Grenade Explosion Method for Maximum Weight Clique Problem

A Dissertation submitted to the University of Hyderabad in partial fulfillment of the degree of

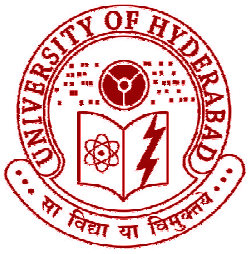
MASTER OF TECHNOLOGY

in

Artificial Intelligence

by

Manohar Pallantla



Department of Computer and Information Sciences

School of Mathematics and Computer/Information sciences

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad – 500 046
Andhra Pradesh
India

CERTIFICATE

This is to certify that the dissertation entitled **"Grenade Explosion Method for Maximum Weight Clique Problem"** submitted by **Manohar Pallantla** bearing Reg. No **09MCMI08** in partial fulfillment of the requirements for the award of Master of Technology in Artificial Intelligence is a bonafide work carried out by him under my supervision and guidance.

The dissertation 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.

Dr Alok Singh,
Supervisor,
Department of CIS,
University of Hyderabad.

Head,
Department of CIS,
University of Hyderabad.

Dean,
School of MCIS,
University of Hyderabad.

DECLARATION

I Manohar Pallantla hereby declare that this Dissertation entitled **"Grenade Explosion Method for Maximum Weight Clique Problem"** submitted by me under the guidance and supervision of **Dr. Alok Singh** is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date:

Name: Manohar Pallantla

Signature of the Student:

Regd. No. 09MCMI08

To,

My Family and Friends

Acknowledgment

I would like to express my sincere gratitude to Dr. Alok Singh, my project supervisors, for valuable suggestions and keen personal interest throughout the progress of my course of research. They encouraged, supported, corrected and guided me during the project. The project has been learning and growing experience for me. Working under their guidance I learned many things.

I am extremely grateful to our Head of the Department, Prof. C.R.Rao, for providing excellent computing facilities and a nice atmosphere for doing my project. I convey my heartfelt thanks to AI Lab staff for their help in completing the project work successfully.

I would like to take this opportunity to thank my friends who have been morale boosters for me, encouraged me to take up this course and supported me throughout this course with their love and affection. Last but not the least my parents and sister to whom I owe this work. I wouldn't have reached this stage without their support and encouragement.

Manohar Pallantla

Abstract

Maximum weight clique problem is an NP-Hard problem which seeks the fully connected sub graph of maximum weight in a given weighted graph G. In this project we are using a recently proposed optimization technique called Grenade Explosion Method (GEM) to solve maximum weight clique problem. GEM was originally designed for continuous optimization problems. To our knowledge this is the first application which uses GEM for the discrete optimization. Computational results on the benchmark instances show the effectiveness of our proposed GEM approach.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Maximum Weight Clique problem is a combinatorial optimization problem that maximizes the total weight of the clique. Clique is nothing but a sub graph that is complete.

1.1 Global Optimization

The task of global optimization is to find a solution in the solution set for which the objective function obtains its smallest value or the largest value, the global minimum or the global maximum. Global optimization thus aims at determining not just “a local minimum” or “a local maximum” but “the smallest local minimum” or “the largest local maximum” with respect to the solution set.

Many important practical problems can be posed as mathematical programming Problems. Let us consider the design of a system that has to meet certain design criterion. The system will contain feature that may be varied by the designer within certain limits. The values given to these features will be the optimization variables of the problem frequently when the system performance is expressed as a mathematical function of the optimization variables, this function, which will sometimes be called the objective function. The problem of writing computer algorithms that distinguish between these local minima or maxima and locate the best local minimum or maximum is known as the global optimization problem.

Optimization problems can be classified in to two categories, problems encoded in real valued variables and problems encoded in discrete valued variables. Combinatorial optimization is a branch of optimization, which falls in to the second category of optimization problems. It deals with the optimization problems where the search space is

a set of feasible solutions, which are discrete or can be reduced to a discrete one, and the goal is to find the best possible solution or optimal solution.

1.2 Clique Problem

A clique in an undirected graph G = (V, E) is a subset of the vertex set C ⊆ V, such that for every two vertices in C, there exists an edge connecting the two. This is equivalent to saying that the subgraph induced by C is complete. Figure 1.1 shows the one of the cliques in the given input graph G.

A maximal clique is a clique that cannot be extended by including one more adjacent vertex, that is, a clique which does not exist exclusively within the vertex set of a larger clique.

A maximum clique is a clique of the largest possible size in a given graph. The clique number ω(G) of a graph G is the number of vertices in a maximum clique in G. The intersection number of G is the smallest number of cliques that together cover all edges of G. Figure 1.1 provides an example of a clique and a maximum clique.

A maximum weight clique is nothing but the maximum clique with maximum clique weight where clique weight is the sum of the weights of the vertices in that clique. Maximum weight clique problem is a generalization of maximum clique problem in which vertices have positive weights and one has to find the clique with maximum weight.

The opposite of a clique is an independent set, in the sense that every clique corresponds to an independent set in the complement graph. The cliques cover problem concerns finding as few cliques as possible that include every vertex in the graph. A related concept is a biclique, a complete bipartite sub graph. The bipartite dimension of a graph is the minimum number of bicliques needed to cover all the edges of the graph.

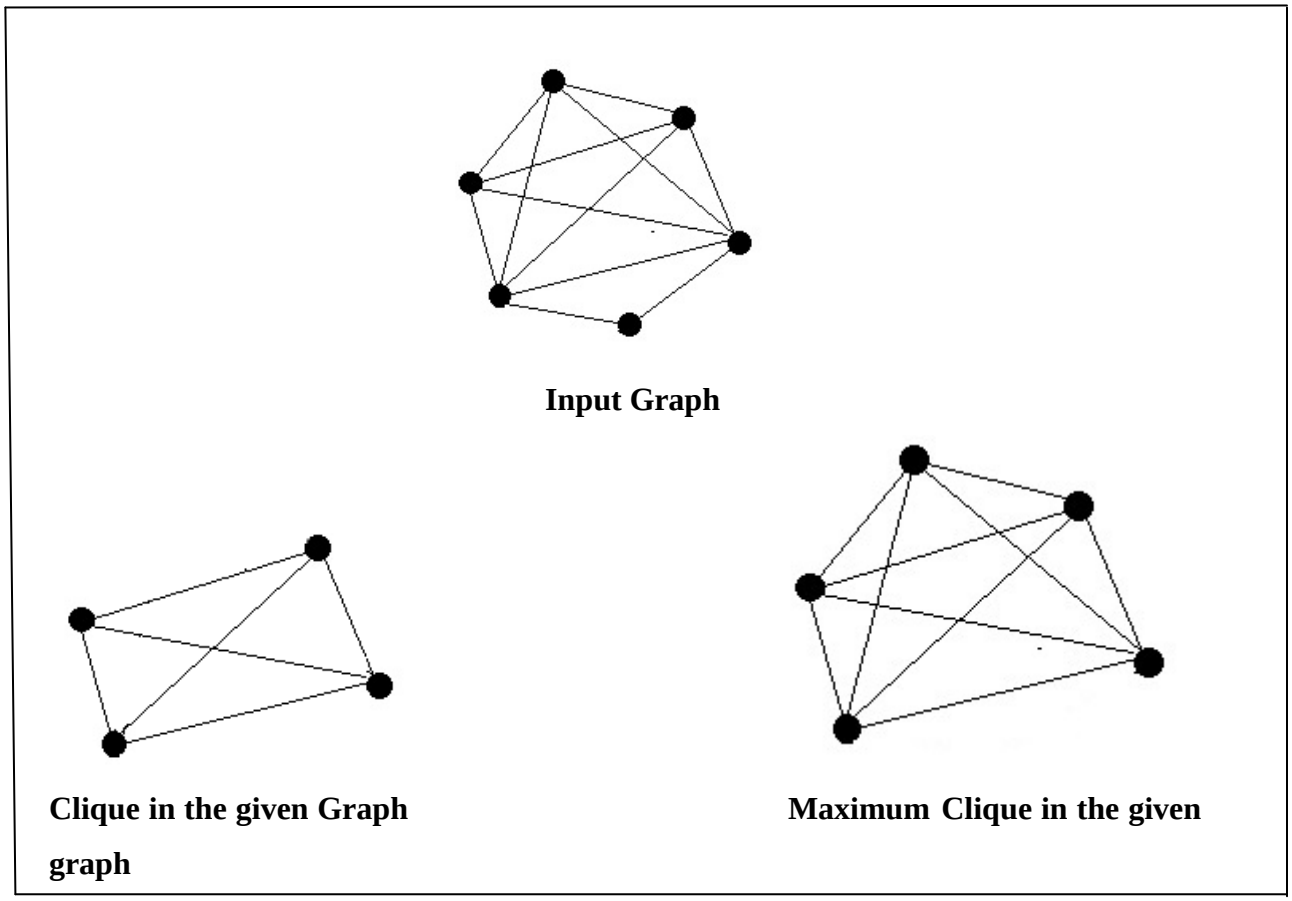


Figure 1.1 Clique and Maximum clique in a graph

Both maximum clique and maximum weight clique problems are NP-Hard [1]. Therefore, exact algorithms for these problems can return a solution only in time that grows exponentially with number of vertices in the graph. Hence, even on moderately large problem instances exact algorithms can take prohibitive time. These problems are also hard to approximate. Khot [2] proved that unless NP = ZPP, there is no polynomial algorithm that can approximate the clique within a factor of $n/2^{(\log n)^{(1-\varepsilon)}}$ or any $\varepsilon > 0$ where n is the number of vertices in the graph.

