ARTILLERY TARGET ASSIGNMENT PROBLEM WITH TIME DIMENSION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURÇİN SAPAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2008

Approval of the thesis:

ARTILLERY TARGET ASSIGNMENT PROBLEM WITH TIME DIMENSION

submitted by **BURÇİN SAPAZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Müslim Bozyiğit
Head of Department, **Computer Engineering** ________________

Prof. Dr. İsmail Hakkı Toroslu
Supervisor, **Computer Engineering Department, METU** ________________

Examining Committee Members:

Prof. Dr. Faruk Polat
Computer Engineering Department, METU ________________

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU ________________

Prof. Dr. Göktürk Üçoluk
Computer Engineering Department, METU ________________

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU ________________

M. Sc. Mustafa Kemal Kaplan
Central Bank of the Republic of Turkey ________________

Date: ________________

Name, Last Name:	BURÇİN SAPAZ
Signature :	

ABSTRACT

ARTILLERY TARGET ASSIGNMENT PROBLEM WITH TIME DIMENSION

Sapaz, Burçin
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. İsmail Hakkı Toroslu

December 2008, 66 pages

In this thesis, we defined a new assignment problem and named it as the artillery target assignment problem(ATAP). The artillery target assignment problem is about assigning artillery weapons to targets at different time instances while optimizing some objectives. Since decisions at a time instance may affect decisions at other time instances, solving this assignment problem is harder than the classical assignment problem. For constructing a solution approach, we defined a base case and some variations of the problem which reflects sub-problems of the main problem. These sub-problems are investigated for possible solutions. For two of these sub-problems, genetic algorithm solutions with customized representations and genetic operators are developed. Experiments of these solutions and related results are presented in this thesis.

Keywords: Assignment problem, weapon target assignment, artillery target assignment problem, genetic algorithms

ÖZ

ZAMAN BOYUTUNDA TOPÇU HEDEF TAHSİSİ PROBLEMİ

Sapaz, Burçin
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. İsmail Hakkı Toroslu

Aralık 2008, 66 sayfa

Bu tezde, yeni bir eşleştirme problemi tanımladık ve bunu, topçu hedef eşleştirme problemi (THEP) olarak isimlendirdik. Topçu hedef eşleştirme problemi, bazı amaçlar optimize edilirken farklı zaman dilimlerinde topçu silahlarının hedeflerle eşleştirilmesiyle ilgilidir. Bir zaman dilimindeki karar başka zaman dilimlerindeki kararları etkileyebildiğinden, bu eşleştirme problemini çözmek klasik eşleştirme problemini çözmekten daha zordur. Bir çözüm yaklaşımı kurmak için ana problemin alt problemlerini yansıtan bir temel durum ve problemin bazı varyasyonlarını tanımladık. Bu alt problemleri muhtemel çözümler için inceledik. Bu alt problemlerden ikisi için, özelleştirilmiş gösterim ve genetik operatörlerle genetik algoritma çözümleri geliştirdik. Bu çözümlerin deneyleri ve ilgili sonuçlar bu tezde sunuldu.

Anahtar Kelimeler: Eşleştirme problemi, silah hedef eşleştirmesi, topçu hedef eşleştirme problemi, genetik algoritmalar

To my family and friends

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. İsmail Hakkı Toroslu for his support, guidance, patience, critics and valuable advices during this study.

I would also like to thank to Prof. Dr. Faruk Polat, Prof. Dr. Göktürk Üçoluk, Asst. Prof. Dr. Tolga Can, Murat Yükselen, Utku Erdoğdu, Ali Galip Bayrak, Bahar Pamuk and Mehmet Gülek for their support, critics and ideas in the project of METU-TAF MODSIMMER.

This study is financially supported by TÜBİTAK.

I wish to express my thanks to all members of Computer Engineering Department, being one of them has always been a great pleasure. Specifically, I want to thank Dr. Meltem Turhan Yöndem for her guidance during my education. Also I want to thank Assoc. Prof. Dr. Veysi İşler for his support and patience during my graduate study.

Finally, I want to express my thanks to my family and to all of my dear friends, especially Murat Yükselen, Sevgi Yaşar, Duygu Saraçoğlu, Sinan Kalkan and Esin Saka, for their endless support during this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this thesis, we worked on assignment problems related to the artillery branch of military. These problems mainly concern about assigning artillery weapons to targets at different time instances by considering some domain related constraints. With this overall definition, in this thesis we introduced an assignment problem and named it as Artillery Target Assignment Problem (ATAP). This problem is derived from the domain of artillery branch of military and from the related real life military principles.

From a general point of view, this problem is a variation of the general assignment problem and more specifically weapon target assignment problems. When we take a deeper look in the problem, we will see that each distinct property of the general problem will lead to a very different sub-problem which has different complexities in it.

In the following chapter, artillery branch of military is briefly introduced in order to give a first glance at the challenges in this area. Then in the third chapter, related assignment problems and current studies in the literature are introduced with a summary. At the fourth chapter, on top of this background knowledge, the artillery target assignment problem (ATAP) and its variations are defined in detail. At the fifth chapter, for two variations of the main problem, our solution approaches which employ customized genetic algorithms are explained. Then in the sixth chapter, information about implementation of our solutions, created test environment and results of our experiments are presented. In the last chapter, a conclusion on our study is given.

CHAPTER 2

ARTILLERY BRANCH OF MILITARY

In this chapter, we will try to summarize the general process that is being used in military for artillery usage including assignment and shooting. Mainly because of the characteristics and abilities of its weapons, artillery branch of military has its own dynamics at each step of usage process and thus differ from other branches of military.

Most artillery weapons are not very precise since they are usually used for long range shootings. However even imprecise shootings of artillery weapons can create damage on targets. As a result of these, one of the primary goals of artillery branch of military is generally shooting up as many targets as possible while creating as much damage as possible on targets, rather than precisely destructing many targets.

In military, artillery weapons are hierarchically organized to form artillery batteries. The hierarchy is a two level structure; the batteries are composed of sections and sections are composed of single artillery weapons. In general, each battery is composed of two sections and includes four to eight single artillery weapons. A single artillery weapon can be a cannon, a howitzer or a mortar. Each of these types has different kinds of weapons resulting in different abilities. However, each battery and thus each section include only one kind of single artillery weapon. Regardless of the hierarchical size of the weapon, generally we will refer each of these (i.e. battery, section, single artillery weapon) as artillery weapons.

Artillery weapons can only shoot when they are emplaced to an artillery position. However because of being stationary, any such weapon is an easy target for opposing forces. Therefore frequently changing emplacements of artillery weapons under some circumstances is a usually preferred tactic. Each such change of emplacement requires assemble (i.e. preparation before moving) time, movement time and a set up time which as a total causes a delay for the consecutive upcoming shooting. Moreover, even if the artillery weapon does not change its emplacement, still it is possible that a preparation time for the consecutive shooting is required. To sum up, each change of emplacement and each shooting may cause a delay for the consecutive shooting. Therefore each artillery weapon may have time periods in which the weapon is unavailable to shoot.

The area inside which an artillery weapon is able to shoot is named as the effective area of this weapon and size of the effective area is related to the abilities of the weapon. Different kinds of artillery weapons may own different types of artillery shells and fuzes. Moreover some shells or fuzes may not exist in the current configuration of the related weapon. Likewise in general military tactics, in artillery each target must be shot by appropriate ammunition (i.e. shell and fuze). As a result of effective area and appropriate ammunition, each artillery weapon may not be a good choice to shoot each target.

Targets of artillery are determined beforehand by previous intelligence (i.e. gathered information) and current tactics or determined on runtime during the war by requests (call for fire) sent by various observers such as forward observers, infantry units and specialized acquisition radars. The targets are usually defined by rectangular areas which may enclose grouped static objects like bridges, buildings or movable enemy units like a group of tanks, infantries or artillery weapons. Close targets with similar kind of enclosed elements are grouped together to create larger targets. Targets have different priorities which are determined by value of the related target. The value of a target is specified according to many different parameters which are mainly related to current tactics and possible effectiveness on friendly forces. Some examples of such parameters are previously defined precedence groups of targets, tactical criticalness, closeness to friendly forces, possible effectiveness according to type of forces, which friendly force observed the target, which friendly force is threatened by the target and which friendly forces are supported with a higher priority. Targets with higher priority (i.e. with higher target values) are shot up by artillery weapons. With these shootings, these targets may or may not be destructed since a target may need more than one shot in order to be destructed. With this situation in mind, sometimes it is also possible to guess the number of shootings required to destruct a target and plan a fixed number of shootings for the related target beforehand.

There are various sub-decisions which must be given during the decision for a shooting from artillery weapons to a target. One of them is the decision for appropriate ammunition that will be used in the shooting to the related target. Another one is about physical dynamics of the shooting like determining vertical shooting angle, calculating the required lead (i.e. trying to calculate the future position of the target when the shooting will be hitting) and taking into account the amount of possible deviation. For such calculations, sometimes a preceding shooting named as adjust fire is made prior to the original one and gathered information is used in the original shooting. Also at this point it is important to keep in mind that even a close shot is actually considered to be successful for artillery weapons.

When fired, each single artillery weapon can create damage in a circular area centered at the position where the shell explodes. This circular area can be referred as the coverage area of this single weapon. When single weapons are grouped hierarchically to create sections and batteries, the coverage area of this larger artillery unit is the combined coverage areas of all single artillery weapons. The method of combination is determined by the concept of sheaf which defines the positioning of each single weapon with respect to each other and with respect to the target position. There are various sheaf types like converged sheaf, parallel (regular) sheaf, open sheaf and special sheaf. Width of sheaf is the sidelong length of the coverage area created by using the related sheaf with current artillery weapon. As an example, for parallel sheaf, coverage areas of each weapon are positioned side by side without any space between and thus the width of parallel sheaf can be easily calculated by summing up diameters of coverage areas. Each sheaf type creates different coverage areas with different widths and as a result each creates different amount of damage on same target. For example, converged sheaf creates more damage than parallel sheaf, whereas open sheaf creates less damage than parallel sheaf. Selecting the appropriate sheaf is another sub-decision that must be given during the shooting process.

If the area of a target can not be covered with one shot of an artillery weapon, then multiple consecutive shots have to be planned in order to cover the related target. Such multiple consecutive shots have been referred as volley fire (group fire). Volley fire can have any number of shots and these shots can be fired from one or more artillery weapons. As an example, in order to cover a large target, a volley fire with four shots can be required and these four shots can be fired from two different artillery weapons which even do not belong to the same hierarchy. There is a decision issue between two choices about timing of consecutive shots of a volley fire. In the first choice, each shot can be fired rapidly as soon as the related artillery weapons are ready to fire. In the second choice, before consecutive shots there can be some delay in time for tactical reasons. As a total, planning of a volley fire with number of shots, related artillery weapons, assignment of weapons to related shots and timing considerations is another sub-decision of the shooting process.

CHAPTER 3

RELATED STUDIES ON ASSIGNMENT PROBLEMS

The assignment problem is a well-known optimization problem and there are many different variations of the assignment problem in different domains. In a general manner, assignment problems can be defined as optimization problems where between some sets of elements a matching is constructed while optimizing some cost functions and satisfying some constraints.

The well known classical assignment problem, also known as linear assignment problem, can be exemplified by the personnel assignment problem. In [16], personnel assignment problem is defined as “the problem of choosing an optimal assignment of n men to n jobs, assuming that numerical ratings are given for each man’s performance on each job. An optimal assignment is one which makes the sum of the men’s ratings for their assigned jobs a maximum”.

The classical assignment problem can be modeled as a weighted bipartite graph and then the problem becomes finding a maximum weighted matching on this graph. There is a polynomial time algorithm named as Kuhn-Munkres algorithm for the solution of this problem [10] [16]. Efficient implementations of the algorithm can solve the problem in $O(n^3)$ time.

The classical assignment problem is a two index assignment problem since two sets of elements are matched. When there are more sets of elements to be matched in a problem, the problem is a Multi Index Assignment Problem (MUIP) [19]. As an example, consider a variation of the personnel assignment problem. Assume in this variation we also have n tools which also affect ratings and we are trying to match each person to the best proper tool to do the best job so that total ratings will be maximized. This example of assignment problem is a Three Index Assignment Problem since we are trying to match three sets of elements. A formal definition for the Three Index Assignment Problem can be found at [2].

In the literature, multi index assignment problem is also referred as multi dimensional assignment problem. Similarly, three index assignment problem is also referred as three dimensional assignment problem.

In research about military issues, Weapon Target Assignment (WTA) Problem is a well known assignment problem. Although there are many variations of the WTA, there are two main classes; static WTA (SWTA) and dynamic WTA (DWTA). According to [9], definition of the most widely studied WTA problem, which is referred as asset-based SWTA is given as assigning weapons to targets with an example objective of minimizing “the expectation of the loss of assets of the defensive”. For this purpose, weapons, assets and targets are defined with asset values, with kill probability of weapons on targets and with destroying probability of targets on assets.

Since WTA is a complex optimization problem, in the literature there are various studies which try to solve the problem with genetic algorithm approaches. Studies presented in [12], [13] and [14] can be given as examples of such approaches.

In the literature, there are also studies related to capacity issues in assignment problems. A well known example is the Generalized Assignment Problem (GAP). [5] gives a clear definition of the generalized assignment problem. There are two sets B and S. B has m bins each having a capacity and S has n items. Each bin and item tuple has been defined a size and a profit. The objective is to find a subset of S that can be feasibly packed in B so that the total profit is maximized.

Among many different versions of WTA problems, “The Generalized Weapon Target Assignment Problem” defined in [18] is interesting since it “extends the basic WTA problem by allowing for multiple target assignments per weapon”.

In the literature, there are not many studies on Weapon Target Assignment Problem which specifically considers different aspects of the artillery branch. One rare example is the study “Targeting and Scheduling Problem for Field Artillery” in which targeting and fire sequencing problems about artillery are investigated [11].

In [8], assignment problem is extended by definition of hierachy in the matching sets. As a result, some new variations are defined for the assignment problem and possible solutions are investigated. For one of the variations which has been named as “maximum-weighted tree matching problem”, a genetic algorithm solution is presented. In this solution, repair strategy is preferred for the offsprings created by crossover.

CHAPTER 4

ARTILLERY TARGET ASSIGNMENT PROBLEM

4.1 Overall Problem Definition

In military, assignment of artillery weapons for targets is a complex decision process which includes many different parameters to be considered. Mainly at each discrete time instance, available weapons are assigned to proper targets with the best possible assignment in order to perform shooting to these targets. Although, it is already difficult to determine the best assignment for a single time instance, the main challenge is that the decisions for a time instance usually affects the decisions for other time instances.

In this problem, there are there main kinds of elements to be defined;

Target: Targets are possible artillery targets and shooting up these targets is the primary goal in artillery assignment. There are m targets; T_1, T_2, T_3,... T_i, ..., T_m

Weapon: Weapons are artillery weapons and at each discrete time instance each of them fires a shot. There are n weapons; W_1, W_2, W_3,... W_j,..., W_n

Time: Time is expressed by discrete time steps. It is assumed that the decision process for artillery branch of military starts and ends between each consecutive discrete time steps. As a result, at each discrete time step a weapon can shoot. Discrete time steps starts at 0 and ends at p; Z_0, Z_1, Z_2,..., Z_k,..., Z_p

Artillery target assignment problem (ATAP) is about assigning *n* weapons to *m* targets in *(p+1)* discrete time instances with best possible cost by means of required optimization parameters.

