



MSc Thesis in

Computer Science - Mobile Computing

*Adaptive complex system modeling for realistic
modern ground warfare simulation analysis based
on evolutionary multi-objective meta-heuristic
techniques*

Diogo Alexandre Breites de Campos Proença

MSc Thesis directed by Professor Silvio Priem Mendes of the School of Technology and Management, Polytechnic Institute of Leiria and co-directed by Juan Ant3nio Gomez-Pulido of the University of Extremadura, Spain.

Leiria, 2011

This Page Intentionally Left Blank

To my family

This Page Intentionally Left Blank

Abstract

The battlefield is a harsh and inhuman environment, where deaths and destruction take lead role. Through many millennia there was blood shed all over the world, people who many time died in a battle that sometimes they didn't even care about.

Today, the battle field is very different, machines take most damage and there are less casualties, this is because of the advancements made in the fields of aeronautics, weaponry, nautical, vehicles, armor, and psychology.

Also there is another important party that throughout the last decades made a special and decisive advantage to the side which is more advanced in this field, it is intelligence and simulation. Intelligence today gives enormous advantage to one country as you “see and feel” the battlefield hundreds or thousands kilometers away. Then, with the data provided by intelligence, countries can simulate the battle in order to deploy the most efficient units into battle.

In this thesis we propose a warfare simulator analysis tool using a multi-objective approach and artificial intelligence. Further on, the 1991 Gulf war scenario is used to simulate and the results are presented and analyzed.

The approach used in this thesis is difficult to be used in games due to its processing complexity and computing demands.

Keywords: Meta-heuristic, Warfare simulator, Multi-objective optimization, Artificial intelligence, Evolutionary algorithms.

This Page Intentionally Left Blank

Resumo

O campo de batalha é um meio adverso e inumano, onde a morte de seres humanos e a destruição têm o papel principal. Desde há muito tempo que sangue é derramado por todas as partes do globo, e muitos desses seres humanos morrem numa guerra que não é sua e pela qual não têm o mínimo apreço.

Atualmente, o campo de batalha é muito diferente, as máquinas é que sofrem o maior dano e há menos mortes. Isto deve-se aos avanços feitos nas áreas da aeronáutica, do armamento, da náutica, dos veículos terrestres, da proteção e da psicologia.

Nas últimas décadas, a informação secreta e as simulações, também tem tido um papel preponderante para as nações mais desenvolvidas. As informações secretas do campo de batalha trazem uma grande vantagem para as nações uma vez que podem “sentir” o campo de batalha a centenas ou milhares de quilómetros de distância. Depois, com a informação recolhida pelos serviços secretos, os corpos militares podem simular o campo de batalha, para que deste modo possam mobilizar as unidades de combate mais eficientes para a batalha em questão.

Nesta dissertação é proposto um simulador de guerra e uma ferramenta de análise utilizando métodos de otimização multi-objectivo e inteligência artificial. A batalha da guerra do golfo de 1991 é utilizada para simular e os resultados são posteriormente apresentados e analisados.

Os métodos utilizados nesta dissertação dificilmente poderão ser utilizados em jogos devido à sua complexidade de processamento e requisitos de computacionais.

Palavras-chave: Meta heurística, Simulador de combate, Otimização Multi-objectivo, Inteligência artificial, Algoritmos evolucionários.

This Page Intentionally Left Blank

List of figures

Fig. 1. US Deaths in Vietnam and Iraq [37].....	2
Fig. 2. Terrain representations in highly aggregated constructive simulations [38].	7
Fig. 3. Units case example	16
Fig. 4. The general scheme of an Evolutionary Algorithm as a flow-chart [46].....	23
Fig. 5. Typical progress of an EA illustrated in terms of population distribution [47].	32
Fig. 6. Typical progress of an EA illustrated in terms of development of the best fitness (objective function to be maximized) value within population in time [47].	33
Fig. 7. Illustrating why heuristic initialization might not be worth. Level a show the best fitness in a randomly initialized population, level b belongs to heuristic initialization [47].	34
Fig. 8. Illustrating why long runs might not be worth. X shows the progress in terms of fitness increase in the first half of the run, Y belongs to the second half [47].	34
Fig. 9. 1980's view on EA performance after Goldberg [58][47].	35
Fig. 10. Illustration of Wright's adaptive landscape with two traits [47].	37
Fig. 11. KMeans working example [62].....	46
Fig. 12. KMeans working example 2 [62].....	47
Fig. 13. MOEGWAO static model.....	49
Fig. 14. MOEGWAO Meta-heuristic logic overview	50
Fig. 15. Example of combat CRT mapping.....	58
Fig. 16. Desert Storm historical deployment [31]	70
Fig. 17. Unit Legend	72
Fig. 18. Configurations Explanation Map.....	73
Fig. 19. Comparison between units destroyed and targets destroyed.....	82
Fig. 20. Comparison between targets destroyed and occupation zone fitness	83
Fig. 21. Time slots required by configuration	84
Fig. 22. Comparison between configurations regarding targets destroyed, targets deserted and units destroyed	84
Fig. 23. Comparison between configurations regarding oilfield occupation	85
Fig. 24. Comparison between configurations regarding occupation zone	86
Fig. 25. Comparison between waypoint fitness and enemy fitness.....	86
Fig. 26. Comparison between weather region fitness and waypoint fitness	87
Fig. 27. Comparison between ammo fitness and supplies fitness	88
Fig. 28. Comparison between configurations regarding supplies spent per time slot (days)	89
Fig. 29. Comparison between configurations regarding ammo spent per time slot (days)	90
Fig. 30. Comparison between occupation zone fitness and oilfield fitness.....	91

This Page Intentionally Left Blank

List of tables

Table 1. High-resolution constructive simulation systems.	6
Table 2. Highly aggregated constructive simulation systems.	8
Table 3. Canonical Complexity Classes.....	12
Table 4. NP Sub Classes	13
Table 5. List of well-known multi-objective EA	41
Table 6. CRT Table [30]	59
Table 7. Weather Region Penalty with adapted values from [30]. Values adapted for use with the A-star algorithm.	60
Table 8. Main objectives to be analyzed	71
Table 9. Unit Sizes Legend	72
Table 10. Mechanized Unit Types Legend	72
Table 11. Non-Mechanized Unit Types Legend	72
Table 12. Map Legend	75
Table 13. Simulation Example	75
Table 14. Time slots required per configuration	92
Table 15. Enemy fitness per configuration	92
Table 16. Waypoint fitness per configuration	93
Table 17. Weather Region fitness per configuration.....	93
Table 18. Supplies fitness per configuration	93
Table 19. Ammo fitness per configuration.....	94
Table 20. Oilfield occupation fitness per configuration.....	94
Table 21. Occupation Zone fitness per configuration	94
Table 22. Targets destroyed per configuration.....	95
Table 23. Targets deserted per configuration	95
Table 24. Units destroyed per configuration	95

This Page Intentionally Left Blank

Code Listing

Listing 1. Pseudo-code of an EA general scheme	23
Listing 2. Procedure for finding the k means.....	45
Listing 3. Pseudo-code of the attack builder operator.....	51
Listing 4. Pseudo-code of the group state machine operator.....	53
Listing 5. Pseudo-code of the group state machine operator (continued)	54
Listing 6. Pseudo-code of the group state machine operator (continued)	55
Listing 7. Pseudo-code of the movement operator.....	56
Listing 8. Pseudo-code of the ranking operator	57
Listing 9. Pseudo-code of the attack operator	58

This Page Intentionally Left Blank

Acronyms

ATKIS Authoritative Topographic Cartographic Information System

AWAC Airborne early warning and control

C2 Command and Control Systems

CIA Central Intelligence Agency

CRT Combat Results Table

DHM Digital Height Model

DFAD Digital Feature Analysis Data

DTED Digital Terrain Elevation Data

DTEF Digital Terrain Elevation Format

EA Evolutionary Algorithm

EP Evolutionary Programming

ES Evolution Strategies

GA Genetic Algorithm

GeoTIFF Geographic Tagged Image File Format

GNP Gross National Product

GP Genetic Programming

NL Logarithmic Space

NP Non-deterministic polynomial Time

P Polynomial Time

PDF Probability distribution function

RAM Random-access machine

TM Turing machine

VMAP Vector Map

This Page Intentionally Left Blank

Table of Contents

ABSTRACT	IV
LIST OF FIGURES	VIII
LIST OF TABLES	X
CODE LISTING	XII
ACRONYMS	XIV
TABLE OF CONTENTS	XVI
INTRODUCTION	1
1.1. <i>WARFARE SCOPE</i>	1
1.2. <i>THESIS STRUCTURE</i>	3
RELATED WORK	5
2.1. <i>HIGH-RESOLUTION CONSTRUCTIVE SIMULATIONS</i>	5
2.2. <i>HIGHLY AGGREGATED CONSTRUCTIVE SIMULATIONS</i>	7
APPROACH	9
3.1. <i>DESIGN CHALLENGES AND CONSIDERATIONS</i>	9
3.1.1. <i>Problem Complexity</i>	9
3.1.2. <i>Multi-objective optimization</i>	16
BACKGROUND CANONICAL MODELS	21
4.1. <i>SUPPORTING PROCEDURES AND MECHANISMS</i>	21
4.1.1. <i>Evolutionary algorithms</i>	21
4.1.2. <i>Multi-objective evolutionary algorithms</i>	39
4.1.3. <i>A-star pathfinder algorithm</i>	42
4.1.4. <i>KMeans clustering algorithm</i>	44
THE MOEGWAO META-HEURISTIC	49
5.1. <i>ATTACK GROUP BUILDER OPERATOR</i>	51
5.2. <i>GROUP STATE MACHINE OPERATOR</i>	53
5.3. <i>MOVEMENT OPERATOR</i>	56
5.4. <i>RANKING OPERATOR</i>	57
5.5. <i>ATTACK OPERATOR</i>	58
5.6. <i>WEATHER REGIONS OPERATOR</i>	60
5.7. <i>PATH RELINK OPERATOR</i>	61
PROBLEM INSTANCE	63
6.1. <i>SIMULATION APPROACH</i>	63
EMPIRICAL SIMULATION RESULTS	75
7.1. <i>SIMULATION EXAMPLE</i>	75
EMPIRICAL RESULT ANALYSIS MODELS	81
8.1. <i>GENERAL EFFECTIVENESS MODEL</i>	81
8.2. <i>OPERATION ANALYSIS MODEL</i>	86
8.3. <i>LOGISTIC ANALYSIS MODEL</i>	88

8.4.	<i>GENERAL OBJECTIVE MODEL</i>	91
8.5.	<i>DETAILED OBJECTIVE ANALYSIS TABLES</i>	91
CONCLUSIONS AND FUTURE WORK		97
9.1.	<i>CONCLUSIONS</i>	97
9.2.	<i>FUTURE WORK</i>	98
BIBLIOGRAPHY		101

This Page Intentionally Left Blank

Introduction

1.1. Warfare Scope

Contemporary warfare paradigms and the complexity of operations introduce new challenges for the decision-making and operational planning processes and operating procedures of headquarters. Operational headquarters are often composite organizations made up of international military staff augmented by governmental and nongovernmental, national or international, organizations. This fact exacerbates new challenges introduced by the new generation of warfare, which makes the training of headquarters more and more complex. Emerging combat modeling and information technologies offer effective approaches that can tackle the complexities of this task. Therefore, computer-assisted simulation exercises aim to immerse the training audience in an environment as realistic as possible and to support exercise planning and control personnel in such a way that they can steer the exercise process toward the exercise objectives as effectively as possible. It has become the main tool for the headquarter training.

With aim on Researchers, military strategists and analysts, this thesis introduces the reader to Adaptive complex system modeling for realistic modern ground warfare simulation analysis based on evolutionary multi-objective meta-heuristic techniques.

The term warfare simulation can be used to cover a wide spectrum of activities, ranging from full scale field exercises to abstract computerized models that can proceed with little or no human involvement. This thesis focuses on the computerized models with the objective of being the most realistic a computer model can possibly be. The objective is to provide the analyst or strategist a series of data which he will analyze and derive the best option based on his expertise.

The military area is an area that benefits from the most detailed and realistic simulations, due to enormous resources needed in war, both material and human. If a battle

can be to some extent predicted before it happens the troops will be more effective in reaching their goals, there will be less casualties and the resources used will be optimized.

Figure 1 shows the difference between the Vietnam and Iraqui war regarding the number of deaths.

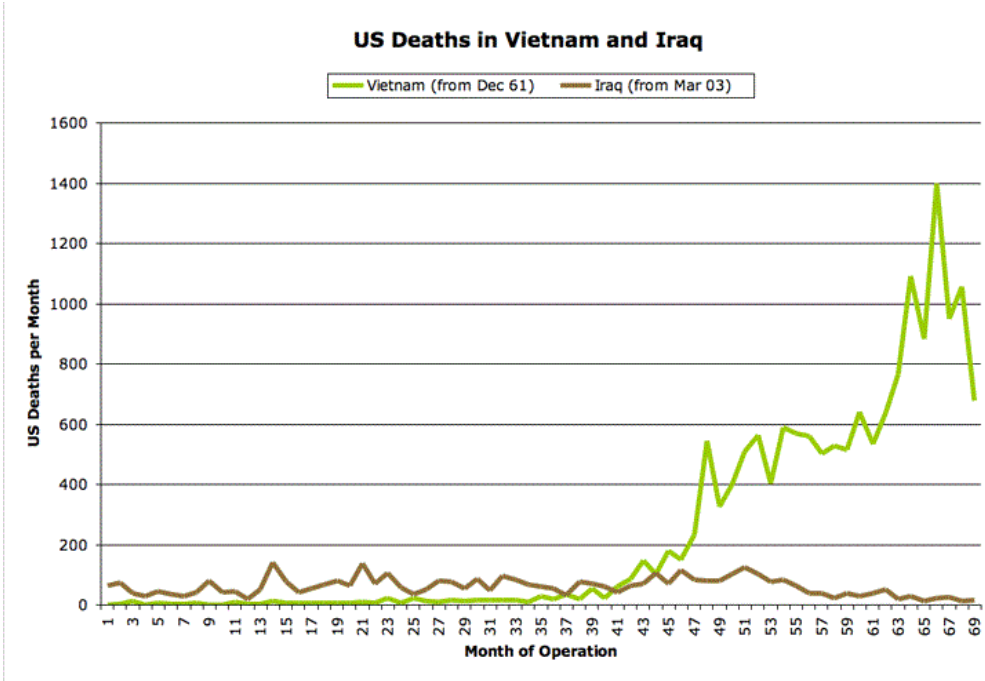


Fig. 1. US Deaths in Vietnam and Iraq [37]

The main difference between these figures is due to intelligence and sophistication in the battle field which reduced dramatically the number of deaths. So, as simulations begin to become more and more realistic this figures tend to low even more as the parts involved in the conflict go better prepared to combat with much more intelligence than ever before.

Due to the overwhelming nature of war planning, this thesis will focus on ground warfare, and will not simulate supply, air warfare or marine operations. It will however take into account the nature of the terrain the battle will evolve and the obstacles on site. In order for this simulation to be the most realistic, the most reliable data must be provided. Also, the approach used in this thesis is difficult to be used in games due to its processing complexity and computing demands.

As with all subjects in the warfare paradigm, this thesis might bring some ethical considerations because it will be widely spread over the internet and other mediums. Anyone can read it, make use of it and even expand the model presented. Following this reasoning, some terrorists, people or groups, might make use of it to plan attacks or learn how to think like military corps. However, this is not the use intended for this thesis.

DISCLAIMER: We are not responsible for, and expressly disclaim all liability for, damages of any kind arising out of use, reference to, or reliance on any information within this thesis. Some of the content found in this thesis may be offensive to some people. We do not have any affiliation with any future, present, or past political parties, military organization, or religious orders.

To solve the problem presented, we are going to propose a meta-heuristic which will make use of known algorithms and meta-heuristics, such as, evolutionary models, A-star pathfinder, KMeans clustering algorithms, among others. Then, we will propose an analysis framework, in order to simplify the analysis of the resulting data.

1.2. Thesis Structure

This thesis is divided into eight sections. First, it will begin by presenting some models already being used by some military corps. The second section will demonstrate the approach used to solve the problem; it will detail the problem complexity and multi-objective optimization. Then, the third section will introduce the background canonical models used, such as, the evolutionary, A-star and KMeans clustering algorithms. Next, the meta-heuristic created as the proposed solution for this problem will be detailed and decomposed into operators. In the fifth section the problem instance will be detailed, this thesis will use the Gulf War of 1991 as the instance to solve. After introducing the instance of the problem, in the seventh section, an example of a solution is given with the resulting maps. Then, in the seventh section, the results achieved will be presented, discussed and an analysis model will be presented. Finally, in the final section, some conclusions will be drawn and some future work will be proposed as continuity for the approach proposed by this thesis.

Related Work

Warfare simulation is still one of the areas which is highly confidential and most time hidden from public view. So, there isn't much information about the systems used by military corps other than names and simple descriptions, this section will list some simulations from two different categories, high-resolution constructive simulations and highly aggregated constructive simulations. The list is far from being exhaustive. The aim is to provide a set of examples to give insight in this area. As a final remark, the examples given in this section only focus on the simulation aspect and are not an analysis tool with well defined metrics.

2.1. High-Resolution Constructive Simulations

High-resolution constructive simulations are typically for tactical levels starting from a single troop up to several brigades. The terrain, weather, and entities are simulated detailed in these models. Each weapon, individual soldier, and combat system can be a simulated entity. Terrain modeling can be as detailed as centimeters, leaves of trees, and furniture in a room. Engagements are modeled typically between entities. Computations can be done for each single bullet shot by a troop.

As the level of detail increases, the more detailed data and the higher hardware capacities (i.e., memory and computational power) are required. Hardware capacities introduce limits on the size of simulation (i.e., the number of entities and the size of the simulation area). Therefore, there is a trade-off between the level of detail and the size of a simulation. As the hardware capacities increase, the limitations on the size of simulation disappears. For example, a typical play box for a high resolution constructive simulation system used to be 200 kilometers x 200 kilometers a decade ago. Nowadays, there are high resolution constructive simulation systems that can simulate as many as 50,000 entities in an area as large as 2000 kilometers x 2000 kilometers [39].

Apart from the hardware constraint, the other factors like the level of planning and the number of operators also affect the selection between a high-resolution or highly aggregated simulations. The higher the level of detail a model has, the more manpower is required to run the model because more details are needed in the commands.

Table 1. High-resolution constructive simulation systems.

Name/Service/Source Nation	Terrain	Play Box in Kilometers	Entities	Virtual	Automated
Virtual Battle Space 2 (VBS2)/Joint/Australia	Rapid terrain generation from DTEP, shape and imagery files. It can import three-dimensional (3-D) models (buildings, vegetation, etc) from 3DS or OpenFlight.	Up to 350x350 for DTED-1. Up to 120x120 for DTED-2. Up to 40x40 for DTED-3.	More than 1000 artificial intelligence (AI) entities. 120-200 human players.	Real-time rendering and highly accurate 3-D representations of objects, forces, and terrain.	Scripted semi-automated behavior.
Gefects Simulation (GESI) or Simulation four Rahmenubungen (SIRA)/Army with air and maritime entities/Germany	Uses TerraVista to read many data formats, such as digital terrain elevation data (DTED), digital height model (DHM), digital feature analysis data (DFAD), authoritative topographic cartographic information system (ATKIS), and geographic tagged image file format (GEOTIFF).	Up to 2000x2000.	Up to 32,000.	3-D view of terrain and entities.	Artificial intelligence to create autonomous forces from selected entities.
Korp-Rahmenmodell (KORA)/Army/Germany	Interface formats are DTEF, DFAD, and GEOTIFF.	Up to 1000x1000.	Not limited.	No.	Behavior agents.
SCIPIO/Army/France	DTED-1, Vector map 0 (VMAPO)/VMAPI, Geo-referenced maps or photos.	No limitations. Currently used in exercises with a play box of 2000x2000.	No hard limitations. Several thousand entities or units.	No.	Semi-automated forces.
CATS-TCT/Army/Sweden	DTED, VMAP, Geo-referenced maps, or photos.	No limitations. Play box enough to cover a brigade level exercise.	No hard limitations. As many entities as can be in a brigade.	3-D viewer.	No.
DEHOS/Navy/Turkey	High resolution from VMAP and shape data.	5 million square nautical miles.	No hard limitations. More than 1000 naval entities.	No.	No.
Joint Conflict and Tactical Simulation (JCATS)/Joint/US	Terrain generation from DTED, shape and imagery files.	4000x4000.	No hard limitations, about 100,000 entities. Up to 10 sides.	No.	No.
One Semi-automated Forces (OneSAF)/Joint/US	Very high resolution (1/12,500). High-resolution buildings (elevator shaft, balcony, stair, etc.).	500x500.	Up to 25 sides. Entities up to brigade level. In high resolution. Up to 500 entities. In low resolution up to 5000 entities.	3-D viewer.	Enhanced semi-automated force behaviors.