1.3 Previous Algorithms

For the maximum clique problem probably no polynomial time algorithm will be possible, but improvements to the existing algorithms can still be effective. Exact algorithms, which can be guaranteed to find the maximum clique, usually use a branch-and-bound approach to the maximum clique problem, searching systematically through possible solutions and applying bounds to limit the search space. The closest bounds come from the vertex-coloring method. This method assigns colors to vertices, so that no two adjacent vertices of a graph *G* are colored with the same color. The number of colors is the upper bound to the size of the maximum clique in graph *G*. Vertex-coloring is also known to be NP-hard, so a graph can only be colored approximately.

Many heuristic and exact algorithms have been proposed to solve the maximum weight clique problem. Nemhauser and Trauter [3] formulated the problem as an integer linear program and used an implicit enumeration algorithm to solve the maximum weight clique problem. Using their work on maximum clique problem, Loukakis and Tsouros [4] developed an algorithm that uses recursive backtracking for the maximum weight clique problem. Carraghan and Pardalos [5] proposed a partial enumerative algorithm for the maximum weight clique problem.

Pardalos and Desai [6] proposed a branch and bound technique, they represented the problem as an unconstrained quadratic 0-1 program and used the proposed algorithm to solve the problem but it was slower than the algorithm of Carraghan and Pardalos [5].Balas and Xue [7] developed an efficient branch and bound procedure using minimum weighted coloring of triangulated graph. After that Balas and Xue [8] developed another branch and bound technique that used as an upper bounding procedure, the heuristic that they developed for the weighted fractional coloring problem.

Balas and Niehaus [9] developed a steady-state genetic algorithm which uses an optimized crossover. The optimized crossover takes two cliques and produces a single child. The child is produced by finding the maximum weight clique in the sub graph formed by the union of vertex sets of two cliques through solving the maximum flow problem in the complement of the subgraph. Babel [10] developed an efficient branch and bound procedure. This method uses upper and lower bounds, for the maximum weight clique which is computed by coloring the weighted graph. Östergard [11] developed a branch and bound based exact method. Bomze et al. [12] proposed a method based on replicator dynamics which uses the continuous formulation of maximum weight clique problem using Motzkin-Strauss theorem [13].

Massaro et al. [14] proposed PBH, which is a complementary pivoting based heuristic which also uses the continuous formulation like in Bomez et al method. Busygin [15] developed QUALEX-MS, a trust region based heuristic, which uses the continuous formulation. Qingfu Zhang, Jianyong Sun, and Edward Tsang, proposed An Evolutionary Algorithm with Guided Mutation for maximum clique problem

The algorithms for finding a maximum clique are frequently used in Chemo informatics, bioinformatics and computational biology applications, where their main application is to search for similarity between molecules. These algorithms are used for screening databases of compounds to filter out molecules that are similar to known biologically active molecules and are feasible to be active themselves. Also, these algorithms are used for comparing protein structures, to provide the information about protein function and also the information about possible interactions between proteins. Searching for the maximum clique is often the bottle-neck computational step in these applications. Maximum weight clique problem has many practical applications in various fields such as computer vision, pattern recognition, robotics where weighted graphs are used to represent high level pictorial information.

1.4 Organization

The project report is organized in to five chapters beginning with the introduction. In chapter 2, we discuss briefly about the Grenade Explosion Algorithm. In chapter 3, we give the details of proposed Grenade Explosion Algorithm for the Maximum Weight Clique problem. In chapter 4 we will present the results obtained by our algorithm. In chapter 5 we will discuss about the future work which can be done and present our conclusions.

Chapter 2

The Grenade Explosion Algorithm

In this dissertation we will discuss in depth about Grenade Explosion algorithm, and will present a brief discussion about techniques like Metaheuristic and Evolutionary Algorithms.

2.1 Metaheuristics:

Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method. Most commonly used metaheuristics are to solve combinatorial optimization problems, but of course can handle any problem that can be converted into that form.

Definition: "A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem."

Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent, biased form. The goal of the Meta heuristic is to efficiently explore the search space in order to find optimal solutions or at least near optimal solutions.

2.1.1 Different Classifications:

Metaheuristics are divided in to two categories by using the characteristic the number of solutions used at the same time. Single-solution metaheuristics, where a single solution is considered at a time and population based metaheuristics, where a number of solutions evolve concurrently [16]. Within these two categories, there are two more sub categories primarily constructive metaheuristics, where a solution is built from raw and improvement metaheuristics, which modifies a solution iteratively. There are lots of techniques which fall in to these two categories like

• Tabu Search, Iterated Local Search, Variable Neighborhood Search, Stimulated annealing (SA), are single solution metaheuristic methods.

• Evolutionary algorithms like Genetic algorithms (GA), Genetic programming (GP), etc., other are Scatter search (SS), Swarm intelligence techniques like Particle Swarm optimization (PSO), Ant colony optimization (ACO) and Artificial Bee colony algorithm (ABC), etc., belongs to population based metaheuristics.

2.1.2 Basic Components of Metaheuristic Techniques:

Based on studies and observations, a clear view of metaheuristics can be constructed. This view contains some basic components common for all metaheuristic techniques these components can be explained as follow.

• **Construction of Initial Population:** This component is almost used by every metaheuristic technique for initial population generation.

• **Recombination:** This recombination is used in GA to generate new children. In this process first two fittest parents are selected and then a child is generated by using these two parents.

- **Modification:** This random modification is used in almost all techniques like GA, ABC to generate new solutions. First a fittest parent is selected after that some variable values in that parent are changed to generate a child.

- **Improvement:** This is used to improve the current solution; for example, by selecting the best solution in the neighborhood or by applying a local search.

- **Memory Update:** This component updates either standard memory in TS, pheromone trails in ACO, populations in GA or reference sets in SS.

- **Dynamic Neighborhood:** This component is used to modify the structure of the neighborhood solutions.

- **Dynamic objective function:** This process is used to escape from local minima by modifying the search landscape. Accordingly, during the search the objective function is altered by trying to incorporate information collected during the search process.

2.2 Evolutionary Algorithms

Among optimization methods, Evolutionary Algorithms (EAs) which are known as general purpose optimization algorithms, have a capability to find the near-optimal solution to the numerical real-valued test problems for which exact and analytical methods do not produce the optimal solution within a reasonable computation time, particularly when the global minimum or maximum is surrounded by many local minima or maxima. EAs don't make use of the derivatives of the objective function. These algorithms are usually formed by observing the phenomenon that are happening in nature, like Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Bee Colony Optimization.

Genetic Algorithm is based on the genetic process of biological organisms [17]. Over many generations, natural populations evolve according to the principles of natural selections that mean survival of the fittest. In this method, a potential solution to a problem is represented as a set of parameters. Each independent variable is represented by a gene. Combining the genes, a chromosome is produced which represents a solutions. In genetic terminology, the set of parameters represented by a particular chromosome is referred to as an individual. During the reproduction phase, the fittest individuals are selected from the population and recombined. Parents are randomly selected from the population using a scheme, which favors the fitter individuals. After selecting two parents, their chromosomes are recombined; by using mechanisms of crossover and mutation. Mutation is usually applied to some individuals, to guarantee population diversity.

