region need not have the same severity level. Based on the severity of

the incident and the capacity of the rescue unit, the rescue units may take different processing times to handle the incident. Travel time of rescue units from one location to another need not be the same, as rescue units may differ in size and traveling speed. These points made the proposed model resemble with the real-world scenario when a disaster strikes.

The authors of [113] developed a large set of heuristics to validate the proposed test instances for *RUASP* in [113]. To be specific, they have proposed eight construction heuristics and five improvement heuristics. Each construction heuristic is combined with an improvement heuristic. Consequently, the set contains many compositions of heuristics. They also proposed *GRASP* metaheuristic based approach for *RUASP*. *GRASP* is a well-known multi-start metaheuristic [137] in which each generation consists of two phases. The first one is the construction phase, while the other is the local search. The *GRASP* metaheuristic proposed here uses the construction heuristics in the first phase and improvement heuristics as the local search in the second phase. To address *RUASP*, two versions were presented in [113], viz., the classical version which is the composition of heuristics only and the other is *GRASP* which uses these compositions of heuristics as its component. Although the classical version uses the same composition of construction and improvement heuristics as *GRASP*, the analysis of the results reveals that the performance of *GRASP* is better only on few instances in comparison with the classical version, i.e., the classical version produced better results on most instances. The performance of these heuristics varies from one instance to another. The Schedule7 heuristic (one of the construction heuristics) performed better on all sets of instances.

If a disastrous event occurs, the precise information related to the incidents is generally not known apriori. Thus, considering uncertain information can add more relevance for the models developed for the decision making by emergency operations centres during the response phase of the disaster. Cunha et al. [114] extended the work presented in [113] by adding uncertainty to the incidents' processing times. They used fuzzy set theory for adding the uncertainty in the processing time. To address this fuzzy version of *RUASP*, authors presented the Biased Random Key Genetic Algorithm (BRKGA) approach and compared the performance of *BRKGA* with the best constructive heuristic (Schedule7) proposed in [113]. *BRKGA* performed better on all the sets of instances, except one of the sets, where Schedule7 performed better.

We investigated the two metaheuristics viz., *GRASP* [113] and *BRKGA* [114] proposed for *RUASP* and observed that none of them is considering the characteristics of the problem and its objective. *GRASP* is one among the highly effective metaheuristics and has been used to solve a wide range of combinatorial optimization problems. Typically, the second phase of *GRASP*

metaheuristic (also known as local search phase) is very crucial for the generation of good quality solutions [138]. In [113], the second phase of *GRASP* uses k-opt moves and multi-unit k opt moves, which are generic operations designed for traveling salesman problem and multiple traveling salesman problems, and, do not exploit the characteristics of the problem at hand and its objective. As a consequence, the *GRASP* approach in [113] does not perform well on most of the instances. The *BRKGA* approach presented in [114] also uses generic crossover and mutation operator and does not take into account the characteristics of the problem and its objective. The *GRASP* approach proposed by Wex et al. [113] takes 25.89 minutes for the largest instance. *RUASP* deals with the allocation and scheduling of rescue units during the response phase of a disastrous event. Thus, the running time of an algorithm is an important factor.

These factors served as the motivation to develop a new metaheuristic approach based on steady-state grouping genetic algorithm. The crossover and mutation operators used in our approach consider the problem-specific knowledge. Our proposed approach uses a combination of greedy and random heuristics to generate initial solutions. Such a combination of heuristics provides not only solutions of superior quality, but also diverse solutions, thereby facilitating the faster convergence of our approach to optimal/near optimal solutions. Thus, our proposed approach is fast and able to find superior quality solutions for all instances.

Although our proposed approach also based on genetic algorithm, still there is no similarity between the proposed approach and *BRKGA* presented in [114]. The *BRKGA* uses an entirely different encoding scheme for solution representation in comparison with the solution encoding of our approach. Additionally, our approach uses a steady-state model, where each generation produces one offspring and the newly generated offspring replaces the solution with the worst fitness in the population. On the other hand, *BRKGA* follows generational model, where each generation produces an entirely new set of solutions as population. The crossover and mutation used in *BRKGA* are also generic operators and do not exploit the characteristics of the problem and its objective.

5.3.3 Overview of related fuzzy theory

The goal of this section is to familiarize the concepts of fuzzy theory that are used in this problem. The *RUASP* addressed in this chapter contains fuzzy processing time, defined as a triangular fuzzy number (TFN). A triangular fuzzy number (TFN) is represented with three

points as: $\widetilde{A}$ = (a, b, c). This representation is interpreted as the membership function $\mu_{\widetilde{A}}(x)$ which can be described as :

$$\mu_{\widetilde{A}}(x) = \begin{cases} \frac{x-a}{b-a}, & \text{if } a \leq x \leq b, \\ \frac{x-c}{b-c}, & \text{if } b \leq x \leq c, \\ 0, & elsewhere. \end{cases}$$

Figure 5.2 is the graphical representation of a TFN, $\widetilde{A}$ = (a, b, c).

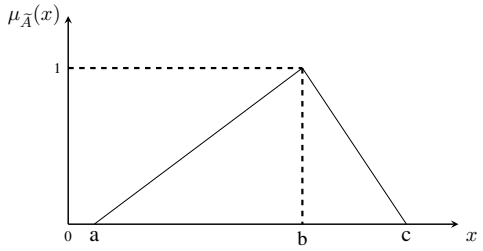


Figure 5.2: Graphical representation of a TFN $\widetilde{A}$ = (a, b, c)

The sum of two TFNs viz. $\widetilde{A_1}$ = (a_1, b_1, c_1) and $\widetilde{A_2}$ = (a_2, b_2, c_2) is computed as:

$$\widetilde{A_1} + \widetilde{A_2} = (a_1 + a_2, b_1 + b_2, c_1 + c_2)$$

The product between a scalar (λ) and a TFN $\widetilde{A}$ = (a, b, c) can be determined as:

$$\lambda\widetilde{A} = (\lambda a, \lambda b, \lambda c)$$

Ranking two fuzzy numbers: In [114], Cunha et al. have used "signed distance" method [139], to rank two fuzzy numbers. The signed distance [140, 141] of a TFN $\widetilde{A}$ = (a, b, c) from the y-axis is computed as follows:

$$d(\widetilde{A}) = \frac{1}{4}(a + 2b + c)$$

Based on this signed distance value, the ranking [140] of two TFNs $\widetilde{\mathcal{A}_1}$ and $\widetilde{\mathcal{A}_2}$ is determined as follows:

$$\widetilde{A_1} \prec \widetilde{A_2} \quad \text{if} \quad d(\widetilde{A_1}) < d(\widetilde{A_2})$$
$$\widetilde{A_1} = \widetilde{A_2} \quad \text{if} \quad d(\widetilde{A_1}) = d(\widetilde{A_2})$$
$$\widetilde{A_1} \succ \widetilde{A_2} \quad \text{if} \quad d(\widetilde{A_1}) > d(\widetilde{A_2})$$

Defuzzification approach: Defuzzification is the method of conversion of a fuzzy value into its corresponding crisp value. The authors in [114] have used centroid method to obtain a defuzzified value corresponding to a fuzzy number. The defuzzified value of a TFN, $\widetilde{A} = (a, b, c)$ is denoted by $\delta(\widetilde{A})$ and computed by the following formula:

$$\delta(\widetilde{A}) = (a + b + c)/3$$

5.4 Steady-state grouping genetic algorithm for *RUASP*

We have developed a grouping genetic algorithm approach for the RUASP, which uses the steady-state population replacement model [34]. In the steady-state model, each generation produces one offspring with the help of crossover and mutation. The fitness of offspring and worst member in the population is compared and if the offspring is better and unique with respect to all the current population members then it is added into the population in place of the worst member. The grouping genetic algorithm was presented by Falkenauer [40, 41] for grouping problems, i.e., problems where the objective is to divide a given set of items into various disjoint groups under some constraints so that a given cost function is optimized. The grouping genetic algorithm takes into account the specific structure of grouping problems and consequently, it performs better for grouping problems. Evidently, *RUASP* is a grouping problem. In addition, it has permutation aspect too. We have designed the crossover and mutation operators by considering the characteristics of the problem as well as the objective. The crossover operator considers both the grouping as well as permutation aspects. On the other hand, our mutation operator is mainly focused on the permutation aspect. Thus, the use of our proposed operators in the grouping genetic algorithm framework can manage the grouping aspect and the permutation aspect both. As a result, our proposed approach performs better than the state-of-the-art approaches available in the literature. Our proposed approach is inspired by the steady-state grouping genetic approach presented in [42]. Henceforth, the proposed approach will be referred to as *SSGGAFP* in this chapter. Our *SSGGAFP* approach uses fuzzy numbers arithmetic discussed in Section 5.3.3. We have also addressed the original version of *RUASP* presented by Wex et al. in [113] and it will be referred to as *SSGGACP*, hereafter. The use of fuzzy numbers arithmetic is the only difference between *SSGGAFP* and *SSGGACP*.

The process commences with initial population generation, which makes use of several heuristics to create the initial solutions. The idea behind using these heuristics is to obtain a

population that has a mixture of some better quality initial solutions along with diverse solutions. In each generation, the *SSGGAFP* approach selects two parents using binary tournament selection (BTS). The probability used in BTS for selection of parent is assigned with a value p_{bts}. The crossover operator takes two parents as input and produces one offspring. The mutation is performed on this offspring and a mutant solution is obtained. The offspring and mutant are evaluated in terms of fitness and the better one is selected as child solution. The fitness of the child solution is compared with the worst solution in the population and it replaces the worst solution if it is better. Also, the replacement is done only if the child solution is unique in population. A solution is considered unique if it is not identical to any solution in the population. The same process is repeated again and again as long as the termination condition remains unsatisfied. It is worth mentioning that the crossover operator will produce the exact copy of the parent in case both parents are the identical. Thus, if BTS yields the same population member as two parents, then only the mutation operator is used and the crossover phase is skipped. The pseudo-code of the proposed *SSGGAFP* approach is provided in Algorithm 17. The other salient features of our *SSGGAFP* approach are provided in subsequent subsections.

5.4.1 Solution encoding

In *RUASP*, a solution is presented as a set of *m* rescue units, where each unit contains the sequence of incidents processed by that unit. In other words, if k^{th} unit contains $|R_k|$ incidents, then the unit R_k contains a linear permutation of $|R_k|$ incidents. The positions in a linear permutation define the order of processing of the incident by the rescue unit. Figure 5.3 shows the solution representation, by assuming an example with 11 incidents (n=11) and 4 rescue units (m=4). The first incident processed by rescue unit 1 is 5 and the second incident is 2 and the last incident is 6. Similarly, other rescue units process the incidents in the same sequence as demonstrated in Figure 5.3.

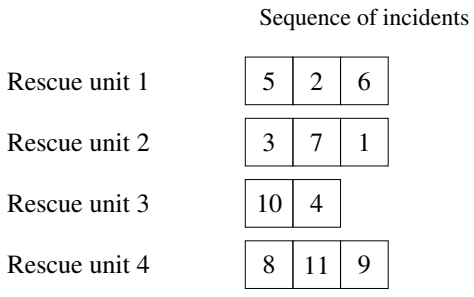


Figure 5.3: Solution representation illustration

5.4.2 Initial population generation

The initial population is generated by using four heuristics. The first heuristic is taken from [113], whereas the others are devised by us considering the characteristics of the problem. A detailed description of these heuristics is presented below.

1. **Schedule7 heuristic:** In [113], the authors presented several constructive heuristics and one of those performed very well for generating better quality solutions for *RUASP*. The authors named this heuristic as **SCHED7** in [113]. The heuristic begins with an empty solution of *m* rescue units. The process iterates for *n* times and in each iteration, one incident is selected and allocated to that rescue unit which incurs the least value of the ratio of completion time to the severity level. Three variables namely, CT_k denotes the current completion time of unit k, θ_k to store the last incident processed by unit k and σ_k represents the sequence of incidents processed by unit k, are used in this heuristic. Algorithm 14 presents the pseudo-code of Schedule7 heuristic.

2. **Greedy heuristic:** Our greedy heuristic is inspired from the greedy heuristic presented in [113]. First, it sorts the incidents in non-increasing order as per their severity levels. The incident whose severity level is highest should be processed first, this is the idea behind sorting. It begins with an empty solution and creates a complete solution iteratively. In each iteration, it selects one incident from the sorted sequence and assigned it to the rescue unit which has the least value of completion time. It differs from the greedy heuristic of [113] only in the allocation of the rescue unit. The greedy heuristic of [113] selects the unit with lowest start time for the incident *i*. The start time of the incident *i* is calculated by adding the previous completion time of unit (k) with the travel (setup) time taken from last incident (θ_k) to the incident *i*. On the other hand, our greedy heuristic selects the unit with lowest completion time for the incident *i* by considering the processing time also along with travel time (i.e., the current completion time of the rescue unit). The allocation of rescue unit to an incident i based on lowest start time may result in an assignment of that rescue unit whose processing time is more and thus results in later completion time, which will eventually affect the completion time of further incidents processed by this rescue unit. Thus, it is important to consider the completion time of the rescue unit rather

than start time. Algorithm 15 presents the pseudo-code of the greedy heuristic.

Algorithm 14: Pseudo-code of Schedule7 heuristic [113]

// The completion time (CT_k) and the sequence of incidents allocated of k^{th} unit initially set as zero, i.e.,
$CT_k \leftarrow 0, \theta_k \leftarrow 0$ and $\sigma_k \leftarrow \phi \; \forall \; k \; \in \; K$ such that $K = \{1, 2, \ldots, m\}$;
// I is initialized as the set of incidents
$I \leftarrow \{1, 2, \ldots, n\}$;
$unassigned\_incd \leftarrow$ n;
$U_{(Cap,i)} \leftarrow \{\text{k} \in K \mid Cap_{ki} = 1\} \; \forall \; i \; \in \; I$;
// $U_{(Cap,i)}$ represents the set of those rescue units which is capable of processing the incident i.

while *($unassigned\_incd \; > \; 0$)* **do**
1. Select incident $i^* \in I$ and unit $k^* \in U_{(Cap,i)}$ which has minimal ratio (ρ) of completion time to the severity level.
$i^*, k^* \leftarrow \underset{i \in I, k \in U_{(Cap,i)}}{\arg\min} \left(\frac{CT_k + S^k_{\theta_k, i} + p^k_i}{w_i} \right)$;
2. Update: $I \leftarrow I \setminus \{i^*\}, \quad CT_{k^*} \leftarrow CT_{k^*} + S^{k^*}_{\theta_{k^*}, i^*} + p^{k^*}_{i^*}$,
$\theta_{k^*} \leftarrow i^*, \quad \sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{i^*\}$
3. $unassigned\_incd \leftarrow (unassigned\_incd \; - \; 1)$;

return $\sigma \leftarrow (\sigma_1, \sigma_2, \ldots, \sigma_m)$;

Algorithm 15: Pseudo-code of Greedy heuristic

// Sorts incidents in non-increasing order of severity. Let the sequence be I.
$I \leftarrow \{i_{\iota_1}, i_{\iota_2}, \ldots, i_{\iota_n}\}$ such that $w_{\iota_1} \geq w_{\iota_2} \geq \cdots \geq w_{\iota_n}$;
// The completion time (CT_k) and the sequence of incidents allocated of k^{th} unit initially set as zero, i.e.,
$CT_k \leftarrow 0, \theta_k \leftarrow 0$ and $\sigma_k \leftarrow \phi \; \forall \; k \; \in \; K$ such that $K = \{1, 2, \ldots, m\}$;

for *($l \; = \; 1 \; : \; n$)* **do**
1. Choose incident $i \leftarrow l$ where $i \in I$;
2. $U_{Cap} \leftarrow \{\text{k} \in K \mid Cap_{ki} = 1\}$;
// U_{Cap} represents the set of those rescue units which is capable of processing the incident i.
3. $k^* \leftarrow \underset{k \in U_{Cap}}{\arg\min} \; (CT_k + S^k_{\theta_k, i} + p^k_i)$;
// k^* is the unit which incurs least value of completion time for i.
4. Update: $I \leftarrow I \setminus \{i\}, \quad CT_{k^*} \leftarrow CT_{k^*} + S^{k^*}_{\theta_{k^*}, i} + p^{k^*}_i$,
$\theta_{k^*} \leftarrow i, \quad \sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{i\}$

return $\sigma \leftarrow (\sigma_1, \sigma_2, \ldots, \sigma_m)$;

3. **Greedy_Rand heuristic:** It differs from the greedy heuristic only in one aspect. Here, the selection of incident for assignment in partial solution is done by selecting a random position in between [0,w_g], from the current position in the sorted sequence of incidents. For example, assume there are seven incidents sorted in descending order as per the severity level, viz. $\{i_5, i_2, i_4, i_1, i_7, i_3, i_6\}$ and let the value of w_g is set as 4. In beginning, the current position will be the first position. A random number $\in [0, 4]$ is chosen. Assume

the random number is 2. The customer after two positions from the current position which is i_4 is selected and assigned to the rescue unit which has the least value of completion time. After assignment, the sequence is updated by shifting all remaining incidents from the current position to the selected position and inserting the selected incident at the current position. The current position will now be shifted to the next position. For example, after selection of incident i_4, the updated sequence will be $\{i_4, i_5, i_2, i_1, i_7, i_3, i_6\}$ and the current position will be at incident i_5.

4. **Random_Sol heuristic:** It begins with an empty solution of *m* rescue units. A complete solution is built by using *n* iterations. In each iteration, a random incident is selected. For the selected incident a pool of rescue units that can process the selected incident is determined. Then, a random rescue unit is selected from this pool and the incident is assigned to this unit. This heuristic produces a purely random solution.

   A population of size *p* is generated by using the initial population generation method. The generated solutions can be classified into two categories. This classification is done based on the quality of the solutions generated by the particular heuristic. Schedule7 heuristic and Greedy heuristic generate solutions by following a purely greedy approach. On the other hand, Greedy_Rand heuristic follows a greedy approach with some randomness. Hence, they generate superior quality initial solutions. The better quality initial solutions may result in faster convergence of the approach to optimal/near optimal solutions. This is the idea behind using better quality initial solutions in the population. The first p_h solutions are generated by these three heuristics. Schedule7 and Greedy heuristic being purely greedy generates only a single solution. Thus, the first two solutions are generated by these two heuristics and the remaining $(p_h - 2)$ solutions are generated by Greedy_Rand heuristic. To generate the remaining solutions, we have used Random_Sol heuristic, which generates purely random solutions. The sole purpose of using Random_Sol heuristic is to add diversity to the population. Uniqueness is ensured for each generated solution, before adding it into the population. In small size instances, we have observed that Greedy_Rand heuristic generates sometimes non-unique solutions also. So whenever Greedy_Rand heuristic generates a non-unique solution, the Random_Sol heuristic is used to generate a new solution. Thus, the first p_h solutions may contain some purely random solutions also. The use of Random_Sol heuristic in such a scenario provides two advantages. First, the solution generated by this heuristic adds diversity to the population. Second, the

Greedy_Rand heuristic may take a large number of attempts to generate p_h*-2* unique solutions, thus it is advantageous to generate some random solutions in lesser amount of time. Algorithm 16 presents the pseudo-code for initial population generation and shows how these heuristics are utilized for generating initial population.

Algorithm 16: Pseudo-code of Initial population generation

```
Input: p (population size), p_h (solutions generated by greedy heuristic)
Output: BEST_sol
   // BEST_sol is the best solution in population.
// u_01 is a uniform variate between [0,1).
p_i ← 1;
Sol_{p_i} ← Schedule7 ();
   // Sol_{p_i} is the solution generated by Scedule7 Heuristic.
p_i ← p_i + 1;
Sol_{p_i} ← Greedy Heuristic();
   // Sol_{p_i} is the solution generated by Greedy Heuristic.
if (Sol_{p_i} is unique) then
    p_i ← p_i + 1;
while (p_i < p) do
    if (p_i < p_h) then
        Sol_{p_i} ← Greedy_Rand Heuristic();
        if (Sol_{p_i} is unique) then
            p_i ← p_i + 1;
        else
            Sol_{p_i} ← Random_Sol Heuristic();
            if (Sol_{p_i} is unique) then
                p_i ← p_i + 1;
    else
        Sol_{p_i} ← Random_Sol Heuristic();
        if (Sol_{p_i} is unique) then
            p_i ← p_i + 1;
return (BEST_sol);
```

5.4.3 Crossover operator

The crossover operator used in our proposed *SSGGAFP* approach is devised by considering the characteristics of the problem as well as the objective. Our crossover operator is inspired by the crossover operators presented in [42, 142]. It takes two parents as input and produces offspring in two stages.

The first stage begins with an empty solution and the offspring is built by following an iterative approach. During each iteration, the first parent is selected with probability p_u and with remaining probability the second parent is selected. The most efficient rescue unit in the selected parent is found. The objective is to minimize the total weighted completion time. If the rescue

unit can process more incidents with less completion time, then selection of such rescue units may provide a solution having minimum total weighted completion time. Hence, the rescue unit having the minimum ratio of the total weighted completion time per incident in the unit is defined as the most efficient rescue unit. The probability p_u is calculated as $\frac{f(p_2)}{f(p_1)+f(p_2)}$, where $f(p_1)$ and $f(p_2)$ are the fitnesses of the first and second parents, respectively. Such a scheme will provide more opportunities to better one among them. The most efficient rescue unit is copied in offspring. This rescue unit is discounted from both the parents and is not considered for the next iteration. It is also possible that the incidents present in this unit may belong to different units in another parent. Hence, the incidents present in the selected rescue unit are removed from their respective units in the other parent one-by-one. After removing an incident, all the subsequent incidents on the affected rescue unit in another parent are shifted one place backwards. The entire process is repeated for *m* times so that a partial solution having *m* rescue units is obtained. It is pertinent to mention that in a solution not all the rescue units need to be processing some incidents. It is also possible that only some of the rescue units can process all incidents by incurring a less objective cost.

Clearly, at the end of the first stage, some incidents may be left unassigned and should be assigned to some rescue units of the partial solution generated by the first stage. The second stage is same as Greedy_Rand heuristic except for two differences. First, we have used parameter w_c in place of parameter w_g. Furthermore, instead of considering the least value of completion time for the assignment of an incident, we have considered here the least value of weighted completion time. Except for these two differences, the second stage follows the same steps as Greedy_Rand heuristic. In Greedy_Rand heuristic, the complete solution was built from empty solution and already the procedure starts with the sorted sequence of incidents as per their severity level. Thus, considering weighted completion time in place of completion time will not produce any major impact. But, in the second stage of crossover the scenario is quite different. Here, we already have a partial solution generated by the first stage, so the sorted sequence of remaining unassigned incidents need not have an equal interval of severity level, since some incidents are already assigned in the first stage. Thus in the second stage, it is better to consider the weighted completion time of the incident for assignment at some position in partial solution obtained by the first stage of the crossover.

Algorithm 17: *SSGGA* framework for *RUASP*

Input: p (population size), RUASP instance, n (number of incidents), m (number of rescue units)
Output: Best solution found so far
// u_{01} represents a random value between [0,1).
$BEST_{sol} \leftarrow$ Initial population generation(p); // $BEST_{sol}$ is the best solution in population.
Itr $\leftarrow$ 0; // Itr represents generations
while $(Itr < ITR_{max})$ **do**
  Par1 $\leftarrow BTS(p)$; // Select parents using binary tournament selection
  Par2 $\leftarrow BTS(p)$;
  offspring $\leftarrow Crossover$(Par1,Par2);
  if $(u_{01} > 0.75)$ **then**
    mutant $\leftarrow$ MUT1(offspring);
  else
    mutant $\leftarrow$ MUT2(offspring);
  Sol $\leftarrow Better_{obj}$(offspring,mutant);
  // Sol is the solution among offspring and mutant whose objective value is better
  if *(Sol is unique) and (better than worst solution)* **then**
    // worst solution is the solution with poorest objective value in the population
    Replace worst solution with Sol;
  if *(Sol is better than* $BEST_{sol}$*)* **then**
    $BEST_{sol} \leftarrow$ Sol;
  $Itr \leftarrow Itr + 1$;
return $(BEST_{sol})$;

5.4.4 Mutation operator

The mutation operator proposed here follows the delete and re-insert strategy and consists of two stages. The first stage selects some incidents and deletes them from the solution. The complete solution is again recreated by following a greedy approach during the second stage. Such a strategy provides a balance between greediness and randomness. In literature, this kind of strategy is successfully used in various optimization problems, especially related to grouping and scheduling domains, e.g. [83, 88].

Two versions of the mutation operator are used in a mutually exclusive manner. The first version is used with probability p_m and it randomly selects one rescue unit. Some incidents $\in$ [1,$n_{sel} - 1$] in this unit are selected randomly and kept in a pool of unassigned incidents. The selected incident is removed from the solution and all the subsequent incidents assigned after the selected incident are shifted one place backwards. Here, n_{sel} is the number of incidents in the selected rescue unit. The second version is used with the remaining probability and it removes n_θ incidents in an iterative manner. In each iteration, one rescue unit is randomly selected and a random incident is selected and added in the list of unassigned incidents. The selected incident

is deleted from the solution in the same manner as mentioned above. The unassigned incidents are inserted in partial solution by the following the same approach as discussed in the second stage of the crossover operator.

5.5 Computational results

In this section, we discussed the computational results of our proposed approach and its comparison with the state-of-the-art approach of [114]. The proposed approach is also compared with the approaches presented in [113]. The proposed approach is coded in C language and all experimental executions were conducted on a Linux based PC with Intel Core i5-7500 CPU, running at 3.40 GHz with 16 GB of RAM. Our *SSGGAFP* approach is executed for $2500 \times n$ generations, where *n* is the number of incidents. The overall size of the population (*p*) is 250, out of which 150 solutions (p_h) are generated by greedy based heuristic and remaining by using random solution based heuristic. The probability p_{bts} in binary tournament selection is assigned with a value 0.7. The first mutation operator is used with probability p_m, which is set as 0.75 and with remaining probability i.e., 0.25, the second mutation is employed. The parameter n_θ used in mutation is assigned a value that is based on the size of the instance. Thus n_θ is set as 4 for all instances with $n \geq 30$, to 3 for instances with $n = 20$ and set as 2 for instances with $n = 10$. The parameters w_g and w_c used in Greedy_rand heuristic and in crossover operator are defined as 4 and 2, respectively. The *SSGGAFP* approach is executed for 10 independent runs with different seed values on each instance like the BRKGA approach of [114]. All these parameter values are chosen based on empirical observations spanning over a large number of trials.

5.5.1 Description of *RUASP* instances

The *RUASP* instances include four sizes $(10, 20, 30, 40)$ of incidents (*n*) and rescue units (*m*). Due to the constraint $m \leq n$, the instances with *m*=10 the value of *n* is $\{10, 20, 30, 40\}$, with *m*=20 the value of *n* is $\{20, 30, 40\}$, with *m*=30 the value of *n* is $\{30, 40\}$ and instances with *m*=40 the only permissible value of *n* is 40. Thus, we have total 10 combinations of instances corresponding to each pair (*m*,*n*). For each combination, 10 different instances were generated, thus total 100 instances generated for *RUASP*. These instances were originally proposed by Wex et al. in [113] and the processing time in these instances contains crisp value. In [114], Cunha et al. proposed a version of *RUASP* with fuzzy processing times. The fuzzy processing times

are created by using the crisp processing times in the instances of [113]. This crisp values of processing times are used as the component *b* of *TFN* in fuzzy processing times, while the other components of *TFN* are computed as : $a \sim U(0.6, 0.9) \times b$ and $c \sim U(1.1, 1.4) \times b$ [114].

The authors of [113] chosen only 40 rescue units and incidents as the upper limit based on feedback that they got from the German Federal Agency for Technical Relief (*THW*) that in case of natural disasters, number of incidents rarely exceeds 40 and each rescue unit consists of several sub units.

The effectiveness of the proposed approach is evaluated by using the same set of instances as used in [113, 114]. To address the version of *RUASP* with fuzzy processing times the modified instances presented in [114] are used by our approach. We have got all these instances as well as the results of the approaches of [113] from first author of [114] through personal communication. Although, the number of instances mentioned in both papers [113, 114] is 100. But the instances received from Cunha et al. [114] contains only 98 instances for *RUASP* with crisp processing times and a total of 86 instances for the version of *RUASP* with fuzzy processing times. We have compared the results of our implemented Schedule7 heuristic with those reported in [114]. The results are matching, which confirms that Cunha et al. have used only 86 instances for their approach in [114].

5.5.2 Experimental results

In this section, we have compared the performance of the proposed approach with other existing approaches for *RUASP* in the literature. We have used the same criteria to measure the performance as used in [114]. Table 5.2 presents the comparison of *SSGGAFP* with *BRKGA* of [114]. In this table, the first column represents the number of units (m) and the second column corresponds to the number of incidents (n). The instances are grouped by size. For example, first row reports the results for $m = 10$ with $n = 10$. As stated in Section 5.5.1, we have a total of 10 combinations of instances corresponding to each pair (*m*,*n*). The third column presents the average value of solutions obtained by the Schedule7 heuristic over ten instances for each pair. The execution time for Schedule7 heuristic is not reported as being a heuristic it is executed only once on an instance and provides the solution within no time. The last six columns provide the results for *BRKGA* and the *SSGGAFP* approaches. The results of each approach for *RUASP* is shown by three columns in order as, the average value of solutions, execution time in seconds and the percentage deviation of the solution (%Dev). The Schedule7 heuristic outperformed all other heuristics presented in [113]. Thus, it is considered as the reference

Table 5.2: Comparison of average results obtained by BRKGA and SSGGAFP over each set of instances

		Schedule7	BRKGA			SSGGAFP		
Units(m)	**Incidents(n)**	**Solution**	**Solution**	**Time** [a]	**%Dev**	**Solution**	**Time**[a]	**%Dev**
10	10	359.68	345.86	44.94	-3.84%	345.86	0.06	-3.84%
10	20	891.75	854.96	64.20	-4.13%	**851.86**	0.18	**-4.47%**
10	30	1570.81	1523.03	83.64	-3.04%	**1494.46**	0.37	**-4.86%**
10	40	2447.86	2421.46	114.60	-1.08%	**2330.41**	0.62	**-4.80%**
20	20	453.67	438.88	107.03	-3.26%	**435.98**	0.25	**-3.90%**
20	30	864.34	844.18	146.62	-2.33%	**817.85**	0.48	**-5.38%**
20	40	1121.98	1128.79	218.88	0.61%	**1073.29**	0.75	**-4.34%**
30	30	579.00	564.91	211.21	-2.43%	**555.45**	0.65	**-4.07%**
30	40	869.66	863.42	285.30	-0.72%	**831.19**	0.97	**-4.42%**
40	40	687.68	679.63	353.80	-1.17%	**656.80**	1.19	**-4.49%**

[a]Time in Seconds

to calculate the percentage deviation of the solution found by an approach from the solution obtained by Schedule7 heuristic. It is computed as, $\%Dev = (\frac{S_A}{S_{sch7}} - 1) \times 100$, where S_A and S_{sch7} are the solutions obtained by the approach A and the Schedule7 heuristic, respectively. We have already mentioned in Section 5.5.1 that, we have received only 86 instances out of 100 instances. Thus, the average value of solutions is computed by considering the number of instances in the set available to us.

In this table, the results of *SSGGAFP* are reported in bold font wherever it performed better than the *BRKGA* approach. The results in Table 5.2 demonstrates the superiority of our proposed approach over state-of-the-art approaches, viz. *BRKGA* and the Schedule7 heuristic. The performance gap between *SSGGAFP* and *BRKGA* widens with the increase in the number of incidents (n) for the same value of the number of rescue units (m), which shows that our approach performs better with the increase in toughness of the instance. The comparison of execution times demonstrates that the *SSGGAFP* approach is very fast in comparison to the *BRKGA* approach. To be specific, the maximum execution time taken by *BRKGA* approach is 353.80 seconds, while *SSGGAFP* takes only 1.19 seconds for the same set of instances. As far as the comparison of execution times are concerned, *BRKGA* were executed on a PC equipped with 3.10 GHz Intel Core i5-4440 and 16 GB RAM which is different from the system used to execute *SSGGAFP*, hence it is not possible to compare the execution times precisely. However, a rough comparison can always be made based on information available in the public domain about the relative speeds of two processors, which indicates our system to be around 1.2 times faster. Even after compensating for this difference in processing speeds, we can safely say that

Table 5.3: Comparison of BRKGA and SSGGAFP on the first instance from each set

			BRKGA					SSGGAFP				
Units	**Incidents**	**Schedule7**	Min_{Sol}	Avg_{Sol}	Max_{Sol}	$MaxMin_{Dev}$	**Time** [a]	Min_{Sol}	Avg_{Sol}	Max_{Sol}	$MaxMin_{Dev}$	**Time** [a]
10	10	282.09	267.44	267.44	267.44	0.00%	44.94	267.44	267.44	267.44	0.00%	0.06
10	20	860.10	807.00	807.49	809.44	0.30%	64.20	807.00	**807.00**	**807.00**	**0.00%**	0.17
10	30	2123.89	2009.28	2022.24	2046.49	1.85%	83.64	**2000.19**	**2000.19**	**2000.19**	**0.00%**	0.34
10	40	2597.37	2502.83	2544.17	2604.37	4.06%	114.60	**2454.94**	**2454.94**	**2454.94**	**0.00%**	0.70
20	20	494.56	463.58	464.45	469.43	1.26%	107.03	463.58	**463.58**	**463.58**	**0.00%**	0.25
20	30	658.63	642.23	675.20	712.41	10.93%	146.62	**635.98**	**635.98**	**635.98**	**0.00%**	0.49
20	40	1117.30	1072.56	1106.72	1191.10	11.05%	218.88	**1056.16**	**1056.16**	**1056.16**	**0.00%**	0.78
30	30	487.58	475.71	482.46	498.24	4.74%	211.21	**474.43**	**474.43**	**474.43**	**0.00%**	0.61
30	40	819.73	786.20	809.92	842.80	7.20%	285.30	**778.11**	**778.11**	**778.11**	**0.00%**	1.04
40	40	702.13	664.55	673.55	682.96	2.77%	353.80	664.55	**664.55**	**664.55**	**0.00%**	1.22

[a]Time in Seconds

the proposed *SSGGAFP* approach is much faster than *BRKGA*. Thus, the overall comparison reveals that the approach presented by us outperforms the existing approaches for *RUASP*.

Any approach is considered to be robust if the results provided by it are consistent in multiple independent runs performed using different random seeds [143]. The robustness analysis of *BRKGA* is performed in [114]. Thus, we have also performed the same analysis for our proposed *SSGGAFP* approach. In this analysis, the first instance from each set of combinations (*m*,*n*) is used. Each of the approaches is executed 10 independent times with different seed values, which generates 10 solutions for each instance. Table 5.3 provides the outcomes of the robustness analysis performed on *BRKGA* as well as on *SSGGAFP*. In this table, the first two columns correspond to the number of units (m) and the number of incidents (n), respectively. The third column shows the solution obtained by Schedule7 heuristic. The last ten columns represent the results of both approaches. Each approach results are reported using five columns in the order: Min_{Sol}, Avg_{Sol}, Max_{Sol}, $MaxMin_{Dev}$ and Time. Min_{Sol} and Max_{Sol} corresponds to the minimum and the maximum solution obtained, while Avg_{Sol} is the average quality of solutions in ten runs. $MaxMin_{Dev}$ is the deviation between maximum and minimum solutions and it is computed as, $MaxMin_{Dev} = (\frac{Max_{Sol}}{Min_{Sol}} - 1) \times 100$. Time reports the average execution time of an approach in ten runs. The maximum deviation for *BRKGA* is found to be 11.05% for the instances with $m = 20$ and $n = 40$, while *SSGGAFP* consistently provides zero deviation for all the instances considered in this analysis. In fact, the *SSGGAFP* approach generates the same solution in all ten runs for all the instances of *RUASP*. This proves that our proposed approach is more robust than the *BRKGA* approach.