2.2. *Highly Aggregated Constructive Simulations*

Examples for the highly aggregated constructive simulation systems are listed in table 2, which is again far from being exhaustive.

The major difference visible to users between high-resolution and highly aggregated simulation systems is the representation of the terrain and environment. In highly aggregated systems, the play box is tessellated with either hexagons or squares, and each of these hexagons or squares represents the following:

- Terrain characteristics (i.e., forest, ocean, desert, etc.)
- Mobility characteristics (i.e., good, bad, no mobility, etc.)
- Altitude or depth

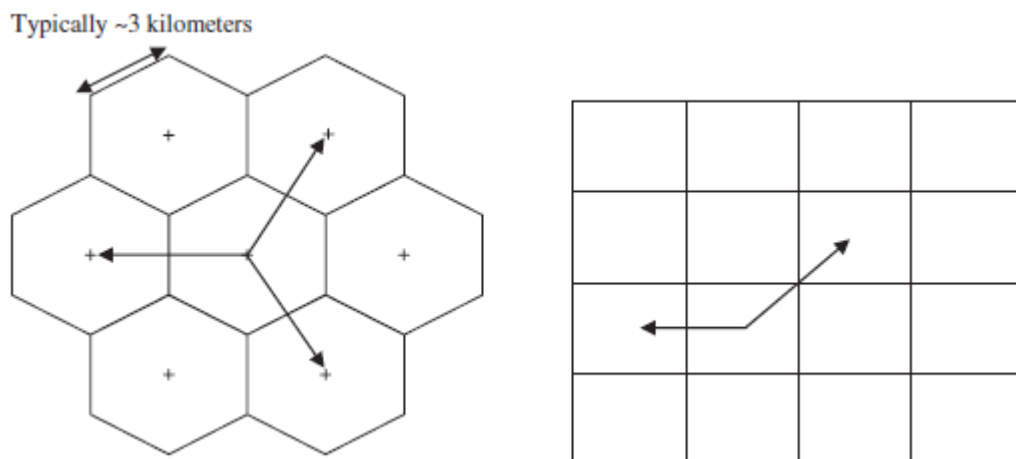


Fig. 2. Terrain representations in highly aggregated constructive simulations [38].

Moreover, the sides of these hexagons or squares are used to introduce obstacles like rivers, tank ditches, shores, minefields, and so on. For example, a river that can be an obstacle for the unit mobility must follow the edges of these geometric shapes. This approach may not look very realistic, and sometimes the results from simulation do not match with the maps and the data in C2 systems. For example, the real location of the river may be several kilometers different from a hexagon edge. Because the model uses the hexagon edge as an obstacle, a unit may stuck somewhere that does not look realistic.

Table 2. Highly aggregated constructive simulation systems.

Name/Service/Source Nation	Terrain	Play Box in Kilometers	Sides/Units	Automated
WAGRAM/Army/France	DTED-1, VMAP0/VMAP1, Geo-referenced maps, or photos.	No limitations. Currently used in exercises with a play box of 2000x2000.	No hard limitations. Several thousand entities or units.	Semi-automated forces.
ORQUE/Navy/France	DTED-1, VMAP0/VMAP1, Geo-referenced maps, or photos.	No limitations. Currently used in exercises with a play box of 2000x2000.	No hard limitations. Several thousand entities or units.	Semi-automated forces.
Simulations Modell fur ubungen Operativer Führung (SIMOF)/Army-Air/ Germany	Interface formats are DFAD and GEOTIFF.	Up to 2500x2000.	Not limited.	No.
Air Land Interactive Conflict Evaluation (ALICE)/ Airforce/Germany	DTED, DFAD, vector, and scanned maps.	4000x4000.	Not limited.	No.
CATS-TYR/Joint/Sweden	DTED, VMAP, Geo-referenced maps, or photos.	No limitations. Play box enough to cover a corps level exercise.	No hard limitations. As many entities as can be in a brigade.	No.
Joint Operational Command and Staff Training System (JOCASTS)/Joint/UK	Hexagons. The size of hexagons can be changed.	Corps size exercises.	No hard limitations.	No.
Joint Theater Level Simulation (JTLS)/Joint/US	Hexagons. The size of hexagons can be changed. The side length can be as short as 1 kilometer. However, when the side length is less than 3 kilometers, the performance of model depends on the scenario and number of units in the scenario.	4000x4000.	Up to 10 sides. As many as 10,000 units (no hard limit).	No.

Approach

This thesis will explore current, and introduce new techniques with the aim of improving the reliability of warfare computer simulation. Such techniques can bring significant advantages in numerous real life scenarios.

After reviewing the state-of-the-art, we reach the conclusion that none of solutions addressed this matter in an efficient way. Also, there is not much information about them as they are closed systems used by governments to simulate battlefield scenarios.

Accordingly, this thesis proposes a new simulation meta-heuristic based on an evolutionary approach with the use of evolutionary algorithms. This meta-heuristic will behave, much as possible, as the military would do in the battlefield. And mainly will be opened to everyone.

3.1. Design challenges and considerations

Warfare simulation, while theoretically a very attractive proposition, faces additional design and implementation hurdles when compared to other types of simulation. Thus, special care must be taken when designing a meta-heuristic which purpose is to simulate the battlefield.

3.1.1. Problem Complexity

Computational complexity is a branch of the theory of computation in theoretical computer science and mathematics that focuses on classifying computational problems according to their inherent difficulty. In this context, a computational problem is understood

to be a task that is in principle possible of being solved by a computer (which basically means that the problem can be stated by a set of mathematical instructions). Informally, a computational problem consists of problem instances and solutions to these problem instances. For example, primality testing is the problem of determining whether a given number is prime or not. The instances of this problem are natural numbers, and the solution to an instance is *yes* or *no* based on whether the number is prime or not.

A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used. The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying the amount of resources needed to solve them, such as time and storage. One of the roles of computational complexity theory is to determine the practical limits on what computers can and cannot do.

Closely related fields in theoretical computer science are analysis of algorithms and computability theory. A key distinction between analysis of algorithms and computational complexity theory is that the former is devoted to analyzing the amount of resources needed by a particular algorithm to solve a problem, whereas the latter asks a more general question about all possible algorithms that could be used to solve the same problem. More precisely, it tries to classify problems that can or cannot be solved with appropriately restricted resources. In turn, imposing restrictions on the available resources is what distinguishes computational complexity from computability theory: the latter theory asks what kind of problems can, in principle, be solved algorithmically.

➤ Complexity classes

- *What is a complexity class?*

Typically, a complexity class is defined by (1) a model of computation, (2) a resource (or collection of resources), and (3) a function known as the *complexity bound* for each resource [40].

The models used to define complexity classes fall into two main categories: (1) machine-based models, and (2) circuit-based models. Turing machines (TMs) and random-access machines (RAMs) are the two principal families of machine models. There are different kinds of (Turing) machines, such deterministic, non-deterministic, alternating, and oracle machines which are out of the scope of this thesis.

When there is the necessity to model real computations, deterministic machines and circuits are our closest links to reality. Then why consider the other kinds of machines? There are two main reasons.

The most potent reason comes from the computational problems whose complexity we are trying to understand. The most notorious examples are the hundreds of natural NP-complete problems [1]. To the extent that we understand anything about the complexity of these problems, it is because of the model of *non-deterministic* Turing machines. Non-deterministic machines do not model physical computation devices, but they do model real computational problems. There are many other examples where a particular model of computation has been introduced in order to capture some well-known computational problem in a complexity class. The second reason is related to the first. Our desire to understand real computational problems has forced upon us a repertoire of models of computation and resource bounds. In order to understand the relationships between these models and bounds, we combine and mix them and attempt to discover their relative power. Consider, for example, non-determinism. By considering the complements of languages accepted by non-deterministic machines, researchers were naturally led to the notion of alternating machines. When alternating machines and deterministic machines were compared, a surprising virtual identity of deterministic space and alternating time emerged.

Subsequently, alternation was found to be a useful way to model efficient parallel computation. This phenomenon, whereby models of computation are generalized and modified in order to clarify their relative complexity, has occurred often through the brief history of complexity theory, and has generated some of the most important new insights [41].

Other underlying principles in complexity theory emerge from the major theorems showing *relationships* between complexity classes. These theorems fall into two broad categories. Simulation theorems show that computations in one class can be simulated by computations that meet the defining resource bounds of another class. The containment of non-deterministic logarithmic space (NL) in polynomial time (P), and the equality of the class P with alternating logarithmic space, are simulation theorems. Separation theorems show that certain complexity classes are distinct.

Complexity theory currently has precious few of these. The main tool used in those separation theorems we have is called diagonalization. This ties in to the general feeling in computer science that *lower bounds are hard to prove*. Our current inability to separate many complexity classes from each other is perhaps the greatest challenge posed by computational complexity theory.

- *Time and Space Complexity Classes*

Fundamental time classes and fundamental space classes, given functions $t(n)$ and $s(n)$:

1. $DTIME[t(n)]$ is the class of languages decided by deterministic Turing machines of time complexity $t(n)$;
2. $NTIME[t(n)]$ is the class of languages decided by non-deterministic Turing machines of time complexity $t(n)$;
3. $DSPACE[s(n)]$ is the class of languages decided by deterministic Turing machines of space complexity $s(n)$;
4. $NSPACE[s(n)]$ is the class of languages decided by non-deterministic Turing machines of space complexity $s(n)$.

- *Canonical Complexity Classes*

Table 3. Canonical Complexity Classes

Complexity class	Time/Space class decomposition	Class Name
L	$DSPACE[\log n]$	Deterministic log space
NL	$NSPACE[\log n]$	Non-deterministic log space
P	$DTIME[nO(1)]$	Polynomial time
NP	$NTIME[nO(1)]$	Non-deterministic polynomial
$PSPACE$	$DSPACE[nO(1)]$	Polynomial space
E	$DTIME[2O(n)]$	
NE	$NTIME[2O(n)]$	
EXP	$DTIME[2nO(1)]$	Deterministic exponential time
$NEXP$	$NTIME[2nO(1)]$	Non-deterministic exponential
$EXSPACE$	$DSPACE[2nO(1)]$	Exponential space

NP is the set of all decision problems for which the instances where the answer is “yes” have efficiently verifiable proofs of the fact that the answer is indeed “yes”. More

precisely, these proofs have to be verifiable in polynomial time by a deterministic Turing machine. In an equivalent formal definition, NP is the set of decision problems where the “yes” instances can be recognized in polynomial time by a non-deterministic Turing machine. The equivalence of the two definitions follows from the fact that an algorithm on such a non-deterministic machine consists of two phases, the first of which consists of a guess about the solution which is generated in a non-deterministic way, while the second consists of a deterministic algorithm which verifies or rejects the guess as a valid solution to the problem [2].

The complexity class P is contained in NP, but NP contains many important problems, the hardest of which are called NP-complete problems, for which no polynomial-time algorithms are known. The most important open question in complexity theory, the P = NP problem, asks whether such algorithms actually exist for NP-complete, and by corollary, all NP problems. It is widely believed that this is not the case [42].

As described before, the complexity class NP can be defined in terms of NTIME as follows:

$$NP = NTIME(n^{O(1)}) = \bigcup_{k \in \mathbb{N}} NTIME(n^k) \quad (1)$$

The NP class has several sub classes, as presented in table 4.

Table 4. NP Sub Classes

NP sub classes
NP-Complete
NP-Hard
NP-easy
NP-equivalent
Co-NP
Co-NP-complete

- *NP-Hard*

NP-hard (non-deterministic polynomial-time hard), is a class of problems that are, informally, “at least as hard as the hardest problems in NP”. A problem H is NP-hard if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H (i.e., $L \leq_T H$). In other words, L can be solved in polynomial time by an oracle machine with an oracle for H . Informally, we can think of an algorithm that can call such an oracle machine as a subroutine for solving H , and solves L in polynomial time, if the subroutine call takes only one step to compute. NP-hard problems may be of any type: decision problems, search problems, or optimization problems.

As consequences of definition (note that these are claims, not definitions) [43]:

- Problem H is at least as hard as L , because H can be used to solve L ;
- Since L is NP-complete, and hence the hardest in class NP, also problem H is at least as hard as NP, but H does not have to be in NP and hence does not have to be a decision problem (even if it is a decision problem, it need not be in NP);
- Since NP-complete problems transform to each other by polynomial-time many-one reduction (also called polynomial transformation), all NP-complete problems can be solved in polynomial time by a reduction to H , thus all problems in NP reduce to H ; note, however, that this involves combining two different transformations: from NP-complete decision problems to NP-complete problem L by polynomial transformation, and from L to H by polynomial Turing reduction;
- If there is a polynomial algorithm for any NP-hard problem, then there are polynomial algorithms for all problems in NP, and hence $P = NP$;
- If $P \neq NP$, then NP-hard problems have no solutions in polynomial time, while $P = NP$ does not resolve whether the NP-hard problems can be solved in polynomial time;
- If an optimization problem H has an NP-complete decision version L , then H is NP-hard.

A common mistake is to think that the NP in *NP-hard* stands for *non-polynomial*. Although it is widely suspected that there are no polynomial-time algorithms for NP-hard

problems, this has never been proven. Moreover, the class NP also contains all problems which can be solved in polynomial time.

The traditional lines of attack for NP-hard problems are the Following:

- Devising algorithms for finding exact solutions (they will work reasonably fast only for relatively small problem sizes);
- Devising “suboptimal” or heuristic algorithms, i.e., algorithms that deliver either seemingly or probably good solutions, but which could not be proved to be optimal; Such algorithms can be: genetic algorithms, tabu search, ant algorithms, among others.
- Finding special cases for the problem (“sub problems”) for which either better or exact algorithms are available.

- *The Units Movement case*

In this case the goal is to minimize the total movement cost between the targets having to pass to each one of them; this case is similar to the traveling salesman problem. So, one wants the best path which corresponds to a sequence of targets. In order to enumerate the set of paths, first is chosen one target, then another one, and so on.

Number of different paths: If n is the number of targets, then, at each step k there can be chosen between $n - k$ targets.

So, *number of paths* = $n - 1 \times n - 2 \dots \times 1 = (n - 1)!$

Complexity: The complexity is $O((n - 1)!)$. As expected, this approach leads to an (hyper-)exponential algorithm. (the factorial function is hyper-exponential: $n! = O((n/2)^n)$)

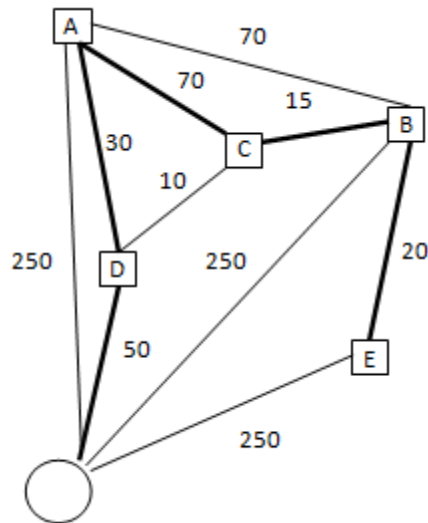


Fig. 3. Units case example

The best route is DACBE costing 185.

Complexity of verification: If a solution (i.e. a sequence of targets) is given, how long is it to compute its cost?

Only one path to explore, with $n - 1$ targets, so the complexity of verification is linear: $O(n)$.

So, this case is in NP and is hard because it is an optimization problem with the objective of finding the least-cost path through all the targets of a weighted map and the decision problem (“given the cost and a number x , decide whether there is a path cheaper than x ”) is NP-Complete. If there was a non-deterministic Turing machine, all the paths could be explored in one go, with linear complexity.

Following this reasoning, the complexity of this optimization involves at least one NP-hard problem making it NP-hard.

3.1.2. Multi-objective optimization

For multiple-objective problems, the objectives are generally conflicting, preventing simultaneous optimization of each objective. Many, or even most, real engineering problems

actually do have multiple objectives, i.e., minimize cost, maximize performance, maximize reliability, etc. These are difficult but realistic problems. EAs are a popular meta-heuristic that is particularly well-suited for this class of problems. Traditional EAs are customized to accommodate multi-objective problems by using specialized fitness functions and introducing methods to promote solution diversity.

There are two general approaches to multiple-objective optimization. One is to combine the individual objective functions into a single composite function or move all but one objective to the constraint set. In the former case, determination of a single objective is possible with methods such as utility theory, weighted sum method, etc., but the problem lies in the proper selection of the weights or utility functions to characterize the decision-maker's preferences.

In practice, it can be very difficult to precisely and accurately select these weights, even for someone familiar with the problem domain. Compounding this drawback is that scaling amongst objectives is needed and small perturbations in the weights can sometimes lead to quite different solutions. In the latter case, the problem is that to move objectives to the constraint set, a constraining value must be established for each of these former objectives.

This can be rather arbitrary. In both cases, an optimization method would return a single solution rather than a set of solutions that can be examined for trade-offs [44]. For this reason, decision-makers often prefer a set of good solutions considering the multiple objectives.

The second general approach is to determine an entire Pareto optimal solution set or a representative subset. A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s). Pareto optimal solution sets are often preferred to single solutions because they can be practical when considering real-life problems since the final solution of the decision-maker is always a trade-off. Pareto optimal sets can be of varied sizes, but the size of the Pareto set usually increases with the increase in the number of objectives.

- *Multi-objective optimization formulation*

Consider a decision-maker who wishes to optimize K objectives such that the objectives are non-commensurable and the decision-maker has no clear preference of the objectives relative to each other. Without loss of generality, all objectives are of the minimization type — a minimization type objective can be converted to a maximization type by multiplying negative one. A minimization multi-objective decision problem with K objectives is defined as follows:

Given an n -dimensional decision variable vector $x = \{x_1, \dots, x_n\}$ in the solution space X , find a vector x^* that minimizes a given set of K objective functions $z(x^*) = \{z_1(x^*), \dots, z_k(x^*)\}$. The solution space X is generally restricted by a series of constraints, such as $g_f(x^*) = b_j$ for $j = 1, \dots, m$, and bounds on the decision variables.

In many real-life problems, objectives under consideration conflict with each other. Hence, optimizing x with respect to a single objective often results in unacceptable results with respect to the other objectives. Therefore, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A reasonable solution to a multi-objective problem is to investigate a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution.

If all objective functions are for minimization, a feasible solution x is said to dominate another feasible solution $y(x \succ y)$, if and only if, $z_i(x) \leq z_i(y)$ for $i = 1, \dots, K$ and $z_j(x) < z_j(y)$ for least one objective function j . A solution is said to be Pareto optimal if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective. The set of all feasible non-dominated solutions in X is referred to as the Pareto optimal set, and for a given Pareto optimal set, the corresponding objective function values in the objective space are called the Pareto front. For many problems, the number of Pareto optimal solutions is enormous (perhaps infinite) [45].

The ultimate goal of a multi-objective optimization algorithm is to identify solutions in the Pareto optimal set. However, identifying the entire Pareto optimal set, for many multi-objective problems, is practically impossible due to its size. In addition, for many problems,

especially for combinatorial optimization problems, proof of solution optimality is computationally infeasible. Therefore, a practical approach to multi-objective optimization is to investigate a set of solutions (the best-known Pareto set) that represent the Pareto optimal set as well as possible. With these concerns in mind, a multi-objective optimization approach should achieve the following three conflicting goals [3]:

1. The best-known Pareto front should be as close as possible to the true Pareto front. Ideally, the best-known Pareto set should be a subset of the Pareto optimal set;
2. Solutions in the best-known Pareto set should be uniformly distributed and diverse over of the Pareto front in order to provide the decision-maker a true picture of trade-offs;
3. The best-known Pareto front should capture the whole spectrum of the Pareto front. This requires investigating solutions at the extreme ends of the objective function space.

For a given computational time limit, the first goal is best served by focusing (intensifying) the search on a particular region of the Pareto front. On the contrary, the second goal demands the search effort to be uniformly distributed over the Pareto front. The third goal aims at extending the Pareto front at both ends, exploring new extreme solutions.