PSO is inspired by choreography of a bird flock. The approach can be viewed as a distributed behavioral algorithm that performs a multi-dimensional search. It makes use of a velocity vector to update the current position of each particle in the swarm. The velocity vector is updated based on the memory gained by each particle, conceptually resembling an autobiographical memory, as well as the knowledge gained by the swarm as a whole. Thus, the position of each particle, which represents a solution to the optimization problem, is updated based on the social behavior of the swarm.

Artificial Bee Colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee swarm. The colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts [18]. Onlooker bees wait in dance area for making decision to choose a food source, employed bees hover around a discovered food source and scout bees try to locate new sources. Bees come into the hive and share the nectar information of the sources with the bees waiting on the dance area within the hive. Then, every employed bee goes to the food source area visited by her at the previous cycle since that food source exists in her memory, and then chooses a new food source by means of visual information in the neighborhood of the present one [18].

Based on the shared information, an onlooker bee is employed at one of the food sources, where probability of recruitment is proportional to profitability of the food source. In the ABC algorithm, the position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution [18].

Ant colony system was inspired by the foraging behavior of real ants. This behavior enables ants to find the shortest path between food sources and their nest. This functionality of real ant colonies is exploited in artificial ant colonies in order to solve optimization problems.

Simulated Annealing (SA) was originally proposed to simulate the annealing process. After generating an initial solution, it attempts to move from the current solution to one of its neighborhood solutions. If the new solution results in a better objective value, it is accepted. However, if the new solution yields a worse value, it can be accepted according to a probability. By accepting worse solutions, SA can avoid being trapped in local optima.

In these algorithms, except SA, a population of agents searches the space. Each agent represents a solution for the optimization problem. During iterations, a set of new solutions are produced, the best solutions are kept and allowed to produce new solutions in the next iterations. This process goes on until the convergence criterion is met.

One of the important disadvantage in population based algorithm, is crowding of the agents, which shows the convergence of the algorithm to a point in the crowded region. If it happens in the early iterations of the algorithm, solution to which the algorithm has converged is probably a local minimum or maxima, because the design space has not been explored thoroughly. And, in the final population, similar agents don't present different solutions, which can be a disadvantage especially when the objective function has several global optimization values.

In EAs, several random numbers are produced each iteration. Consequently to analyze and predict the performance of these algorithms, knowing the probability of an event is important rather than whether it happens or not. Furthermore, the convergence process and probably the final solution may be different in each independent run. To obtain a reliable solution or test the reliability of an optimization algorithm, several independent runs should be executed.

An objective function may have several global optimum values, i.e. several points in which the value of the objective function is equal to the global maximum or minimum value. Furthermore, it may have some local optimum values in which the value of the objective function is close to the global optimum value. As there are usually many simplifications in writing a standard mathematical form for an industrial optimization problem, finding all global optimum values or even these local optimum can present a verity of quite optimal solutions for the decision maker.

The present work uses a novel evolutionary algorithm for optimization problems. This method, which we call Grenade Explosion Method (GEM), is based on the concept present in the next sections. In section 2.3, the method and the concepts behind it are introduced and the detailed algorithm is presented.

2.3 Grenade Explosion Algorithm:

This Algorithm is proposed by Ahrari by observing the explosion of the grenade [19].The idea of the presented algorithm is similar to natural grenade explosion, in which the blasted pieces of shrapnel collides with the objects near the explosion location [19]. The loss caused by each piece of shrapnel is calculated. A high value of loss per piece of shrapnel in an area indicates there are valuable objects in that area. To make more loss, the next grenade has to be thrown where the maximum loss occurs. Although the objects near grenade's location are more likely to be damaged, the probability of destruction is

still kept for farther objects by choosing a high value for the length of explosion along each coordinate, in which the thrown piece of shrapnel may destruct the objects. This process will help us in finding the best place for throwing the grenades, even though shrapnel cannot discover this area in early iterations. Algorithm 2.1 provides the steps in the Grenade Explosion algorithm explained above.

Algorithm 2.1 Steps involved in Grenade Explosion Algorithm

```
1:  Calculate the fitness value of each member of the
    population. Find the best place to place the grenade.
2:  Repeat
3:  Generate Nq points around each grenade
4:  Select the best place to throw the next grenade
5:  Remember the best solution till now
6:  Until (iterations are over or requirement is met)
Nq: Number of shrapnel
```

First we will determine the locations to keep the grenades. Once after all the locations are determined to keep the grenades we will find the loss at each location around each grenade (number of location around each grenade where the loss occurred, is equal to the number of shrapnel). After that we will find the best location in the neighborhood of the current locations that is the location with highest fitness value.

Fitness: The loss caused by destruction of an object is considered as the fitness of the objective function at the object's location.

If the loss at particular location is not improved for a certain number of iterations we will abandon that location and we will decide that the maximum loss occurred at that location as the maximum location in that direction

It is clear from the above explanation that, the main control parameters used in the grenade Explosion algorithm is: The number of locations to place grenades and the number of pieces of shrapnel, the termination criteria.

When we search for a better location around each grenade we will generate some points around each grenade so the points that are generated might not be feasible so if the solution generated around each grenade is not feasible then it is converted into a feasible solution bye the use of a repair procedure the above procedure is explained pictorially in the Figure 2.1.

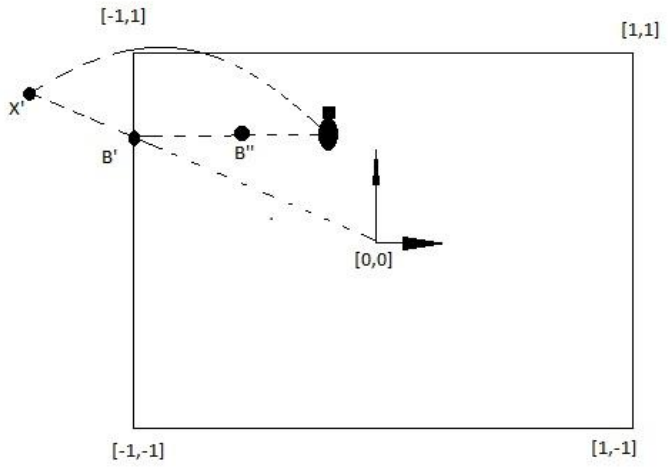


Figure 2.1 Conversion of an unfeasible location into a feasible location

One of the important feature of Grenade explosion algorithm which is not there in other EAs is the agent's territory radius (R_t), which means an agent (in this algorithm agents are grenades) does not allow other agents come closer than a specific distance, which is specified by R_t . The above process is explained in the Figure 2.2. When several agents are exploring the feasible space, a high value for this agent territory radius makes sure that grenades are spread quite uniformly in the feasible space and the whole space is being explored, while a low value for this parameter lets the grenades get closer to search the local region all together.

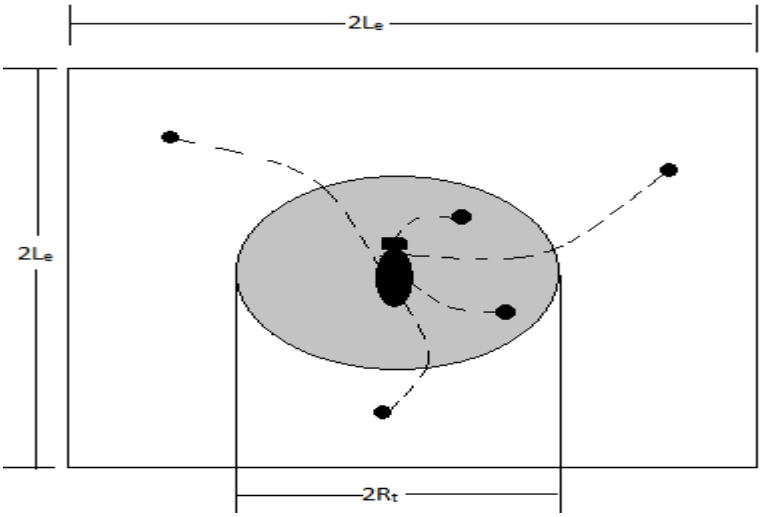


Figure 2.2 Range of explosion and the agent territory radius

Besides the general properties of an ordinary optimization method, GEM shows special abilities in finding solutions for an optimization problem, which can be considered as finding the global minimum while surrounded by many local minima, finding all global minima of functions with multiple global minima and finding high-fitted local minima with a probability which strongly depends on the fitness of that local minimum.