The first author of [114] has also provided the instances and solutions of the version of *RUASP* presented in [113]. To compare our proposed approach with the approaches presented

in [113], the *SSGGAFP* is modified as *SSGGACP*, to make it suitable for considering crisp processing times. The *SSGGACP* addresses the original version of *RUASP*, where the processing times contain the crisp value. In [113], the authors have proposed eight construction heuristics and five improvement heuristics. The Schedule7 heuristic used in this work is one of the construction heuristics proposed in [113]. Each construction heuristic is combined with an improvement heuristic and thus, many compositions of heuristics were presented in [113]. They have also proposed *GRASP* metaheuristic to address *RUASP*. *GRASP* is a well-known multi-start metaheuristic in which each generation consists of two phases. The first one is the construction phase, while the other is the local search. The *GRASP* metaheuristic proposed in [113] has used the construction heuristics in the first phase and improvement heuristics used in the local search. Two versions are proposed in [113], namely, the classical version which is the composition of heuristics only and the other is *GRASP* which is used along with the composition of heuristics. All these approaches were executed once on each instance, and, hence, we have also executed *SSGGACP* only once on each instance. The analysis of the results in [113] concludes that the Schedule7 heuristic provided better quality solutions on all the instances. While different versions produced the best solution for different instances. Table 5.4 presents the comparison of *SSGGACP* with the approaches presented in [113]. For each instance, the best solution and the corresponding execution time are selected by comparing all the solutions generated by the approaches of [113]. This best solution over all the approaches is referred to as $Best_{Wex}$ in the table. The solutions generated by our proposed *SSGGACP* approach is compared with the $Best_{Wex}$. Both the results, viz. $Best_{Wex}$ and *SSGGACP* are reported by using three columns. The organization of this table is done in the same manner as Table 5.2. The solutions generated by Schedule7 heuristic are considered as the reference to calculate the percentage deviation as shown in Table 5.2. In this table, the solutions obtained by *SSGGACP* are reported in bold font. These results clearly highlight the superiority of our proposed approach over all of the composition of heuristics approaches as well as *GRASP* metaheuristic presented in [113]. Our approach is better in terms of both solution quality as well as in terms of running time. In [113], they have mentioned that the average runtime of *GRASP* is 187.63 seconds across all instances while the maximum runtime is 25.89 minutes. Our *SSGGACP* approach being a metaheuristic is very fast in comparison with *GRASP* metaheuristic. Even the maximum execution time of classical version (which is composition of only heuristics) is 20 seconds on the instance of largest size ($m = 40$ and $n = 40$) [113]. While, the maximum execution time by our approach is 1.14 seconds on the same set of largest size instances i.e., with $m = 40$ and $n = 40$. Thus,

Table 5.4: Comparison of average results obtained by $Best_{Wex}$ and *SSGGACP* over each set of instances

		Schedule7	$Best_{Wex}$			***SSGGACP***		
Units(m)	**Incidents(n)**	**Solution**	**Solution**	**Time** [a]	**%Dev**	**Solution**	**Time**[a]	**%Dev**
10	10	359.05	350.60	0.21	-2.35%	**347.48**	0.06	**-3.22%**
10	20	901.78	858.39	64.30	-4.81%	**852.57**	0.16	**-5.46%**
10	30	1579.92	1530.17	35.29	-3.15%	**1508.32**	0.31	**-4.53%**
10	40	2375.07	2279.91	238.03	-4.01%	**2241.28**	0.50	**-5.63%**
20	20	469.67	461.97	214.61	-1.64%	**452.23**	0.23	**-3.71%**
20	30	858.78	836.70	212.31	-2.57%	**821.01**	0.43	**-4.40%**
20	40	1108.01	1085.51	157.67	-2.03%	**1065.61**	0.66	**-3.83%**
30	30	578.32	566.47	12.55	-2.05%	**556.52**	0.60	**-3.77%**
30	40	891.82	866.80	16.45	-2.80%	**852.65**	0.87	**-4.39%**
40	40	689.78	676.71	52.07	-1.89%	**659.03**	1.10	**-4.46%**

[a]Time in Seconds

our proposed approach is faster than all the approaches of [113]. In terms of solution quality, the same pattern can be observed as explained in Table 5.2, i.e., the performance gap between *SSGGACP* and the $Best_{Wex}$ (best solution over all the approaches of [113]) widens with the increase in toughness of the instance.

To visually demonstrate the comparison of solutions, we have used the box plot visualization method. Box plot is a visualization technique used to summarize and compare different sets of data. Box plot visualization reveals more quantitative information than the tabular representation of the data. It uses five points to summarize the data, namely, minimum, first quartile, median, third quartile and maximum. Even without understanding the complete details of the box plot, one can easily draw some important conclusions about the data sets [144]. As already mentioned in Section 5.5.1, there are total 10 sets of combination of *m* and *n* and each set contains 10 instances. Each box plot represents the solutions for one entire set corresponding to the particular combination of *m* and *n*, obtained by the specific approach.

Figures 5.4 and 5.5 presents the comparison of solutions obtained by Schedule7 heuristic and *SSGGACP* for the specific set of instances. We have chosen Schedule7 heuristic for comparison as it is performed better on most instances in comparison to all other approaches (including *GRASP* metaheuristic) presented in [113]. In all box plots, the three levels (minimum, maximum and the median) of *SSGGACP* are always lower than the corresponding levels of Schedule7 heuristic. Also, the boxes of *SSGGACP* shift more downwards with the increase of toughness of instances. Thus, the box plot visualization demonstrates that the *SSGGACP* approach generates superior quality solutions than the Schedule7 heuristic. Similar box plots can not be given for

comparison of *SSGGAFP* with *BRKGA*, because of the unavailability of the instance-by-instance results of *BRKGA*. Likewise, we have not provided the comparison of robustness of *SSGGACP* with the approaches of [113], because of the fact that approaches of [113] were executed only once on each instances, and to make the comparison fair, *SSGGACP* was also executed only once. However, just to check the robustness of *SSGGACP*, we have executed it 10 times on each instance, and found *SSGGACP* converges to same value in each of the 10 runs on all the instances. This clearly demonstrates the robustness of *SSGGACP*.

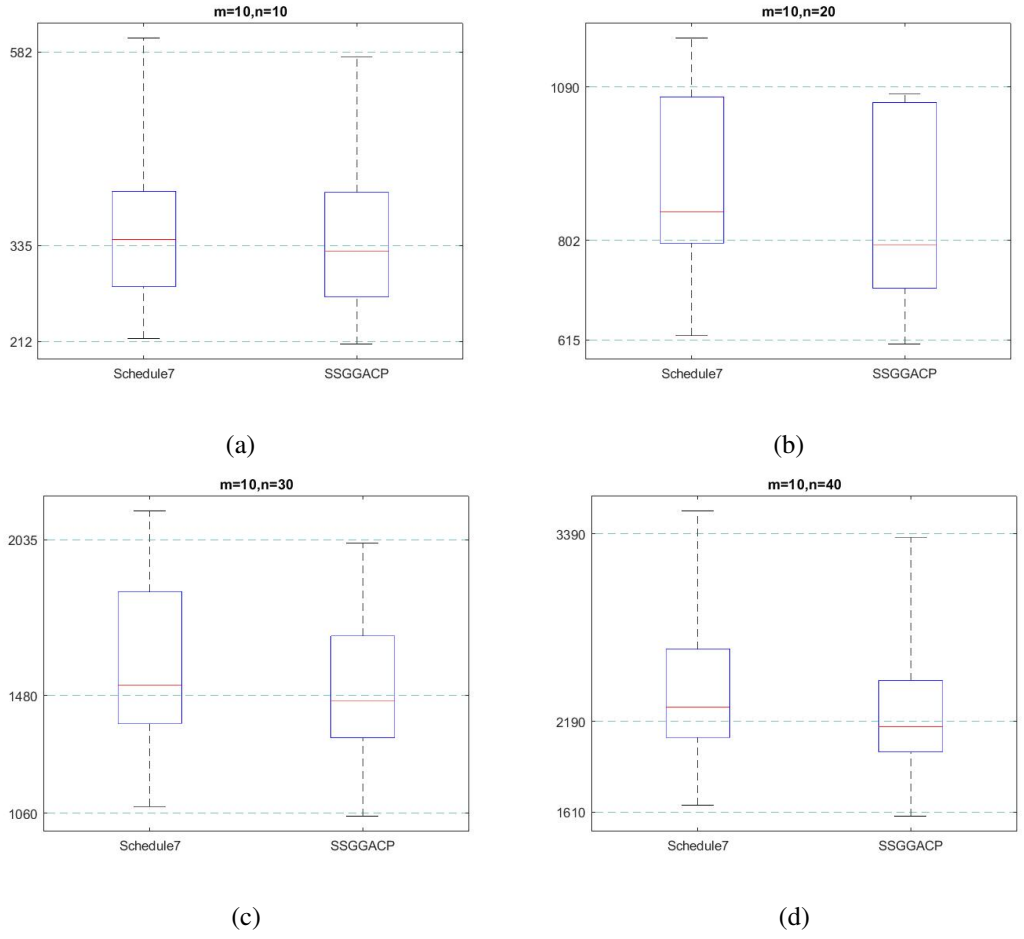


Figure 5.4: Boxplots of the solutions obtained by Schedule7 and *SSGGACP* over the set of ten instances: (a) m=10 & n=10, (b) m=10 & n=20, (c) m=10 & n=30, (d) m=10 & n=40.

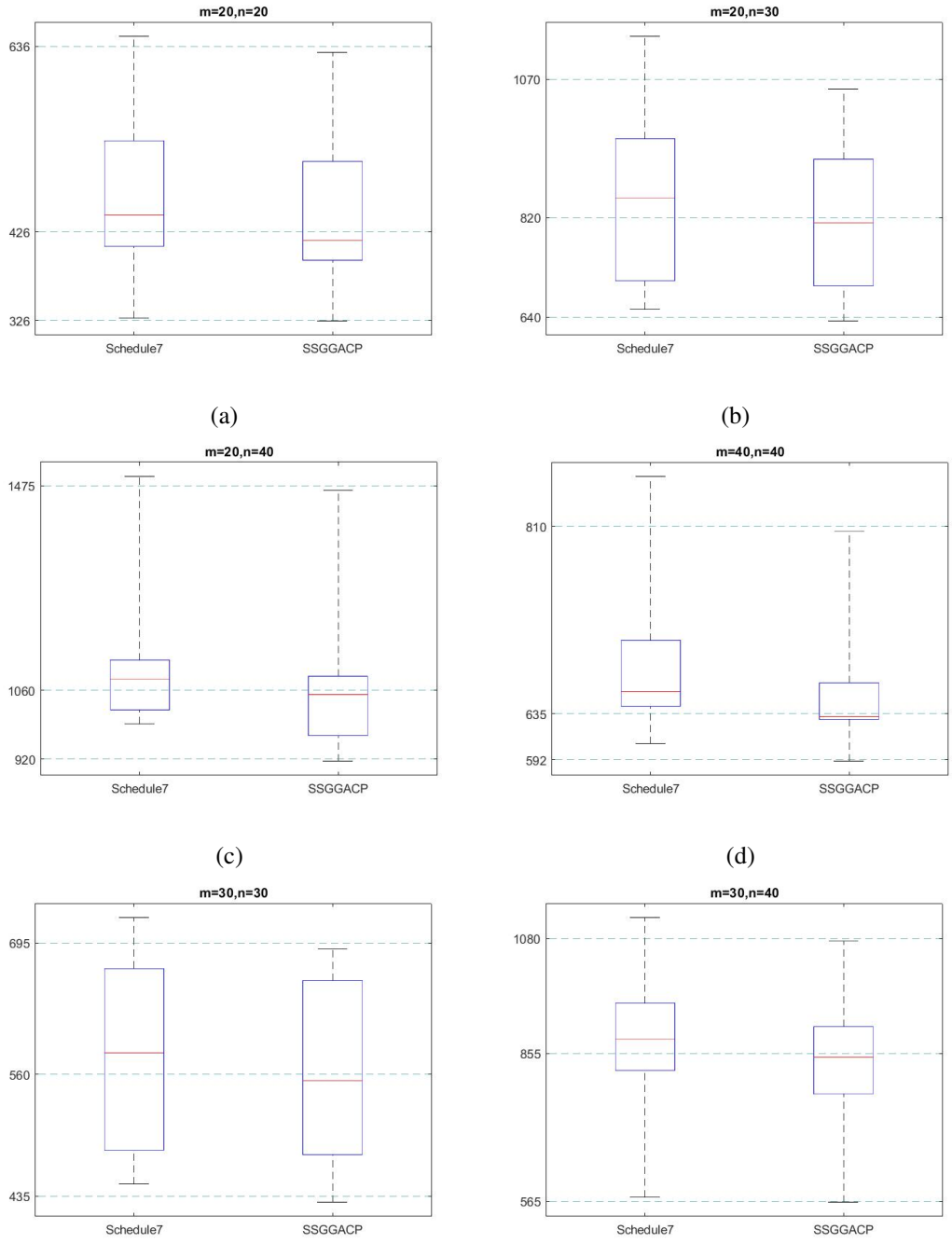


Figure 5.5: Boxplots of the solutions obtained by Schedule7 and *SSGGACP* over the set of ten instances: (a) m=20 & n=20, (b) m=20 & n=30, (c) m=20 & n=40, (d) m=40 & n=40 (e) m=30 & n=30, (f) m=30 & n=40s.

5.6 Conclusions

In this chapter, we presented a steady-state grouping genetic algorithm-based approach for *RUASP*. Our approach uses the crossover and mutation operator that are designed by considering the characteristics of the problem and the objective. We have used a set of greedy and random based heuristics for initial solution generation which provides a superior quality diverse initial solutions. We have compared the solutions generated by proposed approach with the existing state-of-the-art approaches for the addressed problem available in the literature. The computational results show the superiority of the proposed approach both in terms of solution quality and execution time.

Chapter 6

Quality of Service Vehicle Routing Problem with Time Windows

6.1 Introduction

Over the last several decades, vehicle routing problem (VRP) and its variants have attained massive popularity owing to their ability to model a wide range of real-world applications pertaining to various domains. Their applications include transportation planning, supply chain management in logistics networks, production management and so on [145, 146]. The objective of VRP is to design an optimal set of delivery routes for a fleet of vehicles in order to serve a given set of customers of a company's supply chain. It represents the essence of assignment and routing of vehicles with minimum cost in a transportation supply management. Hence, it is a crucial problem in logistics management and also one of the most widely studied problems in the domain of combinatorial optimization. VRP is an $\mathcal{NP}$-hard problem as it generalizes traveling salesman problem which is $\mathcal{NP}$-hard [147, 148]. Since its inception in 1959 [149], many variants of VRP have been proposed in the literature such as [150] and [151], due to the incorporation of various constraints arising in real-world applications. A detailed survey of VRP and its variants can be found in [152]. Some recent works on VRP can be found in [153], [154], [155], and [156].

Most of the VRPs in literature consider minimization of total operational cost as the objective. Usually, the operational cost is the total distance traveled or time [157]. In some route planning problems, the minimization of number of vehicles is also considered as an objective along with operational cost [158]. The objective function can also be designed with reference to customer

satisfaction, and some VRP models contain the cost as service penalties when a customer receives delayed service or an incomplete or a damaged delivery [146].

In the last two decades, supply chain management has gained massive attention owing to their impact on cost effectiveness, quality of customer service and overall competitiveness in global market [159]. Due to the rise in global competition, managements dealing with supply chains have now increased focus on cost reduction strategies and planning for new approaches aimed at increasing the level of customer satisfaction. The cost reduction strategies aid in fetching monetary gains for companies, while approaches for customer satisfaction have emphasis on customer needs, so that more and more customers can be attracted. In transportation service, both aforementioned objectives are crucial, and thus, it is desired to address the both simultaneously. In this regard, it is necessary to focus on the planning of supply and distribution routes along with the proper operational management activities related to supply chain, so that it results in reduction in total costs as well as an improvement in the quality of service (QoS) [160].

With advancement in technology, globalization of markets and increase in demands, any organization, whether small or large, have put on more efforts on implementing customer-focused service strategies. Indeed, quality assurance has become one of the top most priority of management and an important factor for sustainability and growth in business [161]. Customer service with respect to transport industries are mainly focused at timely delivery of goods [162]. In transport logistics, quality is considered as a key element defined in terms of availability, reliability and on time delivery [163, 164].

The stakeholders in transportation focus on timely delivery of goods in unharmed conditions. The importance of these conditions increase manifolds in case of perishable goods which require urgent delivery [165]. Perishable goods include food items such as milk, meat, fish, fruits and vegetables, health related products such as drugs, organs or analytic samples or even a newspaper, value of whose news decreases over time. In most of the applications pertaining to transportation of perishable goods, the urgent delivery of goods is the top most priority, and delay in delivery can result in deterioration in quality of the delivered goods. Thus, importance of urgency in perishable goods related transport is crucial for any business organization dealing with transportation of such goods.

This chapter discusses one of the most commonly addressed variant of VRP known as vehicle routing problem with time windows (VRPTW). VRPTW is a generalization of VRP where a customer can only be served with in its own time window. Thus, it is also $\mathcal{NP}$-hard.

VRPTW is one of the most extensively researched variant of VRP due to its resemblance with various real-world applications. To address VRPTW, numerous approaches have been presented in literature. These include exact, heuristic, and metaheuristic approaches [166]. Since, it is an $\mathcal{NP}$-hard problem, the use of exact approaches [167] for addressing VRPTW is only confined to smaller size problem instances. The metaheuristic approaches are found to be more suitable for solving the large problem instances in a reasonable amount of time [168, 169, 170]. Thus, literature is abound with various metaheuristic approaches [171, 172, 173, 174, 175, 176, 177] for VRPTW.

In [160], Expósito et al. have introduced several definitions aimed at quality of service (QoS) delivered to the customers in transport related applications. Each customer has a time window, and they must be served within this time slot, which provides a guarantee to attain a certain minimum value of quality in service. Moreover, urgency in transportation is also discussed by emphasizing the immediate response in terms of product deterioration. In this regard, authors have proposed the quality of service vehicle routing problem with time windows (*QSVRPTW*). In *QSVRPTW*, they have introduced several objectives aimed at measuring the quality of service and proposed a metaheuristic approach to address the problem. The presented metaheuristic is a *GRASP-VNS* approach which is a combination of a greedy randomized adaptive search procedure (GRASP) and a variable neighbourhood search (VNS). The main purpose of the proposed approach is to analyze the presented objective functions with reference to the quality of service. Furthermore, GRASP-VNS have used the standard neighbourhood structures for traveling salesman problem/multiple traveling salesman problem and does not consider the specific requirements of various objectives.

To address *QSVRPTW*, we have presented a steady-state grouping genetic algorithm approach with crossover and mutation operators designed after considering the characteristics of various objectives. These operators contain a balanced combination of greediness and randomness, and consider the particular requirement of the objective under consideration. We have also proposed several heuristics which are used in initial population generation phase. We have proposed two bounds for each objective. The comparison of experimental results reveals that our proposed approach is able to fetch better solutions, and also faster than the state-of-the-art approach, in terms of execution time. It is also observed that finding an initial solution itself is very tough in case of several instances due to the presence of tight constraints. The proposed heuristics are able to generate feasible solution for all the instances which highlights the additional advantage of the proposed approach to address *QSVRPTW*.

The remaining part of the chapter is organized as follows: Section 6.2 defines *QSVRPTW* formally, whereas Section 6.3 provides an overview of related work and identifies the research gap. In Section 6.4, we proposed two bounds for each objective. The proposed steady-state grouping genetic algorithm approach for *QSVRPTW* is presented in Section 6.5. The computational results and their analysis are presented in Section 6.6. Finally, Section 6.7 summarizes the main contributions and outlines some concluding remarks.

6.2 Problem definition

The VRPTW consists of a set of vertices $v = \{0, \ldots, n\}$, such that the depot is represented by vertex 0 and remaining vertices denote the customers. Each customer i has a demand of goods represented by the quantity $q_i > 0$ and a time window $[e_i, l_i]$, where e_i and l_i are the earliest and latest time for serving the customer i respectively. The time window of a customer is the time slot during which a customer can be served. We have used the terms earliest time and ready time, and, latest time and closing time interchangeably throughout this chapter. Each customer must be served exactly once by some vehicle within its time window. Each customer i has an unloading time u_i which represents the amount of time to serve the customer, i.e., the unloading time of goods once vehicle reaches at customer's place. The depot also has a time window represented by $[0, l_0]$ and all vehicles must start at opening time of depot (at time 0) and eventually come back to depot on or before its closing time (l_0). The demand of goods for depot is assumed to be zero, i.e., $q_0 = 0$. The number of vehicles are assumed to be *m*, and hence, a solution must have *m* routes. Each vehicle has a fixed capacity C, and the total load of each vehicle can not exceed this capacity.

QSVRPTW seeks a set of best possible routes $R = \{r_1, \ldots, r_m\}$ for a fleet of m vehicles according to one of the objective defined below in order to serve a given set of customers. In the context of *QSVRPTW*, the service can be picking up or delivering freight. In this chapter, the considered variant of VRPTW is mainly focused on delivery times (vehicle arrival times) to the customers. Indeed, the arrival time of vehicle at the customer's location with in their time window is a measure of the quality of service.

Let x_{ij}^k is the binary decision variable such that x_{ij}^k = 1, if vehicle k goes from customer i to j and x_{ij}^k = 0 otherwise. Here, $i, j \in$ {0,1,...,n} and $k \in$ {1,2,...,m}. The quality of service is measured by a variable s_i^k which represents the time at which vehicle k starts serving the customer i. If customer j is served immediately after customer i then,

$$s_j^k \;=\; \max\{e_j,\; s_i^k + u_i + t_{ij}\} \tag{6.1}$$

where u_i is the unloading time for customer i and t_{ij} is the travel time from customer i to customer j. For each vehicle, service start time and unloading time at depot are considered as zero, i.e., s_0^k and u_0^k are considered as zero.

In [160], the authors have proposed three objective functions focused at measuring the quality of service and those are described below. Let α_i^k is the binary decision variable such that α_i^k = 1, if the customer i is served in k^{th} route and α_i^k=0 otherwise. Here, $i \in$ {1,. . . ,n} and k $\in$ {1,2,. . . ,m}.

The first objective function is presented as the total amount of the times customers have to wait in their respective time windows. It is the sum total of the waiting times for all customers and is defined as,

$$\min f_c(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n}(s_i^k - e_i)\alpha_i^k \tag{6.2}$$

Since, sum total of ready times of all the customers is a constant, thus minimization of above function is same as minimization of the aggregate of the times at which service starts for each customer (eq. (6.3)). Hence, eq. (6.3) is considered as the first objective for the quality of service in [160], as well as in this chapter.

$$\min f_c(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n} s_i^k \alpha_i^k \tag{6.3}$$

The above objective estimates the quality of service by minimizing the sum total of the times for all the customers at which service for each of these customers starts.

The second objective function (eq. (6.4)) estimates the quality of service by the sum of slack times for all the customers. Slack time for a customer is the difference between closing time of that customer's window and the time at which service for that customer starts. Hence, the second objective function is aimed at maximizing the aggregate of the slack times for all the customers.

$$\max f_s(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n}(l_i - s_i^k)\alpha_i^k \tag{6.4}$$

The eq. (6.4) can be rewritten as:

$$\max f_s(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n} l_i\alpha_i^k - \;\sum_{k=1}^{m}\sum_{i=1}^{n} s_i^k\alpha_i^k$$

$$\text{and,} \quad \sum_{k=1}^{m}\sum_{i=1}^{n} l_i\alpha_i^k \;= \sum_{i=1}^{n} l_i \;=\; constant$$

Hence, eq. (6.4) can be finally rewritten as:

$$\max f_s(x) \;=\; \sum_{i=1}^{n} l_i - \;\min\;\; f_c(x) \tag{6.5}$$

Thus, any solution *x* which minimizes eq. (6.3) also maximizes eq. (6.4). Hence, minimizing function $f_c(x)$ can be considered equivalent to maximizing function $f_s(x)$. This fact is overlooked in [160], and this objective is presented as an independent objective there. We have corrected this mistake.

The third objective which measures the quality of services, takes the length of the customer's time window into consideration. In this function, the waiting time of customer is weighted by the length of customer's time window and thus, it is the average of relative time that customers have to wait within their time window to get the service. It is defined as follows:

$$\min f_r(x) \;=\; \frac{1}{n}\sum_{k=1}^{m}\sum_{i=1}^{n}\frac{(s_i^k - e_i)}{(l_i - e_i)}\alpha_i^k \tag{6.6}$$

It should be noted that, if $(l_i - e_i)$ is same for all customers, then this objective function $(\min\;\; f_r(x))$ is also equivalent to first objective $(\min\;\; f_c(x))$. In all of the above formulations, if customer i is not present in route of vehicle k then it is assumed that s_i^k =$l_i + 1$ which is just a finite infeasible value and this value does not have any effect as α_i^k is always zero in this case.

Let r_k represents the k^{th} route such that, r_k=$\langle c_1^k, \ldots, c_{n_k}^k\rangle$ is the order of the customers in the k^{th} route which contains n_k customers, and c_i^k denotes the customer served at i^{th} place in this route. The variable $a_{c_i^k}$ represents the vehicle arrival time at location of customer i in the k^{th} route. Furthermore, the depot is represented as $c_0^k = c_{n_k+1}^k = 0$.

The objective functions of *QSVRPTW* are subjected to the following constraints:

1. *Demand:* The total demand on any route must not exceed the capacity of the vehicle.

$$\sum_{i=1}^{n_k} q_{c_i^k} \;\leq\; C \quad \forall k \in \{1, 2, \ldots, m\} \tag{6.7}$$

2. *Arrival time:* Any vehicle must reach to a customer's location on or before the closing time window of that customer.

$$a_{c_i^k} \leq l_i \quad \forall i \in \{1, 2, \ldots, n\} \quad \forall k \in \{1, 2, \ldots, m\} \tag{6.8}$$

3. *Return time to depot:* All vehicles after serving the customers must reach to depot on or before its closing time.

$$a_{c_{n_k+1}^k} \leq l_0 \quad \forall k \in \{1, 2, \ldots, m\} \tag{6.9}$$

Hence, we have considered following two objectives with *QSVRPTW*:

- Minimize $f_c(x)$ (eq. (6.3))
- Minimize $f_r(x)$ (eq. (6.6))

subject to the constraints defined by eq. (6.7)–eq. (6.9). This leads to two problem variants, each corresponding to one objective. This differs from [160] where three objectives $f_c(x)$, $f_s(x)$ and $f_r(x)$ are considered, and this difference arises because we have proved two objectives, viz. $f_c(x)$ and $f_s(x)$ of [160] to be equivalent. It is to be noted that we have considered each objective separately exactly like [160], and not in a multiobjective manner. Further, we have presented the results for all the three objectives though we have shown $f_c(x)$ and $f_s(x)$ to be equivalent so that comparison of our proposed approach can be done with approach of [160] on all three objectives. However, the results of $f_s(x)$ for our approach are always computed from corresponding results of $f_c(x)$ by making use of eq. (6.5), and no separate executions have been performed for $f_s(x)$. For the sake of comparison only, we have provided upper bounds for $f_s(x)$ in Section 6.4.

6.3 Related work

This section presents an overview of related work including the state-of-the-art approach for *QSVRPTW* presented in [160], identify the research gap and present the motivation for our approach.

VRPTW is one among the most extensively researched variants of the vehicle routing problem (VRP). Being an $\mathcal{NP}$-hard problem, it poses a challenge for researchers in scientific

community. It considers the constraints which have resemblance with real-world applications from the transportation domain. VRPTW is more relevant in practical applications involving manufacturing, logistics, postal service and so on.

Literature contains some works that aims at the objectives of VRPs concerned with quality of service delivered to the customers. These models contain applications of transportation in school management, logistics in natural disasters, supply chain of perishable goods and in maintenance services. For example, applications of maintenance services are addressed in [178, 179, 180, 181, 182, 183, 184]. The research articles [185, 186] and [187] discuss the applications of transportation in school management. The applications of supply chain of perishable goods are discussed in [188, 189]. All of these applications focus on quality of service delivered to customers and share a common characteristic, that is urgency in supply of goods.

In [190], the authors have analyzed the customer perceptions about performance of companies and presented an empirical study on how to measure the quality of service in the urgent transport services. Several definitions of quality in domain of service sector along with various metrics have been proposed in [191]. Despite of these works, clear definitions of quality of service in transport related services were missing in literature. Expósito et al. [160] analyzed the quality of services in transportation domain and presented various objectives aimed at the quality of services for transport related applications.

In [160], the authors have proposed various objectives measuring quality of services, and also presented a metaheuristic approach to solve the *QSVRPTW* problem. The presented metaheuristic approach *GRASP-VNS* is a combination of a greedy randomized adaptive search procedure (GRASP) with a variable neighbourhood search (VNS). GRASP is a multi-start two phase metaheuristic and it was originally proposed in [137]. The proposed GRASP approach contains construction phase followed by an improvement phase. Variable neighbourhood search (VNS) has been used in improvement phase of GRASP. The VNS was proposed in [16] ,and its working principle is based on systematically varying the neighbourhood structure during the search [192]. The neighbourhood structures used in this VNS are k-chain move, k-edge interchange move and k-swap interchange move. Both GRASP and VNS are highly effective metaheuristics and have been used to solve a wide range of combinatorial optimization problems. However, this GRASP-VNS approach have used generic neighbourhood structures designed for traveling salesman problem and multiple traveling salesman problems and do not exploit the characteristics of the problem at hand as well as the objectives. This made us assert that

a metaheuristic approach designed by keeping in mind these characteristics can outperform GRASP-VNS. As a result, our grouping genetic algorithm approach was born where variation operators (crossover and mutation) are designed by taking into account these characteristics. In addition, our approach uses several problem-specific heuristics. Our proposed approach, as shown in Section 6.6, proved our assertion correct as it is found to be better in terms of solution quality as well as execution time than the GRASP-VNS approach of [160].

6.4 Theoretical study

In this section, we have performed the theoretical analysis of the objectives of *QSVRPTW* and proposed two bounds for each objective. The second bound is tighter than the first.

6.4.1 Lower bounds for objective $f_c(x)$

The two lower bounds for *QSVRPTW* with objective $f_c(x)$ can be computed as follows:

1. **Lower bound 1 (LB_1):** The $f_c(x)$ is defined as :

$$\min f_c(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n} s_i^k \alpha_i^k$$

It is the sum of all times when customer start to be served. From eq. (6.1), one can infer, if a vehicle reaches before the ready time (e_i) of customer *i*, then also it has to wait until the ready time, in order to serve this customer. Thus, the minimum value of s_i^k can be e_i, and hence, sum of all e_i's is the first lower bound for objective $f_c(x)$. So, first lower bound is calculated as,

$$LB_1(f_c(x)) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n} e_i \alpha_i^k \tag{6.10}$$

Since, all customers must be served by some vehicle, thus eq. (6.10) is equivalent to :

$$LB_1(f_c(x)) \;=\; \sum_{i=1}^{n} e_i \tag{6.11}$$

2. **Lower bound 2 (LB_2):** The process for computing this bound begins by determining the minimum unloading time u_{min} among all customers and shortest travel time md between

any two customers or between a customer and the depot. Then all customers are sorted in non-decreasing order as per their ready times, and, an iterative process ensues where during each iteration except possibly the last one, exactly m customers are assigned to routes. Assume, the sorted sequence as $\{c_{\rho_1}, c_{\rho_2}, \ldots, c_{\rho_n}\}$. We also assume routes to be numbered from 1 to m. During first iteration, first m customers from the sorted sequence are inserted at the first position of each of the m routes in such a manner so that customer c_{ρ_i} is assigned to route i ($1 \leq i \leq m$). The time when these m customers start to be served ($s^i_{c_{\rho_i}}, i = 1, \ldots, m$) are set equal to their ready times $e_{c_{\rho_i}}$. In the second iteration, next m customers from the sorted sequence are inserted at the second position of each of the m routes in such a manner that $c_{\rho_{m+i}}$ is assigned to route i ($1 \leq i \leq m$). This iterative process continues till all n customers are assigned. Please note that in the last iteration there can be less than m customers in case m is not a divisor of n.

Now, customer c_{ρ_1} occupies the first position in the first route, the second position of this route is allocated to the $(m+1)^{th}$ customer ($c_{\rho_{m+1}}$) in sorted sequence. The vehicle arrival time at location of this customer ($c_{\rho_{m+1}}$) is computed as ($s^1_{c_{\rho_1}}$ + u_{min} + md). Here, $s^1_{c_{\rho_1}}$ is the time when service for customer c_{ρ_1} starts (which is set to its ready time ($e_{c_{\rho_1}}$)). This customer (c_{ρ_1}) is served for at least the minimum time u_{min} and then vehicle must have taken at least the minimum time md, in order to reach at the location of $c_{\rho_{m+1}}$. Thus, the time ($s^1_{c_{\rho_{m+1}}}$) when customer $c_{\rho_{m+1}}$ start to be served will be equal to $\max\{e_{c_{\rho_{m+1}}},\ s^1_{c_{\rho_1}} + u_{min} + md\}$. The time at which each customer start to be served (presented as s^k_j in eq. (6.1)) for remaining customers can be calculated in the same manner, and, the sum total of all these times provides the lower bound 2 (LB_2) due to the following facts:

- We have taken minimum possible values for unloading time and travel time.
- Service for first customer on each route starts at its ready time. These first customers are those with least value of ready times.
- Assignment of customers to routes in any other sequence will yield a higher or same value of the sum total of s^k_j. This is because any deviation from sort order mentioned above will yield a higher or same sum of s^k_j values of affected customers.

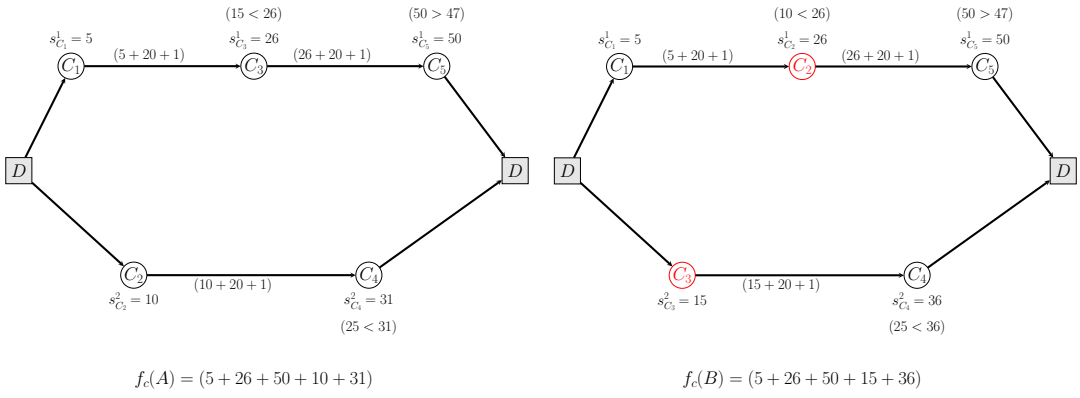
An illustrative example is developed in order to exemplify this lower bound. Figure 6.1 represents the data associated in this example. For sake of illustration, the ready time of

Table 6.1: Particulars of the illustrative example.