Background Canonical Models

4.1. Supporting procedures and mechanisms

As stated before the problem which we are trying to solve with the meta-heuristic described later in this thesis is somehow complex and demanding, so there are several well known canonical models and algorithms used for solving the problem which are described in this section. This section aims that the reader understands all the terms and algorithm principles used throughout the thesis, although it is not exhaustive.

4.1.1. Evolutionary algorithms

➤ **Aims of this section**

The most important aim of this section is to describe what an Evolutionary Algorithm (EA) is. This description is deliberately based on a unifying view presenting a general scheme that forms the common basis of all Evolutionary Algorithm variants. The main components of EAs are discussed, explaining their role and related issues of terminology. Further on the general issues for EAs are discussed concerning their working. Finally, EAs are put into a broader context and their relation is explained with other global optimization techniques.

➤ **What is an Evolutionary Algorithm?**

As the history of the field suggests there are many different variants of EAs. The common underlying idea behind all these techniques is the same: given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and

this causes a rise in the fitness of the population. Given a quality function to be maximized we can randomly create a set of candidate solutions, i.e., elements of the function's domain, and apply the quality function as an abstract fitness measure - the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (the so-called parents) and results one or more new candidates (the children). Mutation is applied to one candidate and results in one new candidate.

Executing recombination and mutation leads to a set of new candidates (the offspring) that compete - based on their fitness (and possibly age) - with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached. In this process there are two fundamental forces that form the basis of evolutionary systems.

- Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty, while
- selection acts as a force pushing quality.

The combined application of variation and selection generally leads to improving fitness values in consecutive populations. It is easy (although somewhat misleading) to see such a process as if the evolution is optimizing, or at least “approximating”, by approaching optimal values closer and closer over its course. Alternatively, evolution it is often seen as a process of adaptation.

From this perspective, the fitness is not seen as an objective function to be optimized, but as an expression of environmental requirements. Matching these requirements more closely implies an increased viability, reflected in a higher number of offspring. The evolutionary process makes the population adapt to the environment better and better.

Note that many components of such an evolutionary process are stochastic. During selection fitter individuals have a higher chance to be selected than less fit ones, but typically even the weak individuals have a chance to become a parent or to survive. For recombination of individuals the choice of which pieces will be recombined is random. Similarly for mutation, the pieces that will be mutated within a candidate solution, and the new pieces

replacing them, are chosen randomly. The general scheme of an EA can be given in listing 1 in a pseudo-code fashion; figure 3 shows a diagram.

```

BEGIN
  INITIALIZE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  END REPEAT
END

```

Listing 1. Pseudo-code of an EA general scheme

It is easy to see that this scheme falls in the category of generate-and-test algorithms. The evaluation (fitness) function represents a heuristic estimation of solution quality and the search process is driven by the variation and the selection operators. EAs possess a number of features that can help to position them within the family of generate-and-test methods:

- EAs are population based, i.e., they process a whole collection of candidate solutions simultaneously;
- EAs mostly use recombination to mix information of more candidate solutions into a new one;
- EAs are stochastic¹.

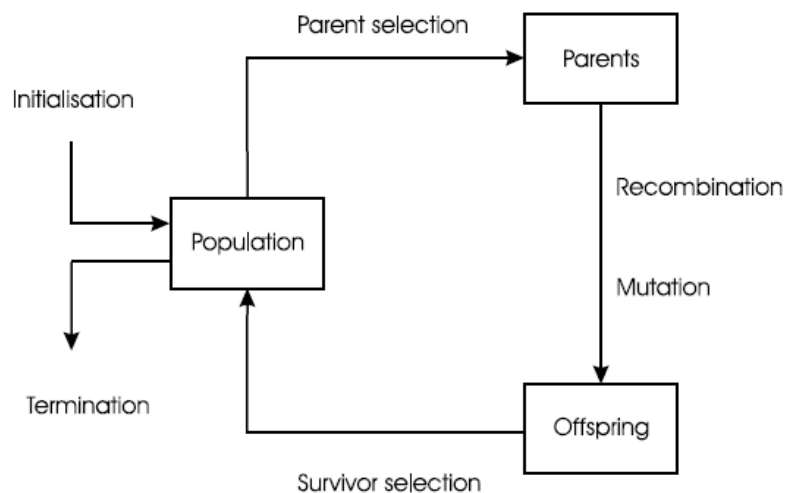


Fig. 4. The general scheme of an Evolutionary Algorithm as a flow-chart [46].

¹ Stochastic refers to systems whose behavior is intrinsically non-deterministic. A stochastic process is one whose behavior is non-deterministic, in that a system's subsequent state is determined both by the process's predictable actions and by a random element.

The various dialects of evolutionary computing that were mentioned previously all follow the general outlines in figure 4, and differ only in technical details. For instance, the representation of a candidate solution is often used to characterize different streams. Typically, the candidates are represented by (i.e., the data structure encoding a solution has the form of) strings over a finite alphabet in Genetic Algorithms (GA), real-valued vectors in Evolution Strategies (ES), finite state machines in classical Evolutionary Programming (EP) and trees in Genetic Programming (GP). These differences have a mainly historical origin. Technically, a given representation might be preferable over others if it matches the given problem better, that is, it makes the encoding of candidate solutions easier or more natural. For instance, for solving a satisfiability problem the straightforward choice is to use bit-strings of length n , where n is the number of logical variables, hence the appropriate EA would be a Genetic Algorithm.

For evolving a computer program that can play checkers, trees are well-suited (namely, the parse trees of the syntactic expressions forming the programs), thus a GP approach is likely. It is important to note that the recombination and mutation operators working on candidates must match the given representation. Thus for instance in GP the recombination operator works on trees, while in GAs it operates on strings. As opposed to variation operators, selection takes only the fitness information into account, hence it works independently from the actual representation. Differences in the commonly applied selection mechanisms in each stream are therefore rather a tradition than a technical necessity.

➤ **Components of Evolutionary Algorithms**

In this section is discussed EAs in detail. EAs have a number of components, procedures or operators that must be specified in order to define a particular EA. The most important components, indicated by italics in listing 1, are:

- representation (definition of individuals);
- evaluation function (or fitness function);
- population;
- parent selection mechanism;
- variation operators, recombination and mutation;

- survivor selection mechanism (replacement).

Each of these components must be specified in order to define a particular EA. Furthermore, to obtain a running algorithm the initialization procedure and a termination condition must be also defined.

➤ Representation (Definition of Individuals)

The first step in defining an EA is to link the “real world” to the “EA world”, that is to set up a bridge between the original problem context and the problem solving space where evolution will take place. Objects forming possible solutions within the original problem context are referred to as phenotypes, their encoding, the individuals within the EA, are called genotypes. The first design step is commonly called representation, as it amounts to specifying a mapping from the phenotypes onto a set of genotypes that are said to represent these phenotypes. For instance, given an optimization problem on integers, the given set of integers would form the set of phenotypes. Then one could decide to represent them by their binary code, hence 18 would be seen as a phenotype and 10010 as a genotype representing it. It is important to understand that the phenotype space can be very different from the genotype space, and that the whole evolutionary search takes place in the genotype space. A solution - a good phenotype - is obtained by decoding the best genotype after termination. To this end, it should hold that the (optimal) solution to the problem at hand - a phenotype - is represented in the given genotype space.

The common EC terminology uses many synonyms for naming the elements of these two spaces. On the side of the original problem context, candidate solution, phenotype, and individual are used to denote points of the space of possible solutions. This space itself is commonly called the phenotype space. On the side of the EA, genotype, chromosome, and again individual can be used for points in the space where the evolutionary search will actually take place. This space is often termed the genotype space. Also for the elements of individuals there are many synonymous terms. A place-holder is commonly called a variable, a locus (plural: loci), a position, or - in a biology oriented terminology - a gene. An object on such a place can be called a value or an allele.

It should be noted that the word “representation” is used in two slightly different ways. Sometimes it stands for the mapping from the phenotype to the genotype space. In this sense it is synonymous with encoding, i.e., one could mention binary representation or binary encoding of candidate solutions [47]. The inverse mapping from genotypes to phenotypes is usually called decoding and it is required that the representation be invertible: to each genotype there has to be at most one corresponding phenotype. The word representation can also be used in a slightly different sense, where the emphasis is not on the mapping itself, but on the “data structure” of the genotype space. This interpretation is behind speaking about mutation operators for binary representation, for instance.

➤ Evaluation Function (Fitness Function)

The role of the evaluation function is to represent the requirements to adapt to. It forms the basis for selection, and thereby it facilitates improvements. More accurately, it defines what improvement means. From the problem solving perspective, it represents the task to solve in the evolutionary context. Technically, it is a function or procedure that assigns a quality measure to genotypes. Typically, this function is composed from a quality measure in the phenotype space and the inverse representation. To remain with the above example, if the goal was to maximize x^2 on integers, the fitness of the genotype 10010 could be defined as the square of its corresponding phenotype: $18^2 = 324$.

The evaluation function is commonly called the fitness function in EC. This might cause a counterintuitive terminology if the original problem requires minimization for fitness is usually associated with maximization. Mathematically, however, it is trivial to change minimization into maximization and vice versa.

Quite often, the original problem to be solved by an EA is an optimization problem. In this case the name objective function is often used in the original problem context and the evaluation (fitness) function can be identical to, or a simple transformation of, the given objective function.

➤ Population

The role of the population is to hold (the representation of) possible solutions. A population is a multiset² of genotypes [48]. The population forms the unit of evolution. Individuals are static objects not changing or adapting, it is the population that does. Given a representation, defining a population can be as simple as specifying how many individuals are in it, that is, setting the population size. In some sophisticated EAs a population has an additional spatial structure, with a distance measure or a neighborhood relation. In such cases the additional structure has to be defined as well to fully specify a population. As opposed to variation operators that act on the one or two parent individuals, the selection operators (parent selection and survivor selection) work at population level. In general, they take the whole current population into account and choices are always made relative to what we have. For instance, the best individual of the given population is chosen to seed the next generation, or the worst individual of the given population is chosen to be replaced by a new one. In almost all EA applications the population size is constant, not changing during the evolutionary search.

The diversity of a population is a measure of the number of different solutions present. No single measure for diversity exists, typically people might refer to the number of different fitness values present, the number of different phenotypes present, or the number of different genotypes. Other statistical measures, such as entropy, are also used. Note that only one fitness value does not necessarily imply only one phenotype is present, and in turn only one phenotype does not necessarily imply only one genotype. The reverse is however not true: one genotype implies only one phenotype and fitness value.

➤ Parent Selection Mechanism

The role of parent selection or mating selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation. An individual is a parent if it has been selected to undergo variation in order to create offspring. Together with the survivor selection mechanism, parent selection is responsible for pushing quality improvements. In EC, parent selection is typically probabilistic. Thus, high quality individuals get a higher chance to become parents than those

² A multiset is a set where multiple copies of an element are possible.

with low quality. Nevertheless, low quality individuals are often given a small, but positive chance, otherwise the whole search could become too greedy and get stuck in a local optimum.

➤ Variation Operators

The role of variation operators is to create new individuals from old ones. In the corresponding phenotype space this amounts to generating new candidate solutions. From the generate-and-test search perspective, variation operators perform the “generate” step. Variation operators in EC are divided into two types based on their arity³ [49].

- Mutation

A unary⁴ variation operator is commonly called mutation. It is applied to one genotype and delivers a (slightly) modified mutant, the child or offspring of it. A mutation operator is always stochastic: its output - the child - depends on the outcomes of a series of random choices⁵. It should be noted that an arbitrary unary operator is not necessarily seen as mutation. A problem specific heuristic operator acting on one individual could be termed as mutation for being unary. However, in general mutation is supposed to cause a random, unbiased change. For this reason it might be more appropriate not to call heuristic unary operators mutation. The role of mutation in EC is different in various EC-dialects, for instance in Genetic Programming it is often not used at all, in Genetic Algorithms it has traditionally been seen as a background operator to fill the gene pool with “fresh blood”, while in Evolutionary Programming it is the one and only variation operator doing the whole search work.

It is worth noting that variation operators form the evolutionary implementation of the elementary steps within the search space. Generating a child amounts to stepping to a new point in this space. From this perspective, mutation has a theoretical role too: it can guarantee that the space is connected. This is important since theorems stating that an EA will (given

³ The arity of an operator is the number of objects that it takes as inputs.

⁴ An operator is unary if it applies to one object as input.

⁵ Usually these will consist of using a pseudo-random number generator to generate a series of values from some given probability distribution. These can sometimes be referred as “random drawings”.

sufficient time) discover the global optimum of a given problem often rely on the property that each genotype representing a possible solution can be reached by the variation operators [55]. The simplest way to satisfy this condition is to allow the mutation operator to “jump” everywhere, for example, by allowing that any allele can be mutated into any other allele with a non-zero probability. However it should also be noted that many researchers feel these proofs have limited practical importance, and many implementations of EAs do not in fact possess this property.

- Recombination

A binary variation operator⁶ is called recombination or crossover. As the names indicate such operator merges information from two parent genotypes into one or two offspring genotypes. Similarly to mutation, recombination is a stochastic operator: the choice of what parts of each parent are combined, and the way these parts are combined, depends on random drawings.

Again, the role of recombination is different in EC dialects: in Genetic Programming it is often the only variation operator, in Genetic Algorithms it is seen as the main search operator, and in Evolutionary Programming it is never used. Recombination operators with a higher arity (using more than two parents) are mathematically possible and easy to implement, but have no biological equivalent. Perhaps this is why they are not commonly used, although several studies indicate that they have positive effects on the evolution [56].

The principal behind recombination is simple - that by mating two individuals with different but desirable features, we can produce an offspring which combines both of those features. This principal has a strong supporting case - it is one which has been successfully applied for millennia by breeders of plants and livestock, to produce species which give higher yields or have other desirable features. EAs create a number of offspring by random recombination, accept that some will have undesirable combinations of traits, most may be no better or worse than their parents, and hope that some have improved characteristics. Although the biology of the planet earth, (where with a very few exceptions lower organisms reproduce asexually, and higher organisms reproduce sexually), suggests that recombination

⁶ An operator is binary if it applies to two objects as input.

is the superior form of reproduction, recombination operators in EAs are usually applied probabilistically, that is, with an existing chance of not being performed.

It is important to note that variation operators are representation dependent. That is, for different representations different variation operators have to be defined. For example, if genotypes are bit-strings, then inverting a 0 to a 1 (1 to a 0) can be used as a mutation operator. However, if possible solutions are represented by tree-like structures another mutation operator is required.

➤ Survivor Selection Mechanism (Replacement)

The role of survivor selection or environmental selection is to distinguish among individuals based on their quality. In that it is similar to parent selection, but it is used in a different stage of the evolutionary cycle. The survivor selection mechanism is called after having created the offspring of the selected parents. As mentioned in the “Population” section, in EC the population size is (almost always) constant, thus a choice has to be made on which individuals will be allowed in the next generation. This decision is usually based on their fitness values, favoring those with higher quality, although the concept of age is also frequently used. As opposed to parent selection which is typically stochastic, survivor selection is often deterministic, for instance ranking the unified multiset of parents and offspring and selecting the top segment (fitness biased), or selecting only from the offspring (age-biased).

Survivor selection is also often called replacement or replacement strategy. In many cases the two terms can be used interchangeably. The choice between the two is thus often arbitrary. A good reason to use the name survivor selection is to keep terminology consistent: step 1 and step 5 in Figure 4 are both named selection, distinguished by an adjective. A preference for using replacement can be motivated by the skewed proportion of the number of individuals in the population and the number of newly created children. In particular, if the number of children is very small with respect to the population size, i.e., 2 children and a population of 100. In this case, the survivor selection step is as simple as to choose the two old individuals that are to be deleted to make place for the new ones. In other words, it is more efficient to declare that everybody survives unless deleted, and to choose whom to

replace. If the proportion is not skewed like this, i.e., 500 children made from a population of 100, then this is not an option, so using the term survivor selection is appropriate.

➤ Initialization

Initialization is kept simple in most EA applications: The first population is seeded by randomly generated individuals. In principle, problem specific heuristics can be used in this step aiming at an initial population with higher fitness. Whether this is worth the extra computational effort or not is very much depending on the application at hand. There are, however, some general observations concerning this issue based on the so-called anytime behavior of EAs.

➤ Termination Condition

As for a suitable termination condition we can distinguish two cases. If the problem has a known optimal fitness level, probably coming from a known optimum of the given objective function, then reaching this level (perhaps only with a given precision $\epsilon > 0$) should be used as stopping condition.

However, EAs are stochastic and mostly there are no guarantees to reach an optimum, hence this condition might never get satisfied and the algorithm may never stop. This requires that this condition is extended with one that certainly stops the algorithm. Commonly used options for this purpose are the following:

1. the maximally allowed CPU time elapses;
2. the total number of fitness evaluations reaches a given limit;
3. for a given period of time (i.e, for a number of generations or fitness evaluations), the fitness improvement remains under a threshold value;
4. the population diversity drops under a given threshold.

The actual termination criterion in such cases is a disjunction: optimum value hit or condition x satisfied. If the problem does not have a known optimum, then we need no disjunction, simply a condition from the above list or a similar one that is guaranteed to stop

the algorithm. Later on in the “Working of an evolutionary algorithm” section the issue of when to terminate an EA will be revisited.

➤ **Example Applications**

- The 8-Queens Problem [52]
- The Knapsack Problem [53 – 54]

➤ **Working of an Evolutionary Algorithm**

EAs have some rather general properties concerning their working. To illuminate how an EA typically works we assume a one dimensional objective function to be maximized. Figure 5 shows three stages of the evolutionary search, exhibiting how the individuals are distributed in the beginning, somewhere halfway and at the end of the evolution. In the first phase, directly after initialization, the individuals are randomly spread over the whole search space, see figure 5, left. Already after a few generations this distribution changes: caused by selection and variation operators the population abandons low fitness regions and starts to “climb” the hills as shown in figure 5, middle. Yet later, (close to the end of the search, if the termination condition is set appropriately), the whole population is concentrated around a few peaks, where some of these peaks can be sub-optimal. In principle it is possible that the population “climbs the wrong hill” and all individuals are positioned around a local, but not global optimum. Although there is no universally accepted definition of what the terms mean, these distinct phases of search are often categorized in terms of exploration (the generation of new individuals in as-yet untested regions of the search space), and exploitation (the concentration of the search in the vicinity of known good solutions).

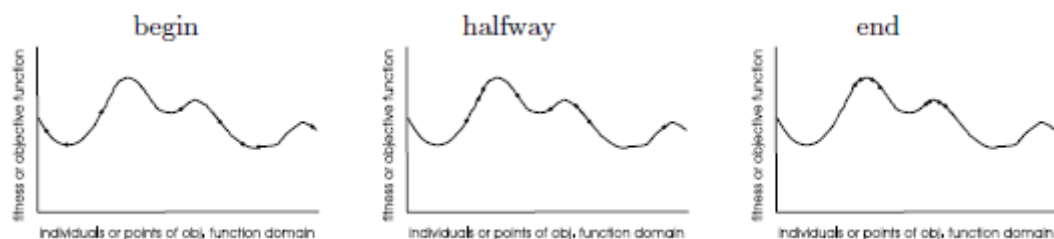


Fig. 5. Typical progress of an EA illustrated in terms of population distribution [47].

Evolutionary search processes are often referred to in terms of a trade-off between exploration and exploitation, with too much of the former leading to inefficient search, and too much of the latter leading to a propensity to focus the search too quickly for a good discussion of these issues). Premature convergence is the well-known effect of losing population diversity too quickly and getting trapped in a local optimum. This danger is generally present in EAs; although there are techniques to prevent it.

The other effect we want to illustrate is the anytime behavior of EAs. This is shown by plotting the development of the population's best fitness (objective function) value in time, see figure 6. This curve is characteristic for EAs, showing rapid progress in the beginning and flattening out later on. This is typical for many algorithms that work by iterative improvements on the initial solution(s). The name “any time” comes from the property that the search can be stopped at any time, the algorithm will have some solution, be it suboptimal.