2.3.1 Pseudo code for Grenade Explosion Algorithm:

The detailed pseudo code of Grenade Explosion algorithm proposed by Ahrari [19] is explained below. First we have to scale the independent variable range to [-1, 1], then generate some grenades in random locations. We have to arrange the grenades according to their fitness values after that we will generate some points around each grenade if its outside the variable range then transport it into variable range if the newly generated point is at a distance of at least R_t from the current grenade then that point is accepted after that we will calculate the fitness of the newly generated points if it is fitter than the existing one then replace the existing one with the new fitter solution. After that update agent radius and the length of the explosion and repeat the above procedure until the maximum number of iterations. Algorithm 2.2 presents the pseudo code for the above explained procedure.

Algorithm 2.2 Pseudo code for grenade Explosion algorithm:

1. Scale the independent variable range to [-1,1].
2. Select problem parameters.
3. Generate N_g grenades in random locations in an n-dimension space.
4. While ite_num<max_iteration.
5. Arrange the Grenades based on their fitness values.
6. Let i=1.
7. While N_q accepted points are generated around i^{th} grenade.
8. Generate a point around i^{th} grenade.
9. If (X′) is outside $[-1,1]^n$ then transport X′ into $[-1,1]^n$.
10. If X′ is at least at a distance of R_t apart then X′ is accepted.
11. End while N_q pieces of shrapnel are thrown.
12. Calculate the fitness of the new generated points around the i^{th} grenade. If the fitness of the best point is better than current location of the i^{th} grenade, move the grenade to the location of the best point.
13. Let i=i+1.
14. If N_g >= i then goto 7.
15. Reduce R_t.
16. Calculate exponent m.
17. Update L_e.
18. Update exponent P.
19. End while % Number of iterations.

N_g = Number of grenades.

N_q = Number of shrapnel.

R_t = Agent territory radius.

L_e = Length of explosion.

We should take care while choosing the number of grenades to be placed because the increasing in the number of grenades more than enough will cause the low convergence rate

The initial value for the radius of the grenade territory ($R_{t)}$ should be large enough so that the possibility of grenade crowding will be eliminated in the early iterations of the algorithm. If initial R_t is very large, it takes a long time to put them in an acceptable configuration, larger values may make it impossible. So we have to choose an optimal value for the agent territory radius depending on the number of grenades that we are going to use.

Chapter 3

Proposed GEM algorithm for Maximum Weight clique problem

3.1 GEM algorithm for Maximum Weight clique problem:

We have developed an approach combining GEM and Maximum clique heuristic. The heuristic transforms the clique into a Maximum weight clique and we used a repair procedure proposed in [20] to convert an infeasible solution obtained in GEM into a feasible solution. Actually the GEM algorithm is designed for the continuous optimization but we are using it for discrete optimization so we made some changes in the algorithm so as to use it to solve the clique problem.

The main features of our Grenade Explosion algorithm like solution encoding, initial population generation, solution selection, heuristic and other features are described in the following subsections.

3.2 Solution Encoding:

We have used an array of length n to represent a solution where n is the total number of nodes in the graph. The values in the solution array forms a clique that is if node i is there in the solution array then the node i is the part of the clique formed by the solution array

If the array consists

2 5 7 9 11 13 15

This represents that the nodes 2, 5, 7, 9, 11, 13, 15 forms a clique.

Algorithm 3.1 Pseudo code of proposed GEM algorithm for maximum weight clique problem

1. Initialize the population S using the Maximum clique heuristic where S is (grenades N) matrix .Grenades is number of initial solutions and N is the number of nodes.
2. Evaluate the fitness value (that is loss at each location) of S that is fitness of all the cliques.
3. Best grenade location and loss at that location is remembered as the global maximum.
4. Cycle 1
5. **Repeat**
6. Generate solutions around each grenade.
7. Evaluate the fitness of S.
8. Best grenade location and loss at that location is remembered as the global maximum.
9. If solution is not improved within limit then remember that grenade location and the loss at that location as the best in that direction.
10. Cycle=cycle+1.
11. **Until** cycle = max cycle value.

The algorithm 3.1 works as follows first we will generate initial population for the grenade explosion using the procedure proposed in [20]. Find the best fit places to place the grenades that are the cliques with maximum weight. And then generate some points around each grenade which is equal to the procedure of shrapnel collision with the locations round each grenade then find the best location around each grenade to place the next grenade. Then the best grenade location and loss that is the clique and its weight are remembered as the best clique and the maximum weight of the cliques in that graph. If the solution is not improved for certain number of iterations the exploration in that

direction will be stopped and that solution will be remembered as the maximum weight in that direction. We will repeat this procedure until the best solution is found or the maximum cycle count is reached. initially we will keep the agent territory radius some large value and gradually we will decrease the agent territory radius this is because initially we have to explore farther regions and then in the latter stages each and every grenade searches in the locations closer to it so that we will avoid the situation of blocking at the local minimum in the initial stages.

3.2.1 Generating initial population

We have used the Maximum clique heuristic and clique algorithm for generating initial solutions [20]. We have also used the repair procedure to convert the unfeasible solutions to convert into a feasible solution. This method generates a solution by following an iterative process.

Algorithm 3.2 algorithm for the initial population

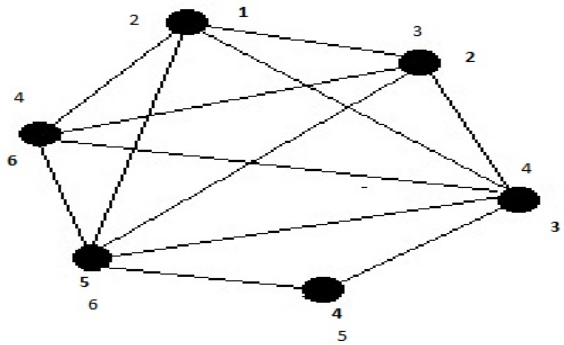
1. For i=1 to N_g
2. Generate some random sub graph from the given graph by randomly including the nodes according to some probability
3. Generated sub graph might not be a clique so use repair procedure to convert the sub graph into a clique
4. Use maximum clique heuristic to convert this clique into a maximum clique
5. Store the clique
6. End for

We used the above algorithm 3.2 for the generation of the initial population for the grenade explosion algorithm this algorithm takes the graph as the input and gives some clique in that graph as the output. First we randomly select some sub graph in the given input graph and this generated graph might be a clique or might not be a clique so we have to convert into a clique for this we use the repair procedure this procedure takes that sub graph and gives a clique after a clique is obtained we will give that clique to the max clique heuristic which converts the input clique to maximum clique.

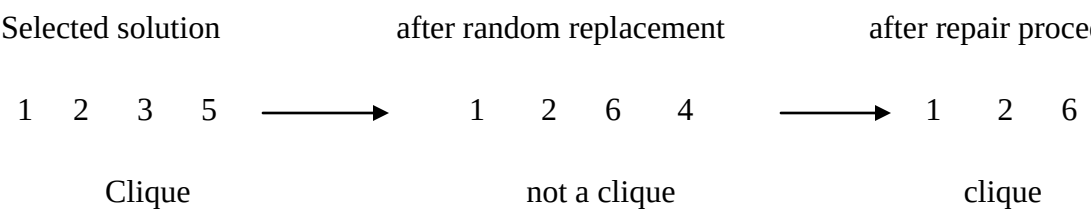
3.2.2 Generating random locations around each grenade

We used these procedures to generate random solutions around each clique. The first algorithm works as follows

We will use the algorithm 3.3 to generate the solutions around each grenade that is we have to find the best solution to place the next grenade around each and every current grenade. First we will take a solution and we will randomly replace some nodes in that solution with some other nodes that are not there in that solution. Like this we will generate some N_q locations around each and every grenade. Here N_q is the number of shrapnel that are used to collide with the nearby locations. The solutions that are generated might be cliques or might not be cliques so we used repair procedure to convert those infeasible solutions to feasible solutions that is cliques. Now check whether the solutions are within the limit of the agent territory radius or not if the solution is in the radius then eliminate that solution. Now the locations around each grenade are in the feasible space now find the best place in all the locations generated to throw the next grenade.