Customer	Ready Time (e_{C_i})	Closing Time (l_{C_i})	Unloading Time (u_{C_i})
D (Depot)	0	100	0
C_1	5	70	22
C_2	10	80	25
C_3	15	65	20
C_4	25	85	26
C_5	50	80	24

customers are considered in ascending order in the illustrative example. In this example, *m* and md are assumed to be two and one respectively. Clearly, u_{min} is 20. In the beginning, C_1 and C_2 are assigned in first positions of first and second route respectively. The $s^1_{C_1}$ and $s^2_{C_2}$ are initialized with e_{C_1} and e_{C_2}. The next customer in sorted sequence as per ready time is C_3 which is assigned on second position of first route. Now, in order to reach location of C_3, the vehicle must have taken time 26 (5+20+1). Since ready time of C_3 is less than 26, hence the C_3 must start to be served at time 26. The same calculation is done for C_4 and C_5 as per their allotted route. The calculation of S^k_i is done when customer i is assigned to route k. Figure 6.1(a) demonstrate the complete procedure of finding LB_2. In order to justify this lower bound, another solution is shown in Figure 6.1(b). In LB_1, we have demonstrated that the minimum value of s^k_i can be e_i. In Figure 6.1(a), the customer C_3 is assigned second position in first route, hence $s^1_{C_3}$ is greater than e_{C_3}. Consider a scenario, where $s^k_{C_3}$ achieves its minimum value i.e., e_{C_3}. Since ready time of C_3 is less than the minimum unloading time, hence $s^k_{C_3}$ equal to e_{C_3} is possible only if we assign C_3 at first position in any route. Figure 6.1(b) shows this assignment and we can see that sum of all times when customers start to be served is increased now. So, minimizing s^k_i of one customer will deteriorate this value for another customer and since customers are sequenced as per ready time, thus exchanging a customer *x* having smaller value of ready time with a customer *y* with larger value of ready time to make s^k_y equal to e_y will yield a larger gap between s^k_x and e_x. Thus, such an exchange will lead to increase in sum of all s^k_i. Hence, the way customers are assigned in Figure 6.1(a) generates the minimum value of sum of all time when customer start to be served and this is a lower bound for $f_c(x)$. It is pertinent to mention that our argument is not relying on the sequence of customers served, but it is based on the fact that if we have more number of customers than routes with ready time less than the unloading time, then definitely some customers will not have s^k_i equal to e_i no matter

in what order they are served, and the values of s_i^k for such customers will also have impact on the customers coming later in the corresponding route. Thus, we will get a lower bound tighter than LB_1. With respect to example under consideration, any customer served at second position of first route will have s_i^1 equal to its ready time if ready time is greater than equal to vehicle arrival time at this position or s_i^1 is equal to vehicle arrival time if ready time is less than vehicle arrival time. In LB_1, we already assumed that s_i^k equal to the ready time (e_i), hence in LB_2 either a customer will have s_i^k equal to the e_i or s_i^k equal to the vehicle arrival time. To make it lowest possible value, sorting is done as per ready time so that the sum of all s_i^k is minimum. In any case if the customer with less ready time value is assigned later, then ultimately the sum of all s_i^k will be increased.



(a) **Solution A** (b) **Solution B**

Figure 6.1: Two solutions of the illustrative example

6.4.2 Upper bounds for objective $f_s(x)$

The two upper bounds for *QSVRPTW* with objective $f_s(x)$ can be computed in the following manner:

1. **Upper bound 1 (UB_1):** The objective $f_s(x)$ is defined as,

$$\max f_s(x) \;=\; \sum_{k=1}^{m}\sum_{i=1}^{n}(l_i - s_i^k)\alpha_i^k$$

In eq. (6.5), we have shown that,

$$\max f_s(x) \;=\; \sum_{i=1}^{n} l_i - \;\; \min \;\; f_c(x)$$

Thus, UB_1 can be computed as,

$$UB_1 \;=\; \sum_{i=1}^{n} l_i - \;\; LB_1 \;\; = \;\; \sum_{i=1}^{n} (l_i - \; e_i) \tag{6.12}$$

2. **Upper bound 2 (UB_2):** Similarly, UB_2 can be computed as,

$$UB_2 \;=\; \sum_{i=1}^{n} l_i - \;\; LB_2 \tag{6.13}$$

6.4.3 Lower bounds for objective $f_r(x)$

The two lower bounds for *QSVRPTW* with objective $f_r(x)$ are as follows:

1. **Lower bound 1(LB_T):** In LB_1 of objective $f_c(x)$, it is shown that the minimum value of s_i^k can be e_i. Substituting this value in eq. (6.6), we will get $f_r(x)$ as zero. Thus the first lower bound is zero for objective $f_r(x)$, which is a trivial lower bound.

2. **Lower bound 2(LB):** The second lower bound is a non-trivial lower bound and it is referred to as LB for this objective. It is calculated by using the values of LB_1 and LB_2, i.e., the lower bounds defined for objective $f_c(x)$. To compute LB, we need to find maximum length of customer's time window among all customers, i.e., mw=$\max\{l_i - e_i\}$ $\forall\, i \in \{1, 2, \ldots, n\}$. In eq. (6.6), $f_r(x)$ is defined as:

$$\min f_r(x) \;=\; \frac{1}{n} \sum_{k=1}^{m} \sum_{i=1}^{n} \frac{(s_i^k - e_i)}{(l_i - e_i)} \alpha_i^k$$

The value of denominator $(l_i - e_i)$ in above equation can be at most mw for any customer. Hence, replacing $(l_i - e_i)$ by mw in above equation provides a lower bound for $f_r(x)$, that is,

$$\min f_r(x) \;\geq\; \frac{1}{n} \sum_{k=1}^{m} \sum_{i=1}^{n} \frac{(s_i^k - e_i)}{mw} \alpha_i^k \;\; = \;\; \frac{1}{n \;\times\; mw} \sum_{k=1}^{m} \sum_{i=1}^{n} (s_i^k - e_i) \alpha_i^k$$

and, $$\frac{1}{n \times mw} \sum_{k=1}^{m} \sum_{i=1}^{n} (s_i^k - e_i)\alpha_i^k = \frac{1}{n \times mw} \sum_{k=1}^{m} \sum_{i=1}^{n} s_i^k \alpha_i^k - \sum_{k=1}^{m} \sum_{i=1}^{n} e_i \alpha_i^k$$

Which is equal to,

$$\frac{1}{n \times mw} (f_c(x) - LB_1)$$

As already demonstrated that the lower bound for $f_c(x)$ is LB_2. Thus, the lower bound LB for $f_r(x)$ can be computed utilizing LB_1 and LB_2 for $f_c(x)$ as follows

$$LB = \frac{1}{n \times mw} (LB_2 - LB_1) \tag{6.14}$$

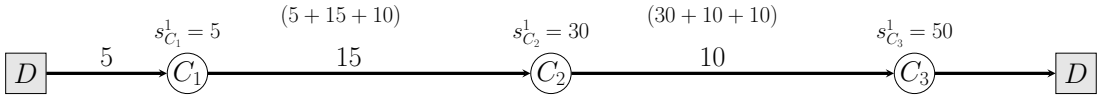
The objectives $f_r(x)$ and $f_c(x)$ have different targets to optimize. The objective $f_c(x)$ seeks a minimization on sum of times when service for each customer starts, while $f_r(x)$ intends to minimize the average of relative time that customers have to wait within their time window to get the service. For illustration, consider the example presented in Table 6.2. Only one route (*m*=1) is assumed in the example, for ease of understanding. Two solutions (X and Y) are demonstrated in Figure 6.2. The solution X represented as Figure 6.2(a) incurs $f_c(X)$=85 and $f_r(X)$=0.533, whereas solution Y represented as Figure 6.2(b) incurs $f_c(Y)$=100 and $f_r(Y)$=0.311. In this example, the ready time of all customers are less than equal to vehicle arrival time, hence the time when customer start to be served is equal to vehicle arrival time. This demonstration provides a scenario, where a solution is better in one objective and worse in another objective. We can observe that, solution X is superior in $f_c(X)$ but inferior in $f_r(X)$, in comparison to solution Y.

Table 6.2: Data of the illustrative example.

Customer	**Ready Time** (e_{C_i})	**Closing Time** (l_{C_i})	**Unloading Time** (u_{C_i})
D (Depot)	0	100	0
C_1	5	80	10
C_2	10	30	10
C_3	20	70	5

Table 6.3: Time matrix of the illustrative example.

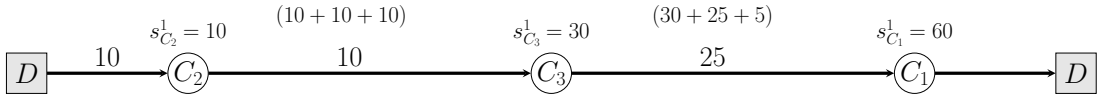
	D	C_1	C_2	C_3
D	0	5	10	15
C_1	5	0	15	25
C_2	10	15	0	10
C_3	15	25	10	0



$$f_c(X) = (5 + 30 + 50) = 85$$

$$f_r(X) = \tfrac{1}{3}((\tfrac{5-5}{80-5}) + (\tfrac{30-10}{30-10}) + (\tfrac{50-20}{70-20})) = 0.533$$

(a) **Solution X**



$$f_c(Y) = (10 + 30 + 60) = 100$$

$$f_r(Y) = \tfrac{1}{3}((\tfrac{10-10}{30-10}) + (\tfrac{30-20}{70-20}) + (\tfrac{60-5}{80-5})) = 0.311$$

(b) **Solution Y**

Figure 6.2: Solutions to illustrate both objectives

6.5 Steady-state grouping genetic algorithm for *QSVRPTW*

We have developed a steady-state grouping genetic algorithm approach with dedicated crossover and mutation operators for each objective to solve the *QSVRPTW*. The proposed approach is inspired from the steady-state grouping genetic algorithm presented in [42]. Henceforth, the proposed approach will be referred to as as *GGA-QOS* in this chapter.

As already mentioned in Section 6.2, the three objectives have been proposed in [160], which measure the quality of service. The first two objectives namely, $f_c(x)$ and $f_s(x)$ represented by eq. (6.3) and eq. (6.4) respectively are found to be equivalent as explained earlier. On the other hand, the third objective ($f_r(x)$) represented by eq. (6.6) is different, if the length of customer's time window, i.e., $(l_i - e_i)$ is not same for all customers. Hence, our approach addresses the first

($f_c(x)$)and third objective ($f_r(x)$) only. For comparison purpose, the results of $f_s(x)$ for our approach are always computed from corresponding results of $f_c(x)$ by making use of eq. (6.5) and no separate executions are performed for $f_s(x)$. We have devised dedicated crossover and mutation operators for each objective by taking into account the specific characteristics of each objective. Thus, there are two versions of *GGA-QOS*. The first version caters to objectives $f_c(x)$ and $f_s(x)$, whereas the second version is developed to address the third objective ($f_r(x)$). These two versions differ only in variation operators, and apart from variation operators, these versions are identical. In fact, both versions evaluate both the objectives as the other objective (objective not catered by the version) is utilized for checking the uniqueness of a solution as explained in the next paragraph. It is to be noted that the number of routes (*m*) is fixed and predetermined for *QSVRPTW* [160]. Expósito et al. [160] set number of routes in their instances for *QSVRPTW* to be equal to number of routes in optimal solution when these instances are solved as VRPTW instances plus 15% rounded to closest integer.

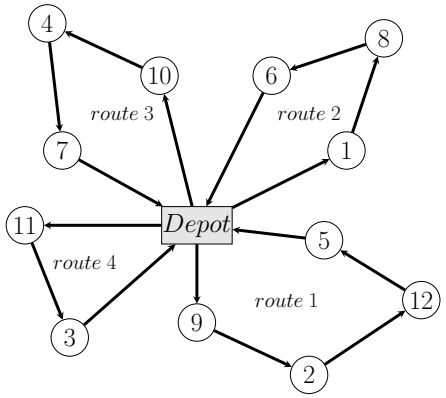
The approach begins with initial population generation which make use of several heuristics to generate these initial solutions. The two parents for crossover have been selected by using the binary tournament selection (BTS). The probability used in BTS for selection of parent is assigned with a value p_{bts}. The crossover is employed on the two parents, which yields an offspring. If the offspring is feasible, it goes through mutation, and a mutant solution is obtained. If the offspring is not feasible then better of the two parent solution is copied as a offspring and subjected to mutation to obtain a mutant solution. If both offspring and the mutant are feasible solutions then the superior one among them is added in population in place of the worst solution of the population, otherwise the feasible solution among them is added in place of worst solution. If both are infeasible then both are discarded and no replacement is made. Uniqueness is checked before adding a solution in population. A solution is considered as unique, if both the addressed objective values do not coincide with the objective values of any other existing member in the population. This same process iterates until the termination condition remain unsatisfied. The probability of producing an infeasible solution through crossover is more in comparison to mutation and this probability increases for those instances where total number of possible feasible solutions is less due to the presence of tight constraints. Hence, to avoid wastage of time due to repeated failures of crossover operator in generating a feasible solution, if crossover operator fails for a total of Mut_x times, then the crossover operator is not applied further and the approach uses only mutation operator. It is to be noted that ascertaining uniqueness of a solution on the basis of different values for both the objectives may discard a solution that is

unique in rare cases. However, it saves lot of computation time as determining uniqueness when objective function values match requires comparison among routes in terms of composition whose contribution to the objective value is same.

The proposed mutation operator possesses exploitative characteristics, thus for the generations coming after Gen_{max} value, only mutation operator is employed and crossover operator is skipped. The mutation operator in these last generations, serves as a local search to some extent and results in further improvement in quality of the solutions in the population. In all the cases where crossover is skipped, the better one among both parents is copied as the offspring and the mutation is applied on this offspring.

On some instances finding the feasible initial solutions can be very tough due to the existence of a few feasible solutions only. Thus, if the initial population generation yields only one feasible solution then again the crossover operator is not used and only mutation operator is employed to generate the solution. For such instances the proposed approach simply works like a hill climbing algorithm. It is pertinent to mention that if both parents are identical then the crossover operator will simply produce the copy of parent. Thus, if BTS procedure selects same member from population as two parents, then also only mutation operator is used and crossover phase is skipped. The pseudo-code of proposed *GGA-QOS* approach is provided in Algorithm 19. If initial population generation yields only one solution then BTS procedure will assign same solution to both parents, hence we have not explicitly mentioned this condition in Algorithm 19.

The crossover and mutation operators build a complete solution from a partial solution by following an iterative process. In this process the unselected customers are inserted iteratively into best position in the partial solution. The definition of best position varies as per the objective and it is defined in Section 6.5.3. While creating a complete solution, it is possible that no feasible position is found for some customer, so that all constraints can be satisfied. In such a scenario, the variation operators used in proposed approach discards the solution. Literature contains some VRPTW problems such as in [193, 194], where number of routes are not static. The aforementioned situation is handled by adding a new route and the customer involved in violation of constraint(s) is inserted at the beginning of newly added route. Thus in such VRPTW variants, the crossover and mutation always generate a constraints satisfying solution. Since in *QSVRPTW*, the number of routes are fixed for each instance, thus the crossover and mutation operator need not necessarily produce a feasible solution always. Hence, there is no choice other than discarding the solution. Subsequent subsections contain description about other salient features of our *GGA-QOS* approach.



(a) **Solution**

Sequence of customers served

1^{st} route	9	2	12	5
2^{nd} route	1	8	6	
3^{rd} route	10	4	7	
4^{th} route	11	3		

(b) **Representation**

Figure 6.3: Solution representation

6.5.1 Solution representation

In *QSVRPTW*, a solution is a set of *m* routes and each route is represented as a linear sequence of customers, i.e., if k^{th} route contains $|r_k|$ customers, then the route r_k contains a linear permutation of $|r_k|$ customers. In a linear permutation, the positions of customers dictates the sequence in which the customers are being served. Since by default any vehicle starts from the depot and after serving the customers eventually returns to the depot, hence depot is not included in solution representation. Figure 6.3 demonstrates the solution representation, by assuming an example with 12 customers (*n*=12) and 4 routes (m=4), namely, $R = \{r_1, r_2, r_3, r_4\}$. Albeit, 0 (depot) is not included in solution representation, but in computation of any value the start and end position of any route is assumed as zero. For example, the first route r_1 is $\langle 0, c_1^1, c_2^1, c_3^1, c_4^1, 0\rangle$, though we represented it as $\langle c_1^1, c_2^1, c_3^1, c_4^1\rangle$. Here, c_1^1 = 9, c_2^1 = 2, c_3^1 = 12 and c_4^1 = 5, i.e., the first customer in this route is customer 9 and the second one is customer 2, and so on. Similarly, 3 customers are served in each of the routes r_2 and r_3 and 2 customers are served in route r_4.

6.5.2 Initial population generation

Several heuristics have been devised for use in initial population generation procedure. Algorithm 18 presents the pseudo-code of initial population generation procedure and the detailed description of these heuristics are presented below.

1. **Sort_Cust heuristic:** First, a pool of customers are created by sorting them as per their

latest time (l_i), in a non-decreasing order. The customer whose latest time is less should be given more priority over others, this is the reason behind this sorting. Thereafter, an empty solution with *m* routes are created. The first customer is picked from the sorted sequence and inserted into empty solution. Now, the partial solution contains the selected customer in first position of the first route. The second customer is selected from the pool and inserted before and after the customer present in solution. It is also inserted at first position of second route. Thus, now we have total three tentative solutions. Out of these three solutions, one having better objective is considered as updated partial solution. The remaining customers are inserted iteratively. In each iteration the i^{th} customer is selected and evaluated at every possible position in the partial solution and then inserted into best position. In this way, a complete solution is generated. It is also possible that some customer cannot be assigned in any route due to violation of constraint(s). In this case, the solution is simply discarded.

2. **Sort_Cust_Rnd heuristic:** It differs from Sort_Cust heuristic only in one aspect. Here, the selection of customer for insertion in partial solution is done in a random manner from a window of size *m*, rather than sequentially. A window of size *m* is considered in pool of sorted customers and a random customer is selected within this window. After selection of the customer, window is slid by one position to the right. For example, assume there are five customers sorted as per their latest time viz., $\{C_5, C_2, C_4, C_1, C_3\}$ and number of vehicles (*m*) is three. Then a window of size three is considered viz., $\{C_5, C_2, C_4\}$ and one customer is randomly selected. For definiteness, let us say the selected customer is C_4 which is at position three. After this selection, the customer at position one will be moved to position three and the window is slid by one position to the right, i.e., the next window will be $\{C_2, C_5, C_1\}$. Using a sliding window in this manner saved us the cost of shifting all customers one position left after the position of selected customer and which is $\mathcal{O}(n)$. Here, also the solution is discarded, if for some customer no feasible place can be found. Last *m-1* customers are selected sequentially in this heuristic.

3. **Rand_Cust heuristic:** Here, a random sequence of customers are considered as pool of customers, instead of a sorted one. Other than this, it is same as Sort_Cust heuristic.

4. **Rand_Sol heuristic:** An empty solution is created and thereafter, a random customer is added at the beginning of the first route. Again, another random customer is inserted into

next position in the current route. Following the same process, the i^{th} customer is added at the next position of previously added customer. If the newly inserted customer does not satisfy the constraint(s), then all the previously existing positions starting from the first position of first route are tried for insertion. If any feasible place is found then the customer is inserted there. Otherwise, a new route is added and this customer is inserted at the beginning of this newly added route. If at any stage the number of routes exceeds *m*, then this process is stopped and solution is discarded. The entire process is repeated until all customers get assigned to some route. After insertion of all customers, if the solution contains exactly *m* routes then this solution is inserted into population, otherwise it is discarded. This heuristic is developed from the diversity perspective only. Here, a new route is created only after trying the insertion at every possible position in existing routes. Thus, this heuristic yields a solution which has fewer routes, and hence, able to generate a feasible solution most of the times.

5. **Feasible_Sol heuristic:** After certain number of trials (100 $\times$ n), if all of the aforementioned heuristics fail to generate a feasible solution, then this heuristic is used. It consists of two phases. In first phase, a population P_γ with γ solutions are generated by using *Sol_Gen* function. These γ solutions are actually infeasible solutions and they satisfy certain criteria, which is explained in the detail in description *Sol_Gen* function. The second phase uses *Make_Sol* function which selects one solution randomly from the P_γ and try to make it a feasible solution. Whenever *Make_Sol* function is successful in generation of a feasible solution, this heuristic terminates and returns the feasible solution.

   (a) **Sol_Gen function:** For each customer, all remaining customers are sorted in non-decreasing order as per their travel time from the customer under consideration. This can be visualized as an *(n $\times$ n)* matrix, where the each cell of the first column is the customer under consideration and remaining *(n-1)* columns of each row are the sequence of customers which is sorted by the travel time from the customer under consideration in non-decreasing order. These *(n-1)* customers are the possible choices which can come after the customer under consideration, in a route. The same sequence of customers is also determined for depot, as depot is the beginning point of any route. This calculation is done as a prerequisite of this function. This function builds a solution in two phases.

Phase 1: The process begins with an empty solution with *m* routes and inserts customers in an iterative manner. Each iteration inserts customers in *m* routes. Since any route begins with depot, so the first reference point of all routes initialized with depot. A random customer from nearest seven customers from depot is selected and inserted into first route. Same process is repeated for remaining *(m-1)* routes. For next iteration, the previously inserted customer in each route acts as new reference point and a random customer from nearest seven customers from new reference point is selected and inserted in each route. If the selected customer is already assigned into some route or insertion of selected customer results in an infeasible solution, then a sequential search is started from the first position of the sorted sequence of reference point customer and an unassigned as well as feasible customer is determined and inserted into the route. Here, the addition of customer is done route wise, i.e., after inserting a customer in route x, the next customer is inserted into route $x + 1$ and after last *(m^{th})* route, we again start from the first route. If in some route, all customers are visited as per sorted sequence and no feasible customer is found which can be inserted into next position of the reference point customer, then the insertion in this route is stopped and remaining routes are tried for filling more customers as per the aforementioned procedure. The first phase ends when insertion in any route is not possible. At the end of this phase, we will get a partial solution with constraint satisfying customers and another set which contains unassigned customers.

This phase is developed with intention to find a partial solution in which customers are inserted by following nearest neighbor with some randomness. This heuristic is developed for finding a feasible solution for very compact instances. The word compact signifies that the instances has very few (optimal) routes, and hence, the customers need to be inserted in a very compact manner. The violation of arrival time constraint is the primary cause behind difficulty in finding a feasible solution in such instances. The sequence of visiting customers which are near may save traveling time, thus such a strategy may provide a partial solution in which the less travel time covers more number of customers.

Phase 2: The output of previous phase contains two sets of customers. First set of customers are present in partial solution and remaining customers are considered as unassigned. In this phase, the unassigned customers are sorted in non-decreasing

order as per their latest time (l_i). The intention behind sorting is same as discussed in Sort_Cust heuristic. Then an iterative process is used to insert the unassigned customers in some route in partial solution of assigned customers. In every iteration, the i^{th} customer is picked randomly from a sliding window of customers beginning at i^{th} position and ending at $(i+4)^{th}$ position in sorted sequence of unassigned customer, and inserted into the best position. This sliding window is managed in the same manner as sliding window of Sort_Cust_Rnd heuristic. If no insertion is possible then this phase halts and the total number of customers in partial solution is returned.

This phase follows the insertion of remaining customers as per their latest time, whereas the former one inserts the customers as per nearest neighbor concept. Both these phases consider some randomness along with the specific idea of selection of customer for insertion. Thus, combination of these two phases represent a mix of two strategies with suitable randomness.

The Sol_Gen function is used to create γ solutions. Here, each solution contains *m* routes and some unassigned customers. Thus these are actually infeasible solutions. If the solution contains 96% or more constraint satisfying customers in *m* routes and remaining customers are unassigned, then this solution is added in population P_γ. Otherwise the solution is discarded and again this function is used to create a new solution. This function can also produce a feasible solution in some cases, i.e., 100% constraint satisfying customers in *m* routes. In such a scenario, *Make_Sol* function is not used and this feasible solution is considered as initial solution. Using this function, γ solutions are created and added in population P_γ. If after a certain number of attempts ($1000 \times n$), no solution is found which satisfy the above criteria, then the 96% condition is relaxed to 94%, for all remaining solutions in P_γ. The value of γ is initialized with 50. If Sol_Gen function fails to provide 50 solutions in *MAXTRY* attempts then γ is reset to the current number of available solutions if their number is at least 30. If this number is less than 30, then the process continues till the number of available solutions reaches 30. The *MAXTRY* is set as ($10000 \times n$).

(b) **Make_Sol function:** This function may take several runs to make a feasible solution. Here, each run begins by selecting a random solution (S_{rnd}) from P_γ. The solution

S_{rnd} is infeasible due to presence of unassigned customers, and it can be converted into a feasible solution by inserting all the unassigned customers at feasible positions in any of *m* routes. The customers in S_{rnd} are divided into two pools viz. assigned and unassigned customers. The customers belonging to *m* routes are considered as assigned customers, whereas the remaining customers are considered as unassigned customers. Afterwards, a random number $\in$ [1,3] of customers are picked randomly from the list of assigned customers and added into pool of unassigned customers. The selection of customers is done iteratively. In each iteration, a random route from *m* routes, is selected and a random customer is picked from this route and inserted into pool of unassigned customers. The customer is picked only if the selected route contains more than one customer, otherwise this iteration is discarded. This step is similar to ruin strategy used in mutation operator. The unassigned customers need to be inserted in *m* routes. These customers are inserted by following an iterative process. In each iteration, one customer from the pool of unassigned customers is selected randomly and tried for insertion in every possible positions in *m* routes. All feasible positions in a route is evaluated and best position in a route is remembered. Similarly, other routes' best positions are also remembered. Thereafter, one route is randomly selected and the customer is inserted into best position in this route. If no feasible place is found for the selected customer then this function go for next run. Before going for next run, the function updates the previously selected solution S_{rnd}, if there is an increment in number of customers in *m* routes, as compared to original S_{rnd} solution. The best position is defined in Section 6.5.3.

This heuristic is particularly useful for those instances, where the number of routes *m* is equal to number of routes in optimal solution of this instance when solved as VRPTW. It is found that the unassigned customers mostly violate arrival time constraint represented by eq. (6.8), in the solutions generated by Sol_Gen function. The arrival time constraint forces a customer to be inserted into a position in some route, where it satisfies the time window of customer. The violation of this constraint implies that no place is found where the customer's time window is satisfied. Since, Make_Sol function also removes some assigned customers, thus inserting the unassigned customer at best place guarantees that this customer is inserted at a position where its service time is least away from starting time (e_i) of the customer. Hence such insertion will have minimal impact on the starting

time of other customers, and hence, further chances of insertion of other unassigned customers may increase. While selection of a random route and insertion at best position of this route is done from the perspective of diversity.

If this function is unable to create a feasible solution after (*MAXTRY* $\times$ 10) runs, then Sol_Gen function is reused to generate a new set of P_γ solutions and then again Make_Sol function is employed to create a feasible solution. In our experiment, this situation had occurred for a few times only.

The proposed heuristics can be classified into three categories. This classification is based on the quality of solutions generated by the particular heuristic. The Sort_Cust, Sort_Cust_Rnd and Rand_Cust heuristics follow a greedy approach with some randomness, hence they generate better quality initial solutions. So, these heuristics are used for superior quality initial solutions in the population. Rand_Sol Heuristic generates purely random solutions. It is used to add diversity in initial solutions. The manner in which these heuristics are utilized for initial population generation is shown in Algorithm 18. Sort_Cust heuristic being purely greedy generates only a single solution, viz. the first solution in the initial population. For generating other solutions, the first preference is given to Sort_Cust_Rnd and Rand_Cust heuristics, which are used with equal probability in a mutually exclusive manner, and only when they fail, Rand_Sol heuristic is used. In tougher instances, we have observed that the greedy based heuristics sometime provide a feasible solution and sometimes they fail to get either a feasible or a unique solution. Whenever these heuristics fail, the Rand_Sol Heuristic is used. The Rand_Sol Heuristic bestows two advantages. First, the solution generated by this adds diversity in population. Second, the greedy heuristics may take a large number of attempts to generate *p* unique solutions, thus it is advantageous to generate some random solutions in less amount of time. Feasible_Sol heuristic is used only in case when all other heuristics have failed in finding any feasible solution. Hence, the sole purpose of this heuristic is to generate a feasible solution.

In order to demonstrate the behavior of the proposed heuristics, an experiment is conducted. Two versions of initial solution generation phase are used in this experiment and objective $f_c(x)$ is considered. In this experiment, we have used the instance *C109* with 100 customers. In the first version, we have used the greedy based heuristics, namely, Sort_Cust, Sort_Cust_Rnd and Rand_Cust heuristics (in

Algorithm 18: Pseudo-code of Initial population generation

```
Input: p (population size), m (number of vehicles)
Output: BEST_sol
  // BEST_sol is the best solution in population.
// u_01 is a uniform variate between [0,1).
p_i ← 1;
Itr ← 1;
Sol ← Sort_Cust Heuristic();   // Sol is the solution generated by Sort_Cust Heuristic.
if (Sol is a feasible solution) then
    p_i ← p_i + 1 ;
while (p_i < p) do
    if (u_01 < 0.5) then
        Sol ← Sort_Cust_Rnd Heuristic();
    else
        Sol ← Rand_Cust Heuristic();
    if (Sol is a feasible solution and unique) then
        p_i ← p_i + 1 ;
    else
        Sol ← Rand_Sol Heuristic();
        if (Sol is a feasible solution and unique) then
            p_i ← p_i + 1 ;
    Itr ← Itr + 1;
      // n is the number of customers.
    if (Itr > 100 × n) then
        if (p_i > 0) then
            p ← p_i;   // p is reset to p_i.
        else
            p ← 1;
            Sol ← Feasible_Sol Heuristic();
return (BEST_sol) ;
```

the same manner as shown in Algorithm 18) to generate the initial population. In this version, Rand_Sol Heuristic is not used at all. On the other hand, in the second version, only Rand_Sol Heuristic is used. Both versions generate population of size p. Figure 6.4 demonstrates the convergence behavior of proposed approach with greedy based heuristics and random solution based heuristic. The greedy heuristic version provides superior quality best solution in initial population, with objective value 40028.25, whereas, the random heuristic yields best solution in initial population, with objective value 47137.75. Thus, greedy heuristics provide better quality initial solutions. The greedy version converges faster and generates the best solution in 2×10^5 iterations, while random heuristic version takes 5×10^5 iterations and still provide a solution inferior than greedy version. The objective value of final solutions yielded by greedy and random versions are 37448.92 and 37620.08, respectively. Thus, greedy based heuristics provide better quality initial

solutions as well as faster convergence with better quality final solution. While, Rand_Sol Heuristic adds diversity in initial population.

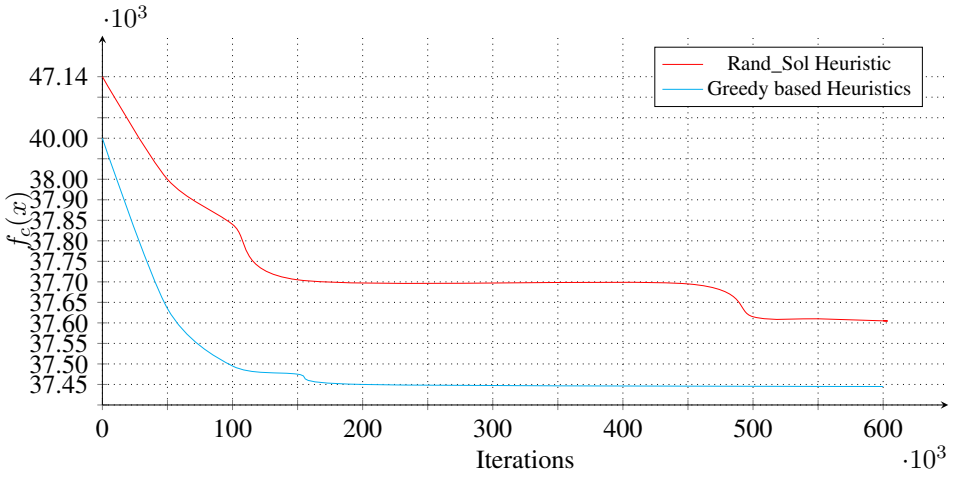


Figure 6.4: Comparison of convergence of approach with greedy based and random based heuristics on instance C109.

6.5.3 Crossover operator

In the proposed *GGA-QOS* approach, the crossover operator is designed by considering the characteristics of *QSVRPTW* as well as the characteristics of the two objectives addressed in this chapter. The crossover operator used here are based on the crossover operators presented in [42, 142]. This crossover operator generates an offspring in two stages.

In the first stage, an empty solution is created and the offspring is constructed in an iterative manner. In each iteration, one parent is picked randomly with equal probability and the most promising route in this parent is found. The most promising route is added in offspring by incrementing the number of routes in the offspring by one. This route is discarded from the list of routes of selected parent. The customers present in the most promising route are removed from their respective routes in the other parent one-by-one. After removing a customer, all the subsequent customers on the affected route are shifted one place backward. The entire procedure is continued for m times, so that a partial solution having m routes is obtained.

Evidently, at the end of first stage, some customers left unallocated and should be allocated in some route of the partial solution created by first stage. The second stage creates a complete solution in an iterative manner. In each iteration, an unallocated customer is selected randomly

and get assigned at the best position in the partial solution. The best position is defined as per the objective under consideration. The selected customer is checked at every position in each route and a set of constraints satisfying positions are determined. Out of all these feasible positions, the position incurring the least objective value is considered as best position. If no position satisfying all the constraints is found then the crossover operator discards the solution and this scenario incurs an unsuccessful crossover operation.

Since two objectives are considered for this problem, hence, the criteria of most promising route and best position vary as per the objective and their details are given below.

$f_c(x)$: The total service time objective minimizes the sum of all times when customers start to be served. If in a route the sum of all service times of customers is less while more number of customers are served, then selection of such route may provide a solution having minimum total service time. Hence, for objective $f_c(x)$, the route having minimum ratio of the total service time per customer in the route, is defined as most promising route. The best position for $f_c(x)$ is the place which leads to the minimum increase in total service time.