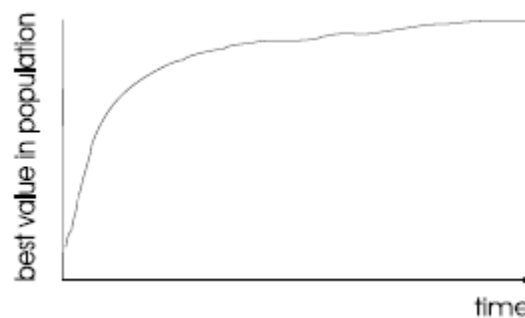


Fig. 6. Typical progress of an EA illustrated in terms of development of the best fitness (objective function to be maximized) value within population in time [47].

Based on this anytime curve some general observations can be made concerning initialization and the termination condition for EAs. As for initialization, recall the question from the “initialization” section whether it is worth to put extra computational efforts into applying some intelligent heuristics to seed the initial populations with better than random individuals. In general, it could be said that the typical progress curve of an evolutionary process makes it unnecessary. This is illustrated in figure 7. As the figure indicates, using heuristic initialization can start the evolutionary search with a better population. However, typically a few (in the figure: k) generations are enough to reach this level, making the worth of extra effort questionable.

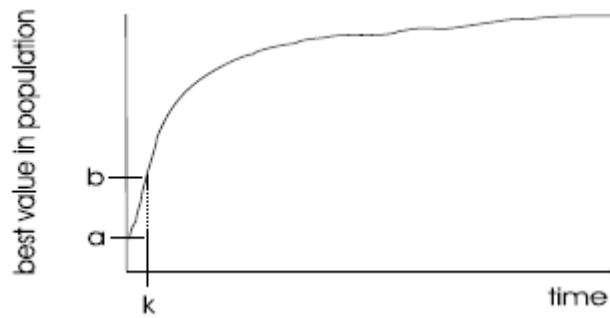


Fig. 7. Illustrating why heuristic initialization might not be worth. Level *a* show the best fitness in a randomly initialized population, level *b* belongs to heuristic initialization [47].

The anytime behavior also has some general indications regarding termination conditions of EAs. In figure 8 the run is divided into two equally long sections, the first and the second half. As the figure indicates, the progress in terms of fitness increase in the first half of the run, *X*, is significantly greater than the achievements in the second half, *Y*. This provides a general suggestion that it might not be worth to allow very long runs: due to the anytime behavior on EAs, efforts spent after a certain time (number of fitness evaluations) may not result in better solution quality.

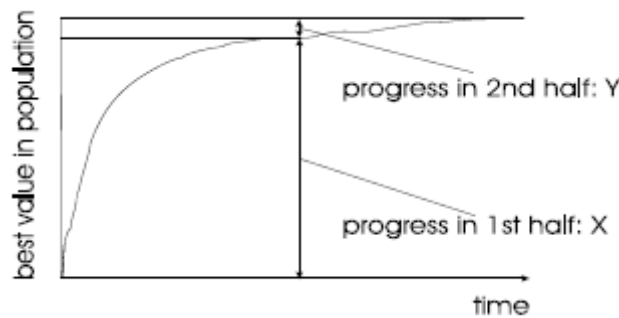


Fig. 8. Illustrating why long runs might not be worth. *X* shows the progress in terms of fitness increase in the first half of the run, *Y* belongs to the second half [47].

This review of EA behavior is closed with looking at EA performance from a global perspective. That is, rather than observing one run of the algorithm, consider the performance of EAs on a wide range of problems. Figure 9 shows the 80's view after Goldberg [58]. What the figure indicates is that robust problem solvers - as EAs are claimed to be- show a roughly even good performance over a wide range of problems. This performance pattern can be compared to random search and to algorithms tailored to a specific problem type. EAs clearly outperform random search. A problem tailored algorithm, however, performs much better

than an EA, but only on that type of problem where it was designed for. As we move away from this problem type to different problems, the problem specific algorithm quickly loses performance. In this sense, EAs and problem specific algorithms form two antagonistic extremes. This perception has played an important role in positioning EAs and stressing the difference between evolutionary and random search, but it gradually changed in the 90's based on new insights from practice as well as from theory. The contemporary view acknowledges the possibility to combine the two extremes into a hybrid algorithm. As for theoretical considerations, the No Free Lunch Theorem has shown that (under some conditions) no black-box algorithm can outperform random walk when averaged over “all” problems [57]. That is, showing the EA line always above that of random search is fundamentally incorrect.

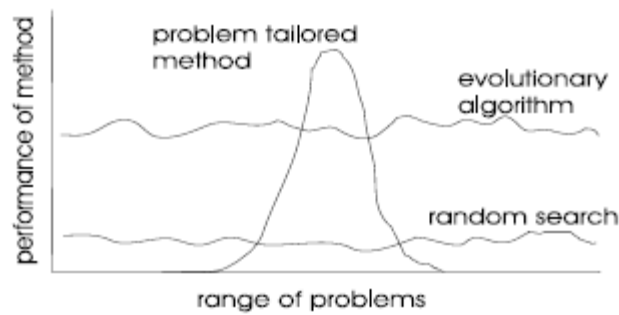


Fig. 9. 1980's view on EA performance after Goldberg [58][47].

➤ Evolutionary Computing and Global Optimization

There has been a steady increase in the complexity and size of problems that are desired to be solved by computing methods and EAs are often used for problem optimization. Of course EAs are not the only optimization technique known, and in this section is explained where EAs fall into the general class of optimization methods, and why they are of increasing interest.

In an ideal world, where the technology and algorithms were possessed that could provide a provably optimal solution to any problem that could suitably pose to the system. In fact, such algorithms exist: an exhaustive enumeration of all of the possible solutions to our problem is clearly such an algorithm. For many problems that can be expressed in a suitably mathematical formulation, much faster, exact techniques such as Branch and Bound Search are well known. However, despite the rapid progress in computing technology, and even if there is no halt to Moore's Law (which states that the available computing power doubles

every one and a half year), it is a sad fact of life that all too often the types of problems posed by users exceed in their demands the capacity of technology to answer them.

Decades of computer science research have taught that many “real world” problems can be reduced in their essence to well known abstract forms for which the number of potential solutions grows exponentially with the number of variables considered. For example many problems in transportation can be reduced to the well known “Travelling Sales Person” problem, i.e., given a list of destinations, to construct the shortest tour that visits each destination exactly once. If we have n destinations, with symmetric distances between them, the number of possible tours is given by $\frac{n!}{2} = n \times (n - 1) \times (n - 2) \times \dots \times 3$, which is exponential in n . While exact methods whose time complexity scales linearly (or at least polynomially) with the number of variables, exist for some of these problems, it is widely accepted that for many types of problems often encountered, no such algorithms exist. Thus despite the increase in computing power, beyond a certain size of problem the search for provably optimal solutions must be abandoned and other methods looked for finding good solutions.

The term Global Optimization will be used to refer to the process of attempting to find the solution x^* out of a set of possible solutions S which has the optimal value for some fitness function f . In other words, if trying to find the solution x^* such that $x \neq x^* \implies f(x^*) \geq f(x)$ (here a maximization problem is assumed, the inequality is simply reversed for minimization).

As noted above, a number of deterministic algorithms exist which if allowed to run to completion are guaranteed to find x^* . The simplest example is, of course, complete enumeration of all the solutions in S , which can take an exponentially long time as the number of variables increases. A variety of other techniques exist (collectively known as Box Decomposition) which are based on ordering the elements of S into some kind of tree and then reasoning about the quality of solutions in each branch in order to decide whether to investigate its elements. Although methods such as Branch and Bound can sometimes make very fast progress, in the worst case (due to searching in a suboptimal order) the time complexity of the algorithms is still the same as complete enumeration.

After exact methods, a class of search methods is found, known as heuristics which may be thought of as sets of rules for deciding which potential solution out of S should next be generated and tested. For some randomized heuristics, such as Simulated Annealing [59 – 60] and (certain variants of) EAs, convergence proofs do in fact exist, i.e., they are guaranteed to find x^* . Unfortunately these algorithms are fairly weak, in the sense that they will not identify x^* as being globally optimal, rather as simply the best solution seen so far.

An important class of heuristics is based on the idea of using operators that impose some kind of structure onto the elements of S , such that each point x has associated with it a set of neighbors $N(x)$. In figure 10 the variables (traits) x and y were taken to be real valued, which imposes a natural structure on S .

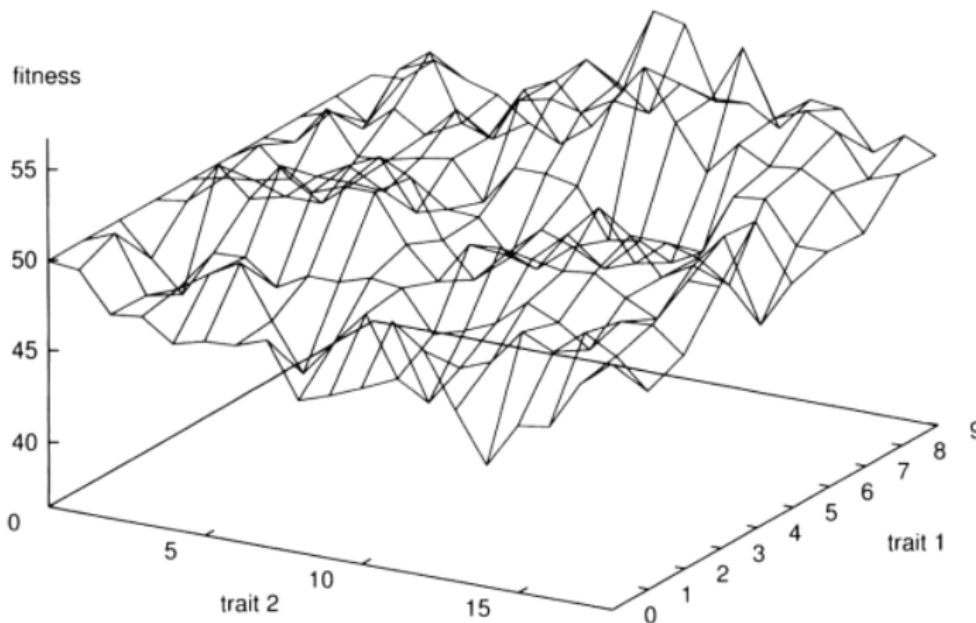


Fig. 10. Illustration of Wright's adaptive landscape with two traits [47].

The reader should note that for many types of problem where each variable takes one of a finite set of values (so-called Combinatorial Optimization) there are many possible neighborhood structures. As an example of how the landscape “seen” by a local search algorithm depends on its neighborhood structure, the reader might wish to consider what a chess board would look like if we re-ordered it so that squares which are possible next moves for a knight are adjacent to each other. Note that by its definition, the global optimum, x^* will always be fitter than all of its neighbors under any neighborhood structure.

So-called Local Search algorithms [59] (and their many variants) work by taking a starting solution x , and then searching the candidate solutions in $N(x)$ for one x' that performs better than x . If such a solution exists, then this is accepted as the new incumbent solution and the search proceeds by examining the candidate solutions in $N(x')$. Eventually this process will lead to the identification of a local optimum: a solution which is superior to all those in its neighborhood. Such algorithms (often referred to as Hill Climbers for maximization problems) have been well studied over the decades, and have the advantage that they are often quick to identify a good solution to the problem (which is in fact sometimes all that is required in practical applications). However, the downside is that frequently problems will exhibit numerous local optima, some of which may be significantly worse than the global optimum, and no guarantees can be offered in the quality of solution found. A number of methods have been proposed to get around this problem by changing the search landscape, either by reordering it through a change of neighborhood function (i.e., Variable Neighborhood Search [61]) or by temporally assigning low fitness to already seen good solutions (i.e., Tabu Search). However the theoretical basis behind these algorithms is still very much in gestation.

There are a number of features of EAs which distinguish them from Local Search algorithms, relating principally to their use of a population. It is the population which provides the algorithm with a means of defining a non-uniform probability distribution function (p.d.f.) governing the generation of new points from S . This p.d.f. reflects possible interactions between points in the population, arising from the recombination of partial solutions from two (or more) members of the population (parents). This contrasts with the globally uniform distribution of blind random search, or the locally uniform distribution used by many other stochastic algorithms such as simulated annealing and various hill-climbing algorithms.

The ability of EAs to maintain a diverse set of points not only provides a means of escaping from one local optimum: it provides a means of coping with large and discontinuous search spaces, and if several copies of a solution can be maintained, provides a natural and robust way of dealing with problems where there is noise or uncertainty associated with the assignment of a fitness score to a candidate solution.

4.1.2. Multi-objective evolutionary algorithms

Being a population-based approach, EAs are well suited to solve multi-objective optimization problems [50]. A generic single-objective EA can be modified to find a set of multiple non-dominated solutions in a single run. The ability of EA to simultaneously search different regions of a solution space makes it possible to find a diverse set of solutions for difficult problems with non-convex, discontinuous, and multi-modal solutions spaces. The recombination operator of EAs may exploit structures of good solutions with respect to different objectives to create new non-dominated solutions in unexplored parts of the Pareto front. In addition, most multi-objective EAs do not require the user to prioritize, scale, or weigh objectives. Therefore, EAs have been the most popular heuristic approach to multi-objective design and optimization problems. Jones et al. [4] reported that 90% of the approaches to multi-objective optimization aimed to approximate the true Pareto front for the underlying problem. A majority of these used a meta-heuristic technique, and 70% of all meta-heuristics approaches were based on evolutionary approaches.

The first multi-objective EA, called vector evaluated EA (or VEGA), was proposed by Schaffer [5]. Afterwards, several multi-objective EAs were developed including:

- Multi-objective Genetic Algorithm (MOGA) [6];
- Niche Pareto Genetic Algorithm (NPGA) [7];
- Weight-based Genetic Algorithm (WBGA) [8];
- Random Weighted Genetic Algorithm (RWGA)[9];
- Non-dominated Sorting Genetic Algorithm (NSGA) [10];
- Strength Pareto Evolutionary Algorithm (SPEA) [11];
- improved SPEA (SPEA2) [12];
- Pareto-Archived Evolution Strategy (PAES) [13];
- Pareto Envelope-based Selection Algorithm (PESA) [14];
- Region-based Selection in Evolutionary Multi-objective Optimization (PESA-II) [15];
- Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [16];
- Multi-objective Evolutionary Algorithm (MEA) [17];
- Micro-GA [18];
- Rank-Density Based Genetic Algorithm (RDGA) [19];

- Dynamic Multi-objective Evolutionary Algorithm (DMOEA) [20].

Note that although there are many variations of multi-objective EA in the literature, these cited EA are well-known and credible algorithms that have been used in many applications and their performances were tested in several comparative studies. Several survey papers [1,11,21–27] have been published on evolutionary multi-objective optimization. Coello lists more than 2000 references in his website [28]. Generally, multi-objective GA differ based on their fitness assignment procedure, elitism, or diversification approaches. In table 5, highlights of the well-known multi-objective with their advantages and disadvantages are given. It is also important to note that although several of the state-of-the-art algorithms exist as cited above, many researchers that applied multi-objective EA to their problems have preferred to design their own customized algorithms by adapting strategies from various multi-objective EA. This observation is a motivation for introducing the components of multi-objective EA rather than focusing on several algorithms [51].

Table 5. List of well-known multi-objective EA

Algorithm	Fitness assignment	Diversity mechanism	Elitism	External population	Advantages	Disadvantages
VEGA [5]	Each subpopulation is evaluated with respect to a different objective	No	No	No	First MOGA Straightforward implementation	Tend converge to the extreme of each objective
MOGA [6]	Pareto ranking	Fitness sharing by niching	No	No	Simple extension of single objective GA	Usually slow convergence Problems related to niche size parameter
WBGA [8]	Weighted average of normalized objectives	Niching. Predefined weights	No	No	Simple extension of single objective GA	Difficulties in non-convex objective function space
NPGA [7]	No fitness assignment, tournament selection	Niche count as tiebreaker in tournament selection	No	No	Very simple selection process with tournament selection	Problems related to niche size parameter Extra parameter for tournament selection
RWGA [9]	Weighted average of normalized objectives	Randomly assigned weights	Yes	Yes	Efficient and easy implementation	Difficulties in non-convex objective function space
PESA [14]	No fitness assignment	Cell-based density	Pure elitist	Yes	Easy to implement. Computationally efficient	Performance depends on cell sizes Prior information needed about objective space
PAES [28]	Pareto dominance is used to replace a parent if offspring dominates	Cell-based density as tie breaker between offspring and parent	Yes	Yes	Random mutation hill climbing strategy Easy to implement Computationally efficient	Not a population based approach. Performance depends on cell sizes
NSGA [10]	Ranking based on non-domination sorting	Fitness sharing by niching	No	No	Fast convergence	Problems related to niche size parameter
NSGA-II [29]	Ranking based on non-domination sorting	Crowding distance	Yes	No	Single parameter (N) Well tested Efficient	Crowding distance. Works in objective space only

(Continued)

Algorithm	Fitness assignment	Diversity mechanism	Elitism	External population	Advantages	Disadvantages
SPEA [11]	Ranking based on the external archive of non-dominated solutions	Clustering to truncate external population	Yes	Yes	Well tested No parameter for clustering	Complex clustering for algorithm
SPEA-2 [12]	Strength of dominators	Density based on the k -th nearest neighbor	Yes	Yes	Improved SPEA Make sure extreme points are preserved	Computationally expensive fitness and density calculation
RDGA [19]	The problem reduced to bi-objective problem with solution rank and density as objectives	Forbidden region cellbased density	Yes	Yes	Dynamic cell update Robust with respect to the number of objectives	More difficult to implement than others
DMOEA [20]	Cell-based ranking	Forbidden region cellbased density	Yes (implicitly)	No	Includes efficient techniques to update cell densities Adaptive approaches to set GA parameters	More difficult to implement than others

After analyzing the components of several EA multi-objective algorithms, we reached the conclusion that they do not fit the problem instance which we are trying to solve in this thesis. This is due to fact that the problem instance is a special case of multi-objective optimization, named many-objective optimization. Many-objective optimization problems are those who have more than 2 or 3 objectives, and the problem depicted in this thesis is clearly a problem in which we can find dozens of concurrent objectives. So, some very known multi-objective evolutionary algorithms, such as, NSGA-II and SPEA are very difficult to apply to this problem [63].

4.1.3. A-star pathfinder algorithm

To find the safest and fastest path to the enemy while avoiding obstacles a A-star pathfinder algorithm variant is used. This algorithm returns the path to the enemy in a grid avoiding any obstacles in the way. The A-star is one of the algorithms with the purpose of finding a solution to a problem that involves state transitions, they are called path finding algorithms.

These algorithms have special impact in problems which have a vast number of solutions for each state or in problems with a large number of states until the target is reached.

The A-star algorithm finds, in each state, the following possible states, making an estimate of the remaining distance to the target (H), and chooses the one that provides the largest progress with the lowest cost to continue testing.

This algorithm implements this principle through the use of three lists:

1. An open list, which contains all the states that were reached but were not tested;
2. A closed list, which contains the evaluated states;
3. A successors list, built at each iteration with the following states of the element in test.

In short, the A-star algorithm works in this way:

- Add the initial state to the open list
- While the open list has elements and the list is not transferred to the closed list
 - Search for the element with the lowest total cost (F)
 - Move to closed list
 - Search for successors
 - For each successor
 - If it is an obstacle or is in the closed list, ignore
 - If not
 - If it is not present in the open list, set the parent as the current node and calculate F,G and H
 - If it is in the open list, calculate again G (Total cost for the current path) if it is less than the previous value replace the parent node, G and F value

The A-star algorithm, for its simplicity and bearing on the problem was nominated as a “help” agent to find the shortest and most reliable path to the target. This becomes strikingly useful in cases where the direction of the target is contrary to the direction that the units have to go to find it.

4.1.4. KMeans clustering algorithm

To use the A-star algorithm for finding the best path to the target, first there is the need to define a point where the unit or units are. When there is only one unit, the point is the unit itself, but when several units are involved, we need to find a point in the grid which is, as much as possible, the nearest to all the units, named centroid. For this we use the KMeans Clustering algorithm.

KMeans (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point we need to recalculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2, \quad (2)$$

where $\|x_i^{(j)} - c_j\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j , is an indicator of the distance of the n data points from their respective cluster centers.

The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids;
2. Assign each object to the group that has the closest centroid;
3. When all objects have been assigned, recalculate the positions of the K centroids;
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the KMeans algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The KMeans algorithm can be run multiple times to reduce this effect.

KMeans is a simple algorithm that has been adapted to many problem domains. As it will be demonstrated, it is a good candidate for extension to work with fuzzy feature vectors.