If for example solution 1 2 3 5 is selected to generate some random locations around it then after random replacement if the solution is changed to 1 2 6 4 then this is not a clique so use repair procedure to convert into clique



The algorithm 3.3 shows the step by step procedure to generate the locations around each grenade

Algorithm 3.3 Pseudo code for method1 for the random generation of new solutions

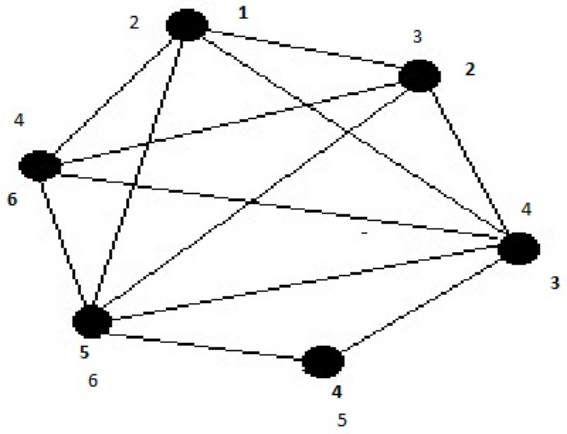
```
1. Select a location to generate points around it.
2. For i=1 to number of shrapnel-1.
3. Randomly select two nodes and replace them with
   some other nodes that are not there in the solution.
4. Make the new solution feasible using the heuristic.
5. If the solution is in the range of agent territory
   radius then eliminate that solution
6. Else
7. Save that solution.
8. End for
9. Find the best location in all the points that are
   obtained around each grenade.
10. If the oldloss<newloss
11. Then replace the old solution with the new
   solution
```

We used another method also to randomly generate random location around each grenade that is first we will select a best grenade to find the solutions around it and then we will randomly select some other solution to use some nodes in that solution to replace in the first selected solution this is similar to cross over in the genetic algorithm.

Algorithm 3.4 shows another procedure to generate random location around each grenade

Algorithm 3.4 Pseudo code for method2 for the random generation of new solutions

```
1. Select a location to generate points around it.
2. For i=1 to number of shrapnel-1.
3. Randomly select a solution and use the nodes in
   that solutions to replace in the solution for which
   we want to generate locations around it
4. Make the new solution feasible using the heuristic.
5. If the solution is in the range of agent territory
   radius then eliminate that solution
6. Else
7. Save that solution.
8. End for
9. Find the best location in all the points that are
obtained around each grenade.
10. If the oldloss<newloss
11. Then replace the old solution with the new
   solution
```



Input Graph

If for example solution 1 2 3 5 is selected to generate some random locations around it and then if we select some other solution 3 4 5 to use some of its nodes to replace in the first selected solution and if after replacement if the solution is changed to 1 2 4 5 then this is not a clique so use repair procedure to convert into clique.

Selected solution	after random replacement	after repair procedure
1 2 3 5		
	⟶ 1 2 4 5	⟶ 1 2 5
3 4 5		
Clique	not a clique	clique

3.2.3 Repair Procedure

In this repair procedure we will take the generated random sub graph as an input and convert it into a sub graph by using the following procedure [20]. This procedure works as follows first we will select a random node from the graph and we will find the list of nodes (S) that are not connected to the random node selected and a random number is generated if that number is less than a certain value(P_{delall}) then we will remove the list s from the graph and if the product of weight and degree of the node V is less than the average of all the nodes in S then remove the random node V from the graph else remove list of nodes S from the generated graph. Algorithm 3.5 shows the procedure for repair.

Algorithm 3.5 Pseudo code for the repair procedure

```
U←C
While (U ≠ ∅) {
v ← random (U)
S ← {u: u ∈ C ∧ (u, v)  ∉ E}
If (u01 ≤ P_delall)
{
C← C − S
U← U − S
}
Else if (deg (v) x w (v) ≤ C_f × ∑ deg(S) x w(S)/|S|)
C ← C − {v}
Else {
C← C − S
U← U − S
}U ← U − {v}
} Return C
```

3.2.4 Maximum clique Heuristic

We used Algorithm 3.6 to convert the clique into maximum clique[20]. First we take the clique as the input and then we will find the list of elements (C_{ad}) that are connect to all the nodes in the input clique then we will generate a random number and if that random number is less than or equal to P_{ad} then we will find the vertex from the set C_{ad} which maximizes the product of its weight and its local degree. Local degree is the numbers of the vertices that the current vertex is connect to in the set $C_{ad.}$ Then we will add that node to the current clique and recomputed the list C_{ad}. If the random number is greater than the value pad then we will randomly choose a node and add to the current clique then recomputed the list $C_{ad.}$ We will repeat the above procedure until the list C_{ad} is empty. Figure 3.1 shows the Maximum weight clique obtained after applying the maximum clique heuristic on the input graph and clique.

Algorithm 3.6 Pseudo code for the Maximum clique heuristic

```
Cad ← {v: v ∉ Cc ∧ (u, v) ∈ E ∀ u ∈ Cc}
  While (|Cad| > 0)
   {
    If (u01 ≤ Pad)
       V ← arg max (w (u) ×| {t: t ∈ Cad ∧ (u, t) ∈ E}|)
   Else
       V ← random (Cad)
      Cc ← Cc ∪ {V}, recalculate Cad
   }
  Return Cc
 Cad = Set of vertices adjacent to current clique
 Cc   = resultant clique
```

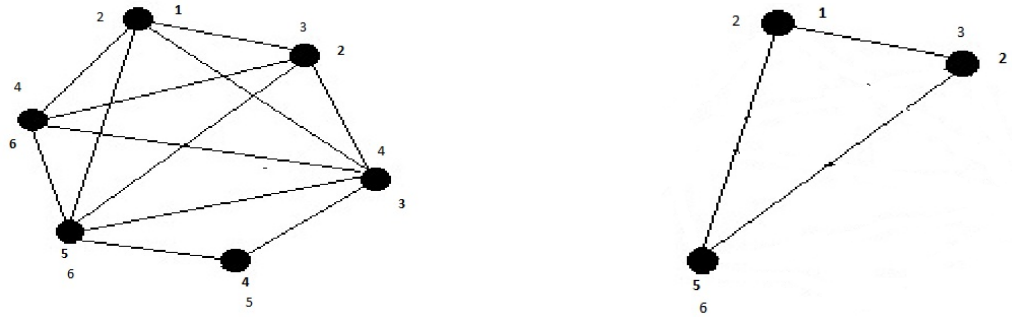


Figure 3.1(a) Input to Maximum clique heuristic

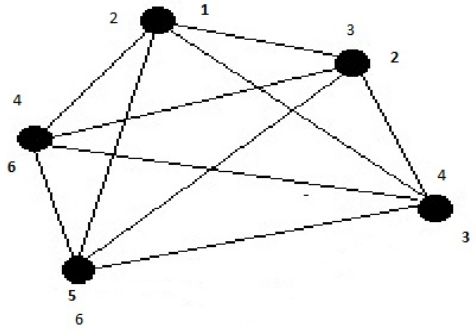


Figure 3.1(b) Output of Maximum Clique Heuristic.

3.2.5 Finding Best Location to Place the next Grenade

We used the algorithm 3.7 to find the best solution among all the solutions generated around each grenade. First find the loss at the first location and then if the loss at the next location is greater than the current location then replace the new loss with the current loss and repeat the above procedure for all the locations around each grenade.

Algorithm 3.7 finding the best solution to place next grenade

```
1. Find the loss at the first location and memorize it as
   oldloss
2. For all the locations around each grenade
3. Find the loss at the next location and remember it as
   curloss
4. If curloss>oldloss
5. Then replace the old solution and loss with the
   current solution and loss
6. End for
```

Table 3.1 shows the variables that we used and the roles that they play in the algorithm and their place of use.

Table 3.1 The roles, places of use and values of different parameters used in GEM for Maximum weighted Clique

Name	Place of use	Role	Value
Number of Grenades	GEM	Gives the population size	50
R_t	GEM	Determines the agent territory radius	1
P_{ad}	Max Heuristic	Gives the probability with which the vertex, having the highest local degree among adjacent vertices, to be included in the clique	0.60
$P_{dellall}$	Repair Procedure	Gives the probability of deleting all those vertices from the clique which are not connected with the selected vertex.	0.50
C_f	Repair Procedure	Determines the weight given to average degree of all those vertices not connected to the selected vertex in comparison to the degree of selected vertex.	1.10

Chapter 4

Results and Discussion

In this chapter we will discuss the results obtained by solving the Maximum weight clique problem by using Grenade Explosion Algorithm.

4.1 Experimental set up

4.1.1 Platform

We have used C programming language to implement GEM algorithm for Maximum weight clique problem and is executed on Intel core 2 quad system with 4 GB RAM running at 2.4 GHz under Linux platform.

4.1.2 Algorithmic Parameters

We have used 50 Grenades. We are running this algorithm for 500 iterations or until we found the optimal solution (if known). The probability of deleting all those vertices from the clique which are not connected with the selected vertex $P_{dellall}$=0.50. The probability with which the vertex, having the highest local degree among adjacent vertices, to be included in the clique P_{ad}=0.60. The weight given to average degree of all those vertices not connected to the vertex in comparison to the degree of selected vertex C_f=1.1. If the solution is not improved further for more than 200 iterations then the solution is abandoned and it is saved as the best solution in that iteration.