$f_r(x)$: This objective also shares the similarity with $f_c(x)$, as here we minimize the average of relative time that customers have to wait within their respective time windows to get the service. Thus, here the most promising route is the route having smallest ratio (s_r) of sum of the relative time that customers have to wait to the number of customers in the route. More precisely, the ratio s_r is represented by eq. (6.15). Let r_k represents the k^{th} route such that, r_k=$\langle c^k_1, \ldots, c^k_{n_k} \rangle$ is the order of the customers in k^{th} route which contains n_k customers and c^k_i denotes the customer served at i^{th} place in this route.

$$s_r \;=\; \frac{1}{n_k} \sum_{i=1}^{n_k} \frac{(s_{c^k_i} - e_{c^k_i})}{(l_{c^k_i} - e_{c^k_i})}. \tag{6.15}$$

Here, the best position in a route is the position which incurs least increment in this objective function.

6.5.4 Mutation operator

The mutation operator used in this work are based on delete and re-insert strategy and consists of two phases. The first stage selects some customers and deletes them from the solution. The complete solution is again recreated by following a greedy approach during the second stage.

In mutation procedure, one or more route is selected randomly and some customers are removed from these route(s). We have used two variants of mutation operator, viz. MUT_1

Algorithm 19: Pseudo-code of *GGA-QOS for QSVRPTW*

Input: p (population size), VRPTW instance, n (number of customers), C (Vehicle capacity), OptVeh (Number of routes in optimal solution of the instance)
Output: Best solution found so far

```
// u01 is a uniform variate between [0,1)
m ← ⌊ OptVeh + OptVeh × 15/100 ⌉;   // m is number of routes in solution
BEST_sol ← Initial population generation(p);   // BEST_sol is the best solution in population
NoImp ← 0;     // NoImp is the count for which no improvement in best solution in successive generations
Itr ← 0;       // Itr represents generations
UnsuccCR ← 0;  // UnsuccCR is the number of times Crossover fails to yield a feasible solution
while (NoImp < N_max) do
    Par1 ← BTS(p);  // Select parents using binary tournament selection
    Par2 ← BTS(p);
    flag_CR ← 1;
    if (Itr > Gen_max) OR (UnsuccCR > Mut_x) OR (Par1 is same as Par2) then
        flag_CR ← 0;
        if (Par1 is same as Par2) then
            offspring ← Par1; // Copying parent solution to offspring
        else
            offspring ← Better_obj(Par1,Par2); // The parent whose objective value is better
    if (flag_CR is equal to 1) then
        C_x ← Crossover(Par1,Par2); // C_x is 1 if Crossover yields a feasible solution
        if (C_x is equal to 1) then
            offspring ← Solution after crossover;
        else
            UnsuccCR ← UnsuccCR +1;
            offspring ← Better_obj(Par1,Par2);
        if (u01 < 0.5) then
            M_x ← MUT1(offspring);
        else
            M_x ← MUT2(offspring);
        if (M_x is equal to 1) then
            mutant ← Solution after mutation;
        // M_x is 1 if Mutation yields a feasible solution
        if ((C_x is equal to 1) and (M_x is equal to 1)) then
            if (offspring is unique) and (better than worst solution as well as mutant) then
                // worst solution is the solution with poorest objective value in the population
                Replace worst solution with offspring;
            else if (mutant is unique) and (better than worst solution) then
                Replace worst solution with mutant;
        else if (C_x is equal to 1) then
            if (offspring is unique) and (better than worst solution) then
                Replace worst solution with offspring;
        else if (M_x is equal to 1) then
            if (mutant is unique) and (better than worst solution) then
                Replace worst solution with mutant;
    if (Newly added solution is better than BEST_sol) then
        BEST_sol ← Newly added solution;
        NoImp ← 0;
    else
        NoImp ← NoImp + 1;
    Itr ← Itr + 1;
return (BEST_sol);
```

and MUT_2 with equal probability in a mutually exclusive manner. The first variant uses two versions of route selection strategy, in a mutually exclusive way. The first version is used with probability p_m which selects a route in a random manner and some customers $\in$ [1,$n_k - 1$] are removed randomly from this route. Here, n_k is the number of customers in selected route. With remaining probability $(1 - p_m)$, second version is used which removes n_c customers in an iterative manner. In each iteration, a random route is selected and a random customer from this

route is removed. If the chosen route contains only one customer then this iteration is discarded, as the number of routes are fixed and removing the only customer from a route will make the solution infeasible.

The second variant of mutation operator selects a route randomly. Assume selected route contains n_k customers, then two positions $\in$ [1,n_k] are selected randomly. The customers between these two selected positions (including) are removed from the route. In case, if the selected positions are first and last positions of the route then the start position is reassigned as second position, so that first customer remains in the route.

The removed customers constitute a list of unassigned customers which need to be inserted into best position. The complete solution is obtained by following an iterative process. Each iteration selects a customer from the list of unassigned customers, in the same sequence in which they have been removed, and assign the selected customer at best position in some existing route. If no feasible place is found due to disagreement of constraint(s), then the solution is discarded. Here, the best position is same as defined for $f_c(x)$ and $f_r(x)$ objectives, in Section 6.5.3.

6.6 Computational results

This section presents the experimental results of the proposed *GGA-QOS* approach and its comparison with the state-of-the-art approach of [160]. Our proposed approach has been implemented in C language and executed on a Linux based PC equipped with 3.10 GHz Intel Core i5 processor and with 8 GB of RAM.

The values for various parameters of *GGA-QOS* that we have used in all our experiments are as follows: *p* is the population size which is set to 100. The probability p_{bts} used in BTS procedure and it is assigned with a value 0.7. The parameter Mut_x and Gen_{max} used to employ only mutation in proposed approach is set as $2000 \times n$ and $15000 \times n$, respectively, where n is the number of customers. The probability p_m and parameter n_c in mutation operator is set as 0.75 and 4, respectively. If there is no improvement in best solution for consecutive N_{max} iterations then the *GGA-QOS* approach terminates and it returns the best solution found so far. The parameter N_{max} is set to $5000 \times n$. Our approach is executed for 10 independent times on each instance like the GRASP-VNS approach of [160]. All these parameter values are obtained by empirically studying the performance of proposed approach in numerous trials. These parameter values are able to fetch good results on most of the instances, but these cannot be assumed to be optimal for all the instances.

We have used same test instances, viz. Solomon dataset [195] as used in [160] to evaluate the performance of the proposed approach. This dataset consists of a total of 56 instances with 100 customers in each. These instances are divided into six classes, namely, C1, R1, RC1, C2, R2 and RC2. These instances are designed by considering four factors: geographical data, the percentage of time constrained customers, quantity of customers which can be served by one vehicle, and tightness and positioning of the time window [160, 195]. On the basis of geographical data, three varieties of instances are generated, instances of C types contains clustered customers, R contains random customers and RC instances have a mix of random and clustered customers. With reference to quantity of customers served by each vehicle, in set 1 (C1, R1 and RC1) fewer customers are allowed to be served, whereas in set 2 (C2, R2 and RC2) more number of customers are served by each vehicle. The number of customers are controlled by the parameter capacity of vehicle. In each type of instances (R/C/RC), the positions of customers are same, but the width of the time window vary from a range of wider window to a very tight window.

As already stated, in *QSVRPTW*, the number of routes (*m*) is fixed. Expósito et al. [160] set the number of routes in their *QSVRPTW* instances to number of routes in optimal solution of classical VRPTW plus 15% rounded to the nearest integer. The *GRASP-VNS* approach of [160] was executed on only five instances by considering 25 and 50 customers. The results of these five instances are made available by the authors of [160] in the Google Drive [1]. The performance of *GGA-QOS* approach is evaluated by making a detailed comparison with *GRASP-VNS* approach presented in [160] on these instances.

Table 6.4 and Table 6.5 present the comparison of *GGA-QOS* and *GRASP-VNS* for 25 and 50 customers, respectively. In each Solomon instance the first customer is the depot, hence first 26 customers are considered in case of instance with 25 customers. In these tables, the first column reports the name of the instances. Each instance results are presented in three rows, where each row corresponds to a specific objective. The third and fourth column represent the bound 1 (B_1) and bound 2 (B_2) respectively. These two bounds for each objective have been discussed in Section 6.4. To be more specific, the B_1 and B_2 for objective $f_c(x)$ represents LB_1 and LB_2, for objective $f_s(x)$ represents UB_1 and UB_2. The first lower bound for objective $f_r(x)$ is zero, and hence, B_1 everywhere for this objective is zero and B_2 represents the lower bound (LB) value for this objective as discussed in Section 6.4. Last six columns reports the results of the two approaches. For each approach, the results are reported in terms of *Best*, *Avg*

[1] https://goo.gl/pXStrU

Table 6.4: Results on instances with number of customers=25.

Instances size = 25 Customers										
					GRASP-VNS			GGA-QOS		
Instance	**m**	**Objective**	B_1	B_2	**Best**	**Avg**	**Time**	**Best**	**Avg**	**Time**
		f_c(x)	7282.00	8863.00	9288.10	9419.63	163.00	9288.10	**9289.56**	1.27
C109	3	f_s(x)	9000.00	7419.00	8185.63	7661.88	186.00	6993.90	6992.44	1.27
		f_r(x)	0.000	0.175	0.011	0.152	93.00	0.223	0.223	1.72
		f_c(x)	39392.00	39467.00	39392.00	39392.00	24.00	39488.25	39488.25	1.69
C201	2	f_s(x)	4000.00	3925.00	3903.75	3903.75	25.00	**3903.75**	**3903.75**	1.69
		f_r(x)	0.000	0.018	0.101	0.113	32.00	**0.024**	**0.024**	1.79
		f_c(x)	1219.00	1329.00	1659.06	1817.38	120.00	1793.34	**1793.34**	0.73
R107	5	f_s(x)	2912.00	2802.00	2443.24	2293.06	170.00	2337.66	**2337.66**	0.73
		f_r(x)	0.000	0.021	0.063	0.159	79.00	0.161	0.161	1.32
		f_c(x)	1981.00	2031.00	2018.39	2018.39	40.56	2425.24	2425.24	0.52
RC106	3	f_s(x)	1500.00	1450.00	2688.17	2329.71	89.00	1055.76	1055.76	0.52
		f_r(x)	0.000	0.033	0.097	0.176	47.40	0.296	0.296	0.60
		f_c(x)	5374.00	5594.00	6776.53	6979.54	102.00	**6155.26**	**6155.26**	1.61
RC203	3	f_s(x)	13288.00	13068.00	12885.00	11167.00	126.00	12506.74	**12506.74**	1.61
		f_r(x)	0.000	0.009	0.033	0.106	124.00	0.034	**0.034**	1.74

and *Time*, where *Best* is the best solution found, *Avg* is the average solution quality and *Time* is the average execution time in seconds over 10 independent runs. In these tables, our results are reported in bold whenever our proposed approach performs better.

Table 6.5: Results on instances with number of customers=50.

Instances size = 50 Customers										
					GRASP-VNS			GGA-QOS		
Instance	**m**	**Objective**	B_1	B_2	**Best**	**Avg**	**Time**	**Best**	**Avg**	**Time**
		f_c(x)	14829.00	17712.00	18946.87	21543.34	284	**18541.7**	**18578.26**	9.50
C109	6	f_s(x)	18000.00	15117.00	17173.46	16752.5	271	14287.3	14250.74	9.50
		f_r(x)	0.000	0.160	0.011	0.1528	169	0.206	0.206	18.02
		f_c(x)	76458.00	76458.00	unknown	unknown	unknown	**76458.31**	**76458.31**	13.38
C201	3	f_s(x)	8000.00	8000.00	unknown	unknown	unknown	**7999.69**	**7999.69**	13.38
		f_r(x)	0.000	0.000	0.072	0.098	41	**0.000**	**0.000**	15.09
		f_c(x)	1988.00	2270.00	2947.34	3517.84	231	3608.55	3650.93	4.10
R107	8	f_s(x)	5631.00	5349.00	4652.78	4297.86	249	4010.45	3968.07	4.10
		f_r(x)	0.000	0.026	0.019	0.024	173	0.250	0.251	7.29
		f_c(x)	3818.00	3852.00	4561.67	4712.56	238	4657.51	**4664.49**	2.69
RC106	7	f_s(x)	3000.00	2966.00	2897.65	2706.27	177	2160.49	2153.51	2.69
		f_r(x)	0.000	0.011	0.012	0.14	93	0.280	0.282	2.87
		f_c(x)	8320.00	8857.00	10973.91	12203.7	239	**10282.08**	**10282.08**	9.24
RC203	5	f_s(x)	25690.00	25153.00	23789.72	21754.8	258	23727.92	**23727.92**	9.24
		f_r(x)	0.000	0.011	0.071	0.132	213	**0.050**	**0.050**	10.96

Table 6.6: C201 instance with 25 customers.

CUST NO.	XCOORD.	YCOORD.	DEMAND	EARLIEST TIME	LATEST TIME	UNLOADING TIME
0	40	50	0	0	3390	0
1	52	75	10	311	471	90
2	45	70	30	213	373	90
3	62	69	10	1167	1327	90
4	60	66	10	1261	1421	90
5	42	65	10	**25**	185	90
6	16	42	20	497	657	90
7	58	70	20	1073	1233	90
8	34	60	20	2887	3047	90
9	28	70	10	2601	2761	90
10	35	66	10	2791	2951	90
11	35	69	10	2698	2858	90
12	25	85	20	2119	2279	90
13	22	75	30	2405	2565	90
14	22	85	10	2026	2186	90
15	20	80	40	2216	2376	90
16	20	85	40	1934	2094	90
17	18	75	20	2311	2471	90
18	15	75	20	1742	1902	90
19	15	80	10	1837	1997	90
20	30	50	10	**10**	170	90
21	30	56	20	2983	3143	90
22	28	52	20	**22**	182	90
23	14	66	10	1643	1803	90
24	25	50	10	116	276	90
25	22	66	40	2504	2664	90

Table 6.6 represents the screen-shot image of *C201* instance with 25 customers. For this instance, the number of routes (m) are 2. Hence, any solution for this instance will have exactly two routes. The minimum value of objective $f_c(x)$ can be the sum of earliest time of all the customers, as discussed in Section 6.4. B_1 is the sum of earliest time of all customers, which is 39392.00 for this instance. In [160], authors claimed that *GRASP-VNS* yields a solution for this instance with the objective $f_c(x)$ is 39392.00, which is not possible. In Table 6.6, there are three customers, viz. customer 5, 20 and 22 whose earliest time is less than the minimum unloading time (90.00) and since, the number of routes are only two, so definitely one of these three customers will be served at second position in one of the route. Thus, vehicle will reach to this customer after serving first customer, hence after the unloading time (90.00). Thus the vehicle arrival time for this customer cannot be its earliest time. Hence, we cannot get value (39392.00) for the objective $f_c(x)$ in case of instance *C201* with 25 customers. We found mistakes in the results obtained by *GRASP-VNS* for some other instances too. These results violate the bounds computed by us and are reported with underline in both the tables. Thus, comparison of the results of *GRASP-VNS* with *GGA-QOS* is not meaningful at least for such

instances (this raises doubt about the correctness of results on other instances too). Also, the closeness of obtained results of *GGA-QOS* with bounds reveals the effectiveness of *GGA-QOS* approach for *QSVRPTW* problem. The result of *GRASP-VNS* for objectives $f_c(x)$ and $f_s(x)$ are not reported for *C201* instance with 50 customers. The instance *C201* contains *m* value equal to the number of routes in its optimal solution, hence it is possible that *GRASP-VNS* might not have found any feasible solution for any of these objectives. Our approach is able to find solutions for all Solomon instances, which also adds as an advantage of *GGA-QOS* approach for *QSVRPTW* problem. As far as comparison of execution times are concerned, *GRASP-VNS* were executed on a PC equipped with 3.16 GHz Intel Core 2 Duo processor and 4 GB RAM which is different from the system used to execute *GGA-QOS*, hence it is not possible to compare the execution times precisely. However, rough comparison can always be made based on information available in public domain about the relative speeds of two processors, which indicates our system to be around 2 times faster. Even after compensating for this difference in processing speeds, we can safely say that the proposed *GGA-QOS* approach is faster than *GRASP-VNS*.

We have executed our approach on additional larger Solomon instances with all the customers, i.e., 100 customers. Following Expósito et al. [160] we have considered only those instances for which optimal solution for VRPTW is known for *n*=100. Out of 56 instances, the optimal solution is known for 37 instances and these solutions are available online [1]. The number of routes in these instances are also set equal to number of routes in optimal solution of classical VRPTW plus 15% rounded to the nearest integer like Expósito et al. [160].

Table 6.7 and Table 6.8 reports the results obtained by *GGA-QOS* for all 37 Solomon instances. As already mentioned in Section 6.5, there are two versions of *GGA-QOS*, the former is developed to address $f_c(x)$ and $f_s(x)$ objectives, while the latter one is for $f_r(x)$ objective. The results of *GGA-QOS* for $f_c(x)$ and $f_s(x)$ are reported in Table 6.7, whereas results for $f_r(x)$ are reported in Table 6.8. In Table 6.7, the first column represents the name of instance. The remaining four columns provides the values of two bounds, best solution and average solution quality for each objective, respectively. The last column presents the average execution time in seconds. In Table 6.8, the first column represents the instance, second column provides the number of routes in optimal solution of the corresponding VRPTW instance and the third column is the number of routes (*m*) calculated as number of routes in optimal solution plus 15% rounded to the nearest integer. The last four columns provides the lower bound value, best solution, average quality solution and the average execution time in seconds, respectively, for

[1] `http://web.cba.neu.edu/~msolomon/problems.htm`

each instance. The first lower bound for $f_r(x)$ is zero for all instances, hence it is not reported in Table 6.8.

Table 6.7: Results of GGA-QOS for objectives $f_c(x)$ and $f_s(x)$ respectively.

	$f_c(x)$				$f_s(x)$				
Instance	LB_1	LB_2	**Best**	**Avg**	UB_1	UB_2	**Best**	**Avg**	**Time**
C101	42680	42896	43517.93	43640.04	6076	5860	5238.07	5115.96	10.56
C102	30083	35499	38536.86	38976.59	32569	27153	24115.14	23675.41	29.68
C103	21537	34485	37693.38	37899.46	58849	45901	42692.62	42486.54	35.09
C104	10258	33413	36994.25	37125.15	85294	62139	58557.75	58426.85	78.45
C105	39916	40647	41494.29	41621.78	12161	11430	10582.71	10455.22	28.16
C106	38374	39354	40865.69	41094.36	15615	14635	13123.31	12894.64	25.82
C107	37298	38563	39398.06	39490.65	18000	16735	15899.94	15807.35	30.93
C108	34722	36943	38735.59	38869.18	24328	22107	20314.41	20180.82	36.48
C109	29833	35342	37388.12	37537.03	36000	30491	28444.88	28295.97	44.18
C201	147016	148965	154679.88	154690.27	16000	14051	8336.12	8325.73	84.64
C202	112008	145939	154679.88	155728.80	93774	59843	51102.12	50053.20	324.72
C203	73647	145685	154659.93	154927.02	171482	99444	90469.07	90201.98	1770.00
C204	39959	145684	154653.24	155910.50	249258	143533	134563.76	133306.50	511.74
C205	139323	146689	154621.35	154621.35	32000	24634	16701.65	16701.65	82.37
C206	131290	146528	154621.35	154623.73	48664	33426	25332.65	25330.27	371.19
C207	124373	146528	154612.80	154615.17	61232	39077	30992.20	30989.83	195.75
C208	124264	146528	155709.05	155812.92	64000	41736	32554.95	32451.08	104.48
R101	9648	9648	9666.68	9667.59	1000	1000	981.32	980.41	18.29
R102	7503	7543	8120.35	8127.89	5739	5699	5121.65	5114.11	23.28
R103	5063	5615	7494.01	7585.22	10299	9747	7867.99	7776.78	37.50
R104	2615	4447	7390.93	7425.51	14831	12999	10055.07	10020.49	37.96
R105	8673	8673	8922.19	8939.16	3000	3000	2750.81	2733.84	31.02
R106	6769	6869	7943.20	7985.49	7239	7139	6064.80	6022.51	28.78
R107	4559	5339	7893.94	8003.69	11299	10519	7964.06	7854.31	27.39
R109	7301	7301	8094.81	8152.65	5889	5889	5095.19	5037.35	26.64
R110	6018	6027	7603.75	7687.16	8650	8641	7064.25	6980.84	32.58
R111	6135	6147	7635.76	7686.08	9310	9298	7809.24	7758.92	39.83
R201	39121	39121	39164.75	39164.75	11596	11596	11552.25	11552.25	72.77
RC101	9182	9182	9595.73	9637.24	3000	3000	2586.27	2544.76	208.43
RC102	7113	7203	8547.58	8572.31	7146	7056	5711.42	5686.69	31.17
RC103	4788	5556	8641.01	8732.35	11250	10482	7396.99	7305.65	21.25
RC105	8259	8259	8797.74	8837.93	5433	5433	4894.26	4854.07	24.38
RC107	6412	6494	8336.66	8408.92	8821	8739	6896.34	6824.08	20.28
RC108	5509	5858	8329.06	8398.78	11233	10884	8412.94	8343.22	23.16
RC201	37118	37118	37134.33	37134.33	12000	12000	11983.67	11983.67	63.62
RC202	28979	29224	30372.60	30375.62	31896	31651	30502.40	30499.38	64.40
RC205	32796	32797	33080.57	33080.57	22306	22305	22021.43	22021.43	72.14

The *GRASP-VNS* approach is tested only on five instances and that too for only upto 50 customers. Our proposed approach is tested on all 37 Solomon instances (with known optimal solution when solved as classical VRPTW) and that too with maximum size, i.e., 100 customers, thus *GGA-QOS* approach can be considered as first approach suitable for all instances for *QSVRPTW* problem. The execution time is very less as compared to *GRASP-VNS* approach, thus our approach can find a better quality solution in shorter running time on most of the

instances. On some instances, it is found that finding a feasible solution is very tough due to the presence of tight constraints and in such cases *Feasible_Sol* heuristic is used. These situation occurs particularly for the instances where number of routes in optimal solution is equal to *m*, such as in C2 types instances. Thus, sometimes finding a feasible solution in initial population itself is very hard and *Feasible_Sol Heuristic* tries to find a feasible solution by making several attempts, and hence, *GGA-QOS* approach consumes relatively large time on such instances. For example, the execution time for *C203* is more in comparison to other instances.

Table 6.8: Results of GGA-QOS for objectives $f_r(x)$.

Instance	OptVeh	m	$f_r(x)$			
			LB	Best	Avg	Time
C101	10	12	0.024	0.147	0.161	8.21
C102	10	12	0.047	0.163	0.171	35.88
C103	10	12	0.113	0.195	0.207	46.58
C104	10	12	0.203	0.273	0.278	98.04
C105	10	12	0.041	0.138	0.149	33.29
C106	10	12	0.025	0.151	0.169	33.76
C107	10	12	0.070	0.117	0.125	71.54
C108	10	12	0.062	0.155	0.167	92.37
C109	10	12	0.153	0.212	0.218	131.65
C201	3	3	0.121	0.479	0.480	85.70
C202	3	3	0.103	0.160	0.169	349.52
C203	3	3	0.218	0.253	0.257	1807.04
C204	3	3	0.321	0.355	0.361	593.43
C205	3	3	0.230	0.478	0.478	85.57
C206	3	3	0.215	0.473	0.473	403.44
C207	3	3	0.176	0.391	0.394	208.87
C208	3	3	0.347	0.491	0.492	100.42
R101	20	23	0.000	0.019	0.020	12.55
R102	18	21	0.001	0.045	0.046	25.52
R103	14	16	0.025	0.162	0.167	29.89
R104	11	13	0.085	0.272	0.279	64.81
R105	15	17	0.000	0.081	0.089	25.97
R106	13	15	0.004	0.147	0.156	37.91
R107	11	13	0.036	0.254	0.262	38.40
R109	13	15	0.000	0.127	0.145	37.40
R110	12	14	0.000	0.184	0.192	39.89
R111	12	14	0.000	0.166	0.174	49.64
R201	8	9	0.000	0.003	0.003	79.79
RC101	15	17	0.000	0.138	0.151	154.03
RC102	14	16	0.004	0.152	0.158	21.84
RC103	11	13	0.034	0.274	0.279	23.84
RC105	15	17	0.000	0.089	0.093	26.09
RC107	12	14	0.005	0.217	0.226	25.92
RC108	11	13	0.019	0.254	0.262	30.95
RC201	9	10	0.000	0.001	0.001	70.09
RC202	8	9	0.002	0.019	0.019	77.50
RC205	7	8	0.000	0.010	0.010	95.10

6.7 Conclusions

In this chapter, we have proposed a steady-state grouping genetic algorithm based approach called *GGA-QOS* for *QSVRPTW*. The crossover and mutation operators used in our approach are developed by exploiting the characteristics of QSVRPTW as well as of the objectives. We presented two bounds for each objective, which provide an assessment of the quality of solutions obtained by an approach. The computational results signify the superiority of the proposed *GGA-QOS* approach over the state-of-the-art approach for *QSVRPTW*. The computational results show that our perform well both in terms of quality of solution as well as the execution time. We have devised several heuristics which is able to provide feasible solution for tougher instances also. This is another advantage of our proposed approach.

Chapter 7

Multiobjective Vehicle Routing Problem with Time Windows

7.1 Introduction

The vehicle routing problem with time windows (VRPTW) intends to find the assignments and routing of vehicles under the given constraints so that the cost is optimized. In VRPTW, the cost can be number of vehicles/routes, total distance traveled by all vehicles, total travel time, etc. Several previous studies considered VRPTW as a single objective problem. For example, [177] and [176] presented approaches based on simulated annealing and genetic algorithm respectively to minimize the total travel distance as the sole objective. Literature also contains several research studies in which VRPTW is addressed as a multiobjective problem. Often, two objectives, viz. minimization of total travel distance and minimization of number of routes are considered. In most of such problem variants, multiple objectives are transformed into a single-objective by using scalar techniques like the one presented in [173].

The multiobjective VRPTW, referred to as *MOVRPTW* in the literature, may have objectives so that the optimization of one objective may lead to deterioration in the values of other objectives. Further, relative importance of different objectives may vary from one domain to another or from one scenario to another. Therefore, instead of finding a single solution for VRPTW, it is desirable to present a set of solutions having trade-offs among different objectives, as the preference of decision maker may not be known a priori [196]. Thus, VRPTW is primarily a multiobjective optimization problem (MOP). However, there exist only a few works in the literature which address multiobjective VRPTW. The main characteristics of multiobjective

approaches is to handle all the objectives with the same preference, and hence, these approaches should generate a set of Pareto optimal solutions. [196], [197], [198], [199] and [200] report approaches which considered VRPTW as a bi-objective problem. However, these approaches were tested on the widely used Solomon's problem instances [195]. The Solomon's problem instances [195] were originally proposed for single-objective VRPTW, i.e., minimizing the total travel distance as the sole objective. [201] and [202] analyzed the conflicting nature of various objectives in Solomon's instances [195], and concluded that the Solomon's instances do not represent the ideal benchmark scenarios for multiobjective VRPTW due to the following reasons [201]:

1. Dependence relationships between objectives are weak in Solomon's instances.
2. The data in Solomon's instances are based on Euclidean distance for both travel distance and travel time, which does not represent a realistic scenario with respect to real-world applications.

This chapter deals with a general *MOVRPTW* with five objectives that are used commonly in various multiobjective VRPTW variants in the literature. This problem was presented by Castro-Gutierrez *et al.* in [201] and the five objectives are as follows:

1. Minimizing the number of vehicles,
2. Minimizing the total travel distance by all the vehicles,
3. Minimizing the makespan, i.e., longest travel time among all routes (from/to depot),
4. Minimizing the total waiting time due to early arrivals
5. Minimizing the total delay time due to late arrivals

The significance of these objectives can vary from domain to domain. For example, with reference to food delivery and healthcare industries, delay time is of utmost importance. Goods transportation industry may consider total distance traveled as a crucial objective to minimize than other objectives as consumption of fuel is directly proportional to distance traveled, and minimizing the total distance traveled by all the vehicles is important from the perspective of monetary considerations. For small scale industries, minimization of number of vehicle may be of highest priority in comparison to other objectives. Castro-Gutierrez *et al.* [201] also presented

a set of *MOVRPTW* instances utilizing the data from a distribution company in Tenerife, Spain. These *MOVRPTW* instances are more realistic as well as truly multiobjective in nature as far as *MOVRPTW* is concerned. In [201], a nondominated sorting genetic algorithm II (NSGA-II) [203] based approach is presented which is also the first approach using the new *MOVRPTW* instances. These instances are publicly available for download [204].

The NSGA-II approach presented in [201] uses generic crossover and mutation operators to generate offspring. Generic operators do not exploit the structure of the problem and the characteristics of the objectives, and therefore may fail to yield high quality solutions. A new local search-based multiobjective algorithm (LSMOVRPTW) is presented in [193], which is the second approach for *MOVRPTW* in the literature. In LSMOVRPTW approach, for each objective, one local search procedure is devised by considering the problem structure of *MOVRPTW* with corresponding objective. First, a solution is randomly selected from the archive and then all local search procedures are applied iteratively. The newly generated solutions are updated in the archive by using ϵ-dominance concept [205, 206]. Computational results and their analysis presented in [193] show that LSMOVRPTW performed better than NSGA-II approach presented in [201].

MOVRPTW is a relatively understudied problem as the only approaches mentioned in the previous paragraph are available in the literature. The LSMOVRPTW uses local search approach to generate new solutions. Usually, local search approaches are considered as exploitative in nature and lacks the exploration characteristics. NSGA-II algorithm is a Pareto-based multiobjective optimization technique that is among most successful techniques in the literature for addressing multiobjective problems. Despite these facts, LSMOVRPTW is able to beat NSGA-II approach of [201] quite comfortably due to the generic nature of variation operators (crossover and mutation) used in the latter approach. These generic variation operators do not make use of characteristics specific to *MOVRPTW* and its various objectives, and, hence lack appropriate exploitation characteristics. As a result, this genetic algorithm yields inferior solutions in comparison to LSMOVRPTW. This stressed the need for the variation operators appropriately utilizing the characteristics of *MOVRPTW* and its various objectives. This has motivated us to develop such variation operators and a new approach based on NSGA-II utilizing such variation operators. As our crossover and mutation operators are designed as per the characteristics of *MOVRPTW* and the characteristics of each objective, we have a dedicated crossover operator and a dedicated mutation operator for each objective. Hence, our approach uses five different crossover operators and five different mutation operators. Another key feature

of our approach is that for the last 25% generations, crossover is not used and only mutation is used. This is due to the fact that our mutation operators have better exploitation ability than our crossover operators. Each mutation operator at these final iterations acts as a local search to some extent for its respective objective and improves the solution quality further in the population with respect to its objective. In addition to these two key features, our new NSGA-II approach differs from the NSGA-II approach of [201] in the manner in which initial population is generated and parents are selected for crossover and mutation. NSGA-II approach of [201] uses a constructive method for initial population generation that aims at satisfying first the customers farthest from the depot. On the other hand, we have used a method similar to the method used by [193] to generate an initial solution. NSGA-II approach of [201] uses deterministic binary tournament selection, whereas our method has used probabilistic binary tournament selection. Computational results on benchmark instances from the literature demonstrate that our approach is capable of producing superior solutions in comparison to LSMOVRPTW.

The rest of this chapter is organized as follows: Section 7.2 presents the problem formulation of the *MOVRPTW*. In Section 7.3, we present an overview of existing approaches for *MOVRPTW* and the multiobjective optimization. The proposed NSGA-II approach for *MOVRPTW* is presented in Section 7.4. Section 7.5 presents the computational results and their analysis. Finally, Section 7.6 concludes the chapter.

7.2 Problem definition

This section introduces the notational conventions used and provides a formal definition of *MOVRPTW*. We have used the same formulation and notational conventions as presented in [193]. VRPTW contains a set of vertices $v = \{0, \ldots, N\}$, where vertex 0 corresponds to depot and remaining vertices represent customers. The fleet of vehicles depart from the depot in order to serve the customers, and each vehicle has a capacity C. Each customer i is associated with a demand of goods $g_i > 0$ and a time window $[b_i, e_i]$. The time window constraint represents that the customer i must be served in-between and including the time b_i and e_i. In other words, e_i is the latest service time such that the vehicle should arrive at customer i's location. If vehicle arrives at customer i's location before the earliest service time b_i, then vehicle must wait until b_i in order to serve the customer. This situation leads to waiting of the vehicle, and, waiting time will be the difference in time between the earliest service time b_i and the arrival time. Each customer i has a service time s_i, which is the time taken in delivery of goods after arrival of

vehicle at customer's location. The time window constraint for depot is $[0, e_0]$, i.e., vehicles start from depot at time 0 and must return to depot on or before time e_0. It is considered that the demand of goods for depot is $g_0 = 0$. Here, d_{ij} and t_{ij} represent the travel distance and travel time between vertices *i* and *j* respectively. The aim of the *MOVRPTW* is to find the set of M routes $R = \{r_1, \ldots, r_M\}$ with the minimum cost, such that each customer is served exactly once by some vehicle and each vehicle should start at time 0 & return to the depot on or before time e_0. Let r_j represents the j^{th} route such that, r_j=$\langle c(1, j), \ldots, c(N_j, j)\rangle$ is the sequence of customers served in j^{th} route which is having N_j customers and $c(i, j)$ specifies some customer visited at i^{th} position in this route. Furthermore, the depot is represented as $c(0, j) = c(N_j + 1, j) = 0$.

The total travel distance of the j^{th} route is defined as

$$Dist_j = \sum_{i=0}^{N_j} d_{c(i,j)c(i+1,j)}. \tag{7.1}$$

Here, $i = 0 = N_j + 1$ represents the depot and $d_{c(i,j)c(i+1,j)}$ provides the distance between customer i and customer $i + 1$.

To find the total travel time, we need to define arrival time and leaving time of a vehicle at a customer location. Let $a_{c(i,j)}$ be the arrival time and $l_{c(i,j)}$ be the leaving time of vehicle j at the i^{th} customer $c(i, j)$ in this route. It is pertinent to mention that vehicle j represents the j^{th} route.

The arrival time is defined as

$$a_{c(i,j)} = \ l_{c(i-1,j)} + t_{c(i-1,j)c(i,j)}. \tag{7.2}$$

The leaving time of vehicle j from depot is considered as 0, i.e., $l_{c(0,j)} = 0$ and $t_{c(i-1,j)c(i,j)}$ represents the travel time between customer $i - 1$ and i.

If a vehicle arrives early at a customer location then the vehicle needs to wait until the earliest service time of this customer. This scenario incurs a waiting time for the vehicle. The waiting time of vehicle j at i^{th} customer $c(i, j)$ can be computed as

$$w_{c(i,j)} = \begin{cases} 0 & \text{if } a_{c(i,j)} \geq b_{c(i,j)} \\ b_{c(i,j)} - a_{c(i,j)}, & otherwise. \end{cases} \tag{7.3}$$

and the leaving time of vehicle j from i^{th} customer $c(i,j)$ is

$$l_{c(i,j)} = \ a_{c(i,j)} + w_{c(i,j)} + s_{c(i,j)}. \tag{7.4}$$

Hence, the total travel time of the route r_j can be computed as follows

$$T_j = \ \sum_{i=0}^{N_j} (t_{c(i,j)c(i+1,j)} + w_{c(i+1,j)} + s_{c(i+1,j)}). \tag{7.5}$$

The $s_{c(i+1,j)}$ presents the service time of $(i+1)^{th}$ customer in the j^{th} route. Here, waiting time and service time of depot is zero, i.e., $w_{c(N_j+1,j)} = 0$ and $s_{c(N_j+1,j)} = 0$. This is due to the fact there is no service requirement at the depot.