➤ *An example*

Suppose that there is n sample feature vectors x_1, x_2, \dots, x_n all from the same class, and is known that they fall into k compact clusters, $k < n$. Let m_i be the mean of the vectors in cluster i . If the clusters are well separated, a minimum-distance classifier can be used to separate them. That is, x is in cluster i if $\|x - m_i\|$ is the minimum of all the k distances. This suggests the following procedure for finding the k means:

- Make initial guesses for the means m_1, m_2, \dots, m_k
- Until there are no changes in any mean
 - Use the estimated means to classify the samples into clusters
 - For i from 1 to k
 - Replace m_i with the mean of all of the samples for cluster i
 - End For
- End Until

Listing 2. Procedure for finding the k means

In figure 11 is an example showing how the means m_1 and m_2 move into the centers of two clusters.

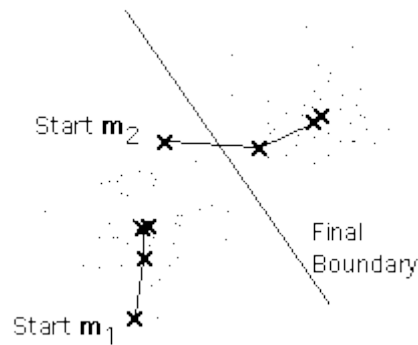


Fig. 11. KMeans working example [62].

➤ *Remarks*

This is a simple version of the KMeans procedure. It can be viewed as a greedy algorithm for partitioning the n samples into k clusters so as to minimize the sum of the squared distances to the cluster centers. It does have some weaknesses:

- The way to initialize the means was not specified. One popular way to start is to randomly choose k of the samples;
- The results produced depend on the initial values for the means, and it frequently happens that suboptimal partitions are found. The standard solution is to try a number of different starting points;
- It can happen that the set of samples closest to m_i is empty, so that m_i cannot be updated. This is an annoyance that must be handled in an implementation;
- The results depend on the metric used to measure $\|x - m_i\|$. A popular solution is to normalize each variable by its standard deviation, though this is not always desirable;
- The results depend on the value of k .

This last problem is particularly troublesome, since we often have no way of knowing how many clusters exist. In the example shown above, the same algorithm applied to the same data produces the following 3-means clustering. Is it better or worse than the 2-means clustering?

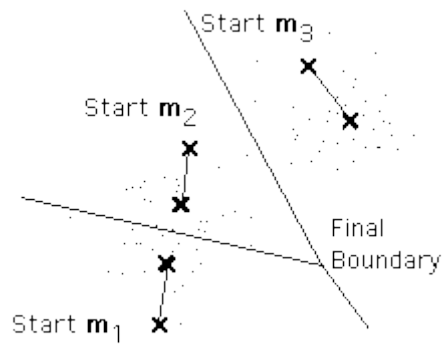


Fig. 12. KMeans working example 2 [62].

Unfortunately there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k classes and choose the best one according to a given criterion (for instance the Schwarz Criterion), but special care must be taken because increasing k results in smaller error function values by definition, but also an increasing risk of overfitting.

In this special case the k is one, as only one cluster of the dataset provided is needed, which will be the start and end for the A-star algorithm.

The MOEGWAO meta-heuristic

This section will focus on the MOEGWAO (Multi-objective Evolutionary Ground Warfare Adaptive Optimizer) meta-heuristic itself, decomposing it into operators for simplicity and ease of perception. In this section the user will see how the meta-heuristics and algorithms described in the section above will be integrated in the model proposed.

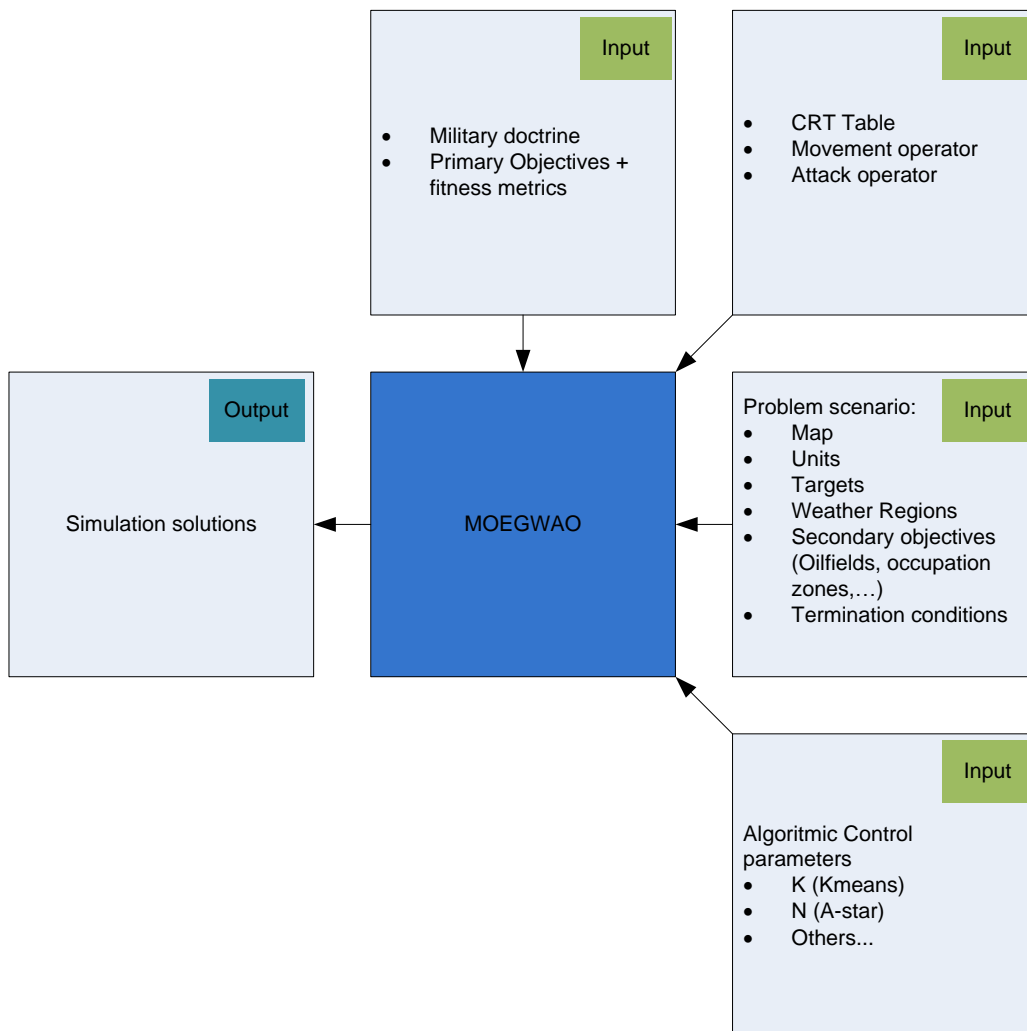


Fig. 13. MOEGWAO static model

Figure 13 shows the static model of MOEGWAO and depicts all the input parameters. There are 5 main groups of input parameters, the military doctrine and primary objectives for the simulation, then another group which consists of the CRT table, movement and attack operators adapted to the military specialist needs. Then the scenario specific parameters, with all the units and targets involved as well as the map, any secondary objectives and termination conditions. Finally, we input the algorithmic control parameters. Which are specific controls, such as the k in the KMeans clustering algorithm or other specific parameters for fine tuning. The logic overview of the meta-heuristic is detailed in the figure 14.

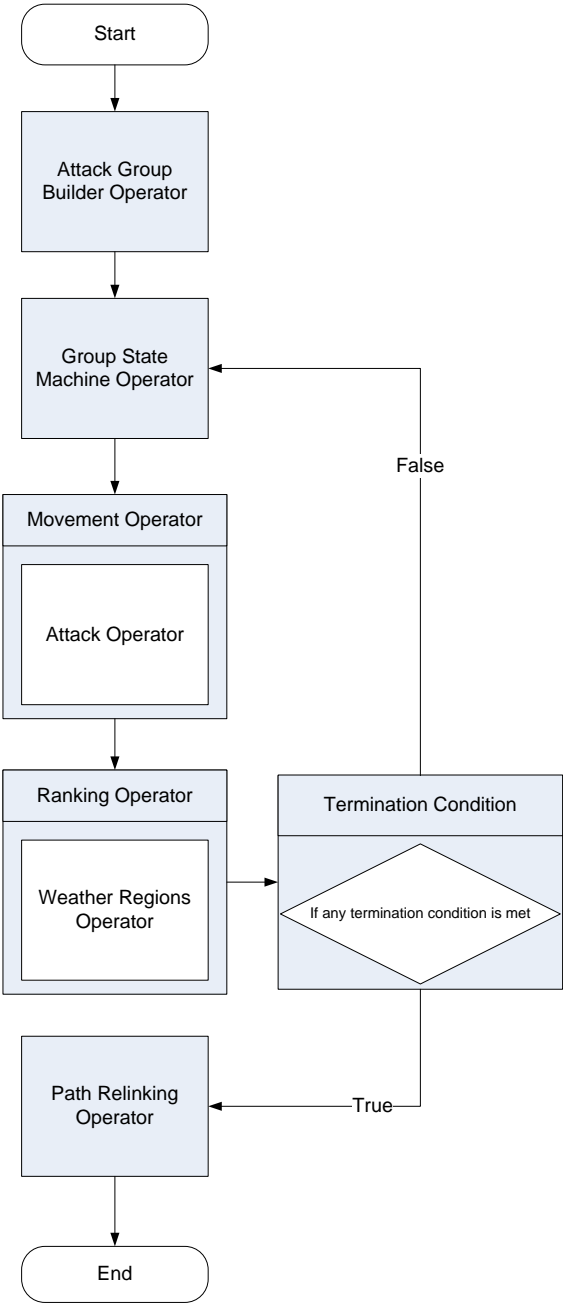


Fig. 14. MOEGWAO Meta-heuristic logic overview

The meta-heuristic begins by forming the attack groups, then the group state machine begins operating. It moves the attack groups throughout the map and simulates the combat between the units and the targets using the attack operator. Then the results are ranked using the ranking operator. This operator also uses the weather region operator in order to calculate the weather region fitness. This terminates a time slot of the simulation. In the end of the time slot, the termination condition is evaluated. If it returns true, the simulation ends but if it return false, then a new time slot is simulated. In the end of the whole simulation we apply the path relinking operator to gather more diversity in the results. And this ends the meta-heuristic operation.

5.1. Attack group builder operator

First, the attack groups are formed using the steps in listing 3. In this operator we use a control parameter, as the number of centroids or K in the KMeans clustering algorithm operator terminology is 1.

```

WHILE there are units available and targets not locked

    GET the available target with the lowest x and highest y position

    GET the available targets with visibility radius greater or equal to the distance
to the main target

    GET nearest unit from the main target

    WHILE units power less or equal to targets power and there are units available

        GET nearest available/eligible unit from main unit

    END WHILE

    IF units power less or equal to targets power THEN

        SET group state to 'MovingToTarget'

    END IF

    ELSE

        SET group state to 'Attacking'

    END ELSE

    GET targets centroid using KMeans

    GET units centroid using KMeans

    GET path from the units centroid to the targets centroid using the A-star variant

    GET random points from the path

    SET noise/entropy on the path

END WHILE

```

Listing 3. Pseudo-code of the attack builder operator

The formed group can be in one of these nine states:

➤ Idle

This state occurs when a group which was attacking destroyed all its targets. In this state a group can move to the “Help other group” state if there are other groups in the “Awaiting near target” state, move to “Attacking” or “Moving to the target” if there isn’t any groups in the “Awaiting near target” state but there are targets which are idle (not being attacked), if there aren’t any targets left it will find the nearest occupation zone and occupy it.

➤ Attacking

This group will move to its targets and destroy them.

➤ Moving to target

If in a certain attack group there are no more units available and the attack power is not enough to destroy the targets the group will be on “Moving to target” state. This group will move to the target but stay out of its visibility radius. When it reaches the target its state will change to “Awaiting near target”.

➤ Help other group

This state occurs when a group has destroyed its targets and there is other group in the “Awaiting near target” state. This group will move to that other group.

➤ Awaiting near target

In this state the group will stay on hold waiting for other group to back up. When another group in the “Help other group” state reaches it will change to “Attacking” state if the combined power of the two has enough power to destroy the targets or will stay in the “Awaiting near target” state if it hasn’t the attacking power needed to destroy the targets.

➤ Destroyed

This state occurs when a group engaged its targets and was destroyed in the battle. This group and its units will not be used again in the simulation.

➤ Disposed

This state occurs when a group in the “Help other group” state reaches its destination. This group and the group which it is backing up combine in one group, and the group which was on the “Help other group” state changes to “Disposed” which means it will be no longer considered.

➤ Moving to occupy

When a group which was on “Idle” is moving towards an occupation zone.

➤ Occupying

When a group which state was “Moving to occupy” reaches the occupation zone and stays there occupying the zone.

5.2. *Group state machine operator*

With groups formed, the simulation begins with steps depicted in Listing 4, 5 and 6. Here another control parameter is used, the termination condition can be defined according to the need of the military specialist who is using the simulator.

```
WHILE termination condition is not met DO
  FOR each attack group
    IF group state is idle
      IF there is a group in the awaiting near target state
        SET attack group state as help other group
        SET destination
      END IF
      ELSE IF exist targets not locked by other group
        SET target
        IF there is enough attacking power to destroy the target
          IF attacking power is more than required to destroy the target
            CREATE another group the remaining units in the idle state
          END IF
        ELSE
```

Listing 4. Pseudo-code of the group state machine operator

```

        SET group state Attacking
    END ELSE
END IF
ELSE
    SET group state Moving to target
END ELSE
END IF
ELSE
    GET the occupation zone with the least units
    SET attack group state Moving to occupy
END ELSE
END IF
IF group state is attacking
    IF group state is near the target
        Simulate combat using the CRT Table
        IF group wins the battle
            SET group state to Idle
        END IF
    ELSE
        Destroy group and its units
    END ELSE
END IF
ELSE
    Move to target
END ELSE
END IF
ELSE IF group state is Moving to target
    IF the group is near the targets visibility zone
        SET group state Awaiting near target
    END IF
ELSE
    Continue moving to target
END ELSE

```

Listing 5. Pseudo-code of the group state machine operator (continued)

```

END ELSE IF

ELSE IF group state is Awaiting near target
    Await for other group
END ELSE IF

IF group state is Help other group
    IF other group is near enough
        Combine the two groups
        IF there is enough power to destroy the target
            SET group state Attacking
        ELSE
            SET group state Awaiting near target
        END IF
    ELSE
        Keep moving to the other group
    END ELSE
END ELSE IF

IF group state is Moving to occupy
    IF group reached destination
        SET group state Occupying
    END IF
    ELSE
        Keep moving to destination
    END ELSE
END ELSE IF

ELSE IF group state is Occupying
    IF there are groups awaiting near target and no other groups
    available or there still exist targets not locked and no units available
        SET group state idle
    END IF
    ELSE
        Keep occupying
    END ELSE
END ELSE IF

END FOR

END WHILE

```

Listing 6. Pseudo-code of the group state machine operator (continued)

5.3. *Movement operator*

After the groups state is refreshed, the movement operator is used, if necessary using the steps in the listing 7.

```
FOR each group
    IF group is still moving
        GET all possible destinations for the group
    END IF
    ELSE
        GET actual position
    END ELSE
END FOR

Combine all possible movements between the groups until the maximum of movements defined
is reached

Create a solution for each combination

FOR each combination
    GET fitness values
END FOR

Rank solutions

Select the defined quantity of non dominated solutions

Select a few random dominated solutions, if they exist
```

Listing 7. Pseudo-code of the movement operator

5.4. *Ranking operator*

For ranking solutions the ranking operator is used as depicted in the listing 8.

```
FOR each solution

    Calculate enemy distance fitness (using Euclidean/Manhattan formula)

    Calculate waypoint fitness (distance to the nearest waypoint)

    Calculate weather region fitness

    Calculate supplies fitness (supplies used in this time slot)

    Calculate ammo fitness (ammo used on this time slot)

    Calculate oilfield fitness (oilfields conquered until this time slot)

    Calculate occupation zone fitness (occupation zones being occupied during this time
    slot with no targets in them)

END FOR

FOR each solution

    Check if there is another solution which is better in all fitness values

    IF exists one better solution

        SET solution as dominated

    END IF

    ELSE

        SET solution as non dominated

    END ELSE

END FOR
```

Listing 8. Pseudo-code of the ranking operator

The meta-heuristic makes use on many objectives simultaneously extensively, although the objectives being used are those described by the fitness values above, it is not limited to these values and more objectives can be added at any time.

5.5. Attack operator

The CRT table is a table used in combat simulation to simulate the combat between two parties. It contains the combat result taking into consideration the attack power of the group and a random number which is named die roll. In this model the CRT table is used as an input parameter, which makes part of the military doctrine.

The attack factor is calculated using the steps depicted in listing 9.

```
IF the group units power is higher than the targets power
    SET combat factor 1 as (units power / targets power)
    SET combat factor 2 as 1
END IF
ELSE IF the group units power is lower than the targets power
    SET combat factor 2 as (targets power / units power)
    SET combat factor 1 as 1
END ELSE IF
ELSE
    SET combat factor 1 and 2 as 1
END ELSE
```

Listing 9. Pseudo-code of the attack operator

If there was a group with attack power of 9 and a target with attack power of 1:

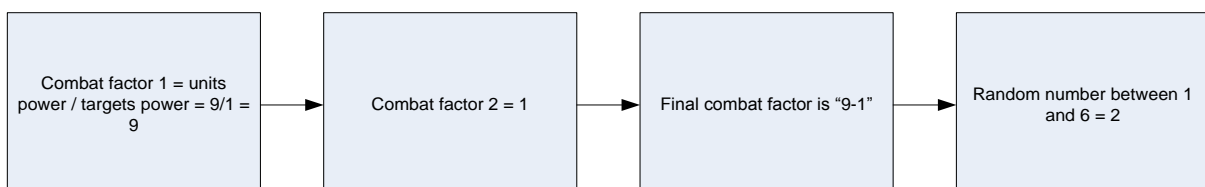


Fig. 15. Example of combat CRT mapping

Table 6. CRT Table [30]

		Combat factors										
		1-3	1-2	1-1	2-1	3-1	4-1	5-1	6-1	7-1	8-1	9-1
Die Roll	1	3/0	2/0	2/1	1/1	1/2	1/1	1/2	1/3	0/3	0/4	0/DS
	2	2/0	2/1	1/1	1/2	1/1	1/2	1/3	0/3	0/4	0/5	0/DS
	3	2/0	1/1	1/2	1/1	1/2	1/3	0/3	0/4	0/5	0/3	0/DS
	4	1/0	1/2	1/1	1/2	1/3	0/3	0/4	0/5	0/3	0/4	0/DS
	5	1/0	1/1	1/2	1/3	0/3	0/4	0/5	0/3	0/4	0/5	0/DS
	6	1/1	1/2	1/3	0/3	0/4	0/5	0/3	0/4	0/5	0/4	0/DS

Using table 6, the result of the combat would be 0/DS, which means the attacker loses 0 units and the defender surrenders (DS) meaning all targets are destroyed, if there was a value instead of DS it would be the targets that were destroyed. This is the table which will be used for the simulation.

Most of the CRT tables also include a retreat value, used for the defender to move a few steps away from battle, but that value will not be used in this simulation. The value will not be used because according to a well-known and respected military magazine named ‘Command’ [30]:

“Our analysis of high-speed, low-drag, 1990s, operational-level combat has led us to conclude such things (at the time/ space scale, anyway) don’t really happen any longer. That is, on the modern, ultra-high intensity battlefield, a brigade-sized unit’s destiny is pretty much determined by the way it enters the battlefield. Firepower has become so overwhelming in effect and precise in direction, units simply don’t have the opportunity to perform functional tactical/operational battlefield retreats as in days of yore. More than ever before, combat has become a matter only of the death of soldiers and the destruction of equipment. Direct, brutal and savage.”

5.6. Weather regions operator

The weather regions are areas in the map where the terrain or weather is different from most of the map, they can have adverse conditions or better conditions than in most of the map.

There are various types of weather regions as detailed in table 7.

Table 7. Weather Region Penalty with adapted values from [30]. Values adapted for use with the A-star algorithm.