4.2 Results

We are able to test GEM Maximum Weight Clique Problem for not only on 100 and 200 but also on 300 vertex graph with edge density up to 0.80 like in WT-HSS.

Table 1 compares the performance of GEM for Maximum Weight Clique Problem with WT-HSS on normal graphs while Table 2 compares the performance of our algorithm on irregular graphs.

Östergard [11] proposed 15 benchmark instances for the maximum weight clique problem based on real life coding theory problems. Number of nodes varied from 132 to 8914 in these instances. Table 3 shows the performance of our algorithm on these benchmark instances. Here GEM for Maximum Weight Clique Problem was executed 20 times on each benchmark instance each time with a different random seed. Table 3 reports the best, average and standard deviation of maximal weight cliques found by GEM for Maximum Weight Clique Problem on each instance. It also reports the average and standard deviation of time to find the best solution. % Opt is the percentage number of runs for which the optimal clique value was found by our algorithm in all runs of the algorithm. GEM for Maximum Weight Clique Problem was able to find the optimal value for all instances. There are nine instances for which %Opt is 100% indicating that our algorithm was able to find the optimal value in all 20 runs. Computation times were also quite small.

Table4.1 Performance of WT-HSS with GEM and without GEM on normal random weight graphs

N	Density	GEM	WT-HSS
		Percentage optimal	Percentage optimal
100	0.10	100.00%	100.00%
100	0.20	100.00%	100.00%
100	0.30	100.00%	100.00%
100	0.40	100.00%	100.00%
100	0.50	100.00%	100.00%
100	0.60	100.00%	100.00%
100	0.70	100.00%	100.00%
100	0.80	100.00%	100.00%
100	0.90	100.00%	99.94%
100	0.95	100.00%	99.98%
200	0.10	100.00%	100.00%
200	0.20	100.00%	100.00%
200	0.30	100.00%	100.00%
200	0.40	100.00%	100.00%
200	0.50	100.00%	100.00%
200	0.60	100.00%	100.00%
200	0.70	100.00%	99.92%
200	0.80	100.00%	99.91%
300	0.10	100.00%	100.00%
300	0.20	100.00%	100.00%
300	0.30	100.00%	100.00%
300	0.40	100.00%	100.00%
300	0.50	100.00%	100.00%
300	0.60	99.91%	99.86%
300	0.70	100.00%	99.15%
300	0.80	99.80%	99.39%

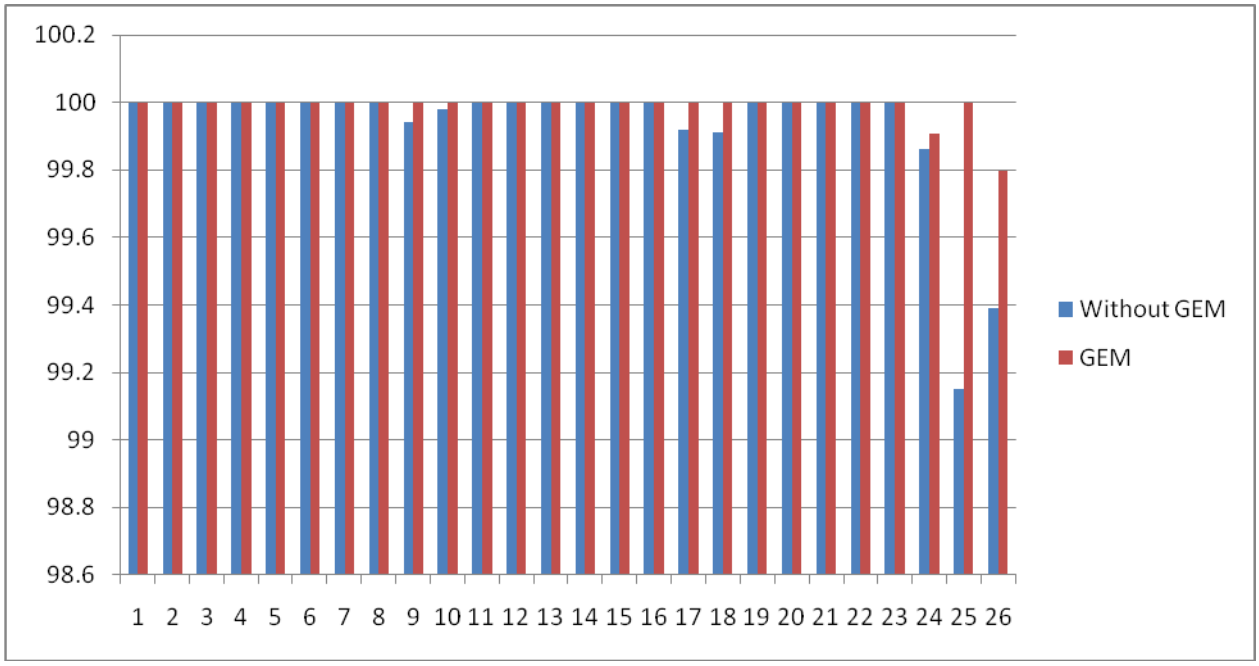
Table 4.2 Performance of WT-HSS with GEM and without GEM on irregular random weight graphs

N	Density	GEM	WT-HSS
		Percentage optimal	percentage optimal
100	0.10	100.00%	100.00%
100	0.20	100.00%	100.00%
100	0.30	100.00%	100.00%
100	0.40	100.00%	100.00%
100	0.50	100.00%	100.00%
100	0.60	100.00%	100.00%
100	0.70	100.00%	100.00%
100	0.80	100.00%	100.00%
100	0.90	100.00%	100.00%
100	0.95	100.00%	100.00%
200	0.10	100.00%	100.00%
200	0.20	100.00%	100.00%
200	0.30	100.00%	100.00%
200	0.40	100.00%	100.00%
200	0.50	100.00%	100.00%
200	0.60	100.00%	100.00%
200	0.70	100.00%	100.00%
200	0.80	100.00%	100.00%
300	0.10	100.00%	100.00%
300	0.20	100.00%	100.00%
300	0.30	100.00%	100.00%
300	0.40	100.00%	100.00%
300	0.50	100.00%	100.00%
300	0.60	100.00%	100.00%
300	0.70	100.00%	100.00%
300	0.80	100.00%	100.00%

Table 4.3 Performance of WT-HSS with GEM on standard benchmark instances of Östergard

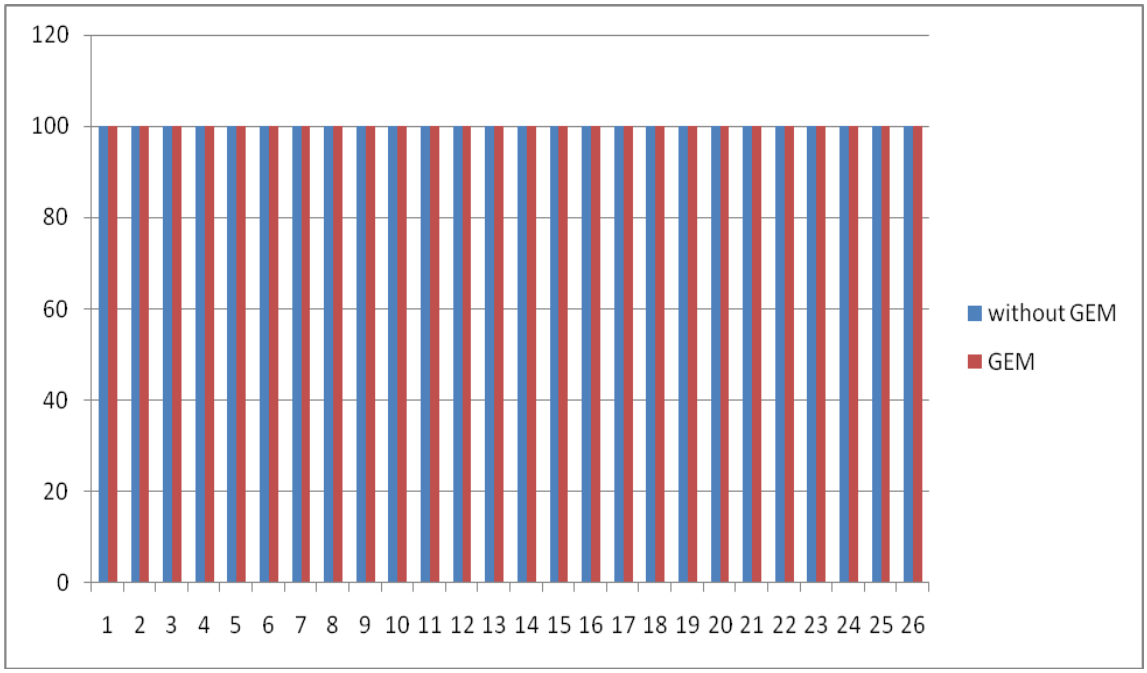
Instance	n	opt	Clique Size				Time
			Best	Avg	St.Dev	%opt	
11-4-4	150	34	34	34.00	0.00	100%	0.029
12-4-6	230	110	110	110.00	0.00	100%	0.4875
14-4-7	223	282	282	280.299	4.1605	75%	1.5025
14-6-6	807	42	42	42.00	0.00	100%	0.204
16-4-5	156	322	322	322.00	0.00	100%	0.2905
16-8-8	2246	30	30	30.00	0.00	100%	0.041
17-4-4	132	156	156	156.00	0.00	100%	0.032
17-6-6	558	70	70	70.00	0.00	100%	1.087
19-4-6	263	1448	1448	1448.00	0.00	100%	0.958
19-8-8	2124	62	62	62.00	0.00	100%	0.9015
20-6-5	1302	83	83	83.00	0.00	100%	0.0645
20-6-6	1490	190	190	189.00	3.00	90%	10.7115
20-8-10	2510	83	83	83.00	0.00	100%	0.065
20-10-9	5098	26	26	26.00	0.00	100%	0.5615
20-10-10	8914	46	46	46.00	0.00	100%	1.7015