Using above notations, the total waiting time of this route is

$$W_j = \ \sum_{i=1}^{N_j} w_{c(i,j)}. \tag{7.6}$$

There are two versions of VRP in the context of latest service time. The first version with *hard time windows* [196, 197] does not allow the vehicle to arrive at a customer location after latest service time of that customer. This version of VRP is not very realistic as the travel time can vary a lot depending on external factors like road and traffic conditions which can be chaotic and unpredictable [207, 208]. The second version with *soft time windows* provides a relaxation to latest service time and allows a small time window violation, since it can not be considered as a critical breach of service requirements in most practical applications. Literature contains so many practical applications of VRP with soft time windows, such as in [209, 210, 211, 212]. It is observed that relaxing the requirements of hard time windows can yield lower cost solutions requiring fewer vehicles, shorter travel distance and less travel time. Furthermore, VRP with soft time windows can be considered as a generalization of VRP with hard time windows [209, 210]. The *MOVRPTW* considered in this chapter addresses VRP with soft time windows, i.e., a vehicle is allowed to arrive late within a certain time after the latest service time, also known as maximum allowed time in the literature. The late arrival causes a delay time for the vehicle. Let md denotes the maximum allowed time that a vehicle is permitted to arrive after the end of time window.

The delay time of vehicle j at i^{th} customer of the route is

$$delay_{c(i,j)} = \begin{cases} 0 & \text{if } a_{c(i,j)} \leq e_{c(i,j)} \\ a_{c(i,j)} - e_{c(i,j)}, & otherwise. \end{cases} \tag{7.7}$$

Hence, the total delay time of this route is

$$Delay_j = \sum_{i=1}^{N_j} delay_{c(i,j)}. \tag{7.8}$$

It is pertinent to mention that, the early arrival of vehicle at a customer's location results in waiting time and late arrival of vehicle incurs the delay time, in the route covered by this vehicle.

The five objectives for *MOVRPTW*, each of which needs to be minimized, can be stated as follows:

f_1 (number of vehicles):

$$\text{Minimize } f_1 = |R| = M; \tag{7.9}$$

f_2 (total travel distance):

$$\text{Minimize } f_2 = \sum_{j=1}^{M} Dist_j; \tag{7.10}$$

f_3 (makespan, i.e., longest travel time among all routes (from/to depot)):

$$\text{Minimize } f_3 = \max\{T_j | j = 1, \ldots, M\}; \tag{7.11}$$

f_4 (total waiting time due to early arrivals):

$$\text{Minimize } f_4 = \sum_{j=1}^{M} W_j; \tag{7.12}$$

f_5 (total delay time due to late arrivals):

$$\text{Minimize } f_5 = \sum_{j=1}^{M} Delay_j; \tag{7.13}$$

The objectives f_1,f_2 and f_3 can be visualized as transportation costs. Objective f_3 is also concerned about balancing the drivers' workloads. On the other hand, f_4 and f_5 can be considered as service costs representing the customer satisfaction [207].

The constraints associated with *MOVRPTW* are stated below:

1. *Demand constraint:* The total demand of each route r_j must be less than or equal to the capacity of the vehicle, i.e.,

$$\sum_{i=1}^{N_j} g_{c(i,j)} \ \leq \ C \quad \forall j \in \{1, 2, \ldots, M\} \tag{7.14}$$

2. *Travel time constraint:* The delay time for all vehicles at any customer location must be less than or equal to the maximum allowed delay time, i.e.,

$$delay_{c(i,j)} \ \leq \ md \quad \forall i \in \{1, 2, \ldots, N\} \quad \forall j \in \{1, 2, \ldots, M\} \tag{7.15}$$

3. *Return time constraint:* All vehicles must return to depot before the closing time, i.e.,

$$a_{c(N_j+1,j)} \ \leq \ e_{c(N_j+1,j)} \quad \forall j \in \{1, 2, \ldots, M\} \tag{7.16}$$

Thus, we can summarize the *MOVRPTW* with five objectives addressed in this paper as:

$$\min f \ = \ \{f_1, f_2, f_3, f_4, f_5\} \tag{7.17}$$

subject to constraint conditions defined by 7.14–7.16.

7.3 Related work

In this section, we delve into the the previously proposed approaches for addressing *MOVRPTW*. Furthermore, we provide a brief description about the basic concepts of multiobjective optimization problems (MOPs) for better understanding the terminology used in the proposed approach.

7.3.1 Existing approaches for *MOVRPTW*

As already mentioned in Section 7.1, *MOVRPTW* with five objectives was introduced by Castro-Gutierrez *et al.* [201]. At the same time, it was observed that no problem instance

exists that model real-world characteristics. Most of the existing datasets for *MOVRPTW* either do not consider real-world characteristics or are derived from the classic single objective datasets. In [201], authors proposed a real-world problem instances derived from the data of a distribution company located in Tenerife, Spain. The *MOVRPTW* instances presented in [201] are considered as ideal benchmark instances for *MOVRPTW*. These datasets are having real-world characteristics, for example in real-world the travel distance and travel time are often distinct and asymmetric. One can easily notice the fact that the travel time in urban areas and in rural areas has a significant difference for the same distance. Similarly, travel time from location *A* to *B* need not be same as travel time taken in traveling from *B* to *A*. Furthermore, [201] considers the triangle inequality violations for both travel distance and travel time, i.e., $Dist_{ik} \geq Dist_{ij} + Dist_{jk}$, which are highly prevalent in real-world scenarios. [213] and [214] have studied the effects of asymmetry and triangle inequality violations on realistic VRP. The above mentioned points show that the real-world *MOVRPTW* instances presented in [201] truly reflect multiobjective nature, and hence, are more desirable to evaluate the performance of multiobjective optimization approaches.

Castro-Gutierrez *et al.* [201] used NSGA-II approach on newly proposed *MOVRPTW* instances and on well-known Solomon's problem instances [195]. In NSGA-II approach, the population initialization uses a constructive method in which the customers farthest from depot are serviced first. It used three crossover operators, viz. one-point crossover, edge crossover and generic crossover. It used four kinds of mutation operators, viz. swap, insertion, inversion and displacement. The goal of using NSGA-II approach was to show that the newly proposed instances have truly multiobjective characteristics than commonly used Solomon's problem instances. The authors conducted a pair-wise comparison between all combinations of objectives. [201] concluded that the proposed *MOVRPTW* instances have better dependency relationships in pair-wise comparisons of the objectives, whereas dependence relationships between objectives are weak in Solomon's instances. Hence, the proposed *MOVRPTW* instances represent ideal benchmark scenarios for multiobjective VRPTW.

To address *MOVRPTW*, Zhou and Wang [193] presented a new local search-based multiobjective algorithm (LSMOVRPTW). In LSMOVRPTW approach, one local search procedure is devised for each objective. Initially an archive of solutions is created in an iterative manner. Five local searches each corresponding to one objective are applied iteratively on a randomly generated solution. The local search for f_1 picks the route with fewest customers and try to insert these customers in some other routes where the constraints remain satisfied. The local

search procedures for f_2, f_3, f_4 and f_5 use three neighborhood operators. A solution is selected randomly from the archive and all objective-wise local searches are applied on this solution in an iterative manner. Each objective-wise local search is used ten times in a continuous manner, by randomly selecting a neighborhood operator each time. The newly generated solutions are updated in the archive by using concept of ϵ-dominance [205, 206].

The performance of LSMOVRPTW is compared with the NSGA-II approach presented in [201]. Zhou and Wang [193] re-executed this NSGA-II approach in their computing environment, as the source code of NSGA-II was available for download [204]. Since LSMOVRPTW uses a series of local searches and there is no explicit concept of generation in this approach, hence the running time of NSGA-II is used as stopping criteria for LSMOVRPTW approach. The comparison of results in [193] reveals that the LSMOVRPTW approach outperformed the NSGA-II approach in almost all the instances.

The NSGA-II approach of [201] uses generic crossover and mutation operators to produce offspring. These generic operators do not consider the structure of the problem and the characteristics of the objectives while generating offspring, and thus lack proper exploitation characteristics. This adversely affected the quality of final solutions obtained through this approach. The LSMOVRPTW presented in [193] uses objective-wise local search. In LSMOVRPTW approach, for each objective, one local search procedure is devised by considering the problem structure of MOVRPTW with corresponding objective. Furthermore, local search approaches are considered as exploitative in nature and lacks the exploration characteristics. In spite of this drawback of LSMOVRPTW and the fact that NSGA-II is among the most successful multiobjective optimization techniques, NSGA-II approach of [201] performed much worse than LSMOVRPTW. This created the need for variation operators that can utilize the structure of MOVRPTW and characteristics of its various objectives in a suitable manner. Such variation operators facilitates better exploitation thereby aiding the genetic algorithm in maintaining an adequate balance between exploration and exploitation which is necessary for finding high quality solutions. To bridge this research gap, we have developed a new approach based on NSGA-II for MOVRPTW with desired variation operators. As our variation operators are designed considering the structure of MOVRPTW and characteristics of its various objectives, an exclusive crossover operator and an exclusive mutation operator are designed for each objective leading to a total of five different crossover operators and five different mutation operators. Our proposed NSGA-II approach is able to maintain an adequate balance between exploration and exploitation owing to these variation operators.

7.3.2 Overview of multiobjective optimization

A multiobjective optimization problem (MOP) can be modeled as:

$$\min F(x) \; = \; \{f_1(x), f_2(x), \ldots, f_m(x)\} \; x \in \Omega \tag{7.18}$$

Where Ω denotes the solution space, x is the solution and $f_i(x)$ is the i^{th} objective function. The number of objectives is m, and in most of the cases these objectives conflict one another, i.e., improvement in value of one objective results in deterioration in values of some others objectives. A solution x dominates another solution y (represented as $x \prec y$), iff $f_i(x) \leq f_i(y) \; \forall \; i \in \{1, 2, \ldots, m\}$, and $f_i(x) < f_i(y) \; \exists \; i \in \{1, 2, \ldots, m\}$. In other words, x dominates y, iff no component of x is larger than the corresponding component of y and at least one component is smaller [215]. A solution x^* is said to be *Pareto optimal* if it is not dominated by any other solution $x \in \Omega$. The set of all Pareto optimal solutions constitute the *Pareto set* and the set of their corresponding objective vectors is known as *Pareto front* [216]. Any multiobjective algorithm for an MOP intends to find a set of nondominated solutions which perform well in terms of convergence and diversity. That is, a multiobjective algorithm considered as a better approach, if it provides a set of nondominated solutions close to and widely distributed along the Pareto front. Literature contains numerous multiobjective evolutionary algorithms (MOEAs) to address various MOPs, such as Strength Pareto Evolutionary algorithm 2 (SPEA2) [217], Multiobjective Evolutionary Algorithm based on Decomposition(MOEA/D) [218], Multiobjective Particle Swarm Optimization (MOPSO) [219] , Generalized Differential Evolution 3 (GDE3) [220], Pareto Archived Evolution Strategy (PAES) [221], NSGA-II [203]. So far the most popular MOEA in literature is NSGA-II [222]. The reader is referred to [222] and [223] for a comprehensive survey on various multiobjective evolutionary algorithms.

7.4 Proposed NSGA-II approach for *MOVRPTW*

This chapter presents a nondominated sorting genetic algorithm II (NSGA-II) [203] approach with objective-specific variation operators to address the *MOVRPTW*. Hereafter, our proposed approach will be referred to as *INSGA-II*. Literature reveals that NSGA-II is most commonly used algorithm to address the multiobjective optimization problems due to its salient features which include fast nondomination sorting procedure, fast crowded distance estimation procedure and simple crowded comparison operator. NSGA-II was proposed by Deb *et al.* [203] and the

simulation results of NSGA-II on various benchmark instances show its superiority over other multiobjective optimization techniques [224].

Nondomination sorting procedure iteratively sorts the population into different nondomination levels. First each solution is compared with other solutions. The solutions not dominated by any other solutions in population are considered as nondominated solutions and they constitute the first nondominated front. There must be some solutions which are dominated by only the solutions in first nondominated front. Hence, if we temporarily remove the solutions in the first front, then we will find the solutions in second nondominated front. Thus the solutions in first front are discounted temporarily, and the above procedure is repeated to find the second nondominated front. The same process continues until all solutions in population are divided into different nondomination fronts. The above mentioned procedure has the overall complexity of $\mathcal{O}(MN^3)$, where M is the number of objectives and N is the size of the population. In NSGA-II [203], a fast nondomination sorting procedure is proposed which has the complexity of $\mathcal{O}(MN^2)$. The idea is to compute two entities for each solution ρ: (1) domination count n_ρ, i.e., the number of solutions which dominate the solution ρ and (2) S_ρ, a set of solutions dominated by the solution ρ. The computing of these two entities require $\mathcal{O}(MN^2)$ comparisons. The solutions with domination count as zero will constitute the first nondominated front. In order to find the second nondominated front, the solutions in first front are discounted temporarily. Now, for each solution ρ in first nondominated front, the corresponding domination count of each member of set S_ρ is reduced by one. The solutions which were only dominated by the solutions in first nondominated front, now must have their domination count as zero and these solutions will be selected as second nondominated front. The same process is repeated until all fronts are identified [203].

In the original NSGA [225], the sharing function approach was used to maintain diversity in the population. There are two difficulties in using sharing function. It requires a sharing parameter value set by user, and secondly, the complexity of sharing function is $\mathcal{O}(N^2)$ due to involvement of comparison of each solution with all others solutions in the population. To maintain diversity in the population, [203] proposed the fast crowded distance estimation procedure in NSGA-II. The crowding distance provides the density of solutions surrounding a particular solution in a nondominated set. The solutions in a nondominated front are first sorted according to each objective function value in an ascending order of magnitude. Then, for each objective, the boundary solutions (solutions with smallest and largest values for that objective) are given an infinite distance value. All other intermediate solutions are assigned

a distance value equal to the absolute normalized difference in the corresponding objective function values of two adjacent solutions. This computation is done for each objective. The overall crowding-distance value is the sum of individual distance values corresponding to each objective. The value of each objective is normalized before computing the crowding distance. The overall complexity of crowded distance estimation is $\mathcal{O}(MN\ logN)$ [203].

In this work, we have used the NSGA-II framework with objective-wise crossover and mutation operators, which consider the specific requirements of each objective. The process begins with initial population generation. The probabilistic binary tournament selection (BTS) procedure has been used to select two parents for crossover. In this procedure, two solutions are selected randomly from the parent population. The better solution is selected to be a parent with probability p_{bt}, otherwise worse one is selected. Here a solution is considered better than the other if it dominates the other one, and in case if no one dominates each other, then the solution having more crowding distance is considered as the better solution.

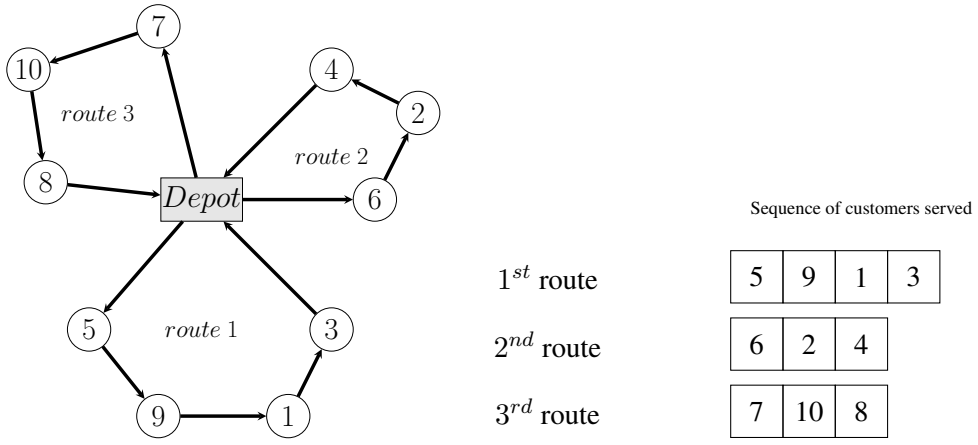
For each generation, we have used an iterative process which is repeated for 50 iterations. In each iteration, the BTS is used to select two parents from the parent population. The selected parents are passed as input to crossover operator, which produces a child solution. The mutation operator is applied on this child solution and a mutant solution is generated. The child solution and mutant are compared and the better one is added in offspring population. There are two possibilities of being a better solution. First, the solution which dominates other is considered better. The other possibility, if no one dominates each other, then solution which has less objective value for the specific crossover/mutation operator is considered better. It is pertinent to mention that the newly generated solution is added only if it is unique. To check uniqueness, we have compared the objective values of a solution from the existing solutions in both the populations, i.e,. parent population as well as offspring population. If all of the five objective values of a newly generated solution do not match with any existing solution's objectives in the population, then it is considered as unique solution. We have used objective-wise crossover followed by mutation, hence, five different crossovers and mutations are used in this framework and consequently each iteration may add upto five solutions in offspring population. At the end of every generation, the parent and offspring population are combined and a new parent population is created for next generation by following the concepts of nondominated sorting and crowding distance presented in [203]. In calculation of crowding distance, the boundary solutions are assigned a very large value, for each objective. In case, if two or more solutions have equal minimum objective value, the solution with minimum value of normalized sum

for all objectives is consider as lowest boundary solution. For example, while calculating crowding distance for first objective, viz. number of vehicles, if more than one solution have same minimum first objective value, then the solution having minimum normalized sum value of all other objectives is selected as lowest boundary solution in this objective. By this method, it is guaranteed that the solution which has minimum value in all objectives will always be included in population, which is also desired.

The proposed approach is executed for N_g generations. Since mutation operator is having better exploitation characteristics in comparison to crossover, hence for the last 25% generations, we have used only mutation operator. The mutation operator at this fag end phase acts as a local search to some extent and improves the solution quality further in the population. In this phase, the parent which has better objective value for the particular mutation operator is selected and copied as child solution and the mutation is applied on this child solution. The following subsection provides the detailed description of different components used in proposed INSGA-II approach. In case, if both parents are same then also we have applied only mutation operator, since the use of crossover is futile under this scenario. Algorithm 20 presents the pseudo-code of proposed *INSGA-II* approach.

7.4.1 Solution representation for *MOVRPTW*

We have used the similar solution representation as presented in [193]. Here, a solution contains several routes and each route is a linear permutation of customers, i.e., if j^{th} route contains $|r_j|$ customers, then route r_j is presented as permutation of $|r_j|$ customers such that the positions of the customers in this route specify the order in which customers are served. It is assumed that by default every route must begin and end with vertex *0*, which represents the depot, and hence, depot is not explicitly included in our solution representation. Figure 7.1 illustrates solution representation, by assuming an example consisting of *N*=10, i.e., 10 customers and M=3 routes, viz. $R = \{r_1, r_2, r_3\}$. Note that 0 is not explicitly included in any route in our representation as shown in Figure 7.1(b), but every route starts and ends at 0. Hence, 0 is made part of a route whenever we process that route to compute various values associated with that route. For example, the first route r_1 is $\langle 0, c(1,1), c(2,1), c(3,1), c(4,1), 0\rangle$, though we represented it as $\langle c(1,1), c(2,1), c(3,1), c(4,1)\rangle$. Here, $c(1,1)$ = 5, $c(2,1)$ = 9, $c(3,1)$ = 1 and $c(4,1)$ = 3, i.e., customer 5 is the first customer served in this route, customer 9 is the second customer served, and so on. Likewise, r_2 and r_3 have 3 customers each, and, their composition can be explained analogously.



(a) **Solution** (b) **Representation**

Figure 7.1: Solution representation

7.4.2 Initial population generation

To generate initial population, we have used an iterative approach similar to initialization procedure presented in [193]. The process begins with creation of a route by selecting a customer randomly. Again the next customer is randomly selected and inserted into next position in the current route. If insertion fails due to constraints violation, then this customer is iteratively tried in previous existing routes, starting from the first route. If a feasible place is found in any route then this customer is inserted there, otherwise a new route is created. This process continues until all customers is inserted into some routes. The generated solution is checked for uniqueness and added in population if it is unique. Using this approach, we have created a population of size p as initial population.

7.4.3 Objective-specific crossover operators

We have developed greedy variation operators which are not only problem-specific, but also objective-specific, i.e., our crossover and mutation operators are designed as per the characteristics of *MOVRPTW* and the characteristics of each objective. Hence, we have a dedicated crossover operator and a dedicated mutation operator for each objective, thereby leading to five different crossover operators and five different mutation operators. The crossover operators used in our approach are inspired from the crossover operators used in [42, 142]. These crossover operators create a child solution in two phases.

The first phase starts with an empty solution and iteratively constructs the child solution. In each iteration, it chooses one of the parent randomly and finds the most promising feasible route in this parent. The most promising route is copied to child solution and number of routes in child is incremented by one. This route is deleted from the parent from which it was taken. In case of other parent, the customers belonging to the most promising route is deleted from their respective routes, by connecting the predecessor of each such customer to its successor and updating the various quantities associated the affected routes accordingly. The process is repeated k times. It is pertinent to mention that, first phase need not always produce a partial solution with k routes, since it is also possible that after few iterations, some or all of the updated routes may become infeasible, and hence, in this case none of the route from the selected parent satisfy the most promising feasible route criteria. In this case, a child solution with less than k routes is produced.

Clearly, after first phase, some customers remain unassigned and need to be inserted in the child solution. The second phase inserts the unassigned customers into best position in an iterative manner, and, a complete child solution is created. During each iteration, it selects an unassigned customer randomly and inserts it at the best position which is defined as per the objective under consideration. To find the best position, the crossover operator begins from the first route and try to insert at every position between the customers in the route. Similarly other routes are also considered one-by-one. If a position is found which satisfy all the constraints, then this position is considered as a feasible position and then all feasible positions are evaluated to find the best position. If the selected unassigned customer cannot be inserted into any existing route due to the violation of any constraint, then a new route is created by inserting the selected unassigned customer at first position.

Since we have used objective-wise crossover operator, and hence, the definition of most promising route, the definition of best position and the value of k vary as per the objective and are specified below for each objective.

Let MIN_R and MAX_R are the minimum and maximum value of the number of routes between both parents. For example, let 8 and 10 are the number of routes in parent 1 and parent 2 respectively. Then, MIN_R = 8 and MAX_R = 10, for the assumed example. The value of k depends on the MIN_R and MAX_R.

f_1: The most promising route is the route which has maximum number of customers. Since f_1 seeks a solution which has minimum number of vehicles (minimum number of routes), hence selecting a route with more number of customers may create a child solution with fewer routes

in comparison to both the parents. Here $k = MIN_R$ -1, due to reason that we may find a child solution which has less number of routes in comparison to both the parents. Here the best position is the first feasible position which satisfy all the constraints.

f_2: Here, the route which has smallest ratio of the distance traveled to the number of customers in the route, is considered as most promising route. If in a route more number of customers are covered by traveling less distance, then selecting such routes may provide a better solution as per objective f_2. This is the idea behind selecting the smallest ratio of the distance traveled to the number of customers in the route. Since we already have a feasible solution (parent 1 or parent 2) with number of routes equal to MIN_R, hence k is assigned here MIN_R. The best position for f_2 is the position which yields least increment in the total traveled distance.

f_3: The objective f_3 is to minimize the longest travel time of a route, and hence, the most promising route is considered as the route which has the minimum travel time. Here, $k = MAX_R + (\lfloor u_{01} \times 10 \rfloor)$, where u_{01} is a uniform random number between [0,1). The initial solution generation procedure provides a solution in which all customers are tried to be accommodated in some existing route instead of creating a new route, and hence, in general, most of the routes have large travel times. Here the value of k is set to a larger value than the number of routes in both parents because the crossover operator used here intends to provide more opportunity to both parents so that a route with less travel time is selected in child solution. Also a large value of k may create more number of routes in child solution, and, it is evident that if a solution has more number of routes then the travel time of each route will be reduced. The best position in a route is considered as a position which leads to least increment in travel time of the route. In case, if there are more than one best position then the route which has less travel time is selected for insertion. It is also possible the least increment travel time position in a route may result in highest makespan, i.e., inserting the customer at this position may makes the concerned route to have longest travel time in comparison to inserting it at respective best positions in all other routes. Hence, in such scenario the second least increment position is considered. If this also result in same highest makespan then third least increment position is considered and so on. If inserting in any route lead to same makespan then instead of creating a new route, we inserted the unassigned customer to the first route considered, so that f_1 should not get deteriorated at the same time.

Objectives f_4 and f_5 are sharing similarity with objective f_2, since all f_2, f_4 and f_5 seek to minimize a overall quantity in a solution. The quantity is total traveled distance, total waiting

time and total delay time for f_2,f_4 and f_5 respectively. Hence the crossover operators for f_4 and f_5 are similar to crossover operator of f_2.

f_4: The most promising route here is the route which has smallest ratio of the waiting time to the number of customers in the route and k = MIN_R. The best position is the position which yields least increment in the total waiting time.

f_5: The most promising route here is the route which has smallest ratio of the delay time to the number of customers in the route and k = MIN_R. The best position is the position which yields least increment in the total delay time.

7.4.4 Objective-specific mutation operators

Our mutation operators are based on destruction and reconstruction strategy, where the solution is partially destroyed and then again reconstructed by using an objective-specific approach which is an appropriate mix of greediness and randomness. These mutation operators use Select_route procedure to select one or more routes and create a pool of unassigned customers by removing some customers randomly, from the selected route(s). There are two versions of Select_route procedure. The mutation operators for f_1 and f_3 use only first version of Select_route procedure, whereas mutation operators for f_2, f_4 and f_5 use both versions of Select_route procedure, in a mutually exclusive manner. In first version of Select_route procedure, a route is selected as per the requirement of objective and some or all customers from this route are removed and considered as unassigned customers. If after removal of customers, the selected route become infeasible, then the route is made feasible by breaking the route into two or more feasible routes. The second version of Select_route procedure works in an iterative manner and used only in mutation operators for f_2, f_4 and f_5. In each iteration, one route is picked randomly and a random customer is selected in this route. If the removal of the selected customer does not result in an infeasible route then this customer is removed from this route and added in pool of unassigned customer.

The complete solution is created in iterative manner. In each iteration, a customer is selected randomly, from the pool of unassigned customers and inserted at best position in some existing route. If no best position is found due to violation of constraints, then a new route is created by inserting the selected unassigned customer at first position in this newly created route.

Like in case of our crossover operators, here also the definition of Select_route and best position are different for various objectives and their descriptions are as follows:

Algorithm 20: Pseudo-code of *INSGA-II* for MOVRPTW

Input: p is the population size
Output: Set of nondominated solutions (Pareto set)
$Population \leftarrow$ Initial population generation(p);
Nondominated Sort(population);
Crowding distance(population);
// Crowding distance is calculated as per the solutions in the particular front
N $\leftarrow$ 0; // N is the generation count & N_g is the total number of generations
while *($N < N_g$)* **do**
  if *(N is less than 75% of N_g)* **then**
    $flag \leftarrow 1$;
  else
    $flag \leftarrow 0$;
  // flag variable is used to perform only mutation for last 25% generations
  // P is the current population of size p
  $p_x \leftarrow p + 1$;
  // p_x is the place where newly generated offspring solution is added in population
  for *($i := 1$ to 50)* **do**
    $Par_1 \leftarrow BTS()$; // Binary tournament selection
    $Par_2 \leftarrow BTS()$;
    // Select parents Par_1 & Par_2 using binary tournament selection
    for *($obj := 1$ to 5)* **do**
      if *(flag is equal to 1)* **then**
        if *(Par_1 and Par_2 are not same)* **then**
          $C \leftarrow Crossover_{obj}(Par_1, Par_2)$;
        else
          $C \leftarrow Par_1$;
        $M \leftarrow Mutation_{obj}(C)$;
      else
        $C \leftarrow Better_{obj}(Par_1, Par_2)$;
        // The parent which is better in particular objective
        $M \leftarrow Mutation_{obj}(C)$;
      if *($M \prec C$)* // M dominates C **then**
        $X \leftarrow M$;
      else if *($C \prec M$)* **then**
        $X \leftarrow C$;
      else
        // Solutions C and M do not dominate each other
        if *($f_{obj}(M) < f_{obj}(C)$)* **then**
          // If Solution M is better than C in the objective *obj*
          $X \leftarrow M$;
        else
          $X \leftarrow C$;
      if *(X is unique in population)* **then**
        Add X in population at p_x;
        $p_x \leftarrow p_x + 1$;
  $P \leftarrow$ Select p solutions from the union of parent (p) and offspring (p_x) population using nondomination sorting and crowding distance;
  $N \leftarrow N + 1$;
return (set of nondominated solutions);

f_1: The Select_route finds a route with least number of customers. If more than one route is having least number of customers then a route is randomly selected from among them. All of the customers in the selected route is considered as unassigned customers. In each iteration, a customer is randomly selected from this pool of unassigned customers and inserted at the best position. Here, the best position is the first feasible position which do not delay the starting time of other customers. The starting time of a customer means the vehicle arrival time to the customer before insertion. If an unassigned customer is inserted at a position which do not delay the starting time of other customers, then such insertion will create possibilities to insert more customers in the route, as the chances of the violation of second and third constraints (7.15 and 7.16) will be less. If no such position is found then the unassigned customer will be inserted at last feasible position. If no feasible position found then the mutation operator simply discard the solution and the mutant will be same as child solution generated by the crossover, since mutation operator fails to decrease the number of routes. Please note that travel times do not follow triangle inequality, and hence, insertion of a new customer may decrease the starting time of subsequent customers. If mutation operator successfully reduces the number of routes by one then next route with least number of customers is again tried for merging with other routes. The mutation operator is stopped when a customer cannot be inserted into any other route.

f_2: The mutation operator for f_2 has used two types of Select_route procedures. The first one which is used with probability 0.7, selects a route randomly and then removes random number of customers from this selected route. For example, if the selected route has 10 customers then a random number of customers between [1, 10] can be selected and a pool of unassigned customers is created using these removed customers. The second type of Select_route procedure is used with remaining probability 0.3 and iteratively removes a random customer from a randomly selected route. It iterates for ν times. Here, the best position is same as used in crossover operator used for objective f_2 i.e, the position which has least increment in the total distance traveled.

f_3: The Select_route finds a route with maximum travel time. If there exist more than one route corresponding to maximum travel time, then a route is randomly selected from among them. A random number $\in [1, \nu]$ of customers is removed randomly from this selected route and a pool of unassigned customers is created. If ν is greater than the number of customers in this route then ν is reset to the number of customers in this route. Since travel times do not satisfy the triangle inequality, so removal of customers from the selected route may result a route with more travel time than the previous maximum travel time. In this case the maximum

travel time is considered as the new travel time value. Here, the best position is the place where the increment in travel time is least after insertion. Like crossover operator here also, if there are more than one best position then the route which has less travel time is selected for insertion. We consider only those positions for insertion which can yield a travel time of route strictly less than the previous maximum travel time.

Here also the mutation operators for f_4 and f_5 work in a similar manner as mutation operator for f_2, due to the reasons mentioned in Section 7.4.3.

f_4: Two types of Select_route procedure is used with different probabilities. First one which is used with probability 0.6, selects a route randomly and then remove random number of customers from this selected route. The second Select_route procedure is used with remaining probability 0.4 and iteratively removes a random customer from a randomly selected route. The number of iterations is ν. The best position is same as used in the crossover operator used for objective f_4, i.e., the position which has least increment in the total waiting time.

f_5: Two types of Select_route procedure is used with different probabilities. First one which is used with probability 0.5, selects a route randomly and then remove random number of customers from this selected route. The second Select_route procedure is used with remaining probability 0.5 and iteratively removes a random customer from a randomly selected route. The number of iterations is ν. The best position is same as used in crossover operator used for objective f_5, i.e., the position which has least increment in the total delay time.

7.5 Computational results

We have implemented the *INSGA-II* approach in C language. A Linux based 2.50 GHz Intel Core i7 processor based system with 8GB of RAM is used to perform all computational experiments. To evaluate the performance of our approach, we have used the same test datasets as used in [193, 201]. Since Solomon's instances do not represent the ideal benchmark scenarios for multiobjective VRPTW. Hence, we have not tested our approach on Solomon's dataset. In order to assess the relative performance of our proposed approach with the state-of-the-art approach viz. LSMOVRPTW [193], we have implemented the LSMOVRPTW approach in C language and executed this approach under the same computing environment as our approach.

The proposed *INSGA-II* approach is executed for 2000 generations i.e., N_g=2000. The population size p is set to 100. The parameter p_{bt} in binary tournament selection is set as 0.7, and the value of ν in mutation operators is set to 4. The proposed approach has been executed

30 independent times on each instance like the LSMOVRPTW approach of [193]. All the parameter values are determined empirically after executing the proposed approach a number of times. These values fetch better results on most of the instances, but they cannot regarded as optimal parameter values for all the instances.

7.5.1 Description of *MOVRPTW* instances

The real-world *MOVRPTW* dataset [201] consists of a total of 45 instances, generated by considering combinations of three different sizes of customers, viz. 50, 150 & 250, five different time windows profiles, viz. 1, 2, 3, 4 & 5, and three kinds of vehicles capacities, each corresponds to a particular value of the parameter δ. The three values of δ are 60, 20 and 5. The depot has 8 hours working time. The five time window profiles represent the availability of customers, and, are based on the experience of the delivery company. These five profiles are described as follows:

1. All the customers are available for all day, i.e., for entire 8 hours (480 min).

2. Three kinds of customers are considered, viz. early customers, mid-day customers and late customers. The total time 480 min is divided into three equal part, viz. 160 min time slots for each kind of customer. The early customers will be served in [0, 160) minutes time slot, mid-day customers will be served in [160, 320) minutes time slot, and late customers are served in [320, 480] minutes time slot.

3. The length of each time window is decreased by 30 minutes. Thus the time slot for early customers is [0, 130], for mid-day customers is [175, 305], and for late customers is [350, 480].

4. The length of each time window is further decreased by 30 minutes. Thus the time slot for early customers is [0,100], for mid-day customers is [190,290] and for late customers is [380,480].

5. Here, a customer can belong to any of the aforementioned time windows profiles.

In order to serve each customer, the minimum capacity of vehicle ($\underline{D}$) must be equal to the maximum demands among the customers. Evidently, the maximum capacity of vehicle ($\overline{D}$) should not be greater than the sum of demands of all the customers. Hence, the capacity of each vehicle C must lie in between [$\underline{D}$, $\overline{D}$]. The capacity of each vehicle is set as $C =$

$\underline{D} + \delta \times \frac{(\overline{D}-\underline{D})}{100}$, where $\delta \in \{60, 20, 5\}$ as mentioned already. Hence, the capacity of vehicle is directly proportional to δ. Each customer i has a demand g_i and requires service time s_i. The demand and service time both are assigned from the set $\{10, 20, 30\}$ with a probability of $\frac{1}{3}$. A maximum delay of 30 min is allowed for each customer after the end of their time window, i.e., $m_d = 30$ min [193, 201].