Terrain Type	Mech Penalty	Non Mech Penalty
<i>Clear</i>	10	10
<i>Rough</i>	40	20
<i>Inundated</i>	P (Movement prohibited)	30
<i>Dunes</i>	20	20
<i>Wadi</i>	20	10
<i>Escarpment</i>	P	P
<i>River</i>	P	P
<i>Stream</i>	20	10
<i>Lake</i>	P	P
<i>Sea</i>	P	P
<i>Fair-weather</i>	10	10
<i>Road</i>	20	20
<i>Road (Paved)</i>	1	1
<i>Multi-lane highway</i>	1	1
<i>Village</i>	1	1
<i>Town</i>	1	1
<i>Oilfield</i>	1	1
<i>City</i>	1	1
<i>Airfield</i>	1	1
<i>Fortification</i>	30	20
<i>Heavy Fortification</i>	40	30

A lower value means that it is the most efficient path, but if it means much more distance than a path with a higher value, a path with higher value might be taken into consideration. These values are used by the A-star variant to find the best route to the targets while trying to avoid these areas, and will be used further on to calculate the weather region fitness.

5.7. *Path Relink Operator*

After getting the non-dominated results a path relink operator was used to get even more variation in the final solutions, this operator tries to create more solutions from the final solutions by combining, for example, the first time slot of solution A and the following time slots of solution B. This mechanism is very important to maximize variability in the population. However, this mechanism doesn't work backwards to recreate, for example the movements of a given group from time X to the beginning of the simulation.

Problem Instance

In this section we present the simulation background and history. For this simulation we use the 1991 Gulf War also known as Desert Storm. For understanding the whole problem instance used it is essential to study the factors that led to war, as well as the map of the region in which the war took place and the units deployed in conjunction with the units characterization, such as, armament. The atmospheric conditions and the way they affect units is also crucial, these are incorporated in the model as weather regions. Also, the main and secondary military objectives are important in order to calculate the fitness values for a solution.

6.1 Simulation Approach

The Gulf War was perhaps the most efficient war in American History, at least when considering the cost in American lives. It proved that U.S. technology and U.S. military doctrine is a potent force when applied to the world stage. Years after the war's end there are disagreements about whether the U.S. was justified in waging war against Iraq and over whether the war was prosecuted far enough.

➤ Factors that Led to the Iraqi Invasion of Kuwait [31]

Any discussion of the Gulf War must begin with the nation of Iraq. It once was a part of the Ottoman Empire, then a British protectorate, then a kingdom and finally a totalitarian state. Saddam Hussein became "President" in 1979 and maintained power through ruthless purges (including even members of his family). The country was also beset by internal strife. In the north the Kurds yearned for independence and in the south, the Shi'ites looked to Iran. The state and the army grew over time to consume most of the GNP. Today, the military alone takes up 35 percent of every dollar earned.

Saddam's expansion of the state's military apparatus was frightening to his neighbors. His investment in nuclear, chemical and biological weapons and corresponding delivery systems even prompted a 1981 attack by Israel in an effort to set back his weapons development program. With the expansion of his military, Saddam attempted to gain hegemony over the Persian Gulf Region. In the 1980's he fought a long, bitter struggle with Iran.

As a result of the war with Iran and the heavy investment in arms and training, the Iraqi military became the dominant force in the region. Led by the Republican Guard it could formidably challenge any of its neighbors. The price of keeping this force active was exorbitant. Iraq borrowed heavily from its oil producing neighbors. The debt coupled with continued investments brought on a 40 percent inflation rate and a stagnant standard of living.

Although Iraq had considerable oil reserves of its own, revenues were not sufficient to meet the demands of its creditors. This problem was exacerbated in 1990 when Kuwait and other oil states began to lower oil prices and increase production beyond agreed upon levels. Iraq was forced to follow suit or lose even more revenues. To make matters worse, Iraq suspected the Kuwaiti's were drilling diagonally from their side of the border to tap Iraqi oil reserves.

Thus Saddam Hussein was now in a precarious position, it was getting more and more difficult to maintain his military power (which he needed to keep down internal opposition as well as to keep up national prestige). He seemed there was an expeditious solution to his problems, a solution involving a foreign adventure.

Saddam Hussein found himself in a tight spot and a quick takeover of Kuwait, his neighbor to the south seemed like a good solution to his problems.

Kuwait was a small country that, like Iraq, had once been part of the Ottoman Empire, then a British Protectorate. When that small country had been granted its independence, its borders had been set in an arbitrary manner, the borders are not readily defensible and the population is not necessarily cohesive. The country was ruled by an Emir of the al-Sabah family.

Like much of the Persian Gulf region, most of the country's revenues derived from the oil industry. The population was small, about 1.9 million, and its military was not a factor in regional politics.

Kuwait was in many ways an irritant to Saddam Hussein in Iraq. Besides lowering oil prices (thus cutting into Iraqi oil revenues), Kuwait had committed the unforgivable sin of loaning Iraq considerable sums during the Iran/Iraq war. Iraq claimed to have saved the entire region from the Iranian steam roller in the 1980's and deserved special consideration amounting to renegotiating or even cancelling the debt. Kuwait refused.

During late July of 1990 Saddam built up his military forces on the border with Kuwait. At 1:00 a.m. on 02 August, three Iraqi divisions of the elite Republican Guard rolled over the border. Resistance was nearly non-existent. The Guard reached the outskirts of the capital, Kuwait City, a mere four and a half hours later. The frontal assault was supported by an airborne special forces division attack directly on Kuwait City itself.

Saddam proclaimed his annexation of Kuwait, built up his forces, and waited to see what the world would say and do about his *fait a compli*.

➤ *The Saudi Invitation*

The Middle East is a region of complex politics involving family ties between rulers, religious strife, socio-economic differences, and human personality. In spite of its often unstable nature, most of the world was shocked by the Iraqi invasion of Kuwait. Iraq justified the move primarily on the grounds that Kuwait was once a part of Iraq and should be again. Of course, it was also a power play by Iraq, an effort to annex some of the worlds richest oil fields. (Between Iraq and Kuwait Saddam now controlled about 20 percent of the worlds oil reserves.)

Once the Republican Guard had secured all of the strategic points in the country, it moved to the Kuwait/Saudi border. Of course, the Saudis were alarmed. It was not in their interests to have a beefed up Iraq to their north; the new build up, containing one of the elite forces in the region, was ominous. Iraq was sending more and more troops streaming into Kuwait, by August 6 there were nearly eleven combat divisions. Intelligence analysts at the time understood that Iraq had enough troops in the area to roll over Saudi Arabia nearly as easily as they had done to Kuwait.

King Fahd of Saudi Arabia recognized his situation as dire and immediately requested aid from his most powerful friend and ally, the United States [32]. President Bush promptly ordered the deployment of U.S. ground and air forces to Saudi territory. U.S. Navy ships were

also deployed to the region. So began the operation to defend Saudi Arabia that would be called "Desert Shield".

➤ *US Interests in the Gulf War [36]*

The response in the United States to Saddam Hussein's moves was first shock and then dismay. Strategists, statesmen and the general public quickly came to understand that the United States had significant interests in making certain that Saudi Arabia was not conquered by Saddam's juggernaut. Having rolled over Kuwait, Saddam already controlled over 20 percent of the world's oil reserves. Saudi Arabia contained an additional 20 percent. Since the world economy was primarily driven by fossil fuels, what Saddam could do with these resources could easily be imagined.

Besides economic factors affecting the daily lives of every American there were other considerations, perhaps even more weighty. Iraq, in its invasion of Kuwait had perpetrated many atrocities on the Kuwaiti people, from summary executions, to wholesale confiscation of movable property, to the torture and degradation of individuals. Such crimes could not be ignored, and Americans had every reason to expect that this kind of behavior would continue and even accelerate should Iraqi forces move into Saudi Arabia.

Further, Iraq had been vigorous in developing weapons of mass destruction. CIA and other intelligent experts estimated that the Iraqis were on the brink of developing a nuclear capability and likely had a biological weapon's capability. There was no question that they had chemical weapons. More ominously, they showed no compunction about using their chemical weapons. They had even done so on villages within their own boundaries in order to put down the Kurdish independence movement.

Economic sanctions had failed to keep Saddam from committing atrocities, they had failed to keep him from developing weapons of mass destruction, they had failed to keep him from invading Kuwait. A majority of Americans understood that military force was not only justified, but absolutely necessary.

➤ *Build Up of Forces*

Saddam Hussein's move into Iraq was so alarming that it galvanized most of the nations in the region to send troops to Saudi Arabia to help oppose the Iraqi build up. The United

Nations had looked askance at Iraqi behavior for some time. At this juncture, the United Nations felt compelled to condemn Iraq and to request an immediate withdrawal of troops from Kuwait. The United Nations would eventually authorize allied use of force in order to forcibly expel Iraq from Kuwait.

General H (Stormin') Norman Schwarzkopf was sent by President Bush, to Saudi Arabia to take command of US forces and defacto command of all the forces in the region [34]. (The Saudis insisted on at least the appearance of joint control.) Sent to the General, via land, sea and air was the best that the United States could provide including the XVIII Airborne Corps (24th Mechanized Infantry Division, 101st Airborne Division, and the 82nd Airborne Division), plus the 1st Marine Division. In time, the United States would send over 500,000 personnel to the region. Other allies, Britain, France, Egypt, Syria even the UAE sent contingents. The force took on an international complexion, with United States leadership.

The build-up was prosecuted as rapidly as possible. Schwarzkopf feared that the Iraqi's would launch an invasion before a proper defense could be constructed. Strategists hypothesize that if Hussein had ordered his troops into Saudi Arabia within a few days of his conquest of Kuwait, there would have been little to stop him from rolling into Riyadh. Saddam hesitated and this hesitation proved his undoing. For it was not until coalition forces had deployed that he decided to test their metal.

On 30 January 1991 the 15th Iraqi Mechanized Infantry Brigade attacked across the border a small town, Al-Khafji, in Saudi Arabia. The attack was swiftly repulsed; it served only to dissuade any wavering allies from any notions that Saddam would be willing to be satisfied with merely taking Kuwait. He would indeed aggrandize all his fellow Arabs.

Operation Desert Shield was meant to defend Saudi Arabia, but in January of 1991 President Bush, advised by Collin Powell and the Joint Chiefs of Staff determined to go on the offensive and take the war to the Iraqis.

➤ *Air War - Operation Desert Storm*

As is usual in modern war, the first objective of the allied force in Saudi Arabia was to gain air superiority. Air superiority gives a military force the ability to indiscriminately attack enemy targets, disrupt enemy lines of supply, to conduct recon, and, of course denies the enemy the ability to do all of these things himself.

The air campaign against Iraq was launched 16 January 1991, the day after the United Nations deadline for Iraqi withdrawal from Kuwait expired. Saddam was given every opportunity to conclude the stand off peacefully, but US/Iraqi talks in Geneva were inconclusive, at best [33].

The magnitude and the power of the air attack was a shock to all concerned. The initial attack swept away much of Iraq's ability to defend against further air assaults. Radar installations were attacked by helicopters, F-117's were sent to the Iraqi capital of Baghdad to destroy command and control centers, air bases and hangars were bombed. U.S. Navy bombers and Tomahawk missiles wreaked havoc on all aspects of Iraqi air defense. The air campaign was conducted not just by the United States, but the Saudi, British, French, Italian, as well as various Arab Air Forces.

The Allied air campaign was thorough and devastating. Realizing that traditional anti-air defense was futile the Iraqis took to psychological methods that included using human hostages as shields for prime targets. They placed their aircraft near ancient historic sites and holy places, knowing the allies would be reticent to attack where there might be significant "collateral damage".

In an effort to demonstrate their own air offensive capability, on 24 January the Iraqis attempted to mount a strike against the major Saudi oil refinery in Abqaiq. Two Mirage F-1 fighters laden with incendiary bombs and two MiG-23s (along as fighter cover) took off from bases in Iraq. They were spotted by US AWACs, and two Royal Saudi Air Force F-15s were sent to intercept. When the Saudis appeared the Iraqi MiGs turned tail, but the Mirages pressed on. Captain Iyad Al-Shamrani, one of the Saudi pilots maneuvered his jet behind the Mirages and shot down both aircraft. After this episode, the Iraqis made no more air efforts of their own, only sending most of their jets to Iran in hopes that they might someday get their air force back. (Iran never returned the jets.)

With Iraqi air defense effectively neutralized, the Allied Air Forces proceeded to pound the Iraqi divisions arrayed in Kuwait and Southern Iraq. Utilizing fuel bombs, cluster bombs, armor piercing guided bombs, missiles and various other ordinance, Allied forces degraded Iraqi ability to fight on the ground. Attacks by B-52 bombers were noted to be especially terrible; entire regiments, brigades and divisions were effectively crushed in a few minute air raid by these powerful though dated bombers.

By late February the Coalition forces were ready to kick off the ground campaign...

➤ *The Ground War - Operation Desert Storm*

There is much argument today about the Ground War phase of the Gulf War. Air advocates claim that the massive yet precise air war in fact defeated the Iraqi forces in Kuwait and that the ground campaign was merely "the great prisoner roundup". Conventional thought, however, recognizes that without forces on the ground it is impossible to hold territory, and engage the enemy on an individual level.

On 24 February 1991 the much feared Marine Divisions kicked off the ground campaign with a thrust into the heart of the Iraqi forces in central Kuwait [35]. The Saudi and Muslim Joint Forces - East attacked up the Kuwaiti coast line. Meanwhile the U.S. 18th Airborne Corps and the French 6th Armored Division, making good use of their high speed and mobility, rushed into Iraq on the far left.

These initial attacks rolled over Iraqi positions and on the 25th of February were followed up with the US VII Corps with the US 1st Infantry Division and the British 7th Armored Division attached.

In effect General Schwarzkopf had designed a strategy based on US doctrine which relied heavily on the flanking maneuver. (The flanking maneuver is a classic and reliable method of creating local superiority of power at a vulnerable point in the enemies line of battle.) Allied Forces occupied Iraqi front line forces while more mobile units encircled the enemy on the left, effectively cutting lines of supply and avenues of retreat. The movement proved to be highly effective and resistance by even battle hardened Iraqi units proved remarkably light.

The ground assault by the allies precipitated a general rout on the part of Iraqi forces positioned in Kuwait. There was basically only one highway out of Kuwait and that was the four lane desert highway that lead from Kuwait City to the Al Jahra' pass. As Iraqi resistance deteriorated the highway became jammed with every nature of vehicle laden with plunder from the Iraqi sack of Kuwait City. This highway was bombed, and thousands of fleeing Iraqis were killed and wounded.

Scenes of destruction of this "Highway of Death" were flashed by news services around the world. Eventually the mood in the Arab countries within the coalition became one of empathy for their brother Arabs on the highway - men they did not want to kill unnecessarily.

As coalition forces moved to completely cut off this last avenue of retreat, Allied leaders, including George Bush and Collin Powell determined that the Allied objective had been all but accomplished. The Iraqis had been turned out of their Kuwaiti conquest. On 28 February President Bush ordered the cessation of offensive military operations before the "Highway of Death" could be completely closed off. While the Iraqis and the Allies negotiated, the remaining Iraqi forces, including intact units of the elite Iraqi Republican Guard streamed out of Kuwait.

➤ *Historical Deployment*

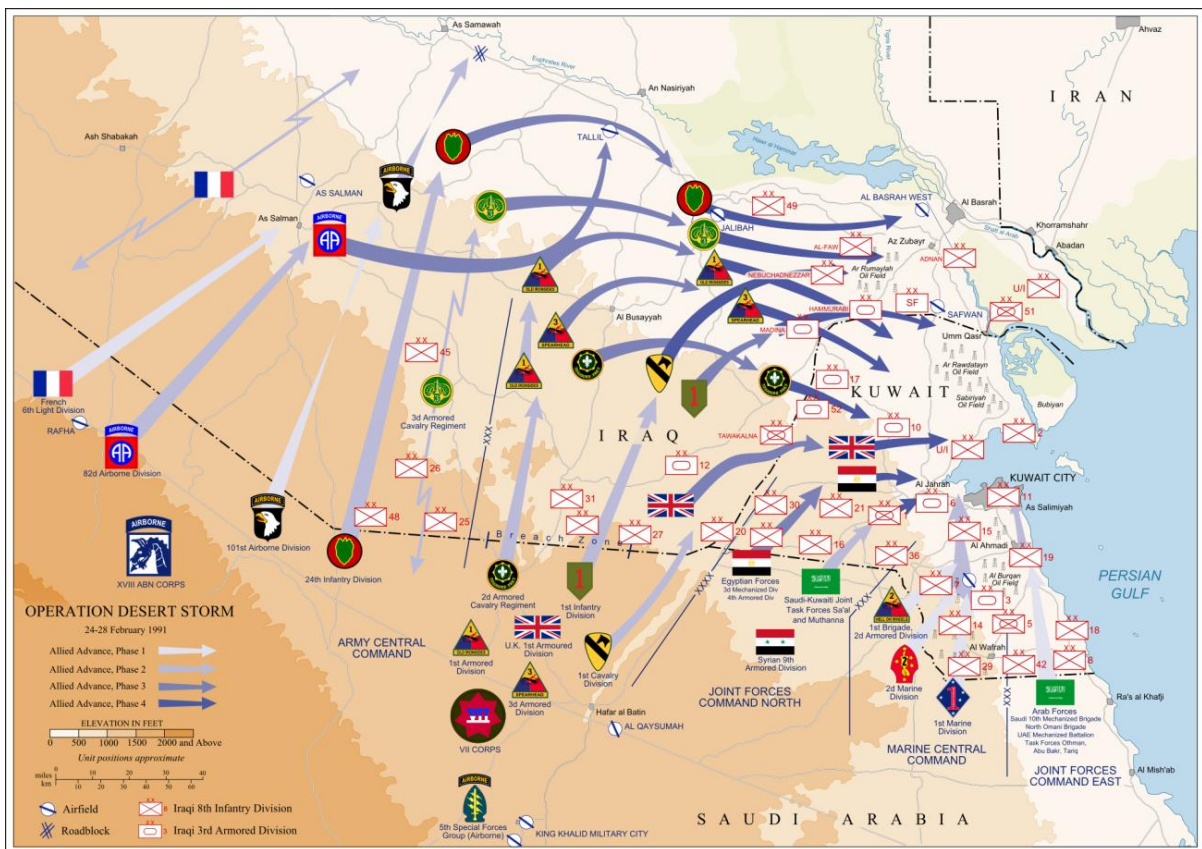


Fig. 16. Desert Storm historical deployment [31]

The results presented in the next section are based on 30 simulations per configuration (these configurations will be detailed further on this section). The map dimensions are 2448 x 1392 pixels, there are 76 units and 70 Targets to destroy. After getting the final solutions the dominated solutions are removed from the final set and then the path relink operator is applied to generate more diversity to the solutions. The combats are simulated using the CRT table described previously. The weather region penalties are those described previously. There are 25 Oilfields to occupy and 4 occupation zones. The main objectives of this simulation are those described in table 8.

Table 8. Main objectives to be analyzed

Objectives
Occupation zones occupied
Oilfields occupied
Time Slots (days) required for the simulation
Targets Destroyed
Targets Deserted
Units Destroyed
Ammo consumed
Supplies Used
Efficiency of the units movement

For easy understating of the maps used in the simulation example in the next section, the map legends are those detailed by the tables 9, 10 and 11 and also figure 17.

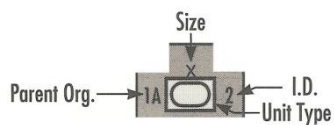


Fig. 17. Unit Legend

Table 9. Unit Sizes Legend

Unit Sizes	
XX	Division
X	Brigade
III	Regiment
II	Battalion

Table 10. Mechanized Unit Types Legend

Mechanized Unit Types	
	Armor or Tank
	Armored Cavalry
	Mech Infantry
	Motorized Infantry
	Motorized Marines
	Motorized Special Operations
	Wheeled Infantry
	Wheeled Airborne Infantry
	Wheeled Marines
	Wheeled Special Operations
	Self-Propelled Artillery
	Wheeled Artillery
	Armored Car
	Combined Arms
	Tracked Anti-Tank
	Attack Helicopters
	Motorized Anti-Tank

Table 11. Non-Mechanized Unit Types Legend

Non-Mechanized Unit Types	
	Infantry
	Mountain Infantry
	Airmobile Infantry
	Special Operations
	Marines

➤ *Initialization control parameter*

To create the graphs depicted in the next section were used 7 different configurations, which belong to the group of control parameters. These are referred as “configs” in the graphs. These “configs” are nothing more than the method used to originally create the first population in order for the simulation to begin, in the initialization step of the EA. These configurations begin by selecting the targets from the specified side of the map, as illustrated in the figure 18.