As given in the definition, trying to find the best assignment in artillery target assignment problem is about optimizing a cost function. The optimization is achieved by minimizing or maximizing the values stated in the cost function. By means of artillery target assignment problem, this cost function can differ according to the related military tactics. Because of this, instead of defining a single cost function for the problem, we will give some examples of possible cost functions. This examples can also be mixed together to define coupled cost functions.

- Maximizing the total value of targets which has been shot up by weapons
- Minimizing the total time that is required to shot up all targets
- Maximizing the total value of shots at a limited time period
- Minimizing the total displacement of all targets that has been shot up by weapons

The artillery target assignment problem is in the family of assignment problems. More specifically, it belongs to the class of weapon target assignment (WTA) problem. By means of our three element definition, the problem can be referred as a three index assignment problem, or more generally as a multi index assignment problem (MUIP). Some variations of the problem, which are actually sub-problems of the main problem, are also related to other types of assignment problems. An example is the variation of limited number of total shots which is related to generalized assignment problem.

In our definition of artillery target assignment problem, each single artillery weapon in the same battery hierarchy is assumed to be the same type. In other words they are equivalent by means of each property like coverage area. Also each battery has the same hierarchy structure resulting in same number of sections and single weapons. Targets can be different from each other resulting in varying target values, shot values and target sizes.

4.2 Base Case of Artillery Target Assignment Problem

We should better start with the base case of artillery target assignment problem before going further with other parameters considered in the decision process. In this base case, each weapon can shoot at each time instance and each target is shot once for destruction. If we construct the analogy to the general assignment problem, each weapon, each target and each time instance is represented by a node. Thus we have three distinct sets of nodes.

Between these three distinct sets, we have two kinds of edges; namely weapon-target edges and target-time edges. Each weapon-target edge EW_{ij} connects a weapon to a target and defines a single shooting from this weapon to the target. A weapon can have more than one weapon-target edge whereas a target can have only one weapon-target edge. Each target-time edge EZ_{ik} connects a target to a time instance and defines when the shooting of the related target will take place. Similarly, a time instance can have more than one target-time edge whereas a target can have only one target-time edge. Actually each weapon-target edge has a related target-time edge and together they define a single shooting. This model is depicted in the below figure (Figure 4.1).

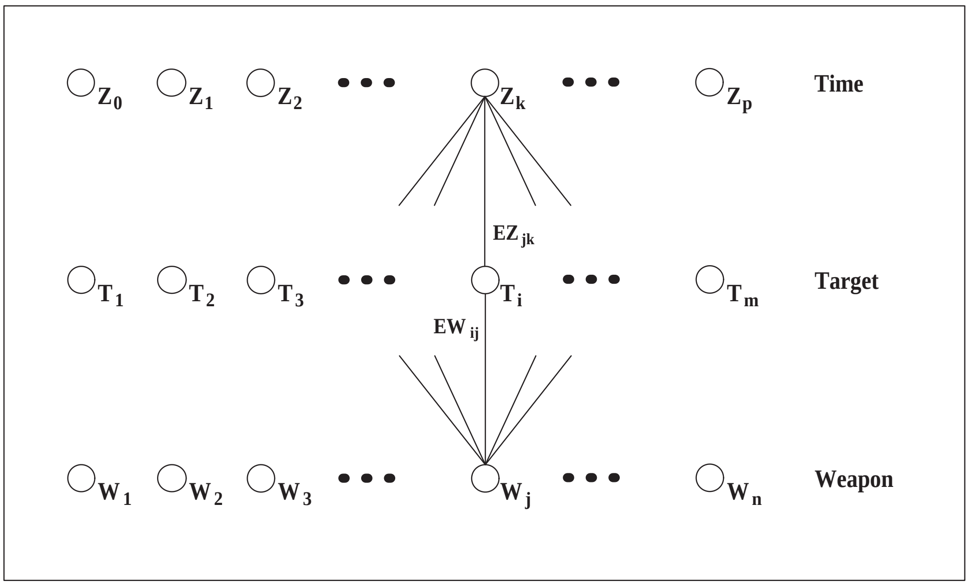


Figure 4.1: Base case of artillery target assignment problem

This model could also be represented by a hypergraph. Then each edge in the hypergraph will be connecting three nodes, one from weapons, one from targets and one from time. Moreover, the model could also be represented by a three dimensional matrix where each dimension represents an element set and each non-zero entry represents a shot. In both ways, it could be easier to express the shots and the whole picture could be more understandable. However, these are neither practical nor advantageous than the current graph representation for defining the variations. In the variations, the current representation will help us to construct the analogy to the general assignment problem and its bipartite graph representation. In other words, the current graph representation is used conceptually to define variations by means of assignment problem.

The cost function for this base case of the artillery target assignment problem is selected to be maximizing the total value of shots at a limited time period. Value of each shot is determined by value of related target node and value of related weapon-target edge.

From a general point of view, the base case of the artillery target assignment problem seems like a three index (three dimensional) assignment problem where weapons, targets and time instances are mapped with each other by edges. However by using a different modeling the above base case of the problem can be simplified to a two index (two dimensional) assignment problem which can be solved in polynomial time. The main idea behind this modeling is that the graph is bipartite in general two index assignment problem; in other words there is a one to one mapping between two sets of elements. The above defined schema for the base case of the problem violates this structure since we have three sets of elements and the mapping is not one to one.

From the definition of the base case, we know that each weapon is able to shoot at each time instance. Therefore, in this new model we can consider each weapon at each time instance separately. Therefore for a weapon W_j in the base case of the problem, in the new model we will have *(p+1)* different instances; W_jZ_0, W_jZ_1, W_jZ_2, $\ldots$, and W_jZ_p. So instead of *m* weapons, we will have *m*(p+1)* weapons. Moreover we will discard the time set, as its effect is directly reflected on the weapon set. Now we have two sets satisfying the above condition, however for simplification to two index assignment problem, we must also show that the mapping between these two sets is one to one. In order to show this, remember that for the base case of the problem, at each time instance each weapon can shoot only once and only to a single target. Therefore in this new model, there can be only one edge E_{ijk}, for each target and for each weapon at a time instance. This guarantees that the mapping between the two sets is one to one.

As a result, this base case of the artillery target assignment problem can be solved with the general two index assignment algorithm of Kuhn-Munkres in polynomial time [10] [16]. This new model is depicted in the below figure (Figure 4.2).

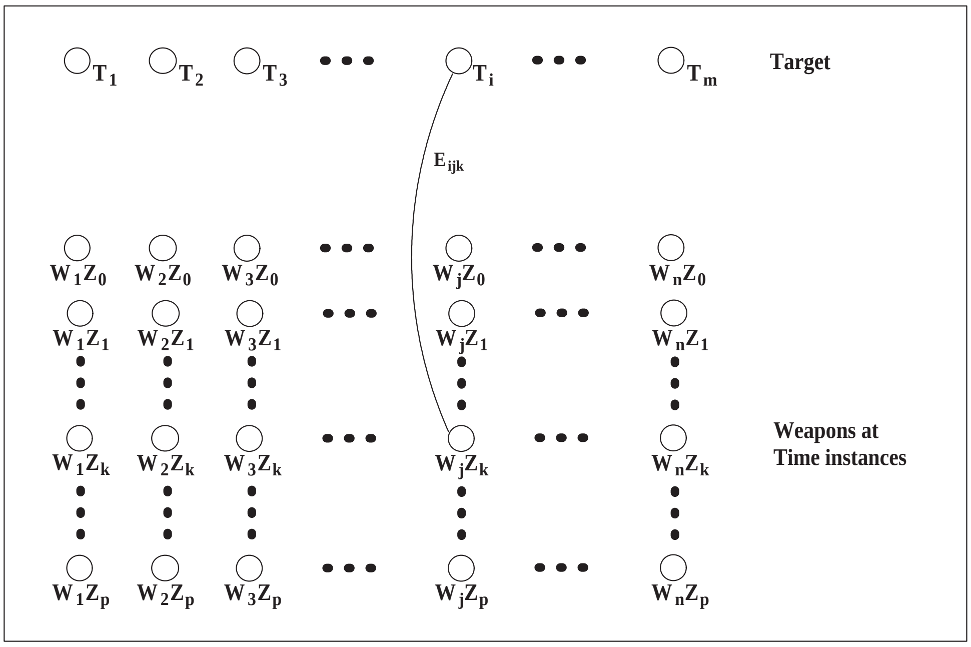


Figure 4.2: New model for base case of artillery target assignment problem

The polynomial time Kuhn-Munkres algorithm can directly be used when two sets to be matched have the same number of elements. However, in this base case, sets may have different number of elements. Therefore, in order to apply the Kuhn-Munkres algorithm, we need to use the method described in [3].

4.3 Main Variations

In this section, we will define main variations of the artillery target assignment problem. Each variation is constructed by adding a few intrinsic properties of the general problem on to the top of the previously defined base case. This way each variation is more understandable, less complex and each variation allows pure concentration on its original property from the main problem.

Being created on to the top of base case, each variation is independent of other variations. In other words, if a variation X has hierarchy of weapons as an intrinsic property, unless explicitly stated, the other variations does not have such a property and they are assumed to have weapons without hierarchy.

As defined in the base case, each weapon is equivalent in all variations where targets may differ.

At each variation, descriptive tags which shortly list the intrinsic properties of the related variation are given at the end of related section.

4.3.1 Variation of Hierarchy in Weapons

In real life, each battery is hierarchically composed of sections and sections are composed of single artillery weapons. According to the target and current conditions, at each time instance the battery can be used as a whole or its subparts can be used individually for shooting. This decision is usually related to the covering issue; if a target can be covered with a smaller artillery weapon unit, then it is usually preferable to use this smaller unit.

As an example, assume we have a battery B which is composed of section S_1 which includes single artillery weapons W_1, W_2, W_3 and section S_2 which includes single artillery weapons W_4, W_5, W_6. If there is a small target which can be covered by just using one section of this battery, then that target is shot with one section, assume S_1 and the other section of the battery, S_2, is used for other shootings. Moreover, it is not a must to use S_2 as a whole; its subparts can be used for different targets individually. One different example for such situation is that W_4 and W_5 can shot one target where W_6 is shooting another target. However, we will not consider the situation depicted in this latter example, because using single artillery weapons together creates unnecessary complexity for the solution.

In the base case of the artillery target assignment problem, each artillery weapon is equivalent in size. In other words, each weapon is a battery or each weapon is a single artillery. However, when the above real life situation is considered, our weapon set becomes heterogeneous by means of weapon sizes. Also the heterogeneity of the set may change among time since at each time instance a battery or its subparts can be used for shooting according to the decision for that time instance.

Thus in this variation, with respect to the base case of the artillery target assignment problem we have a heterogeneous weapon set which changes dynamically among time.

In more formal words, each weapon W_j in weapon set can be divided into sections W_{j-1},

W_{j-2},... W_{j-x} for 2≤ x ≤(number of single weapons at W_j). Each section can be divided into single weapons W_{j-x-1}, W_{j-x-2},...,W_{j-x-y} for 1≤y≤(number of single weapons at W_{j-x}). From the real life, we know that the hierarchy is at most two degrees, in other words there is no other levels of division for a weapon. Also we know that x is usually 2 and y is usually between 2 and 4.

Descriptive Tags: hierarchy of weapons

4.3.2 Variation of Volley Fire with One Weapon

In real life, for shooting a target, a volley fire can be planned in order to cover the target properly. The volley fire consists of multiple shots which can be fired from different artillery weapons. In the basic version of volley fire, only one artillery weapon fires multiple shots which are exactly consecutive in time in order to cover a single target. The number of shots is calculated according to size of target's area and size of the related weapon's coverage area.

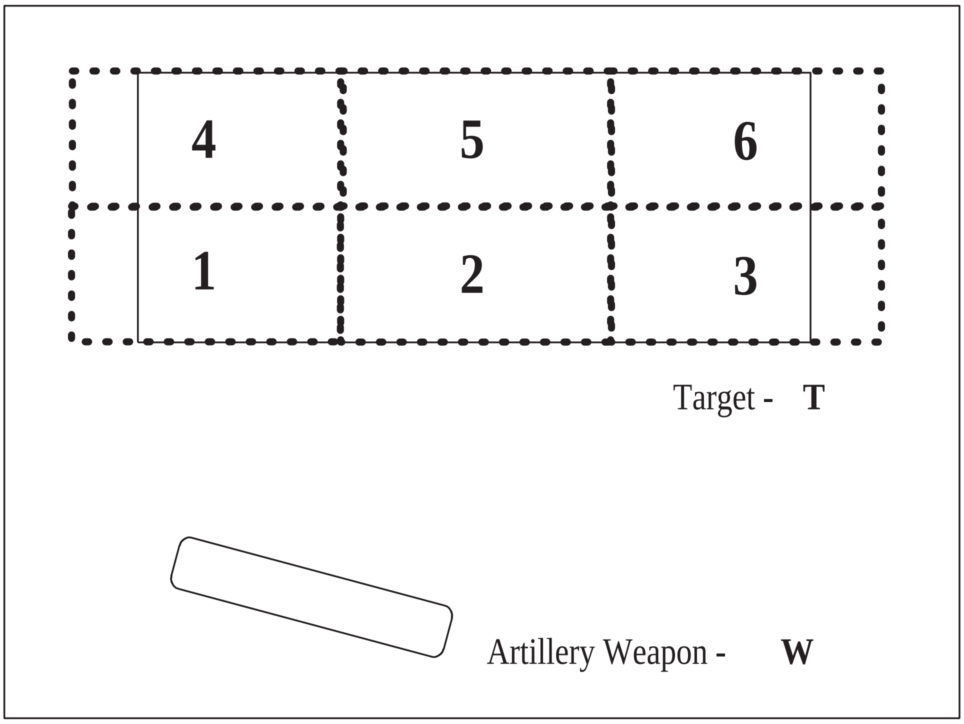


Figure 4.3: Volley fire with one weapon

As an example, assume we have a target T with size *x*y* and an artillery weapon W. W has a coverage area of *(2x/5) * (y/2)* for each single shot. Then in order to cover the target, W has to shot 6 times, calculated by multiplying 3 in width and 2 in height. The situation is depicted in the above figure (Figure 4.3); each shot is represented by a rectangle with dashed lines and numbered from 1 to 6. For the volley fire, order of shots is not critical and can be changed for tactical reasons as long as they are kept to be consecutive. However it is usually preferred to start from bottom leftmost shot, proceed with shots to the left and continue row by row.

In the base case of artillery target assignment problem, each target is shot only once by only one artillery weapon. When the above depicted real life situation, the basic version of volley fire, is considered, a target may be shot multiple times by only one artillery weapon. These shots are fired consecutively in time. The number of shots, F, is only related to the target and the weapon. F does not change among time since targets and weapons are fixed in this variation.

Thus in this variation, with respect to the base case of the artillery target assignment problem we have multiple target-time edges because of multiple shots. These edges are connected to one target node and one or more consecutive time nodes since multiple shots are consecutive.

Descriptive Tags: volley fire, multiple shots, number of shots known, consecutive, one weapon

4.3.3 Variation of Target Joining

In real life, close targets of similar types of elements are grouped together to create larger targets. Usually targets move around in the battlefield and thus change position which results in varying distances between targets. When this distance is close enough, the related targets are considered to be a single target and shootings are planned accordingly. This usually decreases amount of targets and increases total effectiveness of artillery weapons.

As an example, assume we have two targets which are both composed of tanks and they both move towards friendly forces of ours through the same bridge. Also assume that they will arrive at the bridge together and pass through the bridge one by one. For such conditions, tactically it is a good choice to shoot both targets together as one target when they arrive at the bridge at close positions and just before starting to pass through the bridge. Also with this choice, usually probability of creating damage on targets increases.