7.5.2 Performance metrics

In multiobjective optimization, the performance of an algorithm is evaluated in terms of both convergence and diversity. In [215] various performance metrics (quality indicators) are analyzed. It is concluded that no single performance metric is able to provide a comprehensive measure on the performance of a multiobjective optimization algorithm, and thus, it is better to use a combination of quality indicators. To evaluate the performance of our proposed approach, we have used three metrics, viz. inverted generational distance (IGD), hypervolume (HV), coverage metric (C-metric). The first two performance metrics are among most commonly used performance metrics for multiobjective optimization problems [226], whereas the last one was chosen because it is used in [193] to evaluate the relative performance of LSMOVRPTW. These three metrics are described below:

1. Inverted generational distance (IGD): IGD [218] provides a measure to both convergence and diversity of the nondominated solutions obtained by an algorithm in MOP. Let P^* be a set of uniformly distributed solutions in the Pareto front (PF), and X is the set of obtained nondominated solutions. The minimum Euclidean distance between a solution v and the solutions in X is denoted by $d(v, X)$. The IGD approximation of X with respect to P^* is defined as,

$$IGD(P^*, X) = \frac{\sum_{v \in P^*} d(v, X)}{|P^*|} \tag{7.19}$$

   Smaller IGD value represents the closeness of set X towards the Pareto front, and hence, considered as a better set of solutions in terms of convergence and diversity. Figure 7.2 represents the approximation of IGD for a set of nondominated solutions (X) with respect to Pareto front (P^*).

2. Hypervolume (HV): HV metric is proposed by [217]. It represents the n-dimensional volume in the objective space that is contained by nondominated solutions with respect to

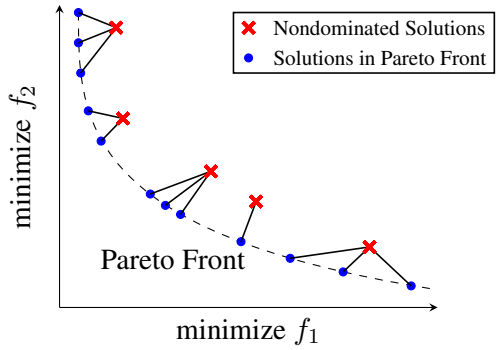


Figure 7.2: Approximation of IGD for the solution set X

a reference point. Figure 7.3 illustrates the approximation of HV for a set of nondominated solutions. A set with higher HV value represents a better set of nondominated solutions approximating the Pareto front, from the point of view of the convergence and diversity. To calculate hypervolume, we have used the HV approach presented in [227], and its associated software tool available in [228]. The point (1.01, 1.01, 1.01, 1.01, 1.01) is used as reference point to calculate the hypervolume.

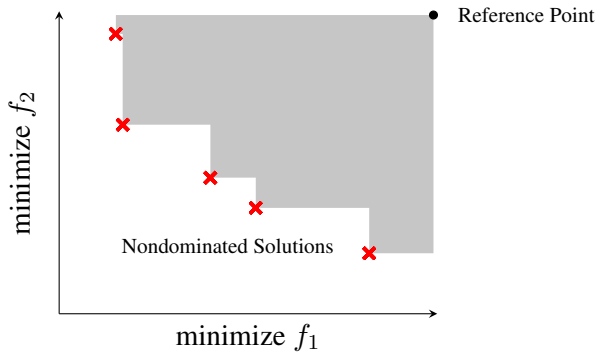


Figure 7.3: Approximation of HV for the solution set X

3. Coverage metric (C-metric): C-metric [229] is used to compare two sets of nondominated solutions obtained by two different approaches. Let A and B are two sets of nondominated solutions, then $C(A, B)$ represent C-metric of A with respect to B, and defined as the percentage of the solutions in B that are Pareto dominated by at least one solution in A. $C(A, B) = 1$ means that all nondominated solutions in B are Pareto dominated by solutions in set A. It is important to mention that the sum of $C(A, B)$ and $C(B, A)$ need not be equal to 1, as some solutions in A and B may not Pareto dominate one another.

As the true Pareto front can not be determined, the P^* is approximated by considering all unique nondominated solutions obtained by LSMOVRPTW and *INSGA-II* in all 30 runs. Since the range of objectives are different, we have normalized all objective values to calculate HV and IGD.

7.5.3 Experimental results

The performance of *INSGA-II* approach is compared with LSMOVRPTW approach [193]. Table 7.1 presents the comparison of results in terms of *IGD*, *HV*, and *C-metric*. In this table, the first column represents the name of the instances in the form "$num_1 - num_2 - num_3$", where num_1 shows the number of customers, num_2 shows the index of δ type with respect to capacity of vehicle, and num_3 indicates the index of time window profile. The second, third, and fourth columns represent the average values over 30 runs of the *IGD*, *HV* and *C-metric* respectively. The values for each metric are reported for LSMOVRPTW and *INSGA-II* approaches both. The last column represents the average execution time (in seconds) over 30 runs taken by *INSGA-II* approach. Since LSMOVRPTW uses a series of local searches and there is no explicit concept of generation in this approach, we have used the running time of *INSGA-II* as stopping criteria for LSMOVRPTW approach for a fair comparison. Similar termination criteria was used in [193] to compare LSMOVRPTW with NSGA-II [201]. The performance of NSGA-II presented in [201] was found to be much worse than LSMOVRPTW [193], and hence, we have not included NSGA-II [201] in this comparison.

Due to the limitation of space, standard deviation of the metrics are not presented in the table. Furthermore, we have used Wilcoxon signed-rank test [230, 231] at 5% significance level, to show the statistically significant difference between the results obtained by our approach *INSGA-II* and LSMOVRPTW. An online calculator [1] is used to perform Wilcoxon signed-rank test. The significantly better values obtained by both the approaches are represented in bold font for ease of identification. The last row of table provides the overall comparison in the format W/S/B, which indicates that the performance of *INSGA-II* is worse than, similar to and better than the LSMOVRPTW in W, S and B instances.

The *INSGA-II* approach significantly outperforms LSMOVRPTW approach for all 45 instances in terms of *IGD*. In terms of *HV*, *INSGA-II* significantly outperformed the LSMOVRPTW in 33 instances and significantly outperformed by LSMOVRPTW in 11 instances. This shows

[1] https://mathcracker.com/wilcoxon-signed-ranks.php

that *INSGA-II* performs better than LSMOVRPTW in terms of both convergence and diversity. C-metric is calculated as the ratio of number of solutions in latter approach that are Pareto dominated by at least one solution in former approach. The LSMOVRPTW approach is a local search based approach in which a series of local searches is performed on solutions present in archive, and hence, the chance of getting some highly converged solutions is high. While the *INSGA-II* is a population based approach without any local search procedure, and, use of objective-specific crossover & mutation operators can provide a better set of nondominated solutions in terms of both convergence and diversity. In terms of C-metric, the *INSGA-II* significantly outperformed the LSMOVRPTW in 10 instances and significantly outperformed by LSMOVRPTW in 29 instances. This can be attributed to the local search based approach in LSMOVRPTW which provides some highly converged solutions, thereby enabling better performance of LSMOVRPTW in terms of C-metric.

In the instances with first time window profile presented as num_3=0 (such as "250-2-0"), the customers are always available, and hence, the objectives f_4 and f_5 are always zero and the problem in this particular time window profile has only three significant objectives, viz. f_1, f_2 and f_3. Evidently, the hardness of the problem increases with increase in number of customers and decrease in capacity of vehicles. This is due to the fact that the problem with more number of customers and with vehicle of smaller capacity will have solution with more number of routes, and hence, to converge on all objectives will be tougher. The instances with fourth time window profile presented as num_3=3 (such as "250-2-3"), the length of time window which presents the availability of customers is smallest. In this case, no customer is available for a total of 178 minutes (two durations of 89 minutes each) out of total 480 minutes (all day duration). Thus the objective f_4, viz. total waiting time of vehicles due to early arrivals at customer place is highest in comparison to all other time window profiles, and hence, the objective f_4 is more significant in these instances. Clearly, all five objectives are equally significant in the instances with fourth time window profile (num_3=3).

The values of HV in Table 7.1 reveals that the performance of *INSGA-II* is always better than the performance of LSMOVRPTW, in first time window profile (num_3=0) and with 250 customers (num_1=250), in all instances . Also the difference in HV values of *INSGA-II* and LSMOVRPTW are marginally more in all instances (particularly with more number of customers) with fourth time window profile presented as num_3=3. This shows that the performance of *INSGA-II* is better than LSMOVRPTW especially on harder instances.

Table 7.1: Average values of IGD, HV, And C-Metric of INSGA-II and LSMOVRPTW

Instance	IGD		HV		C-metric		Time [a]
	LSMOVRPTW	INSGA-II	LSMOVRPTW	INSGA-II	C(LS,IN) [b]	C(IN,LS) [c]	
50-0-0	0.002849	**0.001374**	0.778693	**0.798471**	0.117421	**0.481708**	17.10
50-0-1	0.005578	**0.004443**	**0.460442**	0.434376	**0.290333**	0.029811	12.73
50-0-2	0.013812	**0.005579**	0.293905	**0.372857**	**0.231333**	0.041352	11.26
50-0-3	0.006679	**0.002525**	0.514643	**0.617789**	0.029333	**0.425595**	8.42
50-0-4	0.008824	**0.005600**	**0.943907**	0.939618	**0.306000**	0.215279	14.33
50-1-0	0.003507	**0.001724**	0.799644	**0.819721**	0.125619	**0.539394**	13.00
50-1-1	0.005708	**0.004427**	**0.460791**	0.433047	**0.288333**	0.030735	13.29
50-1-2	0.013848	**0.005465**	0.298750	**0.365217**	**0.223667**	0.034913	11.96
50-1-3	0.006677	**0.002525**	0.514647	**0.617789**	0.029333	**0.444048**	8.48
50-1-4	0.008454	**0.005473**	0.922378	0.923307	0.247000	0.225724	8.91
50-2-0	0.011147	**0.007107**	0.867581	**0.898339**	0.108445	**0.751587**	5.84
50-2-1	0.011408	**0.004776**	**0.734837**	0.701035	**0.306667**	0.055424	6.20
50-2-2	0.019621	**0.004737**	0.368060	**0.615539**	0.148000	0.186794	6.31
50-2-3	0.007721	**0.003965**	0.655798	**0.729146**	0.037667	**0.384127**	6.03
50-2-4	0.019475	**0.009309**	0.783465	**0.872617**	0.003667	**0.914849**	5.53
150-0-0	0.002900	**0.001820**	0.735135	**0.755004**	0.187478	**0.391211**	134.39
150-0-1	0.017247	**0.006331**	0.267536	**0.292883**	**0.419333**	0.003367	93.95
150-0-2	0.012206	**0.007195**	**0.277424**	0.255240	**0.467333**	0.004369	80.57
150-0-3	0.027004	**0.002912**	0.023876	**0.355080**	**0.169333**	0.114975	53.30
150-0-4	0.011028	**0.008906**	**0.469320**	0.397354	**0.653000**	0.012018	96.48
150-1-0	0.002921	**0.001830**	0.735225	**0.755004**	0.188144	**0.390476**	129.30
150-1-1	0.017019	**0.006302**	0.267405	**0.292916**	**0.419000**	0.003345	86.72
150-1-2	0.012152	**0.007187**	**0.277578**	0.255240	**0.467000**	0.004371	80.13
150-1-3	0.026895	**0.002906**	0.023876	**0.355080**	**0.169000**	0.112892	52.98
150-1-4	0.011262	**0.009006**	**0.467632**	0.397354	**0.651333**	0.011976	87.64
150-2-0	0.004479	**0.002366**	0.787328	**0.821469**	0.141612	**0.508714**	45.99
150-2-1	0.015057	**0.006741**	**0.440961**	0.374096	**0.527333**	0.000741	53.81
150-2-2	0.012309	**0.007315**	**0.271850**	0.246393	**0.484333**	0.001149	54.96
150-2-3	0.025342	**0.002870**	0.029056	**0.361563**	**0.156333**	0.098382	48.86
150-2-4	0.009941	**0.008416**	**0.597763**	0.493563	**0.683000**	0.009872	56.62
250-0-0	0.003846	**0.002675**	0.713256	**0.734973**	**0.353986**	0.142155	338.06
250-0-1	0.022698	**0.007819**	0.183124	**0.244897**	**0.489000**	0.001389	211.74
250-0-2	0.023812	**0.007688**	0.137427	**0.218211**	**0.385667**	0.001190	178.88
250-0-3	0.031870	**0.005041**	0.145107	**0.563146**	0.120333	0.096493	111.10
250-0-4	0.017896	**0.010545**	0.232987	**0.471353**	**0.486000**	0.009488	226.10
250-1-0	0.003866	**0.002687**	0.713344	**0.734973**	**0.354666**	0.140602	339.96
250-1-1	0.022437	**0.007742**	0.183535	**0.244658**	**0.490333**	0.001333	234.87
250-1-2	0.023050	**0.007706**	0.145502	**0.218211**	**0.405000**	0.001190	183.26
250-1-3	0.031775	**0.005035**	0.145077	**0.563146**	0.120000	0.096920	105.61
250-1-4	0.017722	**0.010528**	0.233035	**0.471431**	**0.485333**	0.009429	215.09
250-2-0	0.004514	**0.002327**	0.742472	**0.783369**	0.218530	0.284062	179.09
250-2-1	0.022466	**0.007084**	0.176404	**0.231610**	**0.471000**	0.000000	194.70
250-2-2	0.021257	**0.007408**	0.126384	**0.208450**	**0.342667**	0.000000	178.11
250-2-3	0.031753	**0.005106**	0.145161	**0.563510**	0.121667	0.095974	110.86
250-2-4	0.020832	**0.012327**	0.199133	**0.429170**	**0.470667**	0.008397	193.74
W/S/B	0/0/45		11/1/33		29/6/10		

[a] Seconds
[b] C(LSMOVRPTW,INSGA-II)
[c] C(INSGA-II,LSMOVRPTW)

It is pertinent to mention that any solution which has minimum value in any one objective is a nondominated solution irrespective of the values of other objectives. In a real-world scenario, the preference for one objective might be more in comparison to other objectives in MOPs. For example, in *MOVRPTW* with five objectives, the objective f_2, viz. total travel distance might be more important in some perspective than other objectives as distance traveled is directly proportional to consumption of fuel, and hence, it has direct impact on environmental pollution as well as important from monetary considerations. Also for some industrial organizations the objective f_5, i.e., total delay time may be more important than other objectives as delay in service adversely impacts the customers' goodwill, thereby reducing the volume of business. Hence, finding minimum values in all the objectives by an approach is crucial in *MOVRPTW*, since in most of the cases the decision maker's preference is not known a priori. An analysis in this regard is performed in Table 7.2.

There are 45 instances in total and both the approaches have been executed for 30 independent times on each instance. Hence, we have 30 sets of nondominated solutions for each instance and a total 1350 sets of nondominated solutions, generated by each approach. For a particular instance, in each run the five minimum values of each objective obtained by *INSGA-II* and LSMOVRPTW are compared. The comparison results are presented in Table 7.2. In this table, each row except the last two represents the summarized comparison over 30 runs for an instance, in terms of number of runs where the minimum value of each objective found by *INSGA-II* is better (<), equal (=) or worse (>) than the minimum value of corresponding objective found by LSMOVRPTW. Here, the first column represents the name of the instances in the form "$num_1 - num_2 - num_3$". The second, third, fourth, fifth and sixth columns provides the comparison of five objective values respectively, in all 30 runs. For example, second row (instance "50-0-1") and second column (objective f_1) shows that *INSGA-II* is able to find 22 times better, 7 times equal and 1 time worse values for objective f_1 in 30 runs in comparison to LSMOVRPTW. The second last row provides the distribution of 1350 comparisons in better, equal and worse categories. It shows that the minimum objective values obtained by *INSGA-II* is superior than LSMOVRPTW, for objectives f_2, f_3 and f_4, and, equal in objective f_5, but slightly inferior in objective f_1. The last row provides the summary of comparison in terms of number of instances between two approaches for each objective by taking the minimum value of each objective over 30 runs of an approach. The *INSGA-II* is able to find better values on 36 instances, equal value on 1 instance, and worse values on 8 instances for objectives f_2, as compared to LSMOVRPTW. Similarly, *INSGA-II* performs better for objectives f_3 and f_4, and,

equal performance in objectives f_1 & f_5 (except for one instance for f_1 where it performed better). Hence, last two rows of Table 7.2 clearly show that the performance of *INSGA-II* is better than LSMOVRPTW in individual runs as well as in best of all 30 runs.

Table 7.2: Comparison of INSGA-II with LSMOVRPTW in terms of count of the runs as well as in overall 30 runs, on which INSGA-II achieved better (<), equal (=) and worse (>) values in five objectives.

Instance	f1			f2			f3			f4			f5		
	<	=	>	<	=	>	<	=	>	<	=	>	<	=	>
50-0-0	0	30	0	11	10	9	0	30	0	0	30	0	0	30	0
50-0-1	22	7	1	20	0	10	0	30	0	26	0	4	0	30	0
50-0-2	0	30	0	16	0	14	0	30	0	24	1	5	0	30	0
50-0-3	0	30	0	12	0	18	0	30	0	30	0	0	0	30	0
50-0-4	0	30	0	19	2	9	0	30	0	0	30	0	0	30	0
50-1-0	0	30	0	22	0	8	0	30	0	0	30	0	0	30	0
50-1-1	18	10	2	19	1	10	0	30	0	26	0	4	0	30	0
50-1-2	0	30	0	12	0	18	0	30	0	20	4	6	0	30	0
50-1-3	0	30	0	12	0	18	0	30	0	30	0	0	0	30	0
50-1-4	0	30	0	22	1	7	0	30	0	0	30	0	0	30	0
50-2-0	0	30	0	26	0	4	0	30	0	0	30	0	0	30	0
50-2-1	0	30	0	26	0	4	0	30	0	27	0	3	0	30	0
50-2-2	0	30	0	28	0	2	0	30	0	25	0	5	0	30	0
50-2-3	0	30	0	25	0	5	0	30	0	30	0	0	0	30	0
50-2-4	0	30	0	30	0	0	0	30	0	28	0	2	0	30	0
150-0-0	10	18	2	25	0	5	17	12	1	0	30	0	0	30	0
150-0-1	0	30	0	22	0	8	30	0	0	16	0	14	0	30	0
150-0-2	0	30	0	27	0	3	30	0	0	1	1	28	0	30	0
150-0-3	0	30	0	29	0	1	0	30	0	30	0	0	0	30	0
150-0-4	0	6	24	23	0	7	5	25	0	4	2	24	0	30	0
150-1-0	9	19	2	25	0	5	17	12	1	0	30	0	0	30	0
150-1-1	0	30	0	22	0	8	30	0	0	16	0	14	0	30	0
150-1-2	0	30	0	27	0	3	30	0	0	1	1	28	0	30	0
150-1-3	0	30	0	29	0	1	0	30	0	30	0	0	0	30	0
150-1-4	0	6	24	23	0	7	5	25	0	4	2	24	0	30	0
150-2-0	0	30	0	17	0	13	20	10	0	0	30	0	0	30	0
150-2-1	0	30	0	25	0	5	30	0	0	6	0	24	0	30	0
150-2-2	0	30	0	28	0	2	30	0	0	0	0	30	0	30	0
150-2-3	0	30	0	28	0	2	0	30	0	30	0	0	0	30	0
150-2-4	0	30	0	25	0	5	1	29	0	4	1	25	0	30	0
250-0-0	0	30	0	8	0	22	24	5	1	0	30	0	0	30	0
250-0-1	3	22	5	23	0	7	30	0	0	18	1	11	0	30	0
250-0-2	0	30	0	28	0	2	30	0	0	0	0	30	0	30	0
250-0-3	0	30	0	30	0	0	0	30	0	30	0	0	0	30	0
250-0-4	0	3	27	26	0	4	27	3	0	3	0	27	0	30	0
250-1-0	0	30	0	8	0	22	24	5	1	0	30	0	0	30	0
250-1-1	3	22	5	23	0	7	30	0	0	18	1	11	0	30	0
250-1-2	0	30	0	28	0	2	30	0	0	0	0	30	0	30	0
250-1-3	0	30	0	30	0	0	0	30	0	30	0	0	0	30	0
250-1-4	0	3	27	26	0	4	27	3	0	3	0	27	0	30	0
250-2-0	0	30	0	19	0	11	26	3	1	0	30	0	0	30	0
250-2-1	1	24	5	25	0	5	30	0	0	13	1	16	0	30	0
250-2-2	1	29	0	30	0	0	30	0	0	0	0	30	0	30	0
250-2-3	0	30	0	30	0	0	0	30	0	30	0	0	0	30	0
250-2-4	0	3	27	18	0	12	30	0	0	2	1	27	0	30	0
Total	67	1132	151	1027	14	309	583	762	5	555	346	449	0	1350	0
Best value in all 30 runs	1	44	0	36	1	8	13	32	0	21	14	10	0	45	0

As already mentioned in Section 7.3.2, one multiobjective algorithm is considered to be better than another, if the set of nondominated solutions found by the former algorithm is better than latter in terms of convergence and diversity both. Visual representation is often used to compare the solution sets obtained by different approaches in terms of convergence and diversity as it provides ease of understanding. To visually represent the convergence and diversity of the nondominated solutions obtained by LSMOVRPTW and *INSGA-II*, we have used heatmap visualization technique [232, 233]. Heatmap provides the visual representation of solution sets to many-objective problems and facilitates observing of trade-off among various objectives in a clear manner. A heatmap shows the data as a grid of pixels whose colors represent values on a scale from maximal (hot) to minimal (cold) [232]. In heatmap representation, each row is a solution and each column represents an objective. The color of the cells represents the value of an objective for a particular solution. The cooler colors shows the convergence of solution and the distribution of full range of colors shows the diversity of solution. The nondominated solutions obtained by an approach can be considered as good solutions, if the heatmap visualization of the solutions shows the cooler colors as well as distribution of full range of colors.

In order to use heatmaps for visual representation, all objectives must be on the same scale [232], hence, we have normalized all objective values to the range of [0, 1]. Each heatmap is a visual representation of nondominated solutions obtained by an approach on a particular instance in all 30 runs. Each objective value is shown in particular color which is in the range of cold (blue) to hot (red).

As explained in Section 7.5.3, the instances with more number of customers and smallest capacity vehicles are harder than others. Therefore, we have selected instance with num_1= 250 (i.e, 250 customers) and num_2=2 (i.e, third category of vehicle which has least capacity in comparison to other categories) to present the visual comparison of the nondominated solutions obtained by both the approaches. Furthermore, it is observed that the performance of an approach on an instance depends on its time window profile, and hence, all five time window profiles instances with num_1= 250 and num_2=2 are selected for comparison of these two approaches.

Figure 7.4 presents the convergence and diversity of the nondominated solutions obtained by both the approaches using heatmaps on the five instances, viz. 250-2-0, 250-2-1, 250-2-2, 250-2-3 and 250-2-4. The first row and and second row of heatmaps represent the nondominated solutions obtained by *INSGA-II* and LSMOVRPTW respectively on the selected instances. The rows of each heatmap have been rearranged according to f_3 in ascending order. The

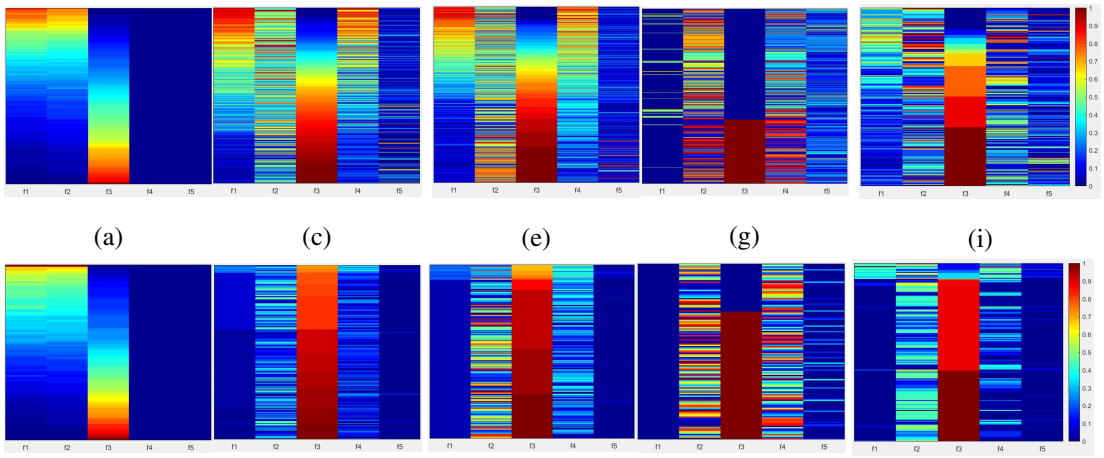


Figure 7.4: Heatmaps of nondominated solutions obtained by INSGA-II and LSMOVRPTW on selected instances (The rows of each heatmap have been rearranged according to f_3 in ascending order): (a) INSGA-II on 250-2-0, (b) LSMOVRPTW on 250-2-0, (c) INSGA-II on 250-2-1, (d) LSMOVRPTW on 250-2-1, (e) INSGA-II on 250-2-2, (f) LSMOVRPTW on 250-2-2, (g) INSGA-II on 250-2-3, (h) LSMOVRPTW on 250-2-3, (i) INSGA-II on 250-2-4, (j) LSMOVRPTW on 250-2-4.

heatmaps of *INSGA-II* show cooler colors than the heatmaps of LSMOVRPTW particularly for objective f_3, on all five instances. Hence *INSGA-II* has better convergence in objective f_3 than LSMOVRPTW in all five instances.

The instance "250-2-0" has only three significant objectives, viz. f_1, f_2 and f_3, as explained in Section 7.5.3. The Figure 7.4 (a) and (b) represents the heatmaps of both the approaches on instance "250-2-0", which clearly show that the *INSGA-II* has cooler colors than LSMOVRPTW in objectives f_1 and f_2, and similar color in objective f_3. Hence, it generates better solutions in terms of convergence and diversity. This is also supported by the values of HV and IGD metrics on instance "250-2-0" in Table 4.1.

In instance "250-2-3", the objective f_4, viz. total waiting time of vehicles due to early arrivals at customer place is highest in comparison to all other time window profiles, and hence, the objective f_4 is more significant in these instances, as explained in Section 7.5.3. The Figure 7.4 (g) and (h) represents the heatmaps of both the approaches on instance "250-2-3", which clearly show that the *INSGA-II* has cooler colors than LSMOVRPTW in objective f_4, and hence, able to find more converged solutions with respect to objective f_4.

The instances with fifth time window profile (presented as num_3=4 in Section 7.5.3, such as "250-2-4") have all variety of customers from different time window profiles, and hence,

they represent more realistic scenarios. The Figure 7.4 (i) and (j) are the heatmaps of both the approaches on instance "250-2-4", which reveals that the nondominated solutions generated by *INSGA-II* distribute more dispersedly with in full range of color than those generated by LSMOVRPTW. Hence, the solutions generated by *INSGA-II* have better diversity.

From all pair of comparisons, it can be inferred that the LSMOVRPTW always lacks diverse solutions and it is more apparent in objectives f_1 and f_5. Also heatmaps of *INSGA-II* show the better convergence in terms of objective f_3 than LSMOVRPTW. Evidently, the objectives conflict each other, and hence, an approach is better if the nondominated solutions yielded by an approach has better diversity, i.e., if the nondominated solutions have a wider spread in full range of color for all the objectives. Thus, the heatmaps visualization reveal that *INSGA-II* performs better than LSMOVRPTW from the perspective of convergence and diversity.

7.6 Conclusions

In this chapter, we presented a nondominated sorting genetic algorithm II (INSGA-II) based approach with objective-specific variation operators for the *MOVRPTW*. This problem is a variant of multiobjective vehicle routing problem (MOVRP) and contains five objectives to minimize. In the proposed *INSGA-II* approach, the crossover and mutation operators are designed by exploiting the problem-specific knowledge as well as the characteristics of each objective. Thus proposed *INSGA-II* approach uses a dedicated crossover operator and a dedicated mutation operator for each objective. To evaluate the performance of our proposed approach, we have used three metrics, viz. inverted generational distance (IGD), hypervolume (HV), coverage metric (C-metric), which are standard metrics used in multiobjective optimization. The computational results reveals that the set of nondominated solutions obtained by the proposed approach is superior than state-of-the-art approach in terms of both convergence and diversity.

Chapter 8

Conclusions and Directions for Future Research

In this thesis, we have addressed six $\mathcal{NP}$-hard permutation based combinatorial optimization problems using evolutionary techniques. These six problems not only have several practical applications in different fields such as wireless sensor networks, transportation, logistics, planning & scheduling, disaster management, but are also challenging from theoretical perspective. Since these problems share a common characteristic, viz. permutation, hence any improvement that can be made while addressing a permutation based problem will provide a scope for improvement for several other related permutation based problems.

To address the problems considered in this thesis, we have used three evolutionary approaches, viz. genetic algorithm (GA), evolution strategy (ES) and discrete differential evolution (DDE). Among them genetic algorithm is the most popular for solving any combinatorial optimization problem. Discrete differential evolution has many successful recent applications for addressing permutation based optimization problems. On the other hand, application of evolution strategy in solving discrete optimization problems is not as common as GA or DDE. In fact, to the best of our knowledge, no recent application of ES for permutation based problem is available in literature. The first three problems have only permutation characteristics. Out of these three, first two problems are addressed via evolution strategy based approach, whereas discrete differential evolution based hybrid approach is used to address the third problem. The last three problems have aspects of permutation as well as grouping. Among these three, the first two problems are solved by grouping genetic algorithm approach, whereas the last problem being multiobjective in nature uses nondominated sorting genetic algorithm II (NSGA-II) based

approach.

Although evolutionary techniques are highly popular for addressing combinatorial optimization problems, it is well-known that a generic evolutionary technique which does not use any problem-specific knowledge usually performs poorly, especially in case of hard problems. This fact is also supported by "no free lunch theorem"[234]. The research area of combinatorial optimization using evolutionary techniques has advanced to a level where any new evolutionary approach for a particular problem can not compete with the state-of-the-art approaches available in the literature for that problem unless and until it incorporates appropriate problem-specific information. This problem-specific information can be can be incorporated anywhere from solution representation, design of variation operators (crossover and mutation), fitness function, initial solution generation and/or in the form of local searches.

The development of new technology and the advancement of human civilization continue to introduce novel and complex real-world problems. Thus in the present age, it is desirable to develop approaches incorporating problem-specific knowledge. Most of the new development in the field of evolutionary techniques for solving combinatorial optimization problem are happening in the context of particular problem only. Our proposed approaches make use of problem-specific knowledge in solution encoding, variation operators, local search and generation of initial solution. The development of these evolutionary approaches, whose performances are superior than the state-of-the-art approaches for their respective problems, is the main contribution of this thesis.

In the following, we summarize the contributions made by various chapters along with the possible research directions which can be explored in future based on the work reported in these chapters.

In Chapter 2, we have proposed an evolution strategy based approach for the cover scheduling problem in wireless sensor networks (*WSN-CSP*). Our approach uses problem-specific heuristic which provide a better quality initial solution. We have used a reshuffle procedure to avoid getting trapped in local optima. *WSN-CSP* is a relatively understudied problem as only the approaches presented in [78] and [79] (namely, GA, ABC and IWO) are available in the literature. GA and ABC are commonly used for solving discrete optimization problems, and hence, their use for solving *WSN-CSP* was quite obvious. However, we have addressed *WSN-CSP* problem via a *two-membered evolution strategy*, whose use in solving discrete optimization problems is not as common as GA or ABC. Our proposed approach is simple and yet able to beat other state-of-art approaches for *WSN-CSP*. The hardness of the problem increases with increase in

breach rate, and the gap in performance of our approach and other approaches widens with increase in breach rate. This indicates the robustness of our approach.

The performance of the proposed approach provides the inspiration for developing similar approaches for other scheduling problems in various domains. We have used three mutation operators in our approach all based on a ruin-and-recreate strategy. Similar mutation operators can be tried in other metaheuristic frameworks to solve *WSN-CSP* and related problems. Its a well-known fact that utilizing the concept of opposite solutions leads to a wider exploration of search space than using only randomly generated solution [64]. However, the concept of opposite solution was defined in the context of optimization problems in continuous domain. We have extended this concept for permutation problems. Future approaches for permutation based problems can utilize the concept of opposite solutions presented here.

Chapter 3 presented a *two-membered evolution strategy* (*(1+1)-ES*) for total rotation minimization problem (TRMP) pertaining to directional sensor networks. The proposed approach for *TRMP* is an extension of *two-membered evolution strategy* approach for *WSN-CSP*. It is incorporated with a pre-processing step, which is used to boost the performance of the evolution strategy (ES). TRMP is an important problem as there is always an emphasis on conserving sensors' battery power and more so in remote or hostile environments where batteries cannot be replaced owing to cost/risks involved. *ES-TRMP* employs a pre-processing step that supplies an improved solution to the evolution strategy by utilizing the NEH heuristic and the concept of opposite solutions. We have compared *ES-TRMP* with the best approach available in the literature, viz. *GA-TRMP+*LS_B [100]. The computational results show the superiority of the proposed approach in terms of solution quality. Not even a single test instance exists where the best or average solution obtained by *GA-TRMP+*LS_B is better than that of *ES-TRMP*. However, *GA-TRMP+*LS_B is faster than our proposed *ES-TRMP* approach.

The fact that even *ES-TRMP*$_{NO\text{-}PP}$ which does not use pre-processing step, obtained solution of better quality than *GA-TRMP+*LS_B in most cases, clearly indicates the suitability of ES based approaches for TRMP and other similar scheduling problems. Hence, similar approaches can be developed for other related problems. We have clearly demonstrated the role of pre-processing step in boosting the performance of evolution strategy. Similar pre-processing steps can be developed for new applications of evolution strategy.

Despite the practical importance of TRMP, it is still an under-studied problem. In addition to present work, only approaches existing in the literature are those presented in [100]. Hence, there is a lot of scope to develop new lower bounds, heuristic, metaheuristic and exact approaches

for TRMP. Further, TRMP is unique from scheduling point of view also as several preceding covers in the schedule, not necessarily including the immediately preceding cover, determine the rotation cost of a cover. To our knowledge, no other scheduling problem exists in the literature in which the computation of scheduling cost of a job is dependent on several preceding jobs. Closest to TRMP are job scheduling problems with sequence dependent setup costs which are widely studied in literature. However, the computation of scheduling cost of a job in these problems is dependent on immediately preceding job only, and, not on several preceding jobs. Hence, TRMP constitutes a class of its own. Investigating the properties of this class of problems will provide a number of opportunities for future work.

Both *TRMP* and *WSN-CSP* share some similarity, viz. finding the optimal permutation of covers. Despite this fact, *TRMP* is more hard to solve due to its aforementioned characteristics. In *TRMP*, the pre-processing step employs *NEH* heuristic and concept of opposite solutions on several random solutions and the best solution among them is fed as input to *ES-TRMP*. To find initial solution for *WSN-CSP*, we used *NEH* heuristic and concept of opposite solutions only on one random solution instead of several ones as considered in *TRMP*. We tried the same pre-processing step as in *TRMP* to find initial solution for *WSN-CSP* also. But the performance of both approaches i.e., single random solution and multiple random solutions, was found to be same. This can be attributed to the difference in the hardness of both problems. *TRMP* is more hard than *WSN-CSP*, so pre-processing steps provide a broader exploration of search space, whereas in *WSN-CSP* such broader exploration is not required.