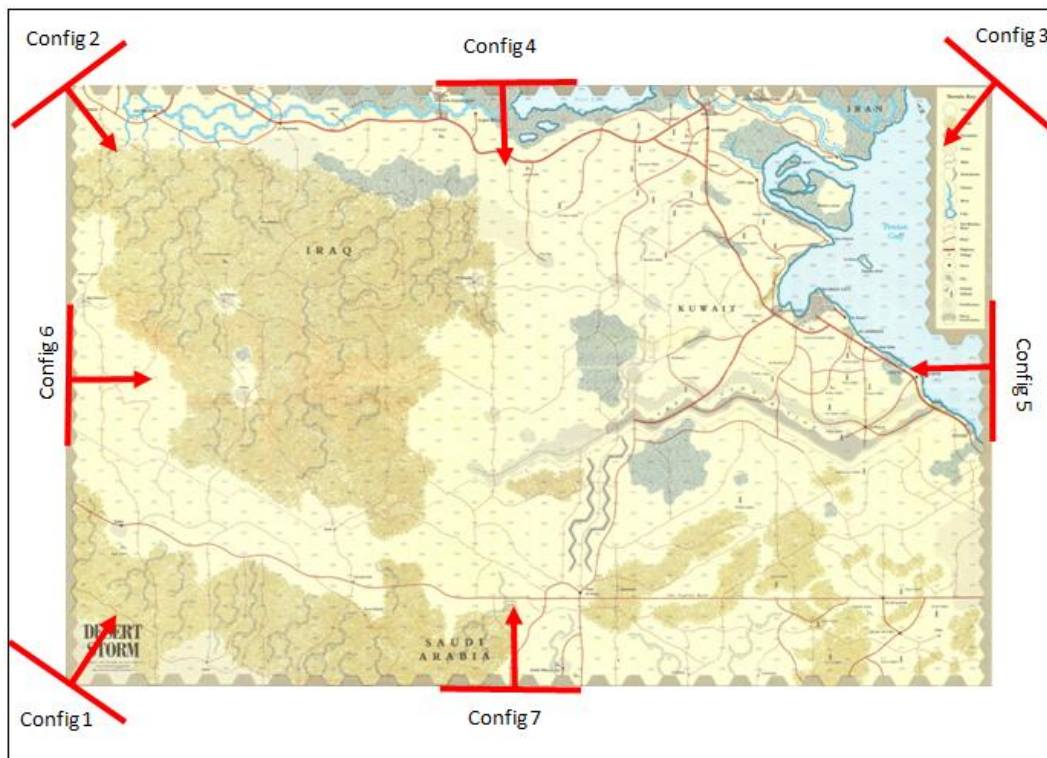


Fig. 18. Configurations Explanation Map

Empirical Simulation Results

This section will begin by giving an example of a solution in table 13, with maps showing the progress through time until the termination condition is met (all targets are destroyed). A solution is a complete simulation of the problem instance, with several time slots until the termination condition is met. The legend for the map is depicted in table 12. Then we will present some graphs and propose an analysis framework, in the next section, in order to analyze the operation of the meta-heuristic.

7.1. Simulation example

Table 12. Map Legend





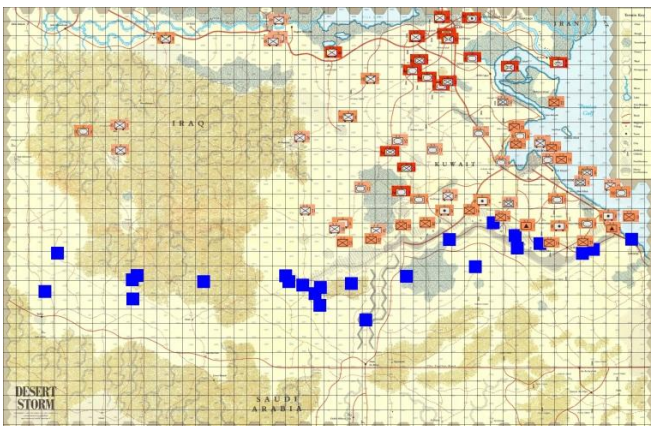
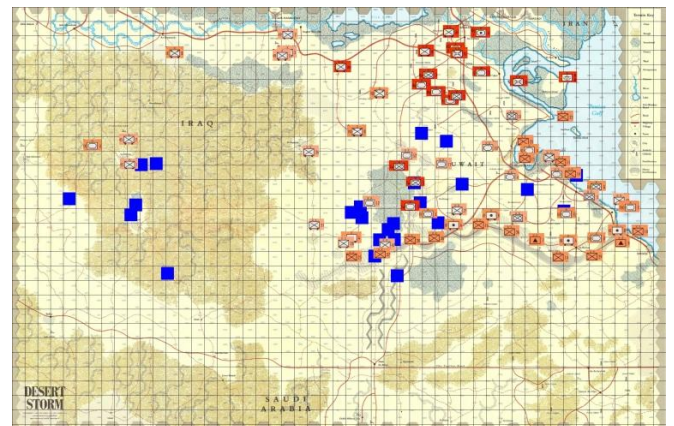
Map Legend	
	Occupied Zone
	Combat
	Target
	Attacking Group

Table 13. Simulation Example⁷

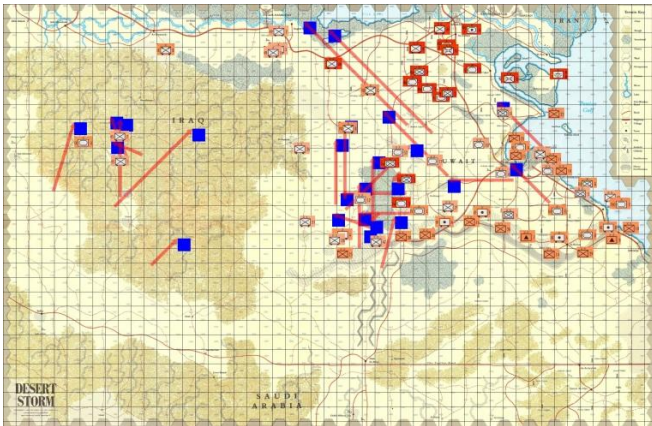


TS:0 – Initial deploy

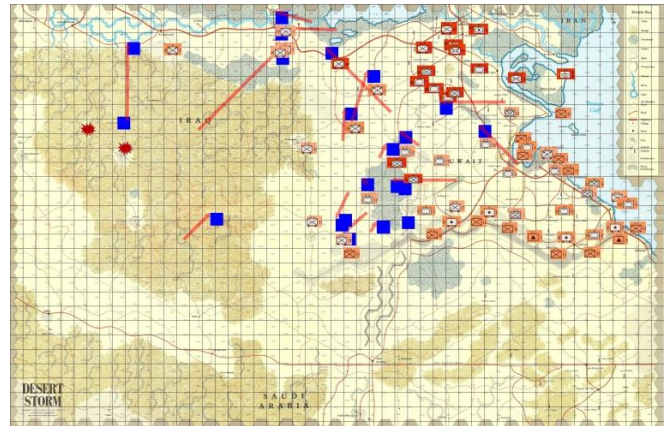


TS: 1 – UD:0 – TD:0 – TDes:0

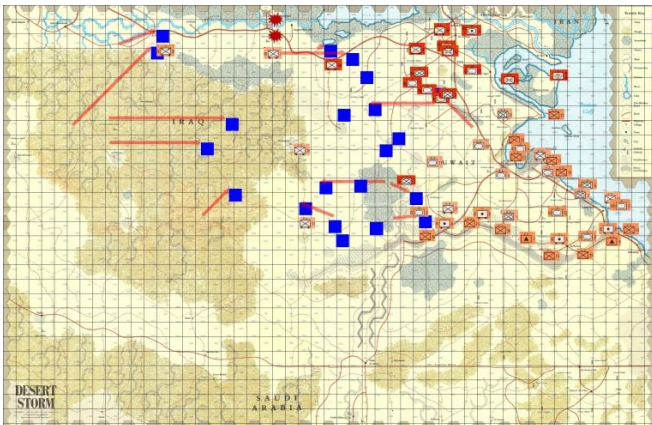
⁷ TS: Time slot, UD: Units Destroyed, TD: Targets Destroyed, TDes: Targets Deserted, OZ: Occupation Zones