When targets are grouped in this way, obviously size of the new target will be different than size of individual targets. Therefore, in real life the new target may require a volley fire and may even require shooting of multiple weapons. In order to keep it less complex, for this variation we will assume that volley fire will not be used. Also value of the new target will probably be different than the individual targets and thus priority will also be different.

In the base case of artillery target assignment problem, we have a constant number of targets with constant properties like size and value of target. However, when the above real life situation is considered, number of targets and targets themselves changes among time. At some time instances, targets are grouped together whereas at some time instances they are treated separately. Also it is possible that at a time instance target T_1 will be grouped with target T_2 and then at another time instance T_1 and T_2 will not be grouped, but T_1 and a target T_3 will be grouped.

Thus in this variation, with respect to the base case of the artillery target assignment problem we have a target set which changes dynamically among time and we have weapon-target edges whose values also change among time.

Descriptive Tags: joining targets

4.3.4 Variation of Unavailability of Weapons

In real life, there is some delay in time between each consecutive shot of an artillery weapon. In other words, the related weapon can not shoot during this time period and it is considered to be unavailable at that time period. Some possible reasons for this unavailability are change of emplacement or preparation time for a next shot.

As an example, assume we have an artillery weapon W and average preparation time for W is t_p. If W shoots at time t_0 then it will be unavailable during t_p time instances. At time t_0+t_p, weapon W will be available to shoot again. However it is probable that at a time t_x between t_0 and t_0+t_p, W might be needed for a more critical shot than the shot at t_0. So it might be better to wait until t_x without shooting and shoot at t_x. In general if there is such a delay for artillery weapons, it is critical to decide when to shoot and when to wait.

In the base case of artillery target assignment problem, each artillery weapon can shoot at each time instance. However, when the above real life situation is considered, weapons become unavailable to shoot at some time instances and this is determined by the previous decisions. Deciding on a shot with a weapon also includes deciding not to shoot with this weapon in some successive time instances. Moreover, such a decision also forces that at some preceding time instances the weapon has not been used which would cause unavailability at that time instance.

Thus, this variation is very different than the base case as it incorporates a strong constraint of unavailability which forces that weapons can shoot only at proper time instances by considering the possible delays after shots. The effect of the situation on the model of the problem can be interpreted in two ways. In the first interpretation, we will interpret the difference with respect to the basic modeling of the base case of the artillery target assignment problem. Then in this variation, our weapons do not have a target-time edge for each time instance. So we have a difference in edges, we have less edges which are dynamic according to this variation. In the second interpretation, we will interpret the difference with respect to the two index modeling of the base case of assignment problem where we have a specific weapon node for each time instance. Then in this variation, our weapon set changes dynamically among time. Instead of absence of nodes or edges, both interpretations can also be expressed by means of weapon-target edge values. This can be achieved by assigning a zero value to an edge value whenever the related weapon is unavailable.

For this variation we need average delay times for consecutive shootings of each weapon. Average delay times can differ according to types of single artillery weapons (i.e. cannon, howitzer). However, as we consider each single artillery weapon to be equivalent by means of each property, delay times are also considered to be the same for each weapon. Also initially at time Z0 all weapons are assumed to be ready to shoot.

Descriptive Tags: unavailability of weapons

4.3.5 Variation of Limited Number of Total Shots

In real life, one of the most important constraints about an artillery weapon is that it can not shoot forever in time. In a more clear explanation, an artillery weapon can fire finite number of shots. This can be considered as a bound of maximum number of shots for the related weapon. The highest possible value of the bound is clearly limited to the number of ammunition, since number of shots can never exceed number of ammunition and number of ammunition is finite. The bound can be less than this value because of tactical reasons. An example for such tactical reasons is the need to preserve some of the ammunition for some reasons like self-protection.

This real life situation is easy to exemplify; assume we have an artillery battery in which each single artillery weapon has ten ammunitions. Then this weapon can shot ten targets and when it finishes these ten shots, the weapon becomes unavailable for all the remaining time instances.

In the base case of artillery target assignment problem, each artillery weapon can shoot once at each time instance which means that the total number of shots possible for a weapon is equal to the total number of discrete time instances. However, when the above real life situation is considered, after a certain finite number of shots (assume s shots and the s'th shot is named as last shot) the weapon will become unavailable to shoot for all the remaining time instances. Therefore it is critical to decide when to fire these s shots, because except these shots the weapon will not be used. To be more precise, a shot fired at a time instance may result in unavailability of the weapon for a better shot at another time instance.

At this variation, time instances which are after the last shot of the weapon can be considered to be similar to the variation of unavailability of weapons. This similarity is related to creating unavailable time instances for weapons, since at those time instances the weapons become unavailable to shoot.

Thus for this variation, the result on the model is very similar to the result at the variation of unavailability of weapons. This time, the constraint forces limited number of weapon-target edges for each weapon at the model. Similar to the variation of unavailability of weapons, effect of this constraint can be reflected to the model by assigning zero value to the edge values of weapon-target edges except limited number of shots.

Descriptive Tags: limited number of shots for weapons, unavailability of weapons

4.3.6 Variation of Multiple Shots to Targets

In real life, it is a preferable and sometimes a desirable artillery tactic to shoot up a target with more than one shot in order to create a desired amount of damage. The number of shots which is required to create the related damage can be roughly forecasted to be a fixed number, beforehand. With this fixed number, multiple shots from one or more weapons are planned for this target. This seems to be similar to the volley fire, but there is a critical difference about timing of shots. In volley fire, multiple shots are fired consecutively, however at this one they can be fired at any proper time.

As an example, assume we have a target which is very strong by means of its armor and we know that our artillery weapons will be unavailable to destroy the target with one shot, although they can already cover the target with one shot. Because of such a reason, we can prefer the above tactic and plan a fixed number of shots, as an example three, to the related target. These three shots can be fired from a single weapon or various weapons. More importantly these shots can be fired at any time instance; first shot at Z_i, second shot at Z_{i+k} and third shot at Z_{i+k+m} where k does not have to be equal to m.

In the base case of artillery target assignment problem, we know that each target is shot once by only one artillery weapon. However when the above tactic is used, the target can be shot with multiple shots from any number of weapons and at any time. If the number of shots is not known beforehand (i.e. it had to be determined according to the results of previous shots), then multiple shots couldn't be planned but they had to be fired real-time. Therefore the key point which allows planning is that the number of multiple shots is determined beforehand.

Thus in this variation, with respect to the base case of the artillery target assignment problem we have multiple target-time edges which is similar to variation of volley fire with one weapon. However at this variation, we do not have the strict constraint which enforces the time nodes of target-time edges of a target to be consecutive.

Because there is no such constraint, actually this variation is easy to solve. We can construct a solution with Kuhn-Munkres algorithm by making use of the two index model which was offered for the base case of artillery target assignment problem. For this purpose, we need an arrangement on the target nodes of the related model. For each target node which needs a multiple shot as given in this variation, we need duplicates of the related target node as much as the number of multiple shots. For example, if we have a target T_i which needs three shots because of tactical reasons, we will change the target set as it will include T_{i1}, T_{i2}, T_{i3} instead of a single T_i. If Kuhn-Munkres algorithm is applied on this model, it will find optimal assignments for this variation.

Descriptive Tags: multiple shots, number of shots known, inconsecutive, any number of weapons

4.3.7 Variation of Volley Fire with Multiple Weapons

In real life, multiple shots of a volley fire can be fired from multiple artillery weapons. In this case some artillery weapons are selected and they fire the individual shots of the volley fire with some synchronization. Selected artillery weapons are preferred to be nearly equivalent in power and coverage area.

As explained before, although order of shots is not critical for volley fire, it is usually preferred to start from bottom leftmost shot, proceed with shots to the left and continue row by row. When multiple weapons are selected for the volley shot, each weapon is assigned a shot with the same order and then they fire together at the same time instances. For example, assume we have the target given in Figure 4.3 and we have selected two proper equivalent artillery weapons, namely W_1 and W_2, in order to shoot this target. At a time Z_i, W_1 shots 1 and W_2 shots 2. Then at time Z_{i+1}, W_1 shots 3 and W_2 shots 4. And lastly at time Z_{i+2}, W_1 shots 5 and W_2 shots 6.

In the base case of artillery target assignment problem, each target is shot only once and just by one artillery weapon. When volley fire with multiple weapons is considered, a target may be shot multiple times and each individual shot can be fired from different weapons. Still the shots of each weapon are consecutive in time.

Thus in this variation, with respect to the base case of artillery target assignment problem we have multiple weapon-target and multiple target-time edges for target nodes. Time nodes of target-time edges are consecutive.

This reflects the required change in the model as a result of this variation. However without any other rules, this new model would be ill-formed by means of edges. As stated earlier, each weapon-target and target-time edge couple expresses a shot. Assume, we need 9 shots for a target and we want to use 2 weapons for these shots. We need 5 time instances to finish shooting and at the last time instance only one weapon will be shooting. For this target, we will have 2 weapon-target and 5 target-time edges. Combination of these means 10 shots which conflicts with the actual 9 shots. However, this is an infrequent situation, because usually for *k*m* shots of volley fire, m weapons are selected where k and m are positive integers. For this variation, we apply this simple rule about number of weapons.

Descriptive Tags: volley fire, multiple shots, number of shots known, consecutive, any number of weapons

4.4 Variation Mixture

Each of the main variations is a part of the main problem and each of them expresses different interesting properties of the main problem. These main variations can be considered as building blocks and logical mixture of some of these building blocks can be used to express sub-problems of the main problem. If this logical mixture includes all sub-problems, then it defines the main problem; artillery target assignment problem(ATAP).

An example of such sub-problems is the following mixture;

- Variation of hierarchy in weapons
- Variation of volley fire with multiple weapons
- Variation of target joining

This given sub-problem is a good example which represents the main skeleton of the original problem.

In general, the artillery target assignment problem is very complex and it is very difficult to solve the problem in polynomial time with any method. In most cases, created mixtures representing sub-problems are even difficult to solve. As a result of these, our approach on this thesis is working on some sub-problems which we have selected as being more interesting than the others.

4.5 Other Possible Variations

The base case and variations of artillery target assignment problem are all defined from the real life military problem by considering the major complexities of the process of artillery assignment and shooting. In real life, there are some other major complexities about the process and it is possible to define other variations from these complexities.

In each of the variation stated above, we observed that they affect the problem from different aspects and mostly they make it harder to solve the new problem. However when these other major complexities are taken into account, the problem does not just become harder to solve, but the nature of the problem changes totally in different aspects. Because of this reason, we considered working on such new problems to be out of the scope of this thesis. Therefore, we will just shortly mention examples of some possible variations resulting in such new problems.

One such possible variation is totally same as the variation of multiple shots to targets except one change; this time number of shots is not known beforehand. In real life, it is really difficult to forecast a fixed number of shots which will create the desired amount of damage. Mostly, damage occurred is observed after each shot and then it is decided to continue with a new shot or not. However if we apply such a manner to our problem, it becomes a real-time problem in which we need actual results of previous problems.

Another such possible variation is about the most major problem in planning the assignment of artillery weapons. In real life, future is not known much, which means that sudden targets may appear or current targets may disappear at any time. This will result in a dynamic set of targets which changes with time and without step by step proceeding in time, it is not possible to know the target set for each time step. Therefore, with this variation again the problem becomes a real-time problem.

The last such possible variation is about targets and their own fire power. Usually many targets in artillery target assignment problem are enemy units. Some of these enemy units have the power to create damage on the artillery weapons. In real life, while artillery weapons are shooting up targets, they also receive damage and sometimes become totally unavailable to shoot. Therefore actually weapon set may become smaller in time and this is determined by results of enemy activities at each time step. Similar to the previous examples, this problem is also a real-time problem.

As seen from the examples, most of the issues that we have considered to be out of the scope of this thesis are related to real-time problems. Solving such real-time problems must also consider the behavior of the enemy units to develop algorithms for different possible cases. Testing possible solutions of such real-time problems requires simulating the whole process since results of previous time steps are required to proceed further in time. This is a rather different area of study than our main concentration in this study. So for all the problems that we deal in this thesis, we kept our scope within strong boundaries which can be simply expressed as ”all the possible states among time are already known at the beginning”.

Besides, there are also studies in the literature dealing with such real-time issues about weapon target assignment problems. One interesting study on real-time weapon target assignment problems presented at [6] considers also the enemy’s actions and proposes a game theoretic approach to the problem. The resulting algorithm is tested in a simulation environment and it is also shown that it is possible to apply the given approach to real-time systems.

CHAPTER 5

SOLUTION APPROACH

5.1 General Solution Approach

The artillery target assignment problem is a complex optimization problem and at most cases it is difficult to find the best solution in polynomial time. Variations of the main problem defined in the Chapter 4 are less complex than the main problem, but it is even difficult to solve most of these variations and their possible mixtures in polynomial time.

Because of this difficulty in finding the optimal solution, our general solution approach for the artillery target assignment problem is to employ customized genetic algorithm methods for this specific problem. However, as stated earlier it is difficult to solve the artillery target assignment problem as a whole, even by using genetic algorithm methods. Because of this reason, in this thesis we worked on two of the main variations of the problem; variation of hierarchy in weapons and variation of target joining. These variations are selected because of their challenging nature about being dynamic in weapon and target domains respectively. For each of these two variations, we developed solution approaches based on customized genetic algorithms. In this chapter, details of these solution approaches are presented.

In general, genetic algorithm approaches are capable of finding optimal or sub-optimal solutions for optimization problems. While developing a genetic algorithm solution, selecting a proper representation, using proper genetic operators and providing enough randomness in the operators are some of the critical issues to take under consideration. For example, if on some selection points deterministic selections are to be made instead of random selections, it is more possible that the solution will got stuck at a local maximum or minimum and will not reach to the desired optimal or suboptimal solutions.

5.2 Proposed Solution for Variation of Target Joining

We define a feasible solution for variation of target joining as a solution in which all targets are shot up, each weapon shots only once at one time instance and at some time instances some joinable target pairs are shot up by one weapon per pair.

Our genetic algorithm approach is based on the idea that each genome at each population must be representing a feasible solution at each generation. Therefore, this feasible solution definition is used at each step of our proposed solution. The proposed representation is adequate for this feasible solution definition and it is easily possible to detect any infeasibilities with this representation. Our crossover strategies may create infeasible solutions. However we prefer to repair such infeasible solutions to create feasible solutions.

Constraint handling is an important concept in genetic algorithms. From penalizing methods to repair algorithms, there are many different constraint handling strategies in the literature. Studies presented in [4] and [15] gives detailed surveys on constraint handling. For our solutions, we preferred repair strategy which is discussed in detail in [1]. In the literature, there is an ongoing discussion on whether repair strategies give better results or not. [17] is an example of studies which says repair stategies give better results.

5.2.1 Representation

In order to represent a solution for the artillery target assignment problem, in general we decided to use a matrix representation. As the problem is about assigning three groups of elements; namely targets, weapons, time, we represent a solution by a three dimensional matrix. Each dimension of the matrix is used for one of these groups. In other words each column represents an artillery weapon, each row represents a target and to the depth of the genome each plane represents a time instance. As a result, each entry in this matrix shows that the related elements are assigned. For this purpose, the matrix has a binary representation where an entry of 1 means a shot from the weapon at the entry's column to the target at the entry's row at the time instance of the entry. In general, this representation is flexible enough for adjustments to reflect the intrinsic properties of the artillery target assignment problem.

For our solution approach of variation of target joining, we use this representation which results in a matrix of dimensions m *x* n *x* (p+1). In order to represent the joined targets, we take advantage of the flexibility of this repsentation. For joined targets it will suffice to use multiple entries at a column. In other words, if a column has more than one entry, this means that the weapon at that column will shoot up more than one target which are joined at that time step. It is important to point out that for a feasible solution of variation of target joining, the representation can have multiple entries at some columns but can not have any multiple entries at any rows.