The above table shows the working of WT-HSS using GEM on Östergard instances. To get the results we gave each file as the input and we run the algorithm for twenty iterations, we saved the best solution in every iteration, after that we calculated the average of the twenty values.



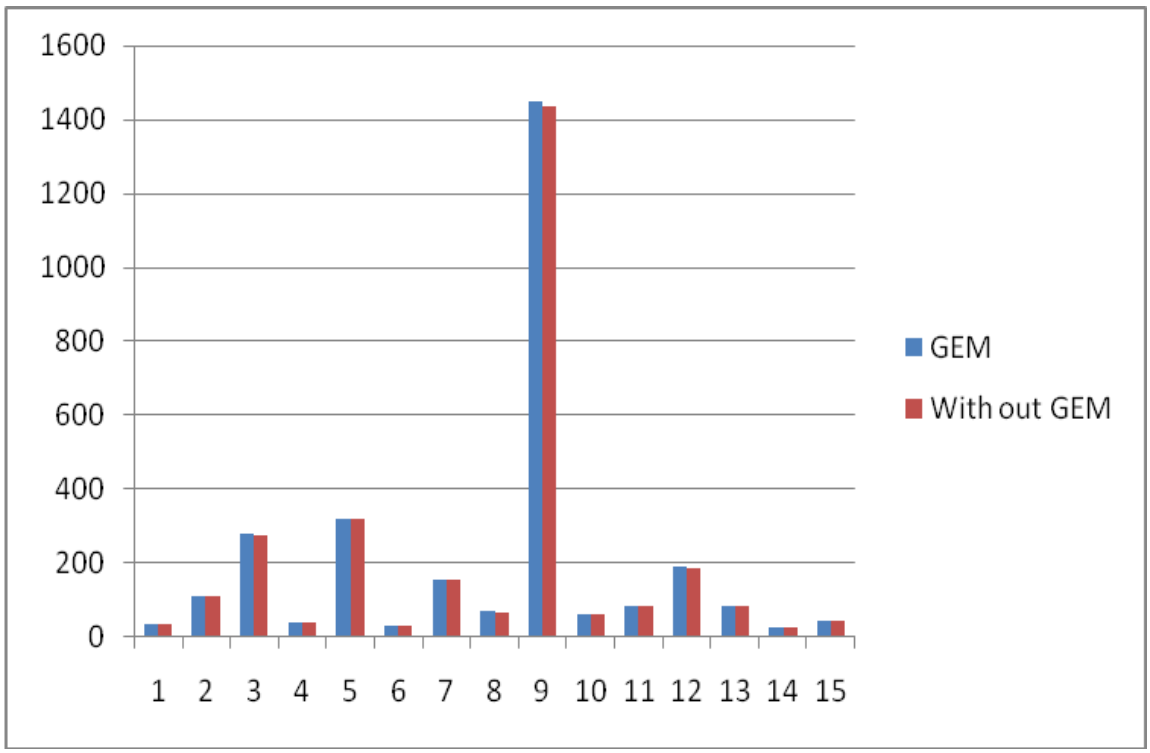
Graph 4.1 Graph between Performance of WH-HSS with GEM and without GEM on random weight graph

The above graph indicates the comparison of the working of WT-HSS with using Grenade explosion and without using Grenade explosion algorithm on the random weight graphs. It clearly shows that the results that are the percentage of getting the exact optimal solution for the given input set of graphs is improved when we solve the problem by using the Grenade Explosion algorithm. In calculating the percentage output we gave twenty graphs with same number of nodes and density as input to the program and we calculated the percentage of number of input graphs for which we got the exact optimal value.



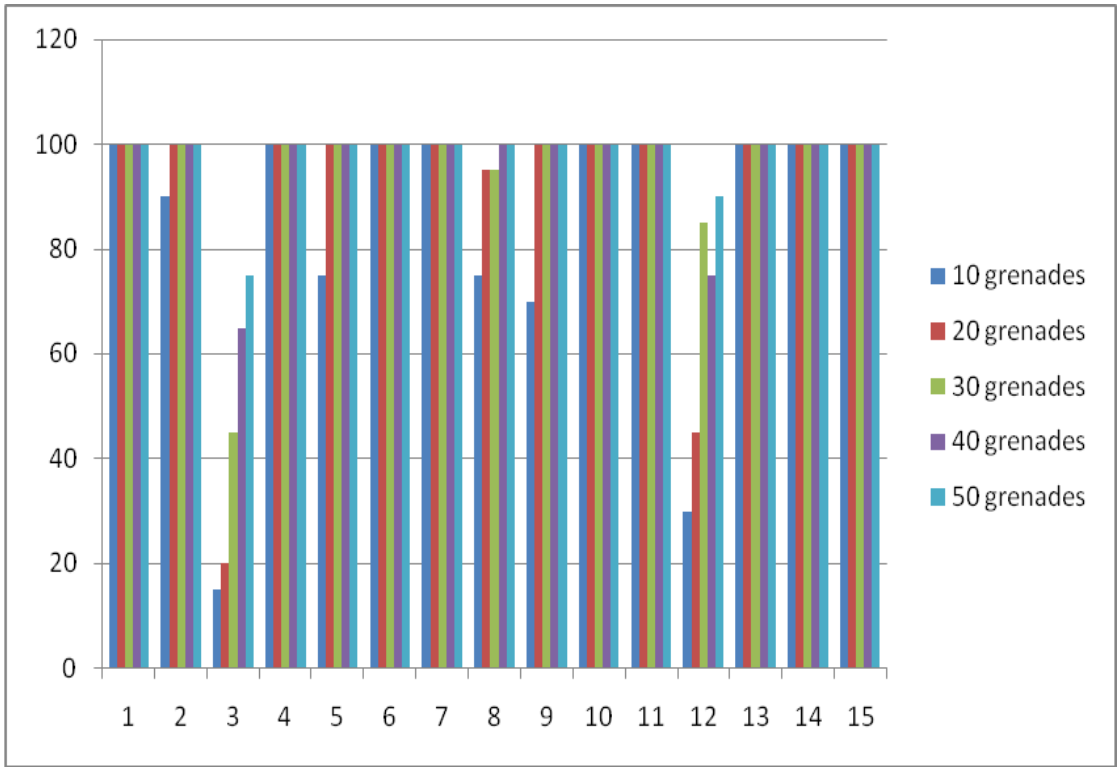
Graph 4.2 Graph between Performance of WT-HSS with GEM and without GEM on Irregular random weight graphs

The above graph indicates the comparison of the working of WT-HSS with using Grenade explosion and without using Grenade explosion algorithm on the irregular random weight graphs. It clearly shows that the results that are the percentage of getting the exact optimal solution for the given input set of graphs is same as the normal WT-HSS when we solve the problem by using the Grenade Explosion algorithm. In calculating the percentage output we gave twenty graphs with same number of nodes and density as input to the program we run the algorithm once for each file and we calculated the percentage of number of input graphs for which we got the exact optimal value.