We also tried multi-membered evolution strategy (population) approach for both the problems. But the performance of two-membered ES is found to be superior than multi-membered ES. This can be due to the fitness landscape of the problem. From the detailed analysis of solutions obtained at different intervals throughout run of multi-membered ES and two-membered ES, we infer that the fitness landscape may have ruggedness and some multi-peaks (local optima) along with the peak of global optima. Thus, application of variation operators on members of population (multi-membered ES) make them oscillate between local optimas. Even, if we apply reshuffle procedure then they jump from one local optima to other neighboring local optima. On the other hand, in case of single initial solution, the two-membered ES performs like a hill climbing algorithm and repeated application of variation operator on a single solution along with reshuffle procedure results in better quality final solution. Finding the characteristics of fitness landscape for a $\mathcal{NP}$-hard problem is very difficult and even not possible in most of the

cases. Thus, it requires experimentation to get some idea about fitness landscape and then select suitable strategy based on the information about fitness landscape.

In Chapter 4, we have addressed single machine total stepwise tardiness problem with release dates (*SMTSTP-R*) via a discrete differential evolution (DDE) based approach hybridized with local search. We have used an additional problem-specific heuristic (*M-Moore*) along with those used in *TRMP* for pre-processing. The idea behind using heuristics is the same, to find better quality initial solutions. As mentioned in Section 1.7, in *DDE* due to application of perturbation before recombination, one parent used in recombination operator is a diverse solution most of the times. This feature of *DDE* is unique and helps in avoiding premature convergence. To address *SMTSTP-R*, we first started with the same *(1+1)-ES* approach as presented earlier. The solution generated by this approach was not able to outperform the solutions generated by state-of-the-art approaches (particularly the *HABC* approach). We observed that, in *SMTSTP-R* problem, the mapping of solution space to the objective space has many-to-one kind of relation, i.e., many distinct solutions yield same objective value. Thus, the *two-membered evolution strategy* which start with single initial solution always stuck in local optima despite using reshuffle strategy. This analysis motivated us to use *DDE* approach for addressing *SMTSTP-R*. The computational results on the standard benchmark instances demonstrate the superiority of *DDE* approach over the state-of-the-art approaches in terms of solution quality.

The superior performance of *DDE* over existing approaches will serve as a motivation for developing analogous approaches for other permutation based problems. The *SMTSTP-R* with multiple identical machines can be investigated, as the future work. Several industries have to bear different amount of costs, if the job is produced before its requirement. One such example is perishable goods. Perishable goods such as as milk, meat, fish, fruits and vegetables, health related products require different types of storage based on how early they are produced before consumption. For example, vegetables such as carrot or peas may require deep freezing, normal freezing, cold storage or room temperature depending on how early they are produced with respect to their date of consumption. Thus, a problem analogous to the *SMTSTP-R* can be modeled in the earliness form, where each job has various earliness dates and the earliness cost increases in stepwise manner as per the earliness dates of jobs. Several examples can also be found where stepwise earliness and stepwise tardiness, both are involved. For example, in production of vaccines earliness can occur at the manufacturing sites and the tardiness can occur at distribution sites. In both side, different costs are involved which increases in stepwise manner over discrete time interval. Thus, this problem can also be extended to “Single machine

total stepwise earliness/tardiness problem with release dates". Future researchers can explore different aspects of these two new problems.

As already mentioned, the first three problems have only permutation characteristics. Out of these three problems, the first two problems have resemblance with *TSP* and can be considered as adjacency based permutation problems. On the other hand, the third one being a job scheduling problem, has resemblance with order-based permutation problems.

In Chapter 5, we have presented a steady-state grouping genetic algorithm based approach to address the rescue unit allocation and scheduling problem (*RUASP*) with fuzzy processing times. The *RUASP* problem intends to find the efficient assignment and scheduling of rescue units for processing the incidents in the event of natural disasters. Thus, it addresses a very critical issue in emergency response management. It can also be generalized as a variant of the unrelated parallel-machine scheduling problem with sequence and machine-dependent setup times. The crossover and mutation operator used in our approach are designed keeping in mind the characteristics of the problem and the objective. Thus, these operators deliver better solutions for *RUASP*. The combination of greedy and random based heuristics in initial solution generation provides a better quality diverse initial solutions. Such a combination of heuristics results in faster convergence of approach and also generates superior quality final solutions. We have compared the solutions obtained by our approach with the existing approaches (heuristics and *GRASP* presented in [113], *BRKGA* of [114]) for *RUASP* available in the literature. This comparison clearly demonstrates the superiority of the proposed approach both in terms of solution quality and execution time.

The superior performance of our approach serves as a motivation to design similar approaches to address other variants of parallel machine scheduling problems with sequence and machine dependent setup times. Analogous evolutionary techniques utilizing a combination of greedy and random heuristics to find better quality diverse initial solutions and the problem characteristics based crossover and mutation operators can be proposed to address other combinatorial optimization problems also.

RUASP can be generalized as the vehicle routing problem with no restriction on the capacity of vehicles. Inclusion of time window restriction to process the incidents is more suitable for a problem involving immediate actions in the event of an emergency. Hence, *RUASP* can be augmented with time window constraints to make the model more realistic. This new version of *RUASP* can be generalized as a particular case of vehicle routing problems with time windows (VRPTW). The literature of VRPTW contains an objective intended for balancing the driver's

workload. This objective seeks minimization of makespan, i.e., the longest travel time among all routes. In the case of *RUASP*, this objective appears to be more relevant. The productivity of rescue units may drop if the workload is not distributed uniformly among the rescue units. As a future work, *RUASP* can be presented as a multiobjective (bi-objective) problem by including two objectives, viz. weighted completion time and minimization of makespan.

Chapter 6 presents a steady-state grouping genetic algorithm based approach named as *GGA-QOS* to solve the vehicle routing problem with time windows designed for objectives seeking the quality of service to the customers *QSVRPTW* which was introduced in [160]. The *GGA-QOS* approach presented here is an extended version of *GGA* approach proposed in the previous chapter. The addressed problem is a variant of vehicle routing problem with time windows (VRPTW) from the perspective of quality of service delivered to customers in transport related problems. *QSVRPTW* has more relevance in real-world transport related applications, especially those involving urgency in supply of goods. In the proposed approach, the crossover and mutation operators are designed by considering the characteristics of QSVRPTW as well as the characteristics of the objectives. We have also proposed two bounds for each objective, which can be used to assess the quality of solutions obtained by an approach. The computational results demonstrate the superiority of the proposed *GGA-QOS* approach over the state-of-the-art approach, viz. GRASP-VNS approach of [160]. The comparison reveals that our proposed approach is not only faster than the state-of-the-art approach, but also fetches better results. In addition, we have reported the results of our approach on large instances. The *GGA-QOS* approach is able to find feasible solution for tougher instances also, which is an additional advantage of the proposed approach.

We have simplified the *QSVRPTW* by showing two of the three objectives equivalent. Further, the bounds proposed by us prove some of the results obtained through GRASP-VNS wrong.

The superior performance of the *GGA-QOS* approach serves as a motivation to design similar approaches to address the other variants of vehicle routing problem with time windows. Analogous evolutionary techniques utilizing problem-specific heuristics to find better quality initial solutions and objective defined variation operators can be proposed to address various combinatorial optimization problems. Instead of considering each objective separately, we can also study *QSVRPTW* by considering all the objectives simultaneously as a multiobjective problem.

Chapter 7 presents a nondominated sorting genetic algorithm II (NSGA-II) based approach with objective-specific variation operators to address the multiobjective vehicle routing problem with time windows (*MOVRPTW*). This problem is a variant of multiobjective vehicle routing problem (MOVRP). *MOVRPTW* has more relevance in real-world applications and consists of five conflicting objectives. In the proposed *NSGA-II* approach, the crossover and mutation operators are designed by exploiting the problem-specific knowledge as well as the characteristics of each objective. The experimental results show that the set of nondominated solutions obtained by the proposed approach is better than *LSMOVRPTW* approach in terms of both convergence and diversity, thereby clearly demonstrating the effectiveness of the proposed *NSGA-II* approach in comparison to the state-of-the-art approach. Performance of our approach, particularly in terms of C-metric, can be improved further at the expense of increased execution times if the objective-specific local searches like those in [193] are incorporated in our approach.

The performance of the proposed approach provides inspiration for developing similar approaches for other variants of multiobjective vehicle routing problem. In particular, the proposed approach can be applied to similar problems presented in [235] and [194]. Similar multiobjective evolutionary algorithm approaches utilizing objective-specific variation operators can be designed for a wide range of MOPs.

Both Chapter 6 and Chapter 7 addressed the problems which are variants of vehicle routing problem with time windows. Still, they have many significant differences in terms of objectives and the constraints. The *QSVRPTW* problem discussed in Chapter 6 considers a specific number of vehicles for each instance, whereas the *MOVRPTW* presented in Chapter 7 has no such restriction. In fact, minimization of number of vehicles is one of the objective in *MOVRPTW*. Due to the restriction of fix number of vehicles in *QSVRPTW*, the application of crossover and mutation operator may result in solutions not satisfying the time window constraints. The possibility of occurrence of this situation is more in tougher instances, where due to less number of vehicles the customers need to be inserted in a very compact manner. Our *GGA-QOS* approach is able to handle such situations also. On the other hand, *NSGA-II* approach handles the aforementioned situation by adding a new route and the customer involved in violation of constraint(s) is inserted at the beginning of newly added route. Thus, in *MOVRPTW*, the application of crossover and mutation always generate a constraints satisfying solution. *MOVRPTW* being multiobjective, our proposed *NSGA-II* approach maintains an adequate balance between exploration and exploitation due to its objective-specific variation

operators. Thus, the developed approaches for both problems can be extended to other variants of the vehicle routing problems.

The six problems considered in this thesis cover a set of recently introduced real-world permutation based problems. Although, all of these problems possess permutation characteristics, the semantic of permutation with respect to objective of the problem is different. The insight gained from this thesis can be useful for other similar real-world permutation based problems. Since, most of the real-world problems differ in objective and the constraints associated with them.

Genetic algorithm is the one of the most popular evolutionary approach for solving combinatorial optimization problems. In fact, it is one among the most successful metaheuristic approaches. Based on the work reported in this thesis, we can say that evolution strategy and discrete differential evolution provide an attractive alternative to genetic algorithm for solving pure permutation based problems. For permutation based problems, already discrete differential evolution has many successful recent applications in literature.

Among the proposed approaches, evolution strategy can be more useful for adjacency-based permutation problems. On the other hand, discrete differential evolution approach can be more suitable for order-based permutation problems. The grouping genetic algorithm incorporated with problem-specific knowledge is found to be highly successful in comparison with other well-known approaches such as *GRASP-VNS*, *BRKGA* etc. The *NSGA-II* approach is already shown as most successful approach for multiobjective problems in the literature, we have made it even more powerful by incorporating objective-specific variation operators. Similar objective-specific variation operators can be incorporated into other multiobjective metaheuristic techniques.

Although, the approaches proposed in this thesis make use of problem-specific knowledge in their various components. However, this fact does not rule out the possibility that for each problem there might be some yet to be explored characteristics that need to be discovered and analyzed further. This, in turn, will provide fresh opportunities to develop new heuristic and metaheuristic approaches and improve existing ones for each problem.

References

[1] K. MENGER. **Das botenproblem**. *Ergebnisse eines mathematischen kolloquiums*, **2**:11–12, 1932. (2)

[2] D.L. APPLEGATE, R.E. BIXBY, V. CHVATAL, AND W.J. COOK. *The traveling salesman problem: a computational study*. Princeton university press, 2006. (2)

[3] G. GUTIN AND A.P. PUNNEN. *The traveling salesman problem and its variations*, **12**. Springer Science & Business Media, 2006. (2)

[4] A.P. PUNNEN. **The traveling salesman problem: Applications, formulations and variations**. In *The traveling salesman problem and its variations*, pages 1–28. Springer, 2007. (2)

[5] S.M. JOHNSON. **Optimal two-and three-stage production schedules with setup times included**. *Naval research logistics quarterly*, **1**(1):61–68, 1954. (2)

[6] M. PINEDO. *Scheduling*, **29**. Springer, 2012. (2)

[7] T.C. KOOPMANS AND M. BECKMANN. **Assignment problems and the location of economic activities**. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957. (2)

[8] G. FINKE, R.E. BURKARD, AND F. RENDL. **Quadratic assignment problems**. In *North-Holland Mathematics Studies*, **132**, pages 61–82. Elsevier, 1987. (2, 3)

[9] J. CEBERIO, E. IRUROZKI, A. MENDIBURU, AND J.A. LOZANO. **A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems**. *Progress in Artificial Intelligence*, **1**(1):103–117, 2012. (3)

[10] H.B. CHENERY AND T. WATANABE. **International comparisons of the structure of production**. *Econometrica: Journal of the Econometric Society*, pages 487–521, 1958. (3)

[11] M. DORIGO, V.O. MANIEZZO, AND A. COLORNI. **Ant system: Optimization by a colony of cooperating agents**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **26**:29–41, 1996. (4, 5)

[12] D. GOLDBERG. *Genetic algorithm in search, optimization and machine learning*. Reading, MA :Addison-Wesley, 1989. (5)

[13] J.H. HOLLAND. *Adaptation in natural and artificial systems: An introductory analysis with applications in biology, control and artificial intelligence*. University of Michigan Press, Ann Arbor, MI, 1975. (5, 6, 7, 11, 13)

[14] M. DORIGO, V. MANIEZZO, AND A. COLORNI. **Positive feedback as a search strategy**, 1991. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy. (5)

[15] D. KARABOGA. **An idea based on honey bee swarm for numerical optimization**, 2005. Computer Engineering Department, Erciyes University, Turkey. (5)

[16] N. MLADENOVIĆ AND P. HANSEN. **Variable neighborhood search**. *Computers & operations research*, **24**(11):1097–1100, 1997. (5, 124)

[17] P. HANSEN AND N. MLADENOVIĆ. **Variable neighborhood search: Principles and applications**. *European journal of operational research*, **130**(3):449–467, 2001. (5)

[18] R. STORN AND K. PRICE. **Differential evolution -a simple and efficient heuristic for global optimization over continuous spaces**. *Journal of global optimization*, **11**(4):341–359, 1997. (5, 25)

[19] M.F. TASGETIREN, Q.-K. PAN, Y.-C. LIANG, AND P.N. SUGANTHAN. **A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single-machine**. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 271–278. IEEE, 2007. (5, 25)

[20] I. RECHENBERG. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann Holzboog Verlag, Stuttgart, 1973. (5, 22, 23)

[21] H.-P. SCHWEFEL. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universität Berlin, 1975. (5, 22)

[22] L.J. FOGEL. **Autonomous automata**. *Industrial research*, **4**:14–19, 1962. (6)

[23] M. DIANATI, I. SONG, AND M. TREIBER. **An introduction to genetic algorithms and evolution strategies**. Technical report, Citeseer, 2002. (6)

[24] D.B. FOGEL. **A comparison of evolutionary programming and genetic algorithms on selected constrained optimization problems**. *Simulation*, **64**(6):397–404, 1995. (6)

[25] M.A. LONES. **Metaheuristics in nature-inspired algorithms**. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 1419–1422, 2014. (6)

[26] R.C. MARTÍ, P.M. PARDALOS, AND M.G.C. RESENDE. *Handbook of heuristics*. Springer, 2018. (6, 7)

[27] R.R. SHARAPOV. *Genetic algorithms: basic ideas, variants and analysis*. IntechOpen, 2007. (7)

[28] T. BLICKLE AND L. THIELE. **A mathematical analysis of tournament selection**. In *ICGA*, **95**, pages 9–15. Citeseer, 1995. (7)

[29] J.D. SCHAFFER, D. WHITLEY, AND L.J. ESHELMAN. **Combinations of genetic algorithms and neural networks: A survey of the state of the art**. In *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37. IEEE, 1992. (10)

[30] M. MITCHELL. *An introduction to genetic algorithms*. Bradford Books, 1998. (11)

[31] J.E. BAKER. **Reducing bias and inefficiency in the selection algorithm**. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987. (11)

[32] D.E. GOLDBERG AND K. DEB. **A comparative analysis of selection schemes used in genetic algorithms**. In *Foundations of Gentic Algorithms*, pages 69–93. Morgan Kaufmann, 1990. (11, 12)

[33] G. SYSWERDA. **Uniform crossover in genetic algorithms**. In *Proceedings of the Third International Conference on Genetic Algorithms*, **3**, pages 2–9. Morgan Kaufmann, 1989. (13)

[34] L. DAVIS. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991. (15, 17, 77, 100)

[35] A.E. EIBEN AND J.E. SMITH. *Introduction to evolutionary computing*. Springer, New york, 2003. (16)

[36] D.E. GOLDBERG AND R. LINGLE. **Alleles, loci and the traveling salesman problem**. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 154–159, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc. (16)

[37] I.M. OLIVER, D.J. SMITH, AND J.R.C. HOLLAND. **A study of permutation crossover operators on the travelling salesman problem**. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987. (18)

[38] D. WHITLEY, T. STARKWEATHER, AND D. SHANER. *The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination*. Citeseer, 1991. (18)

[39] H. MÜHLENBEIN. **Parallel genetic algorithms in combinatorial optimization**. In *Computer science and operations research*, pages 441–453. Elsevier, 1992. (20)

[40] E. FALKENAUER. **The grouping genetic algorithm**. In *State of the art in global optimization*, pages 249–265. Springer, 1996. (20, 21, 22, 100)

[41] E. FALKENAUER. **A hybrid grouping genetic algorithm for bin packing**. *Journal of heuristics*, **2**(1):5–30, 1996. (20, 100)

[42] A. SINGH AND A.K. GUPTA. **Two heuristics for the one-dimensional bin-packing problem**. *OR Spectrum*, **29**(4):765–781, 2007. (21, 100, 105, 131, 142, 167)

[43] E. MEZURA-MONTES, A.H. AGUIRRE, AND C.A.C. COELLO. **Using evolution strategies to solve constrained optimization problems**. In *Evolutionary Algorithms and Intelligent Tools in Engineering Optimization*, pages 1–25. WIT Press, CIMNE Barcelona, 2005. (22)

[44] H.-P. SCHWEFEL. *Numerische optimierung von computer-modellen mittels der evolution-sstrategie: mit einer vergleichenden einführung in die hill-climbing-und zufallsstrategie*, **1**. Springer, 1977. (23)

[45] T. BARTZ-BEIELSTEIN. **Evolution strategies and threshold selection**. In *International Workshop on Hybrid Metaheuristics*, **3636**, pages 104–115. Springer, 2005. (23)

[46] T. BÄCK, F. HOFFMEISTER, AND H.-P. SCHWEFEL. **A survey of evolution strategies**. In *Proceedings of the fourth international conference on genetic algorithms*, **2**, pages 2–9. Morgan Kaufmann, 1991. (23)

[47] J. CAI AND G. THIERAUF. **A parallel evolution strategy for solving discrete structural optimization**. *Advances in Engineering Software*, **27**(1-2):91–96, 1996. (24)

[48] J. CAI AND G. THIERAUF. **Evolution strategies for solving discrete optimization problems**. *Advances in Engineering Software*, **25**(2):177–183, 1996. (24)

[49] A. AHRARI AND O. KRAMER. **Finite life span for improving the selection scheme in evolution strategies**. *Soft Computing*, **21**(2):501–513, 2017. (24)

[50] H.-G. BEYER AND B. SENDHOFF. **Toward a steady-state analysis of an evolution strategy on a robust optimization problem with noise-induced multimodality**. *IEEE Transactions on Evolutionary Computation*, **21**(4):629–643, 2017. (24)

[51] V.N. COELHO, I.M. COELHO, M.J.F. SOUZA, T.A. OLIVEIRA, L.P. COTA, M.N. HADDAD, N. MLADENOVIC, R.C.P. SILVA, AND F.G. GUIMARÃES. **Hybrid self-adaptive evolution strategies guided by neighborhood structures for combinatorial optimization problems**. *Evolutionary computation*, **24**(4):637–666, 2016. (24)

[52] A.H. KASHAN, A.A. AKBARI, AND B. OSTADI. **Grouping evolution strategies: An effective approach for grouping problems**. *Applied Mathematical Modelling*, **39**(9):2703–2720, 2015. (24)

[53] D. WIERSTRA, T. SCHAUL, T. GLASMACHERS, Y. SUN, J. PETERS, AND J. SCHMIDHUBER. **Natural evolution strategies**. *Journal of Machine Learning Research*, **15**(1):949–980, 2014. (24)

[54] R. MALLIPEDDI, P.N. SUGANTHAN, Q.-K. PAN, AND M.F. TASGETIREN. **Differential evolution algorithm with ensemble of parameters and mutation strategies**. *Applied soft computing*, **11**(2):1679–1696, 2011. (25)

[55] Q.-K. PAN, M.F. TASGETIREN, AND Y.-C. LIANG. **A discrete differential evolution algorithm for the permutation flowshop scheduling problem**. *Computers & Industrial Engineering*, **55**(4):795–816, 2008. (25, 26, 45)

[56] S. YUAN, T. LI, AND B. WANG. **A discrete differential evolution algorithm for flow shop group scheduling problem with sequence-dependent setup and transportation times**. *Journal of Intelligent Manufacturing*, pages 1–13, 2020. (25)

[57] L. WANG, Q.-K. PAN, P.N. SUGANTHAN, W.-H. WANG, AND Y.-M. WANG. **A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems**. *Computers & Operations Research*, **37**(3):509–520, 2010. (26)

[58] M.F. TASGETIREN, Q.-K. PAN, AND Y.-C. LIANG. **A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times**. *Computers & Operations Research*, **36**(6):1900–1915, 2009. (26)

[59] Y. YUAN AND H. XU. **Flexible job shop scheduling using hybrid differential evolution algorithms**. *Computers & Industrial Engineering*, **65**(2):246–260, 2013. (26)

[60] M.F. TASGETIREN, Q.-K. PAN, P.N. SUGANTHAN, AND Y.-C. LIANG. **A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion**. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 251–258. IEEE, 2007. (26)

[61] L. TANG, Y. ZHAO, AND J. LIU. **An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production**. *IEEE Transactions on Evolutionary Computation*, **18**(2):209–225, 2013. (26)

[62] A.C. NEARCHOU AND S.L. OMIROU. **Differential evolution for sequencing and scheduling optimization**. *Journal of Heuristics*, **12**(6):395–411, 2006. (26)

[63] M. NAWAZ, E.E. ENSCORE, AND I. HAM. **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**. *Omega*, **11**(1):91–95, 1983. (27, 38, 72)

[64] S. RAHNAMAYAN, H.R. TIZHOOSH, AND M.M.A. SALAMA. **Opposition-based differential evolution**. *IEEE Transactions on Evolutionary computation*, **12**(1):64–79, 2008. (27, 187)

[65] Q. XU, L. GUO, N. WANG, J. PAN, AND L. WANG. **A novel oppositional biogeography-based optimization for combinatorial problems**. In *Natural Computation (ICNC), 2014 10th International Conference on*, pages 412–418. IEEE, 2014. (28)

[66] Q. XU, L. WANG, N. WANG, X. HEI, AND L. ZHAO. **A review of opposition-based learning from 2005 to 2012**. *Engineering Applications of Artificial Intelligence*, **29**:1–12, 2014. (28)

[67] S. RODZIN AND O. RODZINA. **New computational models for big data and optimization**. In *Application of Information and Communication Technologies (AICT), 2015 9th International Conference on*, pages 3–7. IEEE, 2015. (28)

[68] M. ERGEZER AND D. SIMON. **Oppositional biogeography-based optimization for combinatorial problems**. In *2011 IEEE Congress on Evolutionary Computation (CEC)*, pages 1496–1503. IEEE, 2011. (28)

[69] J. ZHAO, L. LV, AND H. SUN. **Artificial bee colony using opposition-based learning**. In *Genetic and evolutionary computing*, pages 3–10. Springer, 2015. (28)

[70] I.F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, AND E. CAYIRCI. **A survey on sensor networks**. *IEEE communications magazine*, **40**(8):102–114, 2002. (36)

[71] V. RAGHUNATHAN, C. SCHURGERS, S. PARK, AND M.B. SRIVASTAVA. **Energy-aware wireless microsensor networks**. *IEEE Signal processing magazine*, **19**(2):40–50, 2002. (36)

[72] L. BENINI, D. BRUNI, A. MACH, E. MACII, AND M. PONCINO. **Discharge current steering for battery lifetime optimization**. *IEEE Transactions on Computers*, **52**(8):985–995, 2003. (36)

[73] K.-Y. CHOW, K.-S. LUI, AND E.Y. LAM. **Wireless sensor networks scheduling for full angle coverage**. *Multidimensional Systems and Signal Processing*, **20**(2):101–119, 2009. (36)

[74] M.X. CHENG, L. RUAN, AND W. WU. **Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks**. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, **4**, pages 2638–2645. IEEE, 2005. (37)

[75] C. WANG, M.T. THAI, Y. LI, F. WANG, AND W. WU. **Optimization scheme for sensor coverage scheduling with bandwidth constraints**. *Optimization letters*, **3**(1):63–75, 2009. (37)

[76] A. ROSSI, A. SINGH, AND M. SEVAUX. **Column generation algorithm for sensor coverage scheduling under bandwidth constraints**. *Networks*, **60**(3):141–154, 2012. (37, 38, 48)

[77] M. GENTILI AND A. RAICONI. **α-coverage to extend network lifetime on wireless sensor networks**. *Optimization Letters*, **7**(1):157–172, 2013. (38)

[78] A. ROSSI, M. SEVAUX, A. SINGH, AND M.J. GEIGER. **On the cover scheduling problem in wireless sensor networks**. In *Proceedings of the 5th International Networks Optimization Conference, Lecture Notes in Computer Science*, **6701**, pages 657–668, Hamburg, Germany, 2011. Springer-Verlag. (38, 39, 41, 42, 43, 48, 50, 186)

[79] V. GOPINADH AND A. SINGH. **Swarm intelligence approaches for cover scheduling problem in wireless sensor networks**. *International Journal of Bio-Inspired Computation*, **7**(1):50–61, 2015. (38, 39, 42, 44, 48, 50, 186)

[80] M. ČREPINŠEK, S.-H. LIU, AND M. MERNIK. **Exploration and exploitation in evolutionary algorithms: A survey**. *ACM Computing Surveys (CSUR)*, **45**(35):1–33, 2013. (44, 61)

[81] C. SOLNON. **Boosting ACO with a preprocessing step**. In *Workshops on Applications of Evolutionary Computation*, **2279**, pages 163–172. Springer, 2002. (44, 62)

[82] P. MERZ AND B. FREISLEBEN. **Fitness landscapes and memetic algorithm design**. *New ideas in optimization*, pages 245–260, 1999. (44, 62)

[83] S.N. CHAURASIA AND A. SINGH. **A hybrid swarm intelligence approach to the registration area planning problem**. *Information Sciences*, **302**:50–69, 2015. (45, 107)

[84] Q.-K. PAN, M.F. TASGETIREN, P.N. SUGANTHAN, AND T.J. CHUA. **A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem**. *Information sciences*, **181**(12):2455–2468, 2011. (45)

[85] F.J. RODRÍGUEZ, M. LOZANO, C. GARCÍA-MARTÍNEZ, AND J.D. GONZÁLEZ-BARRERA. **An artificial bee colony algorithm for the maximally diverse grouping problem**. *Information Sciences*, **230**:183–196, 2013. (45)

[86] J.A. DELGADO-OSUNA, M. LOZANO, AND C. GARCÍA-MARTÍNEZ. **An alternative artificial bee colony algorithm with destructive–constructive neighbourhood operator for the problem of composing medical crews**. *Information Sciences*, **326**:215–226, 2016. (45)

[87] R. RUIZ AND T. STÜTZLE. **A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem**. *European Journal of Operational Research*, **177**(3):2033–2049, 2007. (45)

[88] M.F. TASGETIREN, Q.-K. PAN, P.N. SUGANTHAN, AND A.H. CHEN. **A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops**. *Information sciences*, **181**(16):3459–3475, 2011. (45, 107)

[89] R. SZEWCZYK, A. MAINWARING, J. POLASTRE, J. ANDERSON, AND D. CULLER. **An analysis of a large scale habitat monitoring application**. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226. ACM, 2004. (55)

[90] J. DJUGASH, S. SINGH, G. KANTOR, AND W. ZHANG. **Range-only slam for robots operating cooperatively with sensor networks**. In *Robotics and Automation (ICRA*

2006), Proceedings of IEEE International Conference on, 5, pages 2078–2084. IEEE, 2006. (55)

[91] M. RAHIMI, R. BAER, O.I. IROEZI, J.C. GARCIA, J. WARRIOR, D. ESTRIN, AND M. SRIVASTAVA. **Cyclops: In situ image sensing and interpretation in wireless sensor networks**. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192–204. ACM, 2005. (55)

[92] M.A. GUVENSAN AND A.G. YAVUZ. **On coverage issues in directional sensor networks: A survey**. *Ad Hoc Networks*, **9**(7):1238–1255, 2011. (55)

[93] A. SINGH AND A. ROSSI. **A genetic algorithm based exact approach for lifetime maximization of directional sensor networks**. *Ad Hoc Networks*, **11**(3):1006–1021, 2013. (55)

[94] A. ROSSI, A. SINGH, AND M. SEVAUX. **Lifetime maximization in wireless directional sensor network**. *European Journal of Operational Research*, **231**(1):229–241, 2013. (55, 65)

[95] Y. CAI, W. LOU, M. LI, AND X.-Y. LI. **Target-oriented scheduling in directional sensor networks**. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications.*, pages 1550–1558. IEEE, 2007. (55)

[96] Y. CAI, W. LOU, M. LI, AND X.-Y. LI. **Energy efficient target-oriented scheduling in directional sensor networks**. *IEEE Transactions on Computers*, **58**(9):1259–1274, 2009. (55)

[97] A. MAKHOUL, R. SAADI, AND C. PHAM. **Adaptive scheduling of wireless video sensor nodes for surveillance applications**. In *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 54–60. ACM, 2009. (55)

[98] H. YANG, D. LI, AND H. CHEN. **Coverage quality based target-oriented scheduling in directional sensor networks**. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010. (55)

[99] J.-M. GIL AND Y.-H. HAN. **A target coverage scheduling scheme based on genetic algorithms in directional sensor networks**. *Sensors*, **11**(2):1888–1906, 2011. (55)

[100] A. SINGH AND A. ROSSI. **Group scheduling problems in directional sensor networks**. *Engineering Optimization*, **47**(12):1651–1669, 2015. (56, 57, 58, 61, 65, 187)

[101] C.-T. TSENG AND K.-H. CHEN. **An electromagnetism-like mechanism for the single machine total stepwise tardiness problem with release dates**. *Engineering Optimization*, **45**(12):1431–1448, 2013. (71, 72, 73, 74, 76, 82, 83, 84)

[102] S.N. CHAURASIA, S. SUNDAR, AND A. SINGH. **Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates**. *Operational Research*, **17**(1):275–295, 2017. (71, 73, 74, 76, 79, 82, 83, 84)

[103] J. CURRY AND B. PETERS. **Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives**. *International Journal of Production Research*, **43**(15):3231–3246, 2005. (71)

[104] G. SAHIN. *New combinatorial approaches for solving railroad planning and scheduling problems*. Phd dissertation, University of Florida, 2006. (71)

[105] J.M. MOORE. **An n job, one machine sequencing algorithm for minimizing the number of late jobs**. *Management science*, **15**(1):102–109, 1968. (72, 76)

[106] S.I. BIRBIL AND S.C. FANG. **Electromagnetism-like mechanism for global optimization**. *Journal of Global Optimization*, **25**:263–282, 2003. (73)

[107] J.K. LENSTRA, A.H.G. RINNOOY KAN, AND P. BRUCKER. **Complexity of machine scheduling problems**. In *Annals of discrete mathematics*, **1**, pages 343–362. Elsevier, 1977. (74)

[108] R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, AND A.H.G. RINNOOY KAN. **Optimization and approximation in deterministic sequencing and scheduling: a survey**. In *Annals of discrete mathematics*, **5**, pages 287–326. Elsevier, 1979. (74)

[109] F. WILCOXON, S.K. KATTI, AND R.A. WILCOX. **Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test**. *Selected Tables in Mathematical Statistics*, **1**:171–259, 1970. (87)

[110] A.S. PRASAD AND L.H. FRANCESCUTTI. **Natural disasters**. *International encyclopedia of public health*, **5**(2):215–222, 2017. (89)

[111] R. ABOUNACER, M. REKIK, AND J. RENAUD. **An exact solution approach for multi-objective location-transportation problem for disaster response**. *Computers & Operations Research*, **41**:83–93, 2014. (90)

[112] L.K. COMFORT, K. KO, AND A. ZAGORECKI. **Coordination in rapidly evolving disaster response systems: The role of information**. *American behavioral scientist*, **48**(3):295–313, 2004. (90)

[113] F. WEX, G. SCHRYEN, S. FEUERRIEGEL, AND D. NEUMANN. **Emergency response in natural disaster management: Allocation and scheduling of rescue units**. *European Journal of Operational Research*, **235**(3):697–708, 2014. (90, 91, 92, 93, 96, 97, 98, 100, 102, 103, 108, 109, 111, 112, 113, 114, 190)

[114] V. CUNHA, L. PESSOA, M. VELLASCO, R. TANSCHEIT, AND M.A. PACHECO. **A biased random-key genetic algorithm for the rescue unit allocation and scheduling problem**. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2018. (90, 94, 95, 97, 98, 99, 100, 108, 109, 111, 190)

[115] S. AJAMI AND M. FATTAHI. **The role of earthquake information management systems (EIMSs) in reducing destruction: A comparative study of Japan, Turkey and Iran**. *Disaster Prevention and Management*, **18**(2):150–161, 2009. (95)

[116] N. ALTAY AND W.G. GREEN III. **OR/MS research in disaster operations management**. *European journal of operational research*, **175**(1):475–493, 2006. (95)

[117] P. GASPARINI, G. MANFREDI, AND J. ZSCHAU. *Earthquake early warning systems*. Springer, 2007. (96)

[118] F.N. DE SILVA. **Providing spatial decision support for evacuation planning: A challenge in integrating technologies**. *Disaster Prevention and Management*, **10**:11–20, 2001. (96)

[119] E. POLLAK, M. FALASH, L. INGRAHAM, AND V. GOTTESMAN. **Operational analysis framework for emergency operations center preparedness training**. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, **1**, page 848, 2004. (96)

[120] A. SVENSSON, J. HOLST, R. LINDQUIST, AND G. LINDGREN. **Optimal prediction of catastrophes in autoregressive moving-average processes**. *Journal of Time Series Analysis*, **17**(5):511–531, 1996. (96)

[121] K. SALEEM, S. LUIS, Y. DENG, S.-C. CHEN, V. HRISTIDIS, AND T. LI. **Towards a business continuity information network for rapid disaster recovery.** In *Proceedings of the 2008 international conference on digital government research*, pages 107–116. ACM international conference proceeding series, 2008. (96)

[122] H.D. SHERALI, T.B. CARTER, AND A.G. HOBEIKA. **A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions**. *Transportation Research Part B: Methodological*, **25**(6):439–452, 1991. (96)