A 2D cross-section of this representation which a shows the weapon target assignments at a single time instance is depicted in the figure (Figure 5.1). In the figure an entry of 1 is painted as a grey circle and an entry of 0 is left as empty.

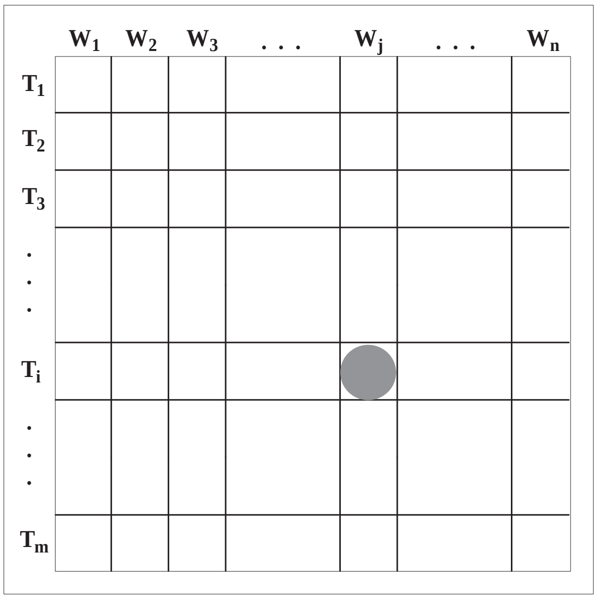


Figure 5.1: Representation of a solution for a single time step

5.2.2 Objective Function

In this solution, we are trying to maximize an objective function. This objective function calculates the total costs of the assigned shots. In equation 5.1, this objective function is expressed in mathematical terms.

$Matrix(i, j, k)$ states the value of the related entry at the current genome which is 1 for a shot and 0 for no shots as defined by the representation. Shooting cost is the cost function of a shot from weapon j to target i at time instance k and value of this function represents the importance of this shot. As stated earlier, many different cost functions can be defined according to the related military tactics, however the objective function does not care about which cost function is used and how its value is calculated. The objective function just tries the maximize the total value of shooting costs for the current shots of a solution.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=0}^{p} Matrix(i,j,k) \times ShootingCost(i,j,k) \tag{5.1}$$

During our experiments, we used a sample cost function which is defined in equation 6.1.

5.2.3 Initialization

Algorithm 1 Initialization of a Single Genome VTJ

```
1: Initialize all bits of the 3D genome to 0
2: Initialize array $W$ as available for all weapons at all time instances
3: Initialize array $T$ as unshot for all targets
4: for all Possible joins do
5:     Randomly decide whether to add this join to this solution or not
6:     if Any of the two targets of this join is already shot up then
7:         Do not add this join to this solution
8:     end if
9:     Using $W$, among all available weapons at join's time instance randomly select a
       weapon
10:    Add the shot for this join to the solution while updating $W$ and $T$ properly
11: end for
12: for all Targets do
13:    if Target is already shot up, checked from $T$ then
14:        Continue with the next target
15:    end if
16:    Using $W$, among all available weapons at all times, randomly select a weapon and time
17:    Add the shot for this target to the solution while updating $W$ and $T$ properly
18: end for
```

Each genome is initialized with the method given in algorithm 1. Firstly, for each genome a solution with no shots is created. Then randomly some joined shots are added to this genome. After adding these joined shots, we need to assign shots for unshot targets since a feasible solution can have no unshot targets. Therefore as a last step, we assign a shot with an available random weapon and time for all unshot targets.

5.2.4 Crossover Operators and Repair Strategies

We planned to use two different crossover operators; one of them is single point crossover on columns and the other one is single point crossover on rows. Nevertheless, both of these operators can produce infeasible solutions. In order to solve this issue, we preferred repair strategy for such individuals and therefore just after applying crossover operators, we repair infeasible individuals.

Although in the literature this kind of crossovers are referred as single point crossovers, we refer our crossover operators single plane crossovers. The reason is that, we apply crossover on four pieces of our genomes created by separating each of the parent genomes into two. As our representation is a 3D matrix, this separation is done by a plane.

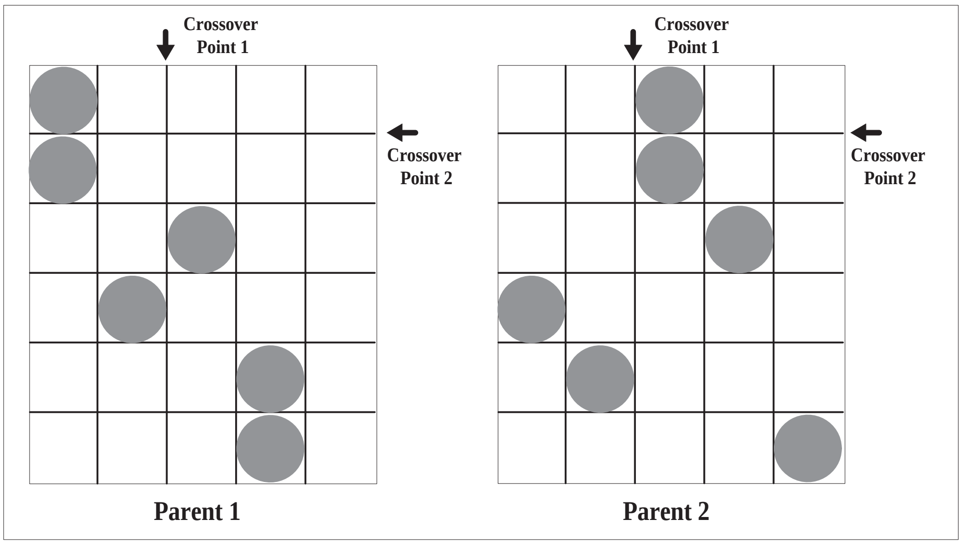


Figure 5.2: Sample Parent Individuals for Variation of Target Joining

Assume we have two sample individuals shown in figure 5.2 and we will apply both types of crossover operators on these individuals.

If we apply single plane crossover operator on columns by using crossover point 1 shown in figure 5.2, two offsprings shown in figure 5.3 will be created. In this figure columns painted with grey comes from parent 2. Both of these offsprings do not represent feasible solutuions since they have multiple entries at rows. So we will use our repair strategy to create feasible solutions from these unfeasible solutions.

For Offspring 1, the target T_5 has two entries which means that the target is shot up by two weapons; W_2 and W_4. Therefore, we need to cancel one of the entries and with a random selection we chose W_2. Actually now Offspring 1 is nearly a feasible solution, but two weapons are not used and two targets are not shot up. As a last step of our repair mechanism we assign these weapons to targets randomly. In figure 5.3, these assignments are shown with unfilled grey circles.

For Offspring 2, the targets T_1 and T_2 has two entries and we know that these targets can be joined at this time step. Therefore we will treat them as a single target and cancel both shots from W_1 or W_3. We randomly select W_1 and cancel related shots. Now Offspring 2 is nearly a feasible solution, however again we have unassigned weapons and targets. Similarly as a last step of our repair mechanism, we assign them randomly as shown with unfilled grey circle in figure 5.3.

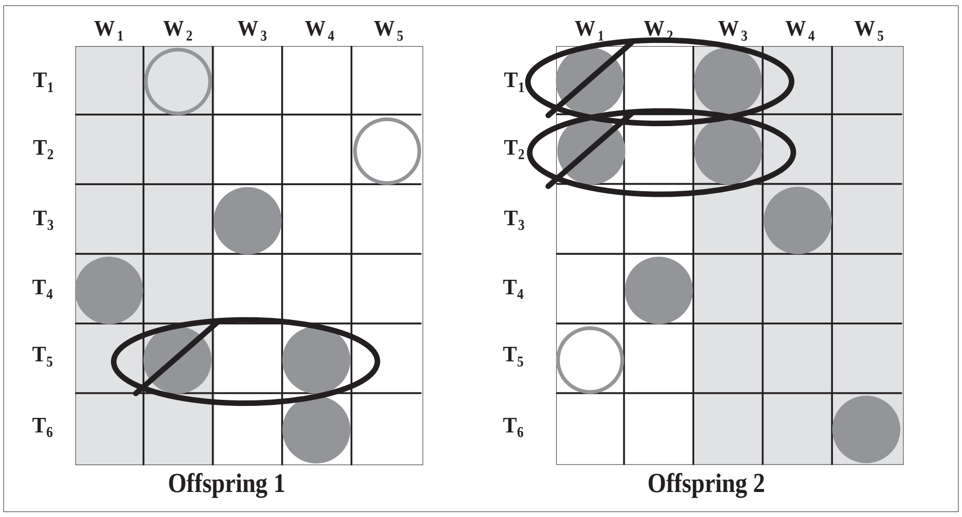


Figure 5.3: Offsprings created after a column crossover and related repairs

If we apply single plane crossover operator on rows by using crossover point 2 shown in figure 5.2, two offsprings shown in figure 5.4 will be created. Similarly in this figure rows painted with grey comes from parent 2. These offsprings seems to represent feasible solutions since they do not have multiple entries at rows. However this crossover creates new joined targets and we need to check if these targets can really be joined at that time step. In this example, for both offsprings we have one new joined targets which are circumscribed in figure 5.4. For offspring 3, we know that at that time step targets T_1 and T_3 can be joined so we do not need to repair offspring 3. However for offspring 4, we know that targets T_1 and T_4 can not be joined at that time step. As a repair strategy, we randomly cancel one of the two shots of W_1. At this example shot for T_1 is cancelled. Now offspring 4 is a close to a feasible solution with one unassigned target. Since all weapons are assigned to targets, we can only shot this target by joining with another target. Among targets that can be joined with T_1, we randomly select T_3 and as W_4 shots T_3 we assign T_1 to W_4 as shown with unfilled grey circle in figure 5.4.

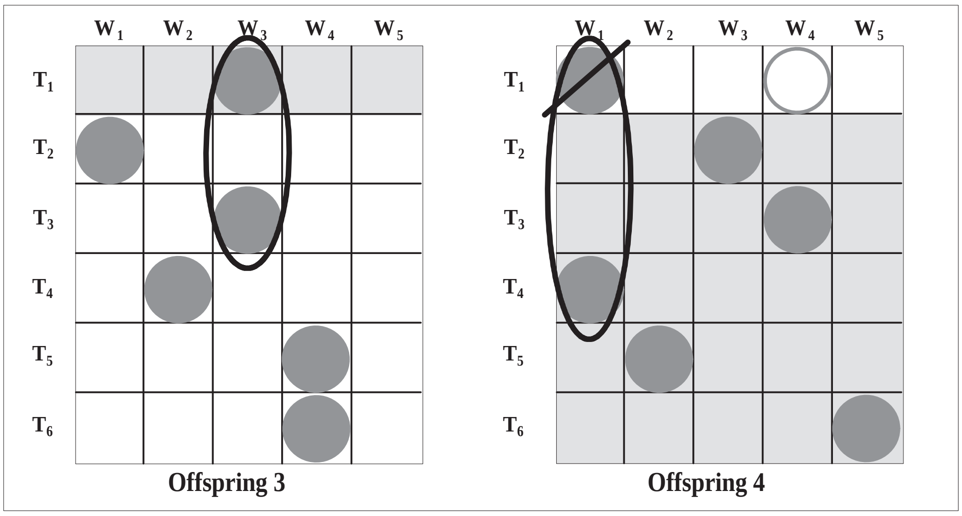


Figure 5.4: Offsprings created after a row crossover and related repairs

Overally our crossover strategy works by the method given in algorithm 2. Chances of randomly selecting row crossover or column crossover are equal.

Algorithm 2 Single Plane Crossover of Two Parents VTJ

```
1: Parent1 $P1$, Parent2 $P2$, Offspring1 $O1$, Offspring2 $O2$
2: Randomly select Row Crossover or Column Crossover
3: **if** Row Crossover **then**
4:     Select a random point $N$ among rows (targets)
5:     Copy rows 0 to $N - 1$ of $P1$ to $O1$
6:     Copy rows $N$ to *end* of $P2$ to $O1$
7:     Copy rows 0 to $N - 1$ of $P2$ to $O2$
8:     Copy rows $N$ to *end* of $P1$ to $O2$
9:     Call *Repair Row Crossover VTJ* for offspring $O1$
10:    Call *Repair Row Crossover VTJ* for offspring $O2$
11: **else if** Column Crossover **then**
12:    Select a random point $M$ among columns (weapons)
13:    Copy columns 0 to $M - 1$ of $P1$ to $O1$
14:    Copy columns $M$ to *end* of $P2$ to $O1$
15:    Copy columns 0 to $M - 1$ of $P2$ to $O2$
16:    Copy columns $M$ to *end* of $P1$ to $O2$
17:    Call *Repair Column Crossover VTJ* for offspring $O1$
18:    Call *Repair Column Crossover VTJ* for offspring $O2$
19: **end if**
```

The repair strategies for row and column crossover operators used in algorithm 2, are explained in detail in algorithm 3 and algorithm 4.

When row crossover is applied, multiple entries at columns may be created. Since in this variation there can be up to two entries at a single column in a feasible solution, after the crossover there can be at most four entries; two entries from each parent. The repair method in algorithm 3 considers this situation. Overally in this method, number of shots for each column, in other words for each weapon at each time instance, is counted and according to the result some shots are cancelled if required.

Algorithm 3 Repair Row Crossover VTJ

```
1:  for all Weapons do
2:      for all TimeInstances do
3:          Initialize number of shots $N_S$ as 0
4:          Initialize targets that are shot up at this column $T_S$ as an empty set
5:          for all Targets do
6:              if There is a shot for this target then
7:                  Add this target to $T_S$
8:                  Increase $N_S$ by one
9:              end if
10:         end for
11:         if $N_S < 2$ then
12:             No infeasibility at this column, so continue with the next column
13:         else if $N_S < 4$ then
14:             Initialize set of possible shots $P_S$ as an empty set
15:             Add individual(unjoined) shots for each target in $T_S$ to $P_S$
16:             for all Possible pairs of targets in $T_S$ do
17:                 if This pair of targets can be joined at this time step then
18:                     Add shot for this pair of targets to $P_S$
19:                 end if
20:                 Randomly select a shot $S$ from $P_S$
21:                 Only keep shot(s) of selected shot $S$ and cancel other shots at this column
22:             end for
23:         end if
24:     end for
25: end for
26: Call *Add Required Shots VTJ* for this genome
```

When column crossover is applied, multiple entries at rows may be created. In this variation there can be only one entry at a single row in a feasible solution. Therefore after the crossover there can be at most two entries at a row; one from each parent. The repair method in algorithm 4 considers this situation. Overally in this method, number of shots for each row, in other words for each target, is counted and if there are two shots, one of the shots is cancelled.

Algorithm 4 Repair Column Crossover VTJ

```
1: for all Targets do
2:   Initialize number of shots $N_S$ as 0
3:   Initialize $S$ as an empty array
4:   for all Weapons do
5:     for all Time Instances do
6:       if There is a shot for this target then
7:         Store weapon and time instance of this shot at $S[N_S]$
8:         if There is a joined target for this shot then
9:           Store joined target of this shot at $S[N_S]$
10:        end if
11:        Increase $N_S$ by one
12:      end if
13:    end for
14:  end for
15:  if $N_S < 2$ then
16:    No infeasibility for this target, so continue with the next target
17:  else if $N_S == 2$ then
18:    Randomly select one of the shots $S[0]$ or $S[1]$
19:    Cancel the selected shot while keeping the other shot
20:    if There is a joined target for selected shot then
21:      Also cancel shot for the joined target
22:    end if
23:  end if
24: end for
25: Call *Add Required Shots VTJ* for this genome
```

Both repair strategies use the same method, add required shots. The reason is that there may be some unshot targets which are created as a result of the crossover or as a result of the repairs. In order to make sure that the solution is feasible, we need to add shots for such unshot targets. The method used for this purpose is given in detail in algorithm 5.