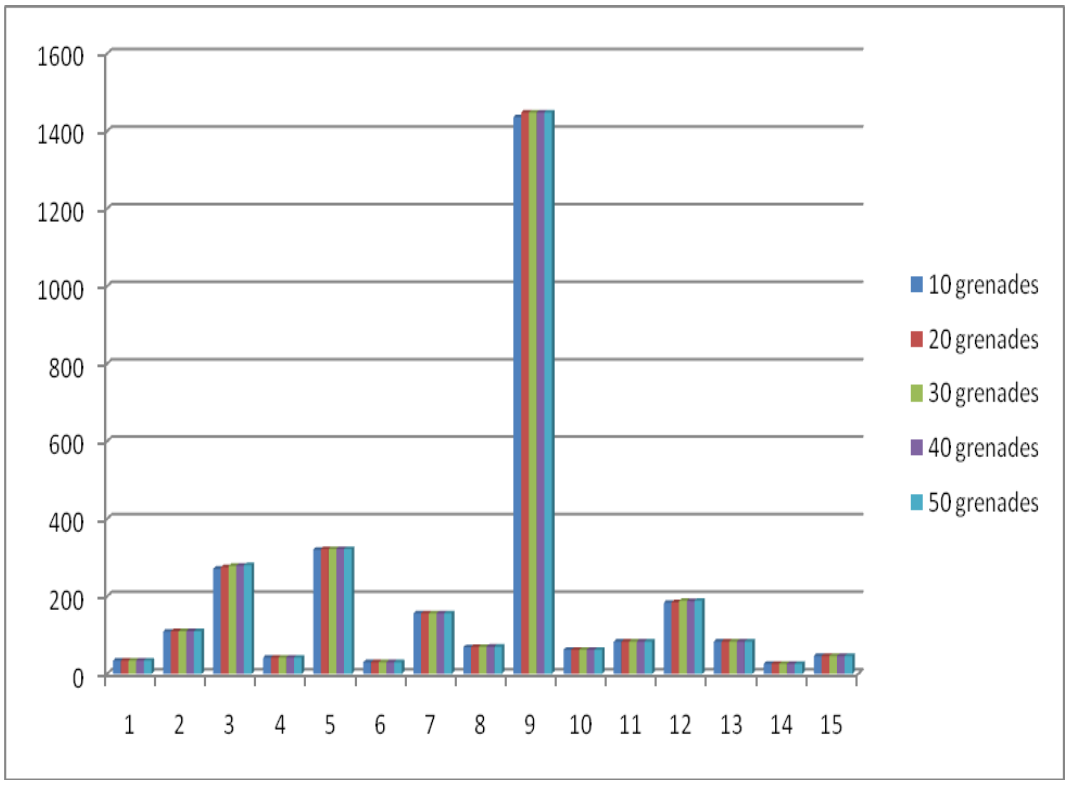
Graph 4.3 Graph between Performance of WT-HSS with GEM and without GEM on standard benchmark instances of Östegard.

The above graph compares the average values of the clique for the given input graph that is calculated using normal WT-HSS and WT_HSS with Grenade Explosion Algorithm on standard benchmark instances of Östegard. It clearly shows that the average clique weight is increased when we use WT-HSS with Grenade Explosion than the normal WT-HSS and the time to compute the average value by using the Grenade Explosion is also similar to the WT-HSS



Graph 4.4 Graph between Percentage optimal values for the standard benchmark instances of Östegard with different number of grenades

We used our algorithm on the standard benchmark instances of Östegard with different number of grenades, and compared the percentage optimal values; it clearly shows that the percentage output of the optimal value is close to best with 50 grenades.



Graph 4.5 Graph between Average values for the standard benchmark instances of Östegard with different number of grenades

We used our algorithm on the standard benchmark instances of Östegard with different number of grenades, and compared the average clique weight; it clearly shows that the average clique weight is close to the best clique weight with 50 grenades.

Chapter 5

Conclusions and Future works

5.1 Conclusions

This project has dealt with the applicability of a new Evolutionary Algorithm called Grenade Explosion Method (GEM) for the Maximum Weight Clique Problem. GEM was originally designed for continuous optimization problems. To our knowledge this is the first application which uses GEM for the discrete optimization. Computational results on the benchmark instances show the effectiveness of our proposed GEM approach. Our approach has outperformed the previous best approach on both Östergard as well as randomly generated instances.

5.2 Future Works

The Grenade Explosion Algorithm worked well on the maximum weight clique problem and it gave good results. However, our approach uses a static agent territory radius. The algorithms performance may be improved with the dynamic agent territory radius so future work can focus in this direction. Another direction for future works lies in exploring the possibility of combining the GEM approach with some exact method for the maximum weight clique problem

Our approach has demonstrated the applicability of GEM for discrete optimization problems. Similar approaches can be designed for other discrete optimization problems also.

References

[1] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: W. H Freeman, 1979.

[2] S. Khot. "Improved Inapproximability Results for Max Clique, Chromatic Number and Approximate Graph Coloring". In Proc. of the 42nd Annual IEEE Syposium on the Foundations of Computer Science, pp. 600-609, 2001.

[3] G. L. Nemhauser and L. E. Trotter. "Vertex Packings: Structural Properties and Algorithms", Math. Programming, VIII, pp. 232-248, 1975.

[4] E. Loukakis and C. Tsouros. "An Algorithm for the Maximally Internally Stable Set in a Weighted Graph", Int J. Computer Math., XIII, pp. 117-129, 1983.

[5] R. Carraghan and P. M. Pardalos. A Parallel Algorithm for the Maximum Weight Clique Problem, Tech. Rep. CS-90-40, Department of Computer Science, Pennsylvania State University, 1990.

[6] P. M. Pardalos and N. Desai. "An Algorithm for Finding a Maximum Weighted Independent Set in an Arbitrary Graph", Int. J. Computer Math., XXXVIII, pp. 163-175, 1991.

[7] E. Balas and J. Xue. "Minimum Weighted Coloring of Triangulated Graphs, with Application to Maximum Weight Vertex Packing and Clique Finding in Arbitrary Graphs", SIAM Journal of. Computing, XX, pp. 209- 221, 1991.

[8] E. Balas and J. Xue. "Weighted and Unweighted Maximum Clique Algorithms with Upper Bounds from Fractional Coloring", Algorithmica, XV, pp. 397-412, 1996.

[9] E. Balas and W. Niehaus. "Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems", Journal of Heuristics, IV, pp. 107-122, 1998.

[10] L. Babel. "A Fast Algorithm for the Maximum Weight Clique Problem", Computing, LII, pp.31-38, 1994.

[11] P. R. J. Östergard. "A New Algorithm for the Maximum Weight Clique Problem," Nordic Journal of Computing, VIII, pp. 424-436, 2001.

[12] I. M. Bomze, M. Pelillo and V. Stix. "Approximating the Maximum Weight Clique using the Replicator Dynamics", IEEE Transactions on Neural Networks, XI, pp. 1228-1241, 2000.

[13] T. S. Motzkin and E. G. Strauss. "Maxima for Graphs and a New Proof of Theorem of Turan", Canadian Journal of Mathematics, XVII, pp. 533-540, 1965.

[14] A. Massaro, M. Pellilo and I. M. Bomze. "A Complementary Pivoting Approach to the Maximum Weight Clique Problem", SIAM Journal of Optimization, XII, pp. 928-948, 2002.

[15] S. Busygin. "A New Trust Region Technique for the Maximum Weight Clique Problem", Discrete and Applied Mathematics 2080-2096, 2006.

[16] M. Gendreau and J.J.- Y.Potvin. Metaheuristics in combinatorial optimization. In Annals of operation research, 2005.

[17] J.F. Goncalves, J.J.M. Mendes, M.G.C. Resende, A genetic algorithm for the resource constrained multi-project scheduling problem, European Journal of Operational Research 189, 1171-1190, 2008.

[18] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of global optimization 39 459-471, 2007.

[19] A. Ahrari, A.A. Atai, Grenade Explosion Method-A novel tool for optimization of multimodal functions, Applied Soft Computing Journal optimization letters, Volume 4, Number4 , Pages 531-541, 2010.

[20] Alok Singh and Ashok Kumar Gupta, A Hybrid Evolutionary Approach to Maximum Weight Clique Problem, International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.2, No.4, pp. 349-355, 2006.