[123] R. CHEN, R. SHARMAN, H.R. RAO, AND S.J. UPADHYAYA. **Coordination in emergency response management**. *Communications of the ACM*, **51**(5):66–73, 2008. (96)

[124] G. AIRY, T. MULLEN, AND J. YEN. **Market based adaptive resource allocation for distributed rescue teams**. In *Proceedings of the 6th conference on information systems for crisis response and management (ISRAM 2009)*. Gothenburg, Sweden, 2009. (96)

[125] T. COMES, C. CONRADO, M. HIETE, M. KAMERMANS, G. PAVLIN, AND N.J.E. WIJNGAARDS. **An intelligent decision support system for decision making under uncertainty in distributed reasoning frameworks**. In *Proceedings of the 7th international conference on information systems for crisis response and management (ISCRAM 2010)*, Seattle, USA, 2010. (96)

[126] J.H. LAMBERT AND C.E. PATTERSON. **Prioritization of schedule dependencies in hurricane recovery of transportation agency**. *Journal of Infrastructure Systems*, **8**(3):103–111, 2002. (96)

[127] H.A. REIJERS, M.H. JANSEN-VULLERS, M. ZUR MUEHLEN, AND W. APPL. **Workflow management systems+ swarm intelligence= dynamic task assignment for emergency management applications**. In *International Conference on Business Process Management (BPM2007)*, pages 125–140. Springer, 2007. (96)

[128] H. TAMURA, K. YAMAMOTO, S. TOMIYAMA, AND I. HATONO. **Modeling and analysis of decision making problem for mitigating natural disaster risks**. *European Journal of Operational Research*, **122**(2):461–468, 2000. (96)

[129] F. FIEDRICH, F. GEHBAUER, AND U. RICKERS. **Optimized resource allocation for emergency response after earthquake disasters**. *Safety science*, **35**(1-3):41–57, 2000. (96)

[130] O. LEIFLER. **Combining technical and human-centered strategies for decision support in command and control: The ComPlan approach**. In *Proceedings of the 5th international conference on information systems for crisis response and management (ISCRAM 2008)*, pages 504–515, Seattle, USA, 2008. (96)

[131] B. VAN DE WALLE AND M. TUROFF. **Decision support for emergency situations**. In *Handbook on decision support systems 2. International handbooks on information systems*, pages 39–63, Berlin, Heidelberg, 2008. Springer. (96)

[132] S. FARAJ AND Y. XIAO. **Coordination in fast-response organizations**. *Management science*, **52**(8):1155–1169, 2006. (96)

[133] M. FALASCA, C.W. ZOBEL, AND G.M. FETTER. **An optimization model for humanitarian relief volunteer management**. In *J. Landgren & S. Jul (Eds.), Proceedings of the 6th international conference on information systems for crisis response and management (ISCRAM 2008)*, Gothenburg, Sweden, 2009. (96)

[134] E. ROLLAND, R.A. PATTERSON, K. WARD, AND B. DODIN. **Decision support for disaster management**. *Operations Management Research*, **3**(1-2):68–79, 2010. (96)

[135] F. WEX, G. SCHRYEN, AND D. NEUMANN. **Operational emergency response under informational uncertainty: A fuzzy optimization model for scheduling and allocating rescue units**. In *L. Rothkrantz, J. Ristvej & Z. Franco (Eds.), Proceedings of the 9th international conference on information systems for crisis response and management (ISCRAM 2012)*, 2012. (96)

[136] F. WEX, G. SCHRYEN, AND D. NEUMANN. **Decision modeling for assignments of collaborative rescue units during emergency response**. In *Proceedings of the 46th Hawaii International Conference on System Sciences*, pages 166–175. IEEE, 2013. (96)

[137] T.A. FEO AND M.G.C. RESENDE. **A probabilistic heuristic for a computationally difficult set covering problem**. *Operations research letters*, **8**(2):67–71, 1989. (97, 124)

[138] P. FESTA AND M.G.C. RESENDE. **GRASP**. In *Handbook of Heuristics*, pages 1–24. Springer, 2016. (98)

[139] J.-S. YAO AND K. WU. **Ranking fuzzy numbers based on decomposition principle and signed distance**. *Fuzzy sets and Systems*, **116**(2):275–288, 2000. (99)

[140] S. BALIN. **Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation**. *Information Sciences*, **181**(17):3551–3569, 2011. (99)

[141] N. VAN HOP. **A heuristic solution for fuzzy mixed-model line balancing problem**. *European Journal of Operational Research*, **168**(3):798–810, 2006. (99)

[142] A. SINGH AND A.S. BAGHEL. **A new grouping genetic algorithm approach to the multiple traveling salesperson problem**. *Soft Computing*, **13**(1):95–101, 2009. (105, 142, 167)

[143] B. ROY. **Robustness in operational research and decision aiding: A multi-faceted issue**. *European Journal of Operational Research*, **200**(3):629–638, 2010. (111)

[144] D.F. WILLIAMSON, R.A. PARKER, AND J.S. KENDRICK. **The box plot: a simple visual method to interpret data**. *Annals of internal medicine*, **110**(11):916–921, 1989. (113)

[145] A. HOFF, H. ANDERSSON, M. CHRISTIANSEN, G. HASLE, AND A. LØKKETANGEN. **Industrial aspects and literature survey: Fleet composition and routing**. *Computers & Operations Research*, **37**(12):2041–2061, 2010. (117)

[146] N. JOZEFOWIEZ, F. SEMET, AND E.-G. TALBI. **Multi-objective vehicle routing problems**. *European journal of operational research*, **189**(2):293–309, 2008. (117, 118)

[147] M.R. GAREY AND D.S. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979. (117)

[148] J.K. LENSTRA AND A.H.G. RINNOOY KAN. **Complexity of vehicle routing and scheduling problems**. *Networks*, **11**(2):221–227, 1981. (117)

[149] G.B. DANTZIG AND J.H. RAMSER. **The truck dispatching problem**. *Management science*, **6**(1):80–91, 1959. (117)

[150] W.H. IP, D. WANG, AND V. CHO. **Aircraft ground service scheduling problems and their genetic algorithm with hybrid assignment and sequence encoding scheme**. *IEEE Systems Journal*, **7**(4):649–657, 2013. (117)

[151] H.-K. CHEN, H.-W. CHOU, C.-F. HSUEH, AND T.-Y. HO. **The linehaul-feeder vehicle routing problem with virtual depots**. *IEEE Transactions on Automation Science and Engineering*, **8**(4):694–704, 2011. (117)

[152] B. EKSIOGLU, A.V. VURAL, AND A. REISMAN. **The vehicle routing problem: A taxonomic review**. *Computers & Industrial Engineering*, **57**(4):1472–1483, 2009. (117)

[153] M. GMIRA, M. GENDREAU, A. LODI, AND J.-Y. POTVIN. **Tabu search for the time-dependent vehicle routing problem with time windows on a road network**. *European Journal of Operational Research*, **288**(1):129–140, 2021. (117)

[154] J.C. MOLINA, J.L. SALMERON, AND I. EGUIA. **An ACS-based memetic algorithm for the heterogeneous vehicle routing problem with time windows**. *Expert Systems with Applications*, **157**:113379, 2020. (117)

[155] E. OSABA, X.-S. YANG, AND J. DEL SER. **Is the vehicle routing problem dead? An overview through bioinspired perspective and a prospect of opportunities**. In *Nature-Inspired Computation in Navigation and Routing Problems*, pages 57–84. Springer, 2020. (117)

[156] R.F. FACHINI AND V.A. ARMENTANO. **Logic-based Benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows**. *Computers & Industrial Engineering*, **148**:106641, 2020. (117)

[157] M.M. SOLOMON AND J. DESROSIERS. **Survey paper-time window constrained routing and scheduling problems**. *Transportation science*, **22**(1):1–13, 1988. (117)

[158] V. KORABLEV, I. MAKEEV, E. KHARITONOV, B. TSHUKIN, AND I. ROMANOV. **Approaches to solve the vehicle routing problem in the valuables delivery domain**. *Procedia Computer Science*, **88**:487–492, 2016. (117)

[159] G.M. GIAGLIS, I. MINIS, A. TATARAKIS, AND V. ZEIMPEKIS. **Minimizing logistics risk through real-time vehicle routing and mobile technologies: Research to**

date and future trends. *International Journal of Physical Distribution & Logistics Management*, **34**(9):749–764, 2004. (118)

[160] A. EXPÓSITO, J. BRITO, J.A. MORENO, AND C. EXPÓSITO-IZQUIERDO. **Quality of service objectives for vehicle routing problem with time windows**. *Applied Soft Computing*, **84**:105707, 2019. (118, 119, 121, 122, 123, 124, 125, 131, 132, 145, 146, 148, 149, 191)

[161] G.S. SURESHCHANDAR, C. RAJENDRAN, AND R.N. ANANTHARAMAN. **The relationship between management's perception of total quality service and customer perceptions of service quality**. *Total Quality Management*, **13**(1):69–88, 2002. (118)

[162] J.T. MENTZER, W. DEWITT, J.S. KEEBLER, S. MIN, N.W. NIX, C.D. SMITH, AND Z.G. ZACHARIA. **Defining supply chain management**. *Journal of Business logistics*, **22**(2):1–25, 2001. (118)

[163] S. LIMBOURG, H.T.Q. GIANG, AND M. COOLS. **Logistics service quality: the case of Da Nang city**. *Procedia engineering*, **142**:124–130, 2016. (118)

[164] J.T. MENTZER, D.J. FLINT, AND G.T.M. HULT. **Logistics service quality as a segment-customized process**. *Journal of marketing*, **65**(4):82–104, 2001. (118)

[165] S.J. JAMES, C. JAMES, AND J.A. EVANS. **Modelling of food transportation systems-a review**. *International Journal of Refrigeration*, **29**(6):947–957, 2006. Issue with special emphasis on data and models on food refrigeration. (118)

[166] O. BRÄYSY AND M. GENDREAU. **Vehicle routing problem with time windows, Part I: Route construction and local search algorithms**. *Transportation science*, **39**(1):104–118, 2005. (119)

[167] R. BALDACCI, A. MINGOZZI, AND R. ROBERTI. **Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints**. *European Journal of Operational Research*, **218**(1):1–6, 2012. (119)

[168] P. TOTH AND D. VIGO. *Vehicle routing: problems, methods, and applications*. SIAM, 2014. (119)

[169] J. BRITO, D. CASTELLANOS-NIEVES, A. EXPÓSITO, AND J.A. MORENO. **Soft computing methods in transport and logistics**. In *Soft Computing Based Optimization and Decision Models*, pages 45–61. Springer, 2018. (119)

[170] J.L. VERDEGAY, R.R. YAGER, AND P.P. BONISSONE. **On heuristics as a fundamental constituent of soft computing**. *Fuzzy sets and systems*, **159**(7):846–855, 2008. (119)

[171] R. BENT AND P. VAN HENTENRYCK. **A two-stage hybrid local search for the vehicle routing problem with time windows**. *Transportation Science*, **38**(4):515–530, 2004. (119)

[172] O. BRÄYSY, G. HASLE, AND W. DULLAERT. **A multi-start local search algorithm for the vehicle routing problem with time windows**. *European Journal of Operational Research*, **159**(3):586 – 605, 2004. (119)

[173] Y. GONG, J. ZHANG, O. LIU, R. HUANG, H.S. CHUNG, AND Y. SHI. **Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach**. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **42**(2):254–267, 2012. (119, 153)

[174] A. LIM AND X. ZHANG. **A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows**. *INFORMS Journal on Computing*, **19**(3):443–457, 2007. (119)

[175] P.P. REPOUSSIS, C.D. TARANTILIS, AND G. IOANNOU. **Arc-guided evolutionary algorithm for the vehicle routing problem with time windows**. *IEEE Transactions on Evolutionary Computation*, **13**(3):624–647, 2009. (119)

[176] Z. URSANI, D. ESSAM, D. CORNFORTH, AND R. STOCKER. **Localized genetic algorithm for vehicle routing problem with time windows**. *Applied Soft Computing*, **11**(8):5375–5390, 2011. (119, 153)

[177] H.C.B. OLIVEIRA AND G.C. VASCONCELOS. **A hybrid search method for the vehicle routing problem with time windows**. *Annals of Operations Research*, **180**(1):125–144, 2010. (119, 153)

[178] N. BJELIĆ, M. VIDOVIĆ, AND D. POPOVIĆ. **Variable neighborhood search algorithm for heterogeneous traveling repairmen problem with time windows**. *Expert systems with applications*, **40**(15):5997–6006, 2013. (124)

[179] R. JOTHI AND B. RAGHAVACHARI. **Approximating the k-traveling repairman problem with repairtimes**. *Journal of Discrete Algorithms*, **5**(2):293–303, 2007. (124)

[180] Z. LUO, H. QIN, AND A. LIM. **Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints**. *European Journal of Operational Research*, **234**(1):49–60, 2014. (124)

[181] S. NUCAMENDI-GUILLÉN, I. MARTÍNEZ-SALAZAR, F. ANGEL-BELLO, AND J.M. MORENO-VEGA. **A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem**. *Journal of the Operational Research Society*, **67**(8):1121–1134, 2016. (124)

[182] I. OME EZZINE AND S. ELLOUMI. **Polynomial formulation and heuristic based approach for the k-travelling repairman problem**. *International Journal of Mathematics in Operational Research*, **4**(5):503–514, 2012. (124)

[183] I. POST AND C. SWAMY. **Linear programming-based approximation algorithms for multi-vehicle minimum latency problems**. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 512–531. SIAM, 2014. (124)

[184] H. TANG, E. MILLER-HOOKS, AND R. TOMASTIK. **Scheduling technicians for planned maintenance of geographically distributed equipment**. *Transportation Research Part E: Logistics and Transportation Review*, **43**(5):591–609, 2007. (124)

[185] L.Y.O. LI AND Z. FU. **The school bus routing problem: a case study**. *Journal of the Operational Research Society*, **53**(5):552–558, 2002. (124)

[186] J. PARK AND B.-I. KIM. **The school bus routing problem: A review**. *European Journal of operational research*, **202**(2):311–319, 2010. (124)

[187] M. SPADA, M. BIERLAIRE, AND T.M. LIEBLING. **Decision-aiding methodology for the school bus routing and scheduling problem**. *Transportation Science*, **39**(4):477–490, 2005. (124)

[188] P. AMORIM AND B. ALMADA-LOBO. **The impact of food perishability issues in the vehicle routing problem**. *Computers & Industrial Engineering*, **67**:223–233, 2014. (124)

[189] Z.-J. MA, Y. WU, AND Y. DAI. **A combined order selection and time-dependent vehicle routing problem with time widows for perishable product delivery**. *Computers & Industrial Engineering*, **114**:101–113, 2017. (124)

[190] L.M. CARO AND J.A.M. GARCÍA. **Measuring perceived service quality in urgent transport service**. *Journal of Retailing and Consumer Services*, **14**(1):60–72, 2007. (124)

[191] J. PAQUETTE, J.-F. CORDEAU, AND G. LAPORTE. **Quality of service in dial-a-ride operations**. *Computers & Industrial Engineering*, **56**(4):1721–1734, 2009. (124)

[192] P. HANSEN, N. MLADENOVIĆ, AND J.A.M. PÉREZ. **Variable neighbourhood search: methods and applications**. *Annals of Operations Research*, **175**(1):367–407, 2010. (124)

[193] Y. ZHOU AND J. WANG. **A local search-based multiobjective optimization algorithm for multiobjective vehicle routing problem with time windows**. *IEEE Systems Journal*, **9**(3):1100–1113, 2015. (133, 155, 156, 161, 162, 166, 167, 173, 174, 175, 177, 192)

[194] J. WANG, Y. ZHOU, Y. WANG, J. ZHANG, C.L.P. CHEN, AND Z. ZHENG. **Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms**. *IEEE transactions on cybernetics*, **46**(3):582–594, 2016. (133, 192)

[195] M.M. SOLOMON. **Algorithms for the vehicle routing and scheduling problems with time window constraints**. *Operations research*, **35**(2):254–265, 1987. (146, 154, 161)

[196] A. GARCIA-NAJERA AND J.A. BULLINARIA. **An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows**. *Computers & Operations Research*, **38**(1):287–300, 2011. (153, 154, 158)

[197] K.C. TAN, Y.H. CHEW, AND L.H. LEE. **A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows**. *Computational Optimization and Applications*, **34**(1):115–151, 2006. (154, 158)

[198] B. OMBUKI, B.J. ROSS, AND F. HANSHAR. **Multi-objective genetic algorithms for vehicle routing problem with time windows**. *Applied Intelligence*, **24**(1):17–30, 2006. (154)

[199] K. GHOSEIRI AND S.F. GHANNADPOUR. **Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm**. *Applied Soft Computing*, **10**(4):1096–1107, 2010. (154)

[200] W.-H. HSU AND T.-C. CHIANG. **A multiobjective evolutionary algorithm with enhanced reproduction operators for the vehicle routing problem with time windows**. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012. (154)

[201] J. CASTRO-GUTIERREZ, D. LANDA-SILVA, AND J.M. PÉREZ. **Nature of real-world multi-objective vehicle routing with evolutionary algorithms**. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 257–264. IEEE, 2011. (154, 155, 156, 160, 161, 162, 173, 174, 175, 177)

[202] J. CASTRO-GUTIERREZ. *Multi-objective tools for the vehicle routing problem with time windows*. PhD thesis, University of Nottingham, 2012. (154)

[203] K. DEB, A. PRATAP, S. AGARWAL, AND T. MEYARIVAN. **A fast and elitist multiobjective genetic algorithm: NSGA-II**. *IEEE transactions on evolutionary computation*, **6**(2):182–197, 2002. (155, 163, 164, 165)

[204] J. CASTRO-GUTIERREZ, D. LANDA-SILVA, AND J.M. PÉREZ. **MOVRPTW dataset**. Available online:`https://github.com/psxjpc/`, 2012. (155, 162)

[205] M. LAUMANNS, L. THIELE, K. DEB, AND E. ZITZLER. **Combining convergence and diversity in evolutionary multiobjective optimization**. *Evolutionary computation*, **10**(3):263–282, 2002. (155, 162)

[206] K. DEB, M. MOHAN, AND S. MISHRA. **Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions**. *Evolutionary computation*, **13**(4):501–525, 2005. (155, 162)

[207] D. TAŞ, N. DELLAERT, T. VAN WOENSEL, AND T. DE KOK. **Vehicle routing problem with stochastic travel times including soft time windows and service costs**. *Computers & Operations Research*, **40**(1):214–224, 2013. (158, 160)

[208] H. HASHIMOTO, T. IBARAKI, S. IMAHORI, AND M. YAGIURA. **The vehicle routing problem with flexible time windows and traveling times**. *Discrete Applied Mathematics*, **154**(16):2271–2290, 2006. (158)

[209] Z. FU, R. EGLESE, AND L.Y. LI. **A unified tabu search algorithm for vehicle routing problems with soft time windows**. *Journal of the Operational Research Society*, **59**(5):663–673, 2008. (158)

[210] W.-C. CHIANG AND R.A. RUSSELL. **A metaheuristic for the vehicle-routeing problem with soft time windows**. *Journal of the Operational Research Society*, **55**(12):1298–1310, 2004. (158)

[211] É. TAILLARD, P. BADEAU, M. GENDREAU, F. GUERTIN, AND J.-Y. POTVIN. **A tabu search heuristic for the vehicle routing problem with soft time windows**. *Transportation science*, **31**(2):170–186, 1997. (158)

[212] J. MÜLLER. **Approximative solutions to the bicriterion vehicle routing problem with time windows**. *European Journal of Operational Research*, **202**(1):223–231, 2010. (158)

[213] A. RODRÍGUEZ AND R. RUIZ. **A study on the effect of the asymmetry on real capacitated vehicle routing problems**. *Computers & Operations Research*, **39**(9):2142–2151, 2012. (161)

[214] C.L. FLEMING, S.E. GRIFFIS, AND J.E. BELL. **The effects of triangle inequality on the vehicle routing problem**. *European Journal of Operational Research*, **224**(1):1–7, 2013. (161)

[215] E. ZITZLER, L. THIELE, M. LAUMANNS, C.M. FONSECA, AND V.G. DA FONSECA. **Performance assessment of multiobjective optimizers: An analysis and review**. *IEEE Transactions on evolutionary computation*, **7**(2):117–132, 2003. (163, 175)

[216] S.-W. CHEN AND T.-C. CHIANG. **Evolutionary many-objective optimization by MO-NSGA-II with enhanced mating selection**. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1397–1404. IEEE, 2014. (163)

[217] E. ZITZLER, M. LAUMANNS, AND L. THIELE. **SPEA2: Improving the strength Pareto evolutionary algorithm**. In *Evolutionary Methods for Design, Optimization and Control*, pages 95–100, Barcelona, Spain, 2002. CIMNE. (163, 175)

[218] Q. ZHANG AND H. LI. **MOEA/D: A multiobjective evolutionary algorithm based on decomposition**. *IEEE Transactions on evolutionary computation*, **11**(6):712–731, 2007. (163, 175)

[219] C.A.C. COELLO AND M.S. LECHUGA. **MOPSO: A proposal for multiple objective particle swarm optimization**. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, **2**, pages 1051–1056. IEEE, 2002. (163)

[220] S. KUKKONEN AND J. LAMPINEN. **GDE3: the third evolution step of generalized differential evolution**. In *2005 IEEE Congress on Evolutionary Computation*, **1**, pages 443–450, 2005. (163)

[221] J.D. KNOWLES AND D.W. CORNE. **Approximating the nondominated front using the Pareto archived evolution strategy**. *Evolutionary computation*, **8**(2):149–172, 2000. (163)

[222] C.A.C COELLO, S.G. BRAMBILA, J.F. GAMBOA, M.G.C. TAPIA, AND R.H. GÓMEZ. **Evolutionary multiobjective optimization: open research areas and some challenges lying ahead**. *Complex & Intelligent Systems*, **6**:221–236, 2020. (163)

[223] A. ZHOU, B.-Y. QU, H. LI, S.-Z. ZHAO, P.N. SUGANTHAN, AND Q. ZHANG. **Multiobjective evolutionary algorithms: A survey of the state of the art**. *Swarm and Evolutionary Computation*, **1**:32–49, 2011. (163)

[224] Y. YUSOFF, M.S. NGADIMAN, AND A.M. ZAIN. **Overview of NSGA-II for optimizing machining process parameters**. *Procedia Engineering*, **15**:3978–3983, 2011. (164)

[225] N. SRINIVAS AND K. DEB. **Multiobjective optimization using nondominated sorting in genetic algorithms**. *Evolutionary computation*, **2**(3):221–248, 1995. (164)

[226] N. RIQUELME, C.V. LÜCKEN, AND B. BARÁN. **Performance metrics in multi-objective optimization**. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11. IEEE, 2015. (175)

[227] C.M. FONSECA, L. PAQUETE, AND M. LÓPEZ-IBÁNEZ. **An improved dimension-sweep algorithm for the hypervolume indicator**. In *2006 IEEE international conference on evolutionary computation*, pages 1157–1163. IEEE, 2006. (176)

[228] C.M. FONSECA, M. LÓPEZ-IBÁNEZ, L. PAQUETE, AND A.P. GUERREIRO. **Computation of the Hypervolume Indicator**. Available online: `http://lopez-ibanez.eu/hypervolume`, 2006. (176)

[229] E. ZITZLER AND L. THIELE. **Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach**. *IEEE transactions on Evolutionary Computation*, **3**(4):257–271, 1999. (176)

[230] F. WILCOXON. **Individual comparisons by ranking methods**. *Biometrics*, **1**(6):80–83, 1945. (177)

[231] J. DERRAC, S. GARCÍA, D. MOLINA, AND F. HERRERA. **A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms**. *Swarm and Evolutionary Computation*, **1**(1):3–18, 2011. (177)

[232] D.J. WALKER, R.M. EVERSON, AND J.E. FIELDSEND. **Visualizing mutually non-dominating solution sets in many-objective optimization**. *IEEE transactions on evolutionary computation*, **17**(2):165–184, 2013. (182)

[233] A. PRYKE, S. MOSTAGHIM, AND A. NAZEMI. **Heatmap visualization of population based multi-objective algorithms**. In *Evolutionary Multi-Criterian Optimization, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds.*, **4403**, pages 361–375. Springer-Verlag, 2007. (182)

[234] D.H. WOLPERT AND W.G. MACREADY. **No free lunch theorems for optimization**. *IEEE transactions on evolutionary computation*, **1**(1):67–82, 1997. (186)

[235] J. WANG, T. WENG, AND Q. ZHANG. **A two-stage multiobjective evolutionary algorithm for multiobjective multidepot vehicle routing problem with time windows**. *IEEE Transactions on Cybernetics*, **49**(7):2467–2478, 2019. (192)

List of Publications

[1] **GAURAV SRIVASTAVA** AND ALOK SINGH. **Boosting an evolution strategy with a preprocessing step: application to group scheduling problem in directional sensor networks.** *Applied Intelligence, **48**: 4760-4774, 2018, Springer.*

[2] **GAURAV SRIVASTAVA**, VENKATESH PANDIRI AND ALOK SINGH. **An evolution strategy based approach for cover scheduling problem in wireless sensor networks.** *International Journal of Machine Learning and Cybernetics, **11**: 1981-2006, 2020, Springer.*

[3] **GAURAV SRIVASTAVA** ALOK SINGH AND RAMMOHAN MALLIPEDDI. **NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows.** *Expert Systems with Applications, **176**: 114779, 2021, Elsevier.*

[4] **GAURAV SRIVASTAVA** ALOK SINGH AND RAMMOHAN MALLIPEDDI. **A hybrid discrete differential evolution approach for the single machine total stepwise tardiness problem with release dates.** *Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC-2021), 652-659, 2021, IEEE.*

[5] **GAURAV SRIVASTAVA** AND ALOK SINGH. **A steady-state grouping genetic algorithm approach for vehicle routing problem with time windows and quality of service objectives.** Communicated to *Applied Soft Computing, Elsevier.*

[6] **GAURAV SRIVASTAVA** AND ALOK SINGH. **Rescue unit allocation and scheduling with fuzzy processing times using a robust evolutionary approach.** Communicated to *Knowledge-Based Systems, Elsevier.*

Evolutionary Techniques for Permutation Based Problems

by Gaurav Srivastava

Submission date: 31-Jan-2022 12:23PM (UTC+0530)
Submission ID: 1751781988
File name: Gaurav_Srivastava.pdf (1.85M)
Word count: 77076
Character count: 374796

Papers at S.No 1 to 4 report the work of the thesis. Excluding these papers, similarity will be 6%.

Evolutionary Techniques for Permutation Based Problems

ORIGINALITY REPORT

(Supervisor) 31/01/2022

44%	6%	43%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Dr. Alok Singh
Professor
School of Computer & Information Sciences
University of Hyderabad
Hyderabad-500 046, Telanagana, India.

PRIMARY SOURCES

1	Gaurav Srivastava, Alok Singh, Rammohan Mallipeddi. "NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows", Expert Systems with Applications, 2021 Publication	15%
2	Gaurav Srivastava, Pandiri Venkatesh, Alok Singh. "An evolution strategy based approach for cover scheduling problem in wireless sensor networks", International Journal of Machine Learning and Cybernetics, 2020 Publication	9%
3	Gaurav Srivastava, Alok Singh, Rammohan Mallipeddi. "A Hybrid Discrete Differential Evolution Approach for the Single Machine Total Stepwise Tardiness Problem with Release Dates", 2021 IEEE Congress on Evolutionary Computation (CEC), 2021 Publication	8%
4	Gaurav Srivastava, Alok Singh. "Boosting an evolution strategy with a preprocessing step:	6%

application to group scheduling problem in directional sensor networks", Applied Intelligence, 2018
Publication

5	Submitted to University of Hyderabad, Hyderabad Student Paper	1 %
6	Airam Expósito, Julio Brito, José A. Moreno, Christopher Expósito-Izquierdo. "Quality of service objectives for vehicle routing problem with time windows", Applied Soft Computing, 2019 Publication	1 %
7	"Handbook of Heuristics", Springer Science and Business Media LLC, 2018 Publication	<1 %
8	cec2021.mini.pw.edu.pl Internet Source	<1 %
9	Victor Cunha, Luciana Pessoa, Marley Vellasco, Ricardo Tanscheit, Marco Aurelio Pacheco. "A Biased Random-Key Genetic Algorithm for the Rescue Unit Allocation and Scheduling Problem", 2018 IEEE Congress on Evolutionary Computation (CEC), 2018 Publication	<1 %
10	freidok.uni-freiburg.de Internet Source	<1 %

11	Felix Wex, Guido Schryen, Stefan Feuerriegel, Dirk Neumann. "Emergency response in natural disaster management: Allocation and scheduling of rescue units", European Journal of Operational Research, 2014 Publication	<1 %
12	Alok Singh. "A new grouping genetic algorithm approach to the multiple traveling salesperson problem", Soft Computing, 01/2009 Publication	<1 %
13	www.springerprofessional.de Internet Source	<1 %
14	"Swarm, Evolutionary, and Memetic Computing", Springer Science and Business Media LLC, 2011 Publication	<1 %
15	aisel.aisnet.org Internet Source	<1 %
16	Ying Zhou, Jiahai Wang. "A Local Search-Based Multiobjective Optimization Algorithm for Multiobjective Vehicle Routing Problem With Time Windows", IEEE Systems Journal, 2015 Publication	<1 %
17	Submitted to South University of Science and Technology of China Student Paper	<1 %

18	Alok Singh, André Rossi. "Group scheduling problems in directional sensor networks", Engineering Optimization, 2014 Publication	<1 %
19	Advances in Intelligent Systems and Computing, 2014. Publication	<1 %
20	Pandiri Venkatesh, Alok Singh. "A Hyper-Heuristic Based Artificial Bee Colony Algorithm for k -Interconnected Multi-Depot Multi-Traveling Salesman Problem", Information Sciences, 2018 Publication	<1 %
21	"Soft Computing for Problem Solving", Springer Science and Business Media LLC, 2021 Publication	<1 %
22	www.inderscience.com Internet Source	<1 %
23	mts.intechopen.com Internet Source	<1 %
24	Sachchida Nand Chaurasia, Alok Singh. "A hybrid evolutionary approach to the registration area planning problem", Applied Intelligence, 2014 Publication	<1 %
	link.springer.com	

25	Internet Source	<1%
26	"Network Optimization", Springer Science and Business Media LLC, 2011 Publication	<1%
27	Lecture Notes in Computer Science, 2007. Publication	<1%
28	Submitted to Universidad de La Laguna Student Paper	<1%
29	epub.uni-regensburg.de Internet Source	<1%
30	eprints.nottingham.ac.uk Internet Source	<1%
31	Shyam Sundar, Alok Singh. "Metaheuristic Approaches for the Blockmodel Problem", IEEE Systems Journal, 2015 Publication	<1%
32	worldwidescience.org Internet Source	<1%
33	Michael Mutingi, Charles Mbohwa. "Grouping Genetic Algorithms", Springer Science and Business Media LLC, 2017 Publication	<1%
34	dokumen.pub Internet Source	<1%

35	Sachchida Nand Chaurasia, Shyam Sundar, Alok Singh. "Hybrid metaheuristic approaches for the single machine total stepwise tardiness problem with release dates", Operational Research, 2016 Publication	<1 %
36	Sina Nayeri, Ebrahim Asadi-Gangraj, Saeed Emami. "Metaheuristic algorithms to allocate and schedule of the rescue units in the natural disaster with fatigue effect", Neural Computing and Applications, 2018 Publication	<1 %
37	mafiadoc.com Internet Source	<1 %
38	Anis Koubaa, Hachemi Bennaceur, Imen Chaari, Sahar Trigui et al. "Robot Path Planning and Cooperation", Springer Science and Business Media LLC, 2018 Publication	<1 %
39	Sahar Trigui, Omar Cheikhrouhou, Anis Koubaa, Anis Zarrad, Habib Youssef. "An analytical hierarchy process-based approach to solve the multi-objective multiple traveling salesman problem", Intelligent Service Robotics, 2018 Publication	<1 %

40	Wex, Felix. "Coordination strategies and predictive analytics in crisis management", Universität Freiburg, 2013. Publication	<1 %
41	André Rossi, Alok Singh, Marc Sevaux. "Column generation algorithm for sensor coverage scheduling under bandwidth constraints", Networks, 2012 Publication	<1 %
42	Submitted to Prince Sultan University Student Paper	<1 %
43	Singh, Alok, and André Rossi. "A genetic algorithm based exact approach for lifetime maximization of directional sensor networks", Ad Hoc Networks, 2012. Publication	<1 %
44	www-usr.inf.ufsm.br Internet Source	<1 %
45	www.ir.nctu.edu.tw Internet Source	<1 %
46	www.oak.go.kr Internet Source	<1 %
47	Kavita Singh, Shyam Sundar. "A new hybrid genetic algorithm for the maximally diverse grouping problem", International Journal of Machine Learning and Cybernetics, 2019 Publication	<1 %

48	W. C. Ng, K. L. Mak, Y. X. Zhang. "Scheduling trucks in container terminals using a genetic algorithm", Engineering Optimization, 2007 Publication	<1 %
49	"Contemporary Computing", Springer Science and Business Media LLC, 2009 Publication	<1 %
50	Ying Zhou, Lingjing Kong, Yiqiao Cai, Ziyan Wu, Shaopeng Liu, Jiaming Hong, Keke Wu. "A Decomposition-Based Local Search for Large-Scale Many-Objective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows", IEEE Systems Journal, 2020 Publication	<1 %
51	Jiahai Wang, Ying Zhou, Yong Wang, Jun Zhang, C. L. Philip Chen, Zibin Zheng. "Multiobjective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms", IEEE Transactions on Cybernetics, 2016 Publication	<1 %
52	www.sintef.no Internet Source	<1 %
53	Submitted to Yonsei University Student Paper	<1 %

54	www.cil.ntu.edu.sg Internet Source	<1 %
55	Submitted to University of Durham Student Paper	<1 %
56	eprints.utm.my Internet Source	<1 %
57	epdf.pub Internet Source	<1 %
58	hal.inria.fr Internet Source	<1 %
59	elea.unisa.it:8080 Internet Source	<1 %
60	Andreas Beham. "Parallel Tabu Search and the Multiobjective Vehicle Routing Problem with Time Windows", 2007 IEEE International Parallel and Distributed Processing Symposium, 2007 Publication	<1 %
61	Submitted to Dayalbag Educational Institute Student Paper	<1 %
62	infoscience.epfl.ch Internet Source	<1 %
63	prism.ucalgary.ca Internet Source	<1 %

64	refubium.fu-berlin.de Internet Source	<1 %
65	www.yumpu.com Internet Source	<1 %
66	Submitted to Erasmus University of Rotterdam Student Paper	<1 %
67	Submitted to University of Strathclyde Student Paper	<1 %
68	ieeexplore.ieee.org Internet Source	<1 %
69	"Neural Information Processing. Theory and Algorithms", Springer Science and Business Media LLC, 2010 Publication	<1 %
70	Submitted to Koc University Student Paper	<1 %
71	shura.shu.ac.uk Internet Source	<1 %
72	thesis.library.caltech.edu Internet Source	<1 %
73	Submitted to University of Sydney Student Paper	<1 %
74	id.scribd.com Internet Source	<1 %

75	www.scribd.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		