Algorithm 5 Add Required Shots VTJ

```
1: Initialize array $W$ as available for all weapons at all time instances
2: Initialize array $T$ as unshot for all targets
3: Traverse all the genome in order to fill $W$ and $T$ properly
4: if All targets are already shot up then
5:     End Add Required Shots VTJ
6: end if
7: for all targets do
8:     if Target is already shot up then
9:         Continue with the next target
10:    end if
11:    Randomly decide on whether to try assigning a joined shot for this target or not
12:    if To try assigning a joined shot then
13:        Call Trying to Add a Joined Shot for this target
14:    end if
15:    if Target is shot up with a joined shot then
16:        Continue with the next target
17:    end if
18:    Using $W$, among all available weapons at all times, randomly select a weapon and time
19:    Add the shot for this target to the solution while updating $W$ and $T$ properly
20:    if All targets are shot up now then
21:        End Add Required Shots VTJ
22:    end if
23: end for
```

The method given in algorithm 5 uses the method given in algorithm 6 in order to try adding some joined shots. However trying to add a joined shot is a preferance which is randomly decided. In our solution, chances of randomly not trying to add a joined shot is greater than trying to add a joined shot.

Algorithm 6 Trying to Add a Joined Shot

```
1: Initialize array $J$ as unavailable for all possible joins
2: Initialize number of available joins $N_J$ as 0
3: for all Possible joins do
4:     if This join does not include this target then
5:         Continue with next join
6:     end if
7:     if The other target in this join is unshot then
8:         if There exists at least one available weapon at join's time, checked from $W$ then
9:             Mark this join as available in $J$ with two unshot targets and increase $N_J$ by one
10:        end if
11:    else if The other target in the join is shot then
12:        if The time of the other target's shot is the same as join's time then
13:            Mark this join as available in $J$ with one shot target and increase $N_J$ by one
14:        end if
15:    end if
16: end for
17: if $N_J > 0$ then
18:    Randomly select one of the available joins from $J$
19:    if Selected join is a join with two unshot targets then
20:        Randomly select an available weapon at this join's time using $W$
21:        Add shots for both targets at join's time with selected weapon
22:        Update $W$ and $T$ properly
23:    else if Selected join is a join with one shot target then
24:        Add a shot for the unshot target with weapon and time of other target's shot
25:        Update $T$ properly
26:    end if
27: end if
```

5.2.5 Mutation

Mutation strategy in this solution randomly cancels a shot for a target. If the target has a joined shot, then the joined shot is also cancelled. After these operations, in order to restore the solution to be a feasible solution, we apply the method given in algorithm 5.

The details of our mutation approach are given in algorithm 7.

Algorithm 7 Mutation VTJ

```
1:  Considering the mutation probability randomly decide whether to mutate or not
2:  if Not to mutate then
3:      End Mutation VTJ
4:  end if
5:  Randomly select a target $T$ to mutate among all targets
6:  Find weapon and time of the shot that is already assigned to $T$
7:  Cancel the shot for $T$
8:  if There exists a joined shot for $T$ then
9:      Cancel the shot for the related joined target
10: end if
11: Call Add Required Shots VTJ for this genome
```

5.3 Proposed Solution for Variation of Hierarchy in Weapons

We define a feasible solution for variation of hierarchy in weapons as a solution in which all targets are shot up, a single weapon shots only once at one time instance and rules related to hierarchy of weapons holds for all shots with all related weapons.

There are two main rules related to hierarchy of weapons. The first rule is that no smaller weapon (eg. a single weapon) shoots while one of its larger weapons (eg. a battery) in its hierarchy is already shooting. Similarly the second rule is that no larger weapon shoots while one of its smaller weapons in its hierarchy is already shooting.

Our genetic algorithm approach follows a similar approach as our approach for variation of target joining. Similarly our strategy is based on the idea that each genome at each population must be representing a feasible solution at each generation. For each method of our approach we try to preserve this property among all genomes.

Besides similarities, because of the problem definitions there are also important differences. One of the most important differences is that although joinable targets change among time, hierarchies of weapons do not change. This has been an advantageous property while creating our solution.

5.3.1 Representation

Similar to our representation for variation of target joining, we could use multiple entries at a row to represent the weapon hierarchies where each hierarchy is represented by only its single weapons in the weapon dimension of the solution. In other words, if a row has more than one entry, this means that the target at that row will be shot up by more than one single weapon which are in the same hierarchy and which create a larger weapon structure like a section or a battery.

However, for randomness in crossover and mutation operators, this would not be a good representation. Because, it is a low probability that shots for some single weapons will combine from two parents and create a shot for a section or battery. Instead, in most cases shots for sections and batteries will be seperated which would higly favour single weapon shots. As stated earlier, such problems in randomness will result in solutions that got stuck in local maximums or minimums.

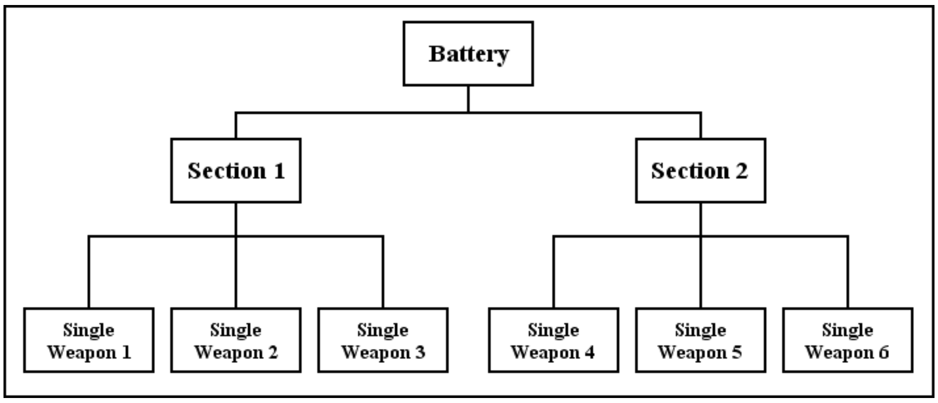


Figure 5.5: The fixed hierarchy which is used in this variation

Because of this situation, we decided to use another 3D matrix representation in which there can be at most one entry at each column and at each row. The target and time instance dimensions of this representation are the same for this representation, however in the weapon dimension we have a modification which will ease our genetic operations. In the weapon dimension, instead of representing all weapons by only their batteries, we represent each weapon with 9 weapons; 6 single weapons, 2 section weapons and 1 battery weapon. This modification is possible since we have a fixed hierarchy that does not change among time. This fixed hierarchy is depicted in figure 5.5.

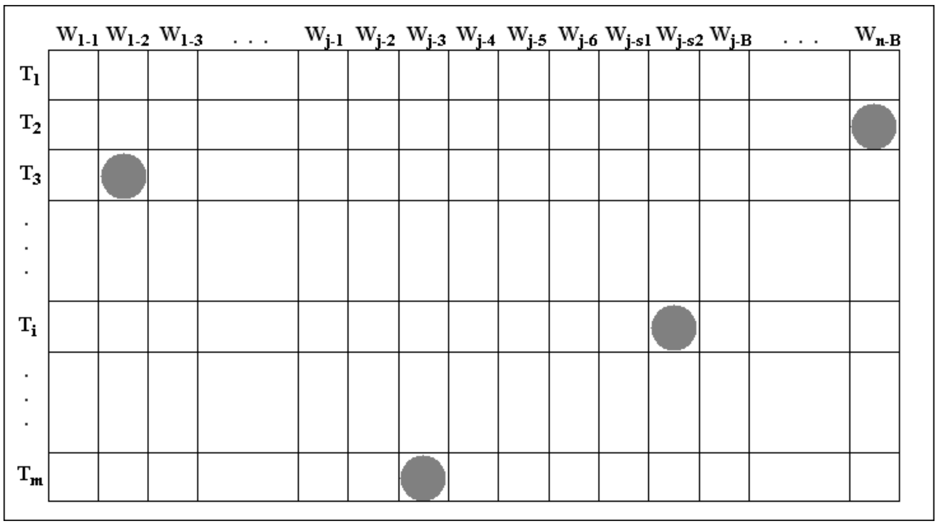


Figure 5.6: Representation of a solution for a single time step

A 2D cross-section of this representation which a shows the weapon target assignments at a single time instance is depicted in the figure (Figure 5.6). In the figure an entry of 1 is painted as a grey circle and an entry of 0 is left as empty. Weapons from W_{j-1} to W_{j-6} represents single weapons, W_{j-s1} and W_{j-s2} represents sections and W_{j-B} represents the battery weapon for weapon W_j.

5.3.2 Objective Function

This solution also tries to maximize the same objective function which was defined for variation of target joining in equation 5.1. However this time, we have different shooting costs for each size of artillery weapons.

During our experiments for this variation, we used the sample cost function which is defined in equation 6.1. This cost function considers the size of the weapon. In general words, if the size of the weapon is proper for the target, the function gives the actual cost. If it is smaller or larger, the cost is decreased as a penalty.

5.3.3 Initialization

Each genome is initialized with the method given in algorithm 8. Firstly, for each genome a solution with no shots is created. Then for each target, we assign a shot with a random weapon and time while keeping track of availability also according to hierarchy information.

Algorithm 8 Initialization of a Single Genome VHW

```
 1: Initialize all bits of the 3D genome to 0
 2: Initialize array $W$ as available for all weapons at all time instances
 3: for all Targets do
 4:     Using $W$, among all available weapons at all times, randomly select a weapon and time
 5:     In order to update $W$, call Mark Weapons as Unavailable for selected weapon and time
 6:     Add the shot for this target to the solution
 7: end for
```

In our solution for variation of hierarchy in weapons, in many cases we need to keep track of availability of weapons and during this process we need to consider weapon hierarchy. For this purpose, we have a method which properly updates a given availability of weapons at time instances array. This method is defined in algorithm 9.

Algorithm 9 Mark Weapons as Unavailable

```
Require: To be passed an array $W$ of availability of all weapons at all time instances
 1: Mark the weapon itself as unavailable at given time on $W$
 2: Find the type of the weapon; battery, section or single weapon
 3: if The weapon is a single weapon then
 4:     Mark section of this single weapon as unavailable at given time on $W$
 5:     Mark battery of this single weapon as unavailable at given time on $W$
 6: else if The weapon is a section then
 7:     Mark all single weapons under this section as unavailable at given time on $W$
 8:     Mark battery of this section as unavailable at given time on $W$
 9: else if The weapon is a battery then
10:     Mark all single weapons and sections of this battery as unavailable at given time on $W$
11: end if
```

5.3.4 Crossover Operators and Repair Strategies

Similar to our approach for variation of target joining, we are using two crossover strategies; single plane crossover on columns and single plane crossover on rows. Crossover mechanisms are nearly the same as mechanisms for variation of target joining, however we have different repair mechanisms for this variation.

An important aspect of these repair mechanisms is that this time we need to check weapon hierarchies besides multiple entries at columns and rows. Checking weapon hierarchies are required for this representation to guarantee that rules related to weapon hierarchies hold for all hierarchies at each time instance.

Assume we have two sample individuals shown in figure 5.7 and we will apply both types of crossover operators on these individuals.

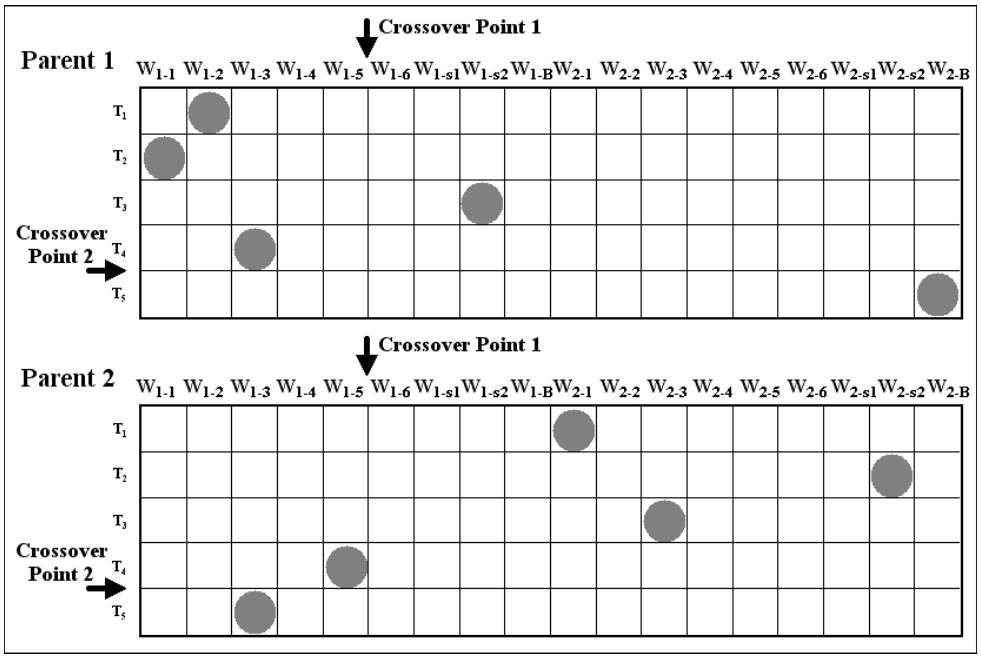
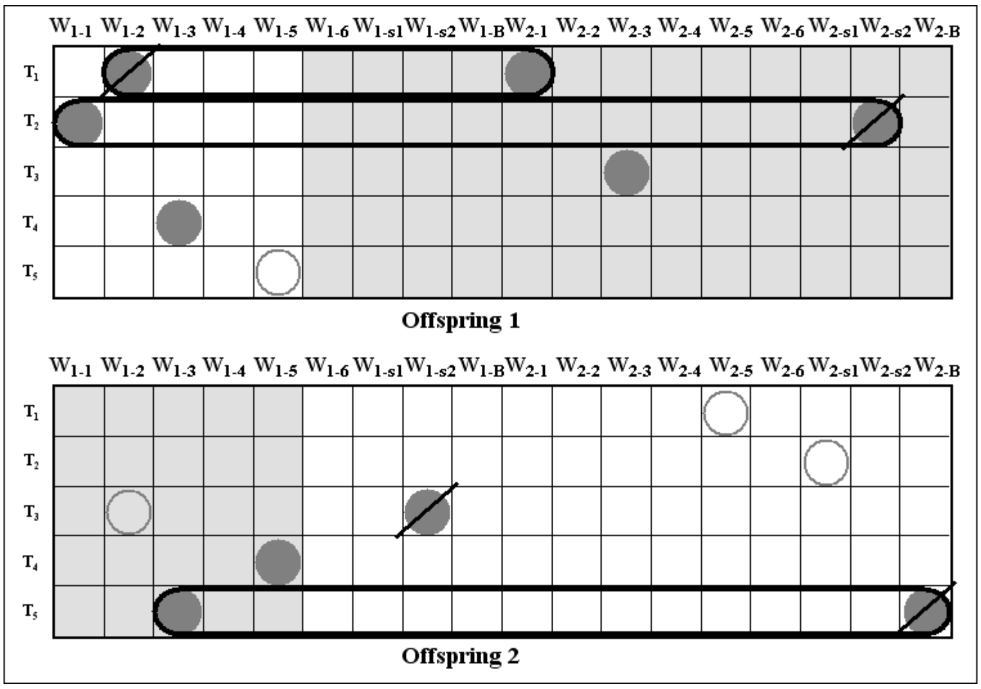

Crossover Point 1
Parent 1
W1-1 W1-2 W1-3 W1-4 W1-5 W1-6 W1-s1 W1-s2 W1-B W2-1 W2-2 W2-3 W2-4 W2-5 W2-6 W2-s1 W2-s2 W2-B
T1
T2
T3
T4
T5
Crossover Point 2
Crossover Point 1
Parent 2
W1-1 W1-2 W1-3 W1-4 W1-5 W1-6 W1-s1 W1-s2 W1-B W2-1 W2-2 W2-3 W2-4 W2-5 W2-6 W2-s1 W2-s2 W2-B
T1
T2
T3
T4
T5
Crossover Point 2

Figure 5.7: Sample Parent Individuals for Variation of Hierarchy in Weapons

If we apply single plane crossover operator on columns by using crossover point 1 shown in figure 5.7, two offsprings shown in figure 5.8 will be created. Similar to our previous examples, columns painted with grey comes from parent 2. These offsprings do not represent feasible solutions and they need to be repaired.



W1-1 W1-2 W1-3 W1-4 W1-5 W1-6 W1-s1 W1-s2 W1-B W2-1 W2-2 W2-3 W2-4 W2-5 W2-6 W2-s1 W2-s2 W2-B
T1
T2
T3
T4
T5
Offspring 1
Offspring 2

Figure 5.8: Offsprings created after a column crossover and related repairs

For Offspring 1, target T$_1$ and target T$_2$ have two entries at their rows which means that these targets are shot up by two weapons. For each of these targets, we need to cancel one of their shots. For targets T$_1$ and T$_2$, we randomly select to cancel shots from W_{1-2} and W_{2-s2}, respectively. Then we check each of the two weapon hierarchies for feasibility and find out that there are no violations. We end up with a solution which do not have any shots for target T$_5$. Therefore, as a last step of our repair mechanism we randomly assign a shot for target T$_5$ to result in a feasible solution. This randomly assigned shot is shown with an unfilled grey circle in figure 5.8.

For Offspring 2, target T$_5$ has been assigned two shots from W_{1-3} and W_{2-B}. We need to cancel one of these shots and we randomly select shot from W_{2-B}. Then we check each of the two weapon hierarchies for feasibility and find out that in the first hierarchy there are shots for both W_{1-5} and W_{1-s2}. Since W_{1-5} is a member of section W_{1-s2}, these two shots are not possible at same time instance. Therefore we need to cancel one of these shots and randomly we select to cancel the shot from W_{1-s2}. After these cancellations, we end up with there unshot targets; T$_1$, T$_2$ and T$_3$. For these targets, we assign shots randomly while considering the feasibility of the solution. Again, these randomly assigned shots are shown with unfilled grey circles in figure 5.8.

If we apply single plane crossover operator on rows by using crossover point 2 shown in figure 5.7, two offsprings shown in figure 5.9 will be created. As in our previous examples, rows painted with grey comes from parent 2.

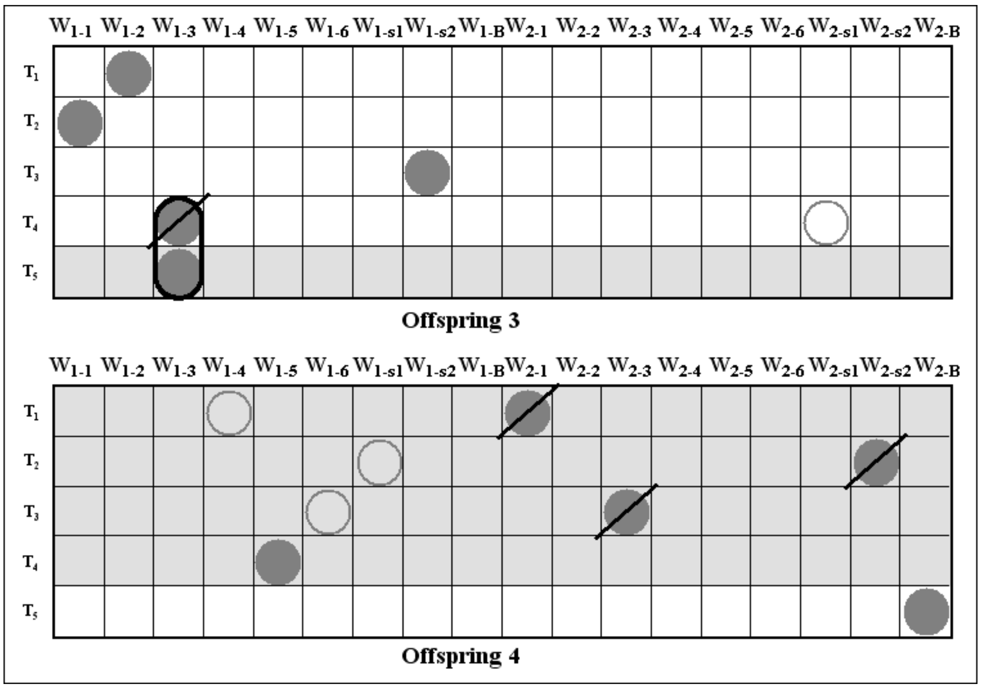

W1-1 W1-2 W1-3 W1-4 W1-5 W1-6 W1-s1 W1-s2 W1-B W2-1 W2-2 W2-3 W2-4 W2-5 W2-6 W2-s1 W2-s2 W2-B
T1
T2
T3
T4
T5
Offspring 3
Offspring 4

Figure 5.9: Offsprings created after a row crossover and related repairs

In Offspring 3, we have multiple shots for weapon W_{1-3} since we have ended up with multiple entries at the related column. We need to cancel one of these shots and we randomly select to cancel the shot for T_4. Then we check each of the two weapon hierarchies for feasibility and find out that there are no violations in this offspring. Although we are close to a feasible solution, we have no shots for target T_4. As a last step of our repair mechanism for a feasible solution, we randomly assign a shot for target T_4. This randomly assigned shot is shown with an unfilled grey circle in figure 5.9.

In Offspring 4, we do not have any multiple entries. However, in weapon hierarchy 2 we have violations on rules related to hierarchy of weapons. There are shots for the battery W_{2-B} of the hierarchy, for one section W_{2-s2} of the hierarchy and for two single weapons, namely W_{2-1} and W_{2-3}, of the hierarchy. We should keep shots for smaller weapons or larger weapons.

Randomly we decide to keep shots for larger weapons and as a result we cancel shots for W_{2-s2}, W_{2-1} and W_{2-3}. After these cancellations, we have three unshot targets; T_1, T_2 and T_3. For these targets, again we assign shots randomly while considering the feasibility of the solution and we show these randomly assigned shots with unfilled grey circles in figure 5.9.

Overall definition of our crossover strategy is given in algorithm 10. This overall definition of our crossover strategy is very similar to the definition for variation of target joining. Similarly, chances of randomly selecting row crossover or column crossover are equal.

Algorithm 10 Single Plane Crossover of Two Parents VHW

```
1:  Parent1 $P1$, Parent2 $P2$, Offspring1 $O1$, Offspring2 $O2$
2:  Randomly select Row Crossover or Column Crossover
3:  **if** Row Crossover **then**
4:      Select a random point $N$ among rows (targets)
5:      Copy rows 0 to $N - 1$ of $P1$ to $O1$
6:      Copy rows $N$ to *end* of $P2$ to $O1$
7:      Copy rows 0 to $N - 1$ of $P2$ to $O2$
8:      Copy rows $N$ to *end* of $P1$ to $O2$
9:      Call *Repair Row Crossover VHW* for offspring $O1$
10:     Call *Repair Row Crossover VHW* for offspring $O2$
11: **else if** Column Crossover **then**
12:     Select a random point $M$ among columns (weapons)
13:     Copy columns 0 to $M - 1$ of $P1$ to $O1$
14:     Copy columns $M$ to *end* of $P2$ to $O1$
15:     Copy columns 0 to $M - 1$ of $P2$ to $O2$
16:     Copy columns $M$ to *end* of $P1$ to $O2$
17:     Call *Repair Column Crossover VHW* for offspring $O1$
18:     Call *Repair Column Crossover VHW* for offspring $O2$
19: **end if**
```

The repair strategies for row and column crossover operators used in algorithm 10, are explained in detail in algorithm 11 and algorithm 12.

Algorithm 11 Repair Row Crossover VHW

```
1: for all Time instances do
2:     for all Weapon Hierarchies do
3:         Call Up Down Strategy for this weapon hierarchy at this time
4:     end for
5: end for
6: Call Add Required Shots VHW for this genome
```

Algorithm 12 Repair Column Crossover VHW

```
1: for all Targets do
2:     Initialize number of shots $N_S$ as 0
3:     Initialize $S$ as an empty array
4:     for all Weapons do
5:         for all Time Instances do
6:             if There is a shot for this target then
7:                 Store weapon and time instance of this shot at $S[N_S]$
8:                 Increase $N_S$ by one
9:             end if
10:        end for
11:    end for
12:    if $N_S < 2$ then
13:        No infeasibility for this target, so continue with the next target
14:    else if $N_S == 2$ then
15:        Randomly select one of the shots $S[0]$ or $S[1]$
16:        Cancel the selected shot while keeping the other shot
17:    end if
18: end for
19: for all Time instances do
20:     for all Weapon Hierarchies do
21:         Call Up Down Strategy for this weapon hierarchy at this time
22:     end for
23: end for
24: Call Add Required Shots VHW for this genome
```

Since in variation of hierarchy in weapons, there can be only one entry at each row and at each column, the resulting offsprings can have at most two entries at a single row or a single column. Repair strategy for column crossover cancels extra shots explicitly by counting shots for each target and cancelling one of the shots randomly if required. Repair strategy for column crossover deals with extra shots by the help of up down strategy.

Actually up down strategy is the core idea of our repair strategies in this variation and it is designed to fix infeasibilities related to weapon hierarchies as explained. However this up down strategy additionaly handles possible multiple entries at a column after a row crossover, since it uses the method defined in algorithm 14.

Both repair strategies makes use of the up down strategy which is given in algorithm 13. Initially this method randomly selects up or down strategy. Both strategies have equal chances in this random selection. Then weapon hierachy is checked for rules related to hierarchy in weapons according to this selection. If up strategy is chosen, smaller weapons (eg. single weapon) are favored and infeasible situations are corrected accordingly. On the other side if down strategy is chosen, larger weapons (eg. battery) are favored and infeasible situations are corrected accordingly.

Algorithm 13 Up Down Strategy

```
Randomly select Down Strategy or Up Strategy
if Down Strategy then
    if Call Check Weapon at Time for battery of this hierachy for this time then
        Cancel all shots of all single weapons and sections of this battery at this time
        End Up Down Strategy
    end if
    for all Sections of this hierarchy do
        if Call Check Weapon at Time for this section for this time then
            Cancel all shots of all single weapons of this section at this time
        else
            Call Check Weapon at Time for each single weapon of this section at this time
        end if
    end for
else if Up Strategy then
    for all Sections in this hierarchy do
        Call Check Weapon at Time for each single weapon of this section at this time
        if There exists at least one shot from any of the single weapons of this section then
            Cancel all shots of this section at this time
            Cancel all shots of the battery of this section at this time
        else
            if Call Check Weapon at Time for this section at this time then
                Cancel all shots of the battery of this section at this time
            else
                Call Check Weapon at Time for battery of this section at this time
            end if
        end if
    end for
end if
```

Algorithm 14 Check Weapon at Time

Ensure: Returns true (there exists at least one shot) or false (no shots)

```
1:  Initialize number of shots $N_S$ as 0
2:  Initialize targets that are shot up at this column $T_S$ as an empty array
3:  for all Targets do
4:      if There is a shot for this target with this weapon at this time then
5:          Add this target to $T_S[N_S]$
6:          Increase $N_S$ by one
7:      end if
8:  end for
9:  if $N_S == 0$ then
10:     Return False
11: else if $N_S == 1$ then
12:     Return True
13: else if $N_S == 2$ then
14:     Randomly select one of the shots $T_S[0]$ or $T_S[1]$
15:     Cancel the selected shot while keeping the other shot
16:     Return True
17: end if
```

Similar to our solution for variation of target joining, after cancelling improper shots, our repair strategies for variation of weapon hierarchy make use of a method called add required shots. This method is given in algorithm 15.

Algorithm 15 Add Required Shots VHW

```
1:  Initialize array $W$ as available for all weapons at all time instances
2:  Initialize array $T$ as unshot for all targets
3:  for all Time Instances do
4:      for all Weapons do
5:          for all Targets do
6:              if There is a shot for this target then
7:                  Mark this target as shot in $T$
8:                  To update $W$, call Mark Weapons as Unavailable for this weapon and time
9:              end if
10:         end for
11:     end for
12: end for
13: if All targets are already shot up then
14:     End Add Required Shots VHW
15: end if
16: for all Targets do
17:     if Target is already shot up then
18:         Continue with the next target
19:     end if
20:     Using $W$, among all available weapons at all times, randomly select a weapon and time
21:     Add the shot for this target to the solution while updating $T$ properly
22:     To update $W$, call Mark Weapons as Unavailable for selected weapon and time
23:     if All targets are shot up now then
24:         End Add Required Shots VHW
25:     end if
26: end for
```

5.3.5 Mutation

Mutation strategy in this solution randomly cancels a shot for a target. After this cancellation, in order to restore the solution to be a feasible solution, the method given in algorithm 15 is applied.

The details of our mutation approach are given in algorithm 16.

Algorithm 16 Mutation VHW

```
1: Considering the mutation probability randomly decide whether to mutate or not
2: if Not to mutate then
3:     End Mutation VHW
4: end if
5: Randomly select a target $T$ to mutate among all targets
6: Find weapon and time of the shot that is already assigned to $T$
7: Cancel the shot for $T$
8: Call Add Required Shots VHW for this genome
```

CHAPTER 6

IMPLEMENTATION AND EXPERIMENTS

6.1 Implementation

In the scope of this thesis, we implemented both of our proposed solutions for variation of target joining and for variation of hierarchy in weapons. For implementations, we used GAlib, “a C++ library of genetic algorithm objects” [7]. This is a widely used genetic algorithms library with many features. In general, GAlib has been a proper choice by means of meeting the requirements of our solutions.

The library provides varying types of representations, genetic operators and genetic algorithms. Moreover, the library is very generic that it provides mechanisms to define new representations, operators and algorithms for different requirements of the user.

As both of our solutions are represented by 3D binary matrices, the 3D binary string genome class, GA3DBinaryStringGenome, in GAlib was a suitable choice for our representations. Just providing proper dimensions for the matrices was sufficient to use instances of this class. We used class member functions which set and return values of an entry in the genome and functions which copy a specified part of a genome to a new genome.

GAlib supports four general types of genetic algorithms; standard simple genetic algorithm, steady-state genetic algorithm, incremental genetic algorithm and deme genetic algorithm. In this study we used steady-state genetic algorithm which is based on overlapping populations. At each generation, a specified portion of the overall population is replaced by new individuals. In our experiments, we used a fixed replacement number in order to specify the amount of individuals to be replaced at each generation. For selection of individuals to be mated in the steady-state genetic algorithm, we used the default selection method; the Roulette Wheel Selection. As a stopping criteria for the steady-state genetic algorithm, we preferred fixed number of generations in our experiments.

Each genome and genetic algorithms class in GAlib uses some genetic operators. For each genetic operator, usually there are more than one implemented choice in the library and moerover as stated before, the user can define new genetic operators. Because of this reason, for each possible kind of genetic operator, the predefined genome and genetic algorithms classes hold function pointers. These function pointers are initialized to the function pointers of default predefined genetic operators and can be changed for any instance of these classes. We used this mechanism to define our own initialization, crossover and mutation operators for each of our solutions.

Because of the base genome class, all genome classes of GAlib are capable of some fundemental abilities. One such ability is the user data pointer in all genomes. The user data pointer is a void pointer and is used for data which is common for all genomes. Only one copy of the related data is stored, however each genome can easily access this data with this common pointer. Since we use cost function values for each weapon target pair at each time instance and additional data like joinable targets at each time instance, this user data pointer has been a very useful ability during the implementations. For this purpose, we defined a class which just serves as an encapsultor for all such data and assigned the pointer of an instance of this class to the user data pointer.

GAlib provides its own random function implementations and for our solutions we used these random function implementations. All such GAlib random function calls can be initialized with a seed which results in the same sequence of random values. This ability gave us the possibility to repeat some experiments although actually there was randomness for each run.

As a last ability, all genetic algorithms parameters in GAlib are encapsulated in GAParameterList objects and can be modified inside the code, from command line or from a text file. This property is especially useful for running different experiments without compiling the code.

6.2 Testbed and Input Data Creation

Our testbed is composed of a data generator, data created by this generator, various configuration files and our implementations of proposed solutions. The data generator creates the data and the implementations use this data as an input together with the related configuration files.

In order to use more realistic data during this testing process, instead of a simple random data generator we implemented a simulator named as Artillery Weapon Target Simulator. During data creation, this simulator considers the dynamics and various aspects of artillery branch of military. Moreover, we are able to create varying sizes of proper test data by simulating target and weapon entities among time instances.

6.2.1 Artillery Weapon Target Simulator

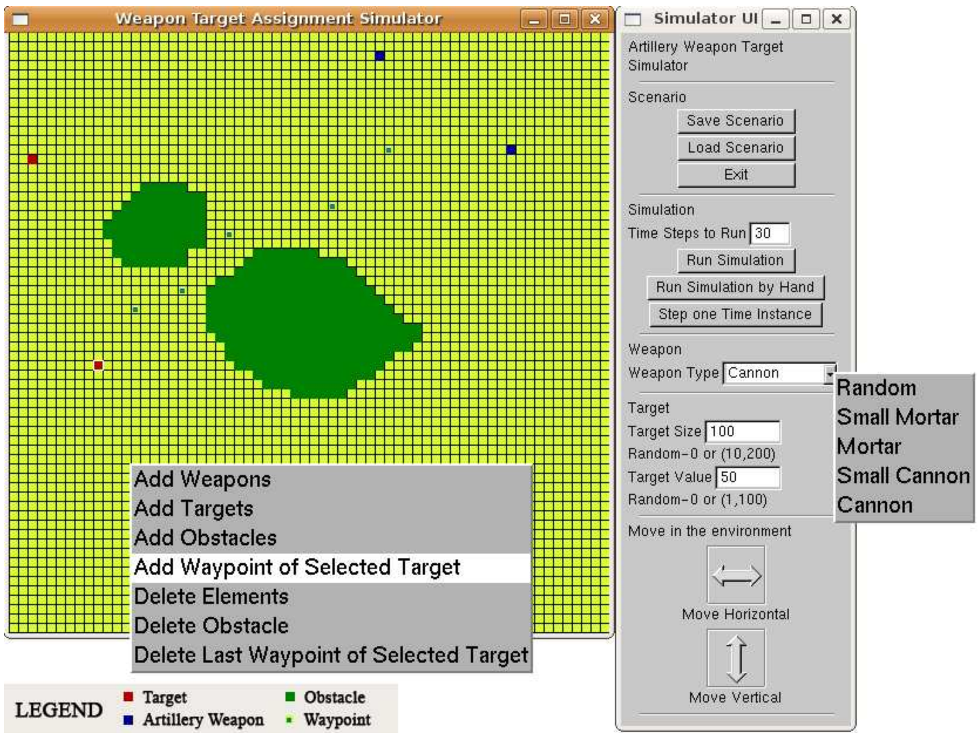
6.2.1.1 Usage of Simulator

The artillery weapon target simulator is a simple tool which provides two usage modes; an editor mode and a simulation mode. These are two distinct modes which are used separately. The Artillery Weapon Target simulator starts in editor mode and whenever operations on the editor mode are finished, the tool is switched to the simulation mode.

In the editor mode, a simulation scenario is created. A simulation scenario is composed of simulation elements which are located on an environment. These simulation elements are weapons, targets and waypoints for each target. The environment is a grid area consisting of grid cells. Some of these grid cells are obstacles which means that targets and weapons can not be located on or moved to these cells. Obstacles are represented with dark green cells where other cells in the environment are represented in light green as shown in figure 6.1.

Simulation elements and obstacles can be created with any sequence. However its better to create the obstacles first and then continue with simulation elements.

Each weapon is located on the environment individually. Each of these weapons has a weapon type which determines its coverage area. Weapon type can be selected from the user interface or can be randomly assigned by the simulator. As shown in the figure 6.1, weapons are represented with blue cells.



Time Steps to Run 30

Weapon Type Cannon

Target Size 100

Random-0 or (10,200)

Target Value 50

Random-0 or (1,100)

Figure 6.1: Artillery Weapon Target Simulator

Each target is also located individually on the environment. Each target is assigned a target value and a target size. These can be selected within previously defined intervals from the user interface or they can be randomly assigned by the simulator. As shown in the figure 6.1, targets are represented with red cells.

For each target, waypoints can be created which define the movement path of the target for simulation. For this purpose, firstly a target is selected and then waypoints are defined in a sequence. As shown in the figure 6.1, waypoints are represented with small green squares in the environment.

The area size of the environment is defined at the begining and it is usually larger than the view area of the simulator window. Because of this reason, there are user controls to move the view area around on the environment.

The created scenario can be saved for further use and loaded back to the simulator later. Any loaded scenario can be modified.

When creation of the scenario is finished, number of simulation time steps is specified and the simulator is switched to the simulation mode to run the scenario. The simulation scenario can be run either automatically or by hand. Automatic run executes simulation time steps in fixed time intervals without any user input. Run by hand executes each simulation time step by a single user input given at any time. This latter simulation method is especially useful for debugging and inspection purposes.

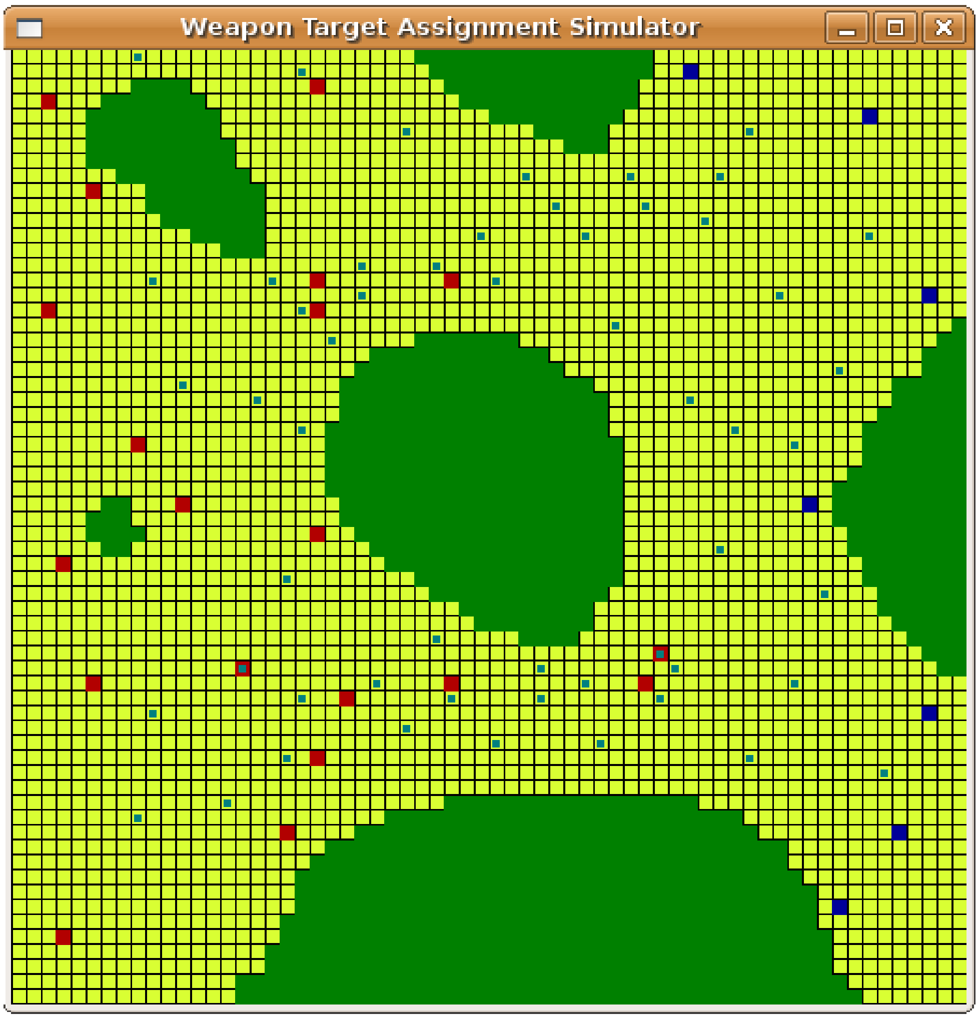


Figure 6.2: A screenshot of the simulation at a time instance

Each discrete time step of the simulation is calculated with respect to the previous time instance. At each time instance there are two things to calculate; target positions and logging data.

With each simulation time step, targets are moved along the paths among their waypoints. A target can move to one of its eight neighbouring grid cells at each simulation time step. The next grid cell is determined according to the linear path between two consecutive waypoints. This linear path is calculated by the well-known Bresenham's line drawing algorithm.

Logging data is composed of the cost function values for all weapon target pairs and additional simulation data like target pairs that can be joined at that time instance. Calculation of cost function values in this simulator is explained in detail at section 6.2.2. Target pairs that can be joined is determined by calculating the distance between targets and comparing this distance with a threshold value.

Figure 6.2 shows a screenshot of a running simulation. In this simulation, some targets are moving whereas some targets are stationary since they have no waypoints or they have already reached the last waypoint. Also, there are some targets which are close enough to be joined.

6.2.1.2 Implementation of Simulator

The simulator is implemented with an object oriented and platform independent design in C++. For rendering related issues OpenGL is used. For event handling, window management and pop-up menu implementation, OpenGL Utility Toolkit (GLUT) is used. For the remaning user interface issues, a GLUT-based user interface library named as GLUI is used.

For this simulator, even though we could have implemented more advanced graphics with advanced rendering techniques ranging from textures to shading methods, we preferred to have simple 2D graphics for visualization. One of the main reasons behind this choice is our goal to increase understandibility with simplicity. Moreover, this simulator is just a helper tool for our study since we are using it for only creating sample proper data for testing our solutions to the problem.

6.2.2 Cost Calculation for Weapon Target Pairs

As stated in previous chapters, for artillery target assignment problem different cost functions can be selected according to different military tactics. For cost calculation in our artillery weapon target simulator, we used a sample cost function. This sample cost function makes use of different tactical choices and combines them into a single value.

We refer this sample cost function as Shooting Cost. Shooting cost is a function of target *i*, weapon j, time instance k and represents the gain of the shot from weapon j to target i at time instance k. The function creates larger values for better shots where betterness is related to some previously defined constraints about tactical choices.

As stated in equation 6.1, we define shooting cost as a multiplication of coverage value and target cost.

$$ShootingCost(i, j, k) = Coverage(i, j) \times TargetCost(i, k) \tag{6.1}$$

Coverage value in equation 6.1 represents the effectiveness of the weapon j over the target i by means of coverage area of the weapon (i.e. width of one shot of the weapon) and size of the target. The relation between these two values can be defined directly from the artillery branch of the military domain.

In military, if your weapon's coverage area is smaller than your target's size, then it means that your weapon can not cover the target properly which results in less damage than a total kill on the target. If your weapon's coverage area is equal to or larger than your target's size, then it means that your weapon can cover your target properly which may result in a total kill on the target. However, if your weapon's coverage area is very large than your target's size, then it means that you are overkilling the target and wasting your valuable resources, especially by means of ammunition.

Therefore, in order to calculte coverage value, firstly we divide weapon's coverage area to the target size. As explained above, we have three possible cases for the result of this division:

- If the result is less than 1, it means that our weapon j can not cover the target i. Then as a coverage value, we use the result of the division itself. By this way, we penalize such shootings the amount of this ratio.

- If the result is between 1 and 1.5, it means that our weapon j can cover the target i without overkill or with insignificant overkill. Then as a coverage value, we use 1. By this way, we favour such shootings.

- If the result is greater than 1.5, it means that it will be an overkill to use weapon j on target i. Then as a coverage value, we use 1 divided by the result of the previous division. As we result in a coverage value less than 1, we also penalize such shootings with this value.

As a result we always have a coverage value between $(0, 1]$ and this value does not depend on time.

Target cost in equation 6.1 is a combination of three main components; target value, closeness to the nearest weapon and time. Calculation of target cost is given as a weighted summation of these three components in equation 6.2.

$$TargetCost(i, k) = [(Value(i) \times P_1) + (Closeness(i) \times P_2) + (Time(k) \times P_3)] \tag{6.2}$$

Each of these three main components are explained below.

- **Target value:** Whenever a target is created in the simulator, a target value is specified via user input or randomly. This value is always between 1 and 100. Higher target values represents more valuable targets where valuable stands for higher priority for shooting and damage.

- **Closeness to the nearest weapon:** In order to calculate the value of this component, firstly for each target, nearest weapon to the target is found and the distance value between the target and this weapon is gathered. Then maximum and minimum of these distance values are found. Target with minimum distance value receives 100 as a closeness value and target with maximum distance value receives 0 as a closeness value. Other targets are mapped to values between 0 and 100 linearly according to the proximity of their distance values to maximum and minimum. Therefore, targets which are closer to some weapons receive higher closeness values. This is a desired result since such targets may be dangerous for related weapons and should be priorly shot up as a military tactic. Actually, value of this component can be considered as a sorting of targets according to a distance metric based on closeness to the nearest weapon.

- **Time:** In order to calculate the value of this component, number of remaining simulation time instances is divided by the total simulation time instances. The gathered value is multiplied by 100 which results in values between 0 and 100 for this component. This gives us a value which favors earlier time instances of the simulation. Since shooting targets as soon as possible is a preferred military tactic in many cases, this is a desired result. It is important to figure out that this component creates a major difference in costs of different time instances of simulation.

P_1, P_2 and P_3 are coefficients representing the importance of these three components. As long as their total is 1.0, any value between 0.0 and 1.0 can be selected for P_1, P_2 and P_3. In our experiments, we selected 0.5 for P_1, 0.3 for P_2 and 0.2 for P_3.

6.2.3 Datasets Created by the Simulator

By using the simulator, we created three different datasets in order to test our solutions. For this purpose, simulation environments are created and simulation elements are located in these environments by user input. Although weapon and target properties are mostly set by user input, random values are used for setting these properties for a small portion of each dataset. By this way with selecting logical values via user input and providing some randomness, we aimed to have proper variance in each dataset. This resulted in datasets which have different types of weapons (i.e. different coverage areas) and different targets by means of their target sizes and target values.

Dataset 1 is a middle sized dataset whereas dataset 2 and dataset 3 are larger datasets. Actually, dataset 2 and dataset 3 have the same simulation environment and elements, but they differ by means of total number of time instances. Detailed information on number of elements in each dataset is given in table 6.1.

Table 6.1: Number of Elements in Datasets

	Dataset 1	Dataset 2	Dataset 3
Number of Weapons for VTJ	5	10	10
Number of Weapons for VHW	45	90	90
Number of Targets	10	100	100
Number of Time Instances	51	51	101
Number of Total Joins for VTJ	32	396	943

6.3 Experiments and Results

We conducted experiments with all of the three datasets for both of our solutions. The genetic algorithm settings that we have used for these experiments are given in table 6.2. These settings change among different datasets, but they are the same for each of our solutions.

For larger datasets, we need to use higher number of generations. During our studies, we also tried larger populations for larger datasets. However this did not have a significant effect on the results for our solutions. Therefore we decided to use the same population size for each dataset.

Table 6.2: Genetic Algorithm Settings for Experiments of Datasets

	Dataset 1	Dataset 2	Dataset 3
Population Size	20	20	20
Number of Replacements	10	10	10
Number of Generations	500	1500	1500
Crossover Probability	0.90	0.90	0.90
Mutation Probability	0.70	0.70	0.70

In each of our experiments, we ran implementations of our solutions with the related datasets for 20 times. Then we took the average of these 20 runs and created graphs of best score versus number of generations. In the rsulting graphs of our experiments, our solutions successfully converged as we expected.

When inspecting our results, it is important to keep in mind that since each of the two solutions that we have proposed actually deals with different sub-problems of ATAP, their results are not comparable with each other.

In the remaining of this chapter, you can find the resulting graphs of our experiments. Figure 6.3, figure 6.4 and figure 6.5 depict the results for Variation of Target Joining whereas Figure 6.6, figure 6.7 and figure 6.8 depict the results for Variation of Hierarchy in Weapons.

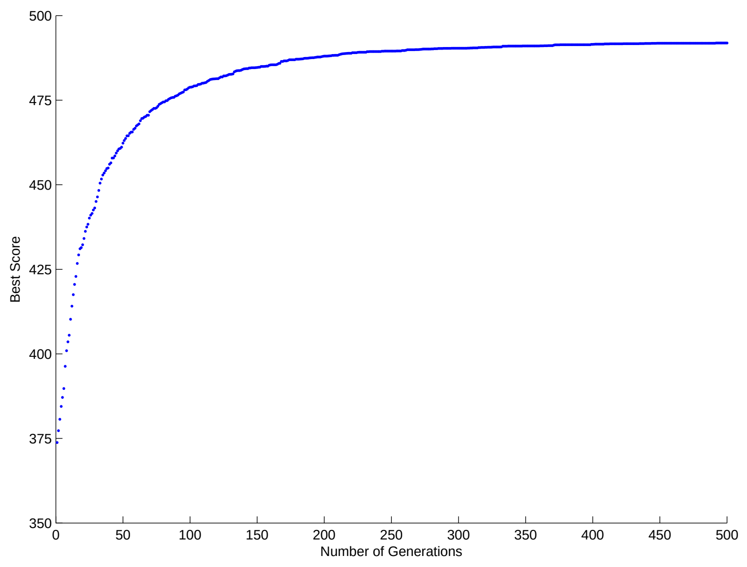


Figure 6.3: Results of dataset 1 for variation of target joining

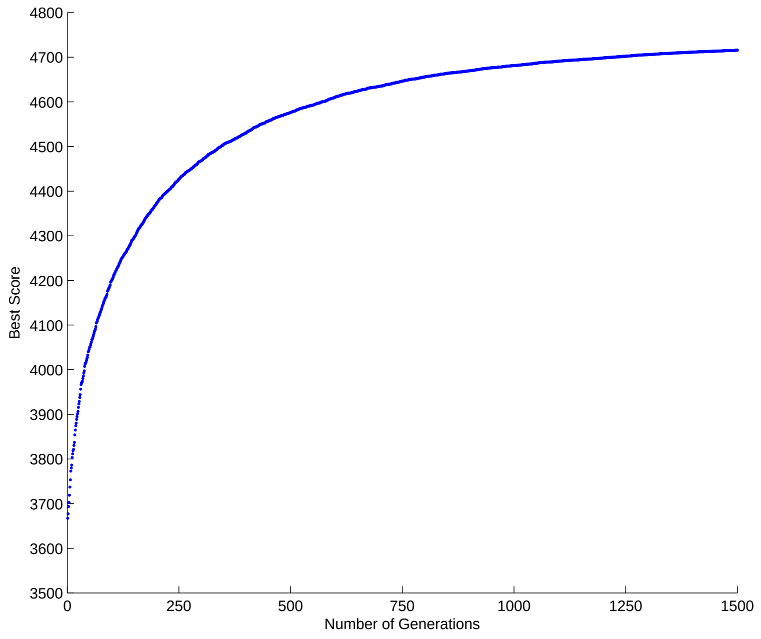


Figure 6.4: Results of dataset 2 for variation of target joining

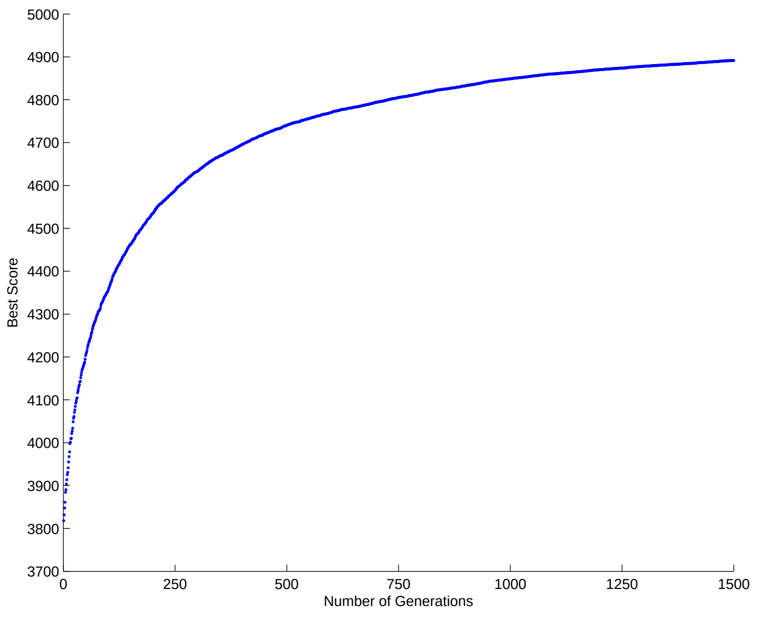


Figure 6.5: Results of dataset 3 for variation of target joining

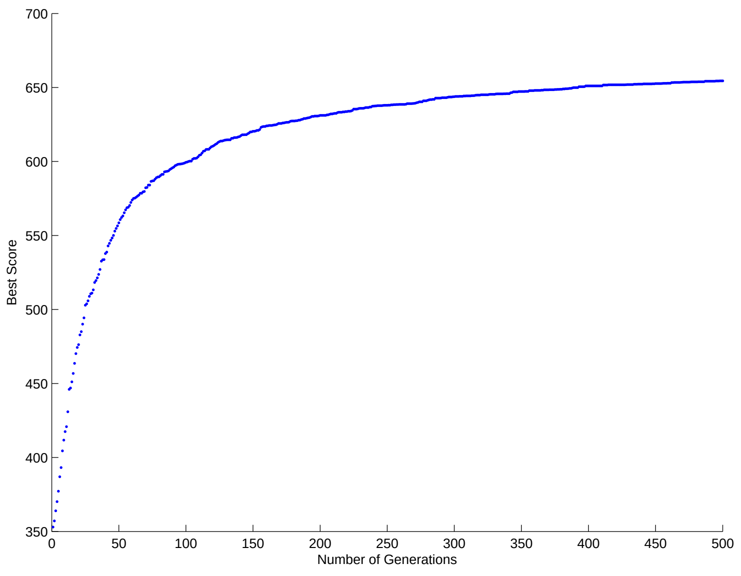


Figure 6.6: Results of dataset 1 for variation of hierarchy in weapons

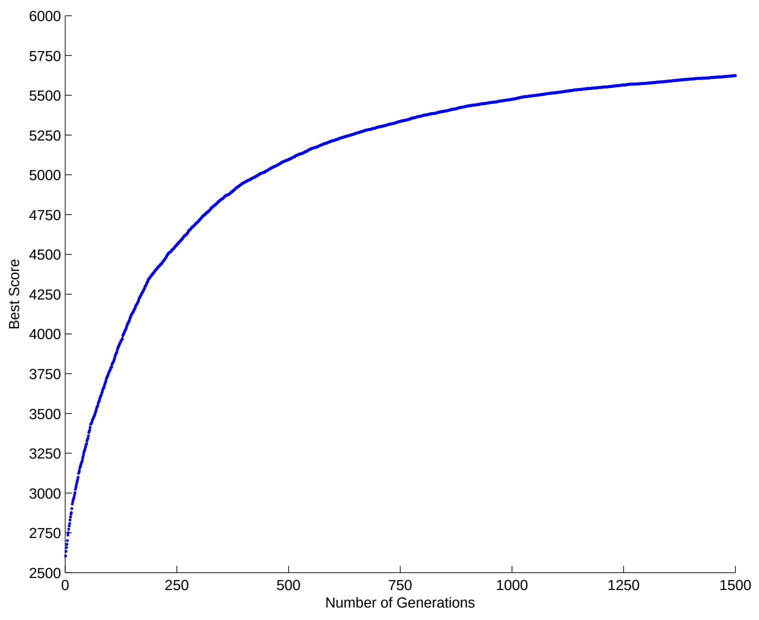


Figure 6.7: Results of dataset 2 for variation of hierarchy in weapons

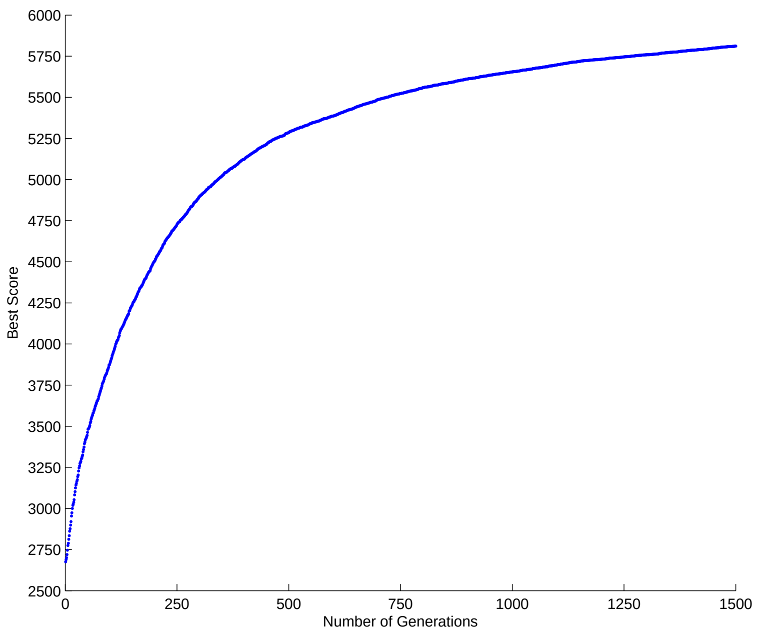


Figure 6.8: Results of dataset 3 for variation of hierarchy in weapons

CHAPTER 7

CONCLUSION

With this study, we defined a new assignment problem and named this problem as the artillery target assignment problem (ATAP). This new problem has many intrinsic properties which do not have much similar examples on the literature. Therefore, artillery target assignment problem can be considered as a novel area of study in the field of assignment problems. Also because of these intrinsic properties, artillery target assignment problem is a very difficult problem by means of finding the best possible solution.

Our study in this thesis has concentrated on solving two special sub-problems of the main artillery target assignment problem. As these sub-problems are highly complex for polynomial time solutions, we worked on efficiently finding sub-optimal solutions for them. For this purpose, our approach was employing customized genetic algorithms with proper representation and operators for each of these sub-problems. Details of these genetic algorithm methods and results of the related experiments are presented in this thesis.

Future work may concentrate on solving other sub-problems and mixture of sub-problems via genetic algorithms. Since nature of each sub-problem is different than others, definition of new representations and operators will probably be required.

REFERENCES

[1] Thomas Baeck, David B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, C5.4 Repair Algorithms. Published by CRC Press, 1997.

[2] Egon Balas, Matthew J. Saltzman. An Algorithm for the Three-Index Assignment Problem. Operations Research, 39(1):150-161, 1991.

[3] François Bourgeois, Jean-Claude Lassalle. An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. Communications of the ACM, 14(12):802-804, 1971.

[4] Carlos A. Coello Coello. Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: a Survey of the State of the Art. Computer Methods in Applied Mechanics and Engineering, 191(11-12):1245-1287, 2002.

[5] Reuven Cohen, Liran Katzir, Danny Raz. An efficient approximation for the generalized assignment problem. Information Processing Letters, 100(4):162-166, 2006.

[6] David G. Galati. Game theoretic target assignment strategies in competitive multi-team systems. Phd Thesis, University of Pittsburgh, 2004.

[7] GAlib: Matthew's Genetic Algorithms Library. http://lancet.mit.edu/ga/

[8] Mehmet Gülek. Assignment Problem and Its Variations. M.S. Thesis, Middle East Technical University. 2007.

[9] Cai Huaiping, Liu Jingxu, Chen Yingwu, Wang Hao. Survey of the research on dynamic weapon-target assignment problem. Journal of Systems Engineering and Electronics, 17(3):559-565, 2006

[10] H. W. Kuhn. The hungarian method for the assignment problem. Naval Research Logistic Quarterly, 2:83-97, 1955.

[11] Ojeong Kwon, Kyungsik Lee, Sungsoo Park. Targeting and Scheduling Problem for Field Artillery. Computers and Industrial Engineering. Vol. 33, Nos 3–4, pp. 693-696. 1997.

[12] Zne-Jung Lee, Wen-Li Lee. A Hybrid Search Algorithm of Ant Colony Optimization and Genetic Algorithm Applied to Weapon-Target Assignment Problems. Lecture Notes in Computer Science 2690, pages 278-285, 2003.

[13] Zne-Jung Lee, Shun-Feng Su, Chou-Yuan Lee. A Genetic Algorithm with Domain Knowledge for Weapon-Target Assignment Problems. Journal of the Chinese Institute of Engineers, 25(3):287-295, 2002.

[14] Zne-Jung Lee, Shun-Feng Su, Chou-Yuan Lee. Efficiently Solving General Weapon-Target Assignment Problem by Genetic Algorithms With Greedy Eugenics. IEEE Trancations on Systems Man and Cybernetics, Part B Cybernetics, 33(1):113-121, 2003.

[15] Zbigniew Michalewicz. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA (1995), pp. 135-155.

[16] James Munkres. Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics, 5(1):32-38, 1957.

[17] David Orvosh, Lawrence Davis. Using a Genetic Algorithm to Optimize Problems with Feasibility Constraints. Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, New York (1994), pp. 548-553.

[18] Jay M. Rosenberger, Hee Su Hwang, Ratna P. Pallerla, Adnan Yucel, Ron L. Wilson, Ed G. Brungardt. The Generalized Weapon Target Assignment Problem. 10th International Command and Control Research and Technology Symposium, The Future of C2, McLean, VA. 2005.

[19] Frits C. R. Spieksma. Multi index assignment problems: complexity, approximation, applications. Nonlinear Assignment Problems, Chapter 1, Springer, First Edition, 2000.