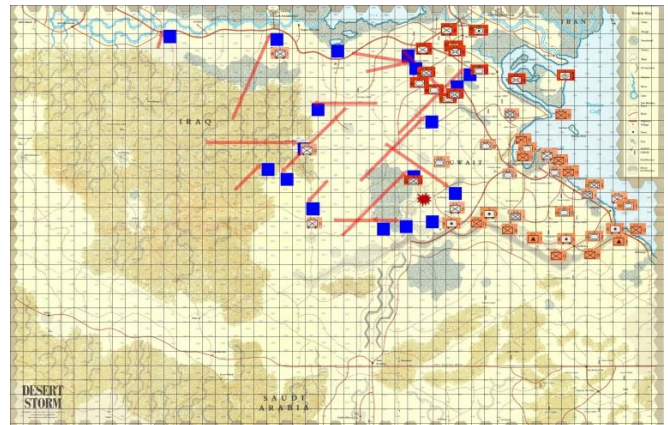
TS: 2 – UD:0 – TD:1 – TDes:0



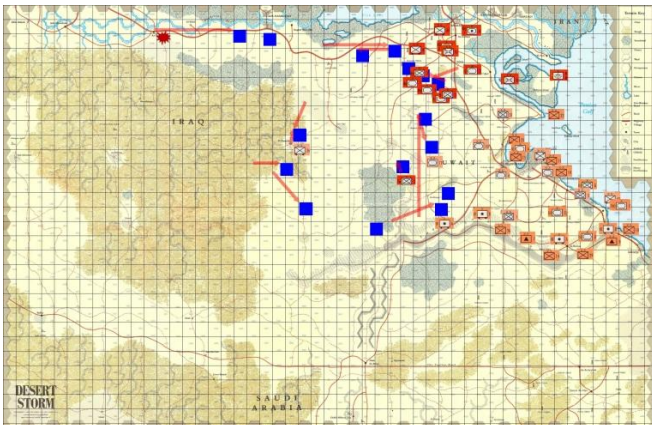
TS: 3 – UD:1 – TD:8 – TDes:0



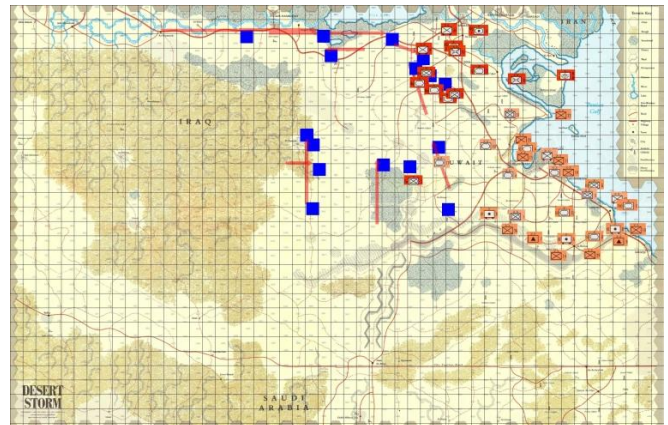
TS: 4 – UD:5 – TD:17 – TDes:0



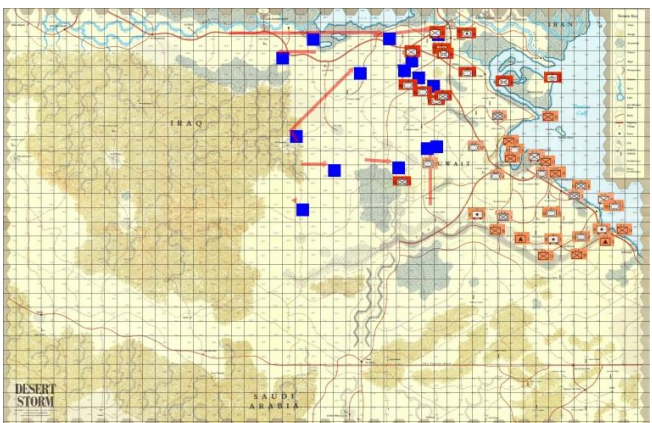
TS: 5 – UD:8 – TD:21 – TDes:0



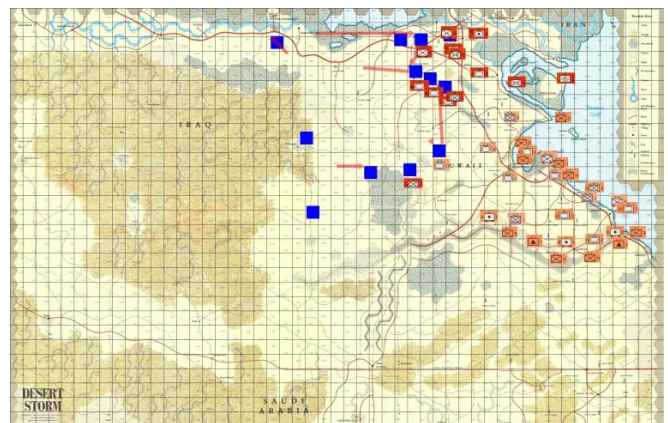
TS: 6 – UD:11 – TD:24 – TDes:1



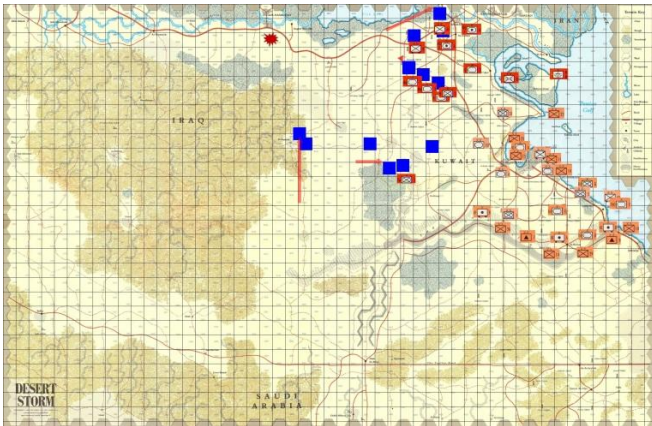
TS: 7 – UD:11 – TD:26 – TDes:1



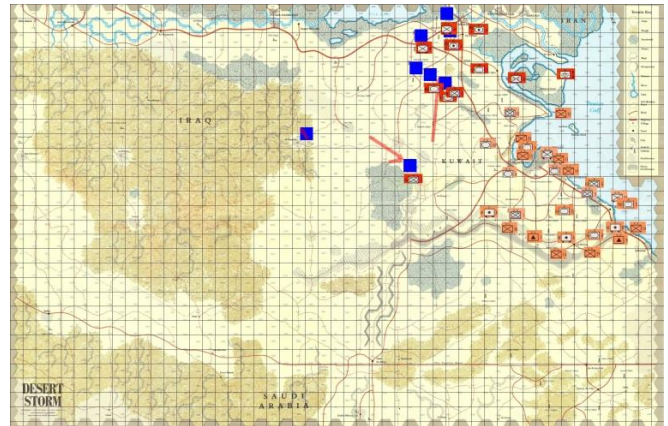
TS: 8 – UD:11 – TD:27 – TDes:1



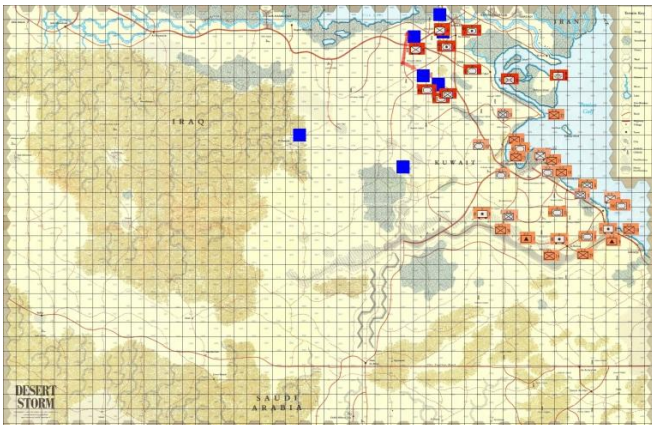
TS: 9 – UD:11 – TD:27 – TDes:1



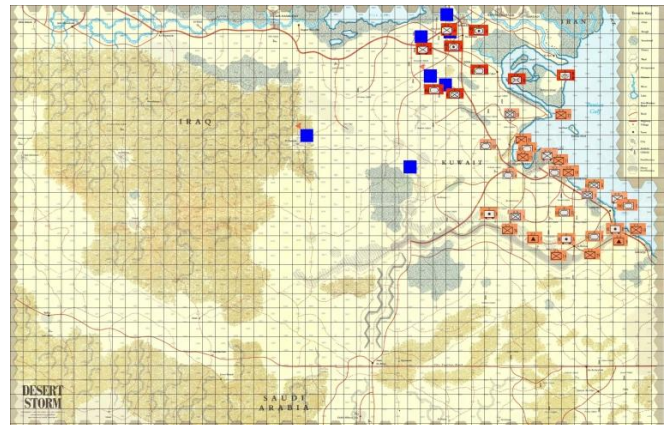
TS: 10 – UD:13 – TD:28 – TDes:2



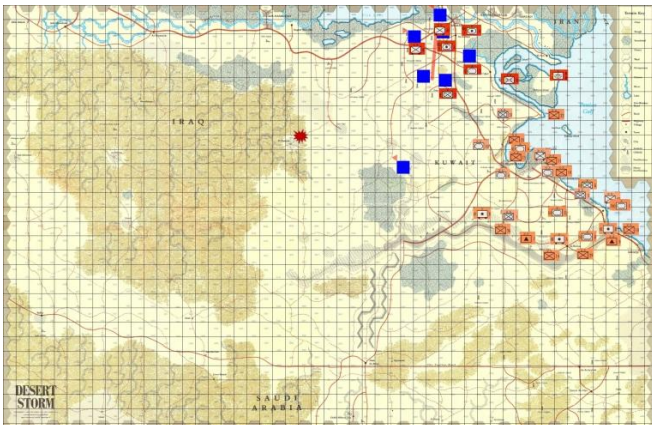
TS: 11 – UD:13 – TD:29 – TDes:2



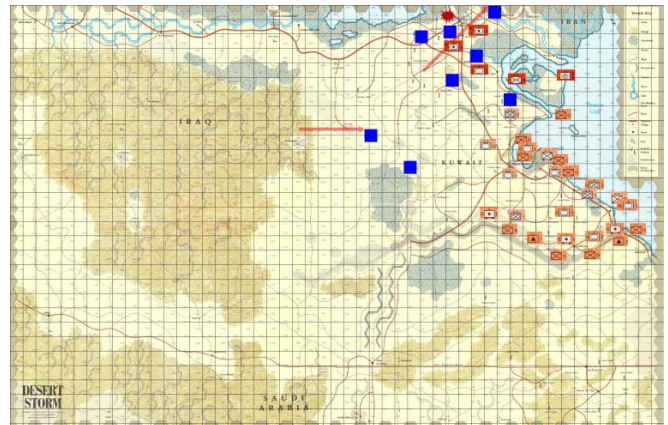
TS: 12 – UD:13 – TD:20 – TDes:2



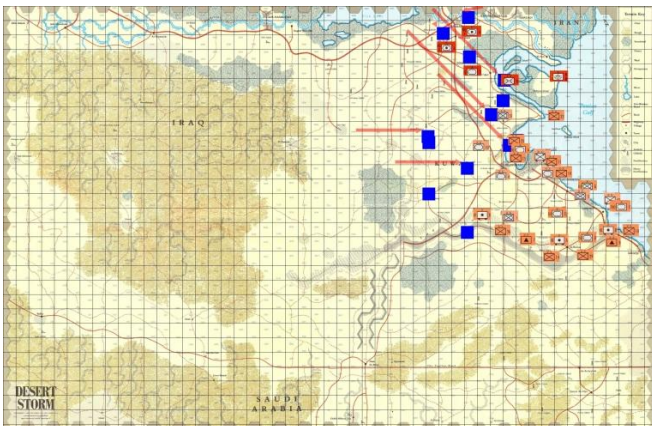
TS: 13 – UD:13 – TD:31 – TDes:2



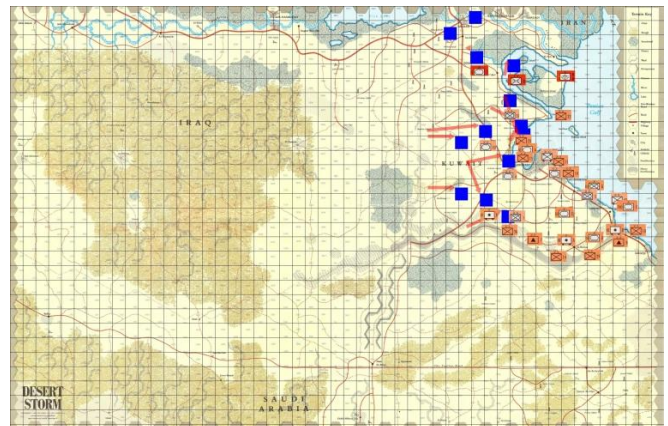
TS: 14 – UD:13 – TD:32 – TDes:2



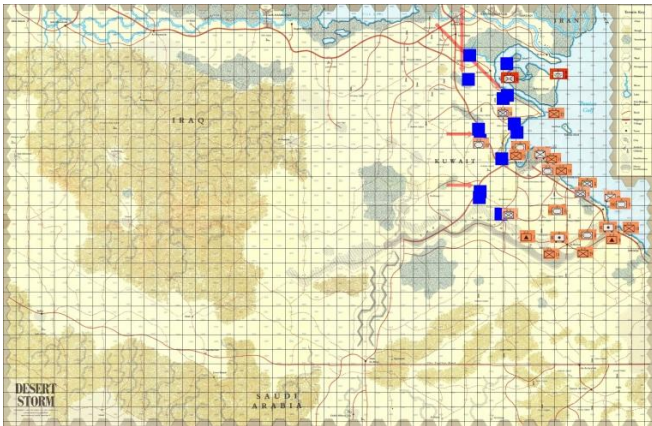
TS: 15 – UD:16 – TD:35 – TDes:2



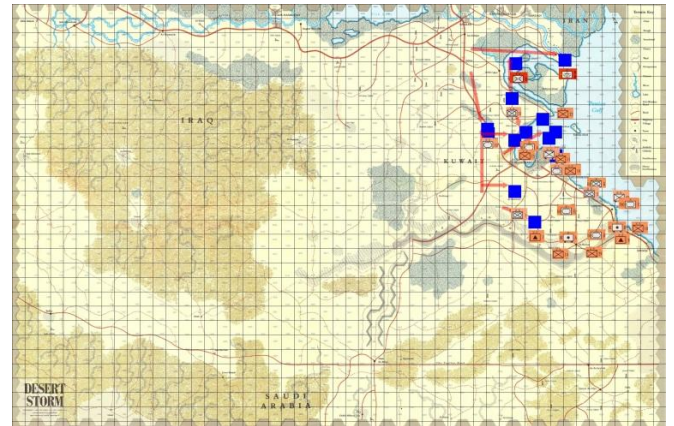
TS: 16 – UD:16 – TD:35 – TDes:2



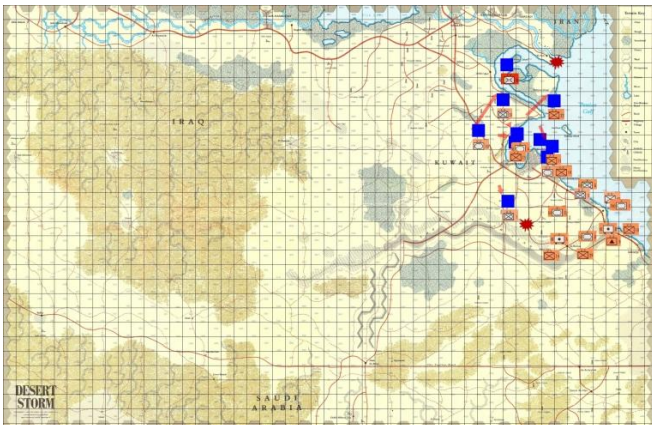
TS: 17 – UD:16 – TD:37 – TDes:2



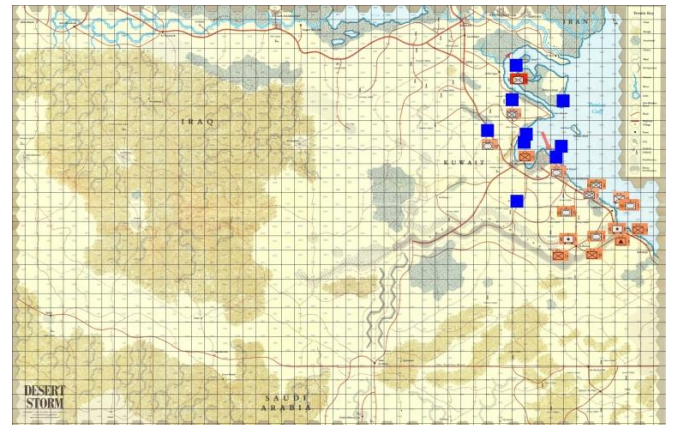
TS: 18 – UD:16 – TD:42 – TDes:2



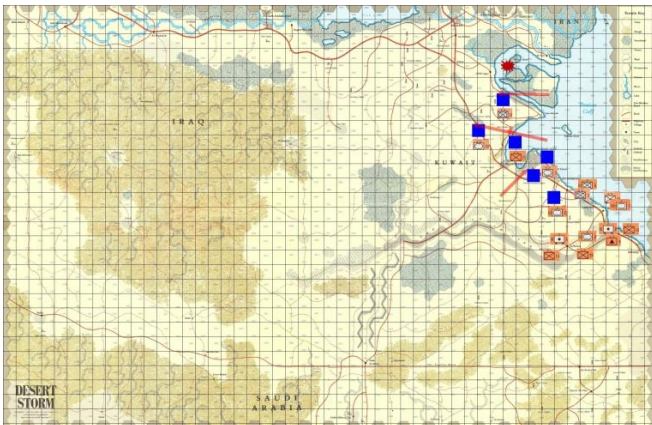
TS: 19 – UD:16 – TD:42 – TDes:2



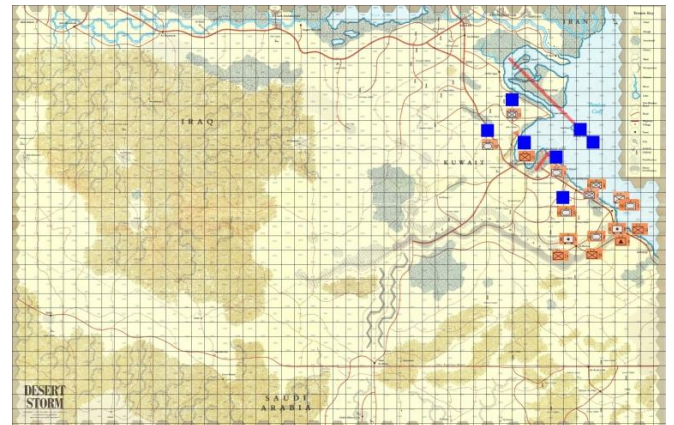
TS: 20 – UD:22 – TD:45 – TDes:2



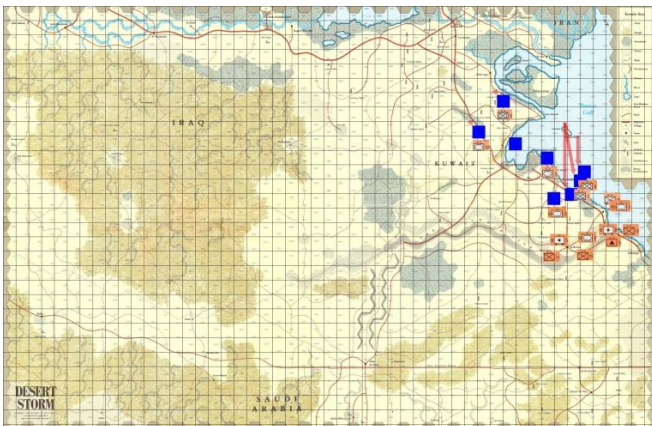
TS: 21 – UD:22 – TD:49 – TDes:3



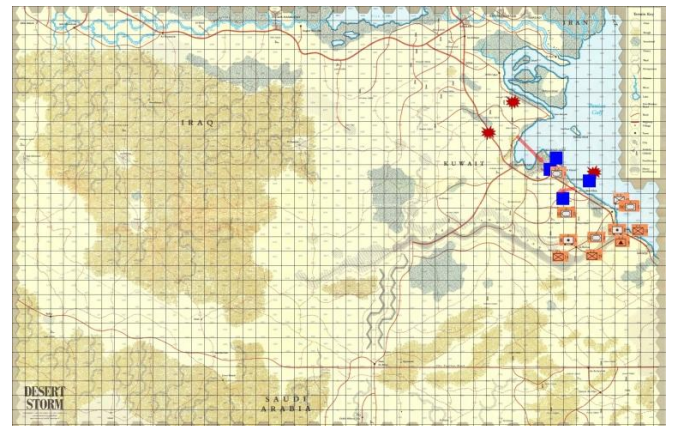
TS: 22 – UD:22 – TD:50 – TDes:3



TS: 23 – UD:22 – TD:50 – TDes:3



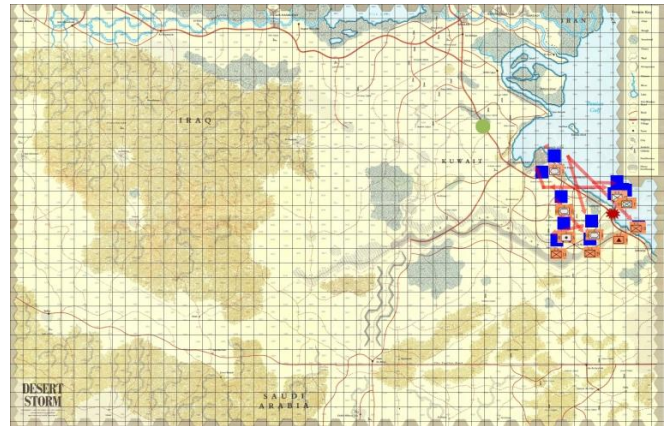
TS: 24 – UD:22 – TD:51 – TDes:3



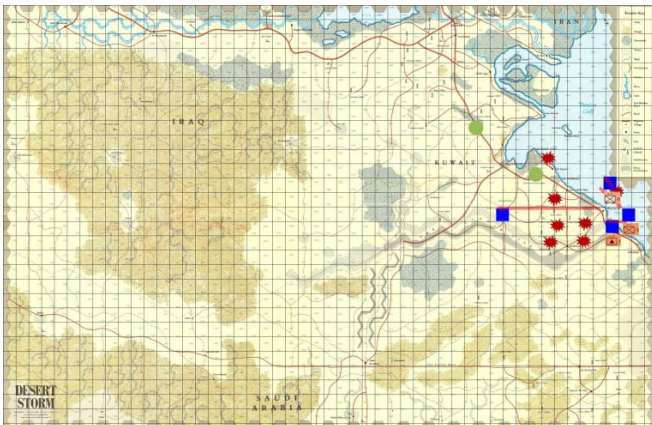
TS: 25 – UD:22 – TD:55 – TDes:3



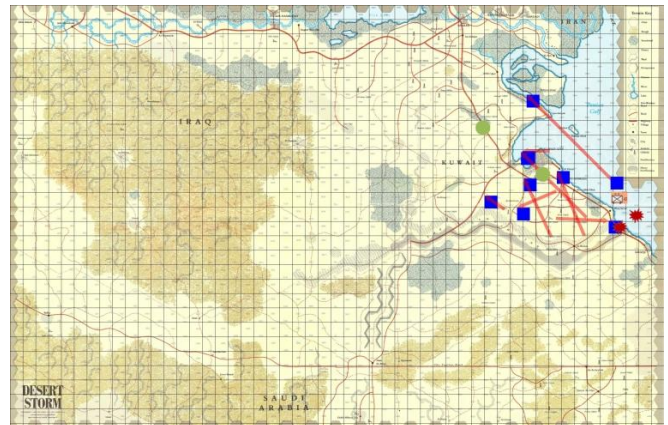
TS: 26 – UD:22 – TD:55 – TDes:3



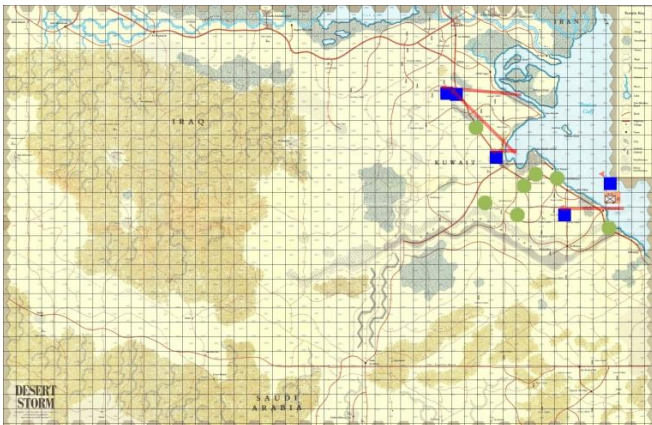
TS: 27 – UD:22 – TD:57 – TDes:3 – OZ: 1



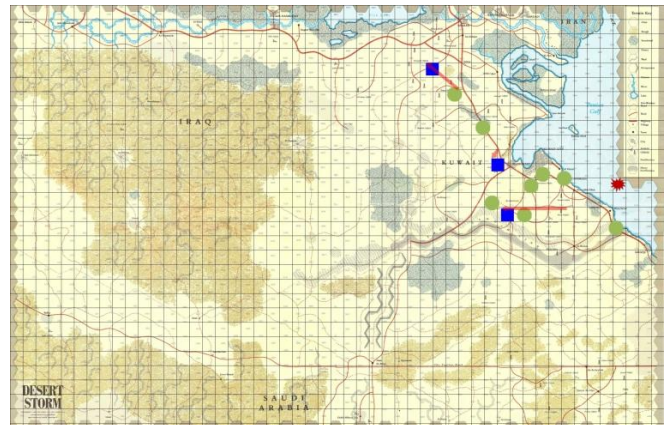
TS: 28 – UD:22 – TD:64 – TDes:3 – OZ: 2



TS: 29 – UD:25 – TD:66 – TDes:3 – OZ: 2



TS: 30 – UD:25 – TD:66 – TDes:3 – OZ: 7



TS: 31 – UD:25 – TD:67 – TDes:3 – OZ: 8

In table 13, in the time slot (TS) 1 we can see the initial historical deploy. From TS 1 to TS 4 the units begin flanking the targets. Then, from TS 5 to TS 26, the units engage in close combat with the targets, in order to achieve the primary and secondary military objectives. From TS 27 until the end of the simulation, the units begin to occupy the occupation zones. The oilfields occupied are not depicted in this table, but they are included in the analysis results.

Empirical Result Analysis Models

Most known problems only have two objectives, as stated in the “Background canonical models” section. Although the scenario depicted in this thesis has clearly more than 2 objectives. The warfare scenario has a multitude of objectives to be taken into account in order to calculate the best set of solutions for the problem. Following this reasoning, is of extreme importance, the creation of an analysis framework, in order to allow the specialist to analyze the data with relative ease. In this thesis we propose an analysis model based on the scenario of warfare. Our framework proposes four integrated models, the general effectiveness model, the operation analysis model, the logistic analysis model and the general objective model, which will be depicted in this section.

8.1. General Effectiveness Model

The general effectiveness model compares the units/targets destroyed between themselves or with primary or secondary military objectives. It can also be used to see if the model reaches the primary and secondary objectives throughout the time slots and between control parameters, for fine tuning.

➤ *Without control parameters*

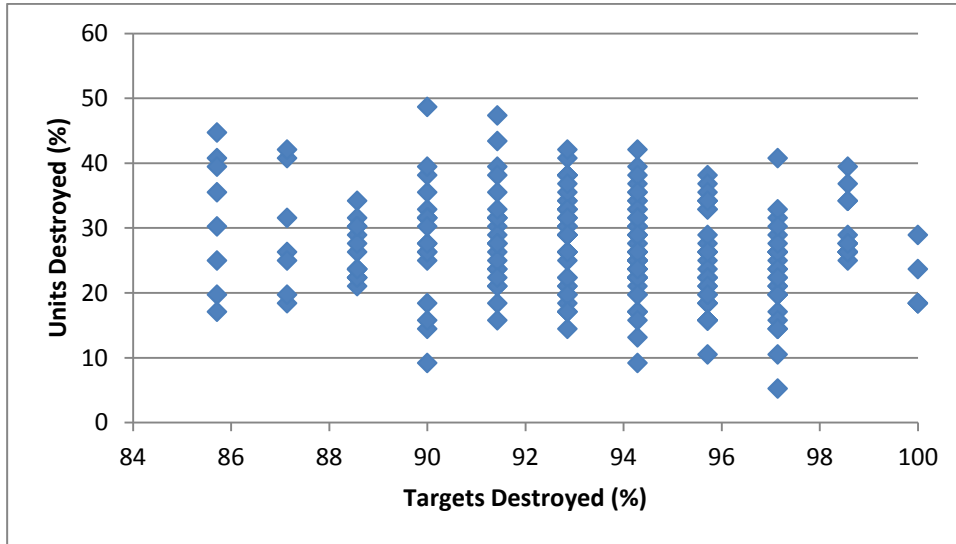


Fig. 19. Comparison between units destroyed and targets destroyed

In the figure 19, we can see that there is a high variety of results. With the data presented here a military strategist could derive the best solution regarding the percentage of targets destroyed and units destroyed. Although there is no correlation between these two variables. We could choose a solution from here according to our objective, if we want a solution that provides the most targets destroyed and less units destroyed, or if we want the most targets destroyed without taking into consideration the units destroyed and even if we want a solution where most units are not destroyed not regarding the targets destroyed. This decisions can only be made by a specialist in the area and must take into account the maps of the solutions he wishes, because although he has the final solution and is the best one according to his expectation, there might be some undesirable movements or decisions through the simulation which he does not approve.

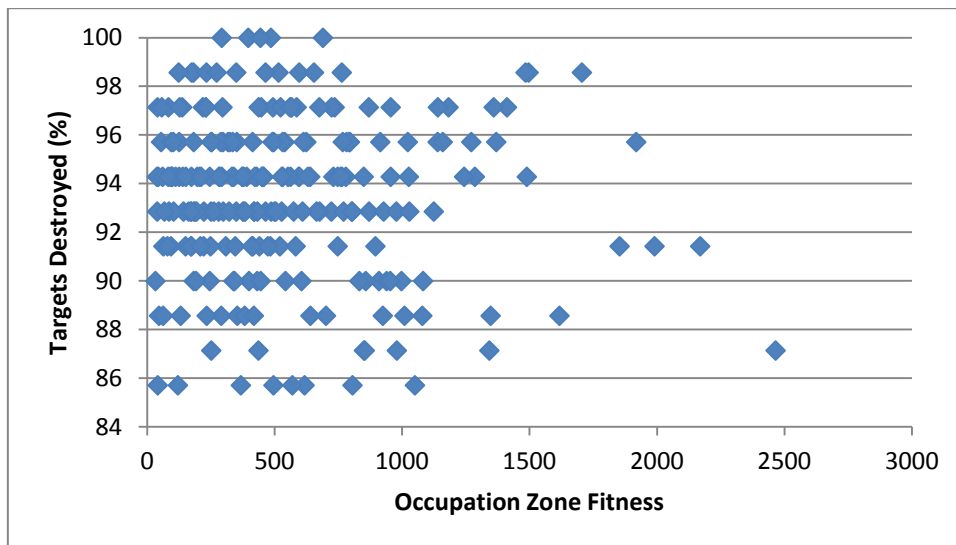


Fig. 20. Comparison between targets destroyed and occupation zone fitness

In the figure 20 we can correlate the percentage of targets destroyed and the occupation zones occupied. We can reach the conclusion that the occupation zones become occupied later in the simulation when most targets are destroyed. This can mean that units are well organized and are being all used throughout the simulation, only becoming available to occupy the occupation zones later when there are less targets still in battle. Higher occupation zone fitness means that the zones become occupied earlier than the ones with lower fitness values.

➤ *With control parameters (fine tuning)*

In the figure 21 we can assert that there are some configurations that are better than others in regard to the time slots required. If a lower value is good or not, is up to the specialist because a configuration which took less time slots might have more casualties. Also, some unit group movements might not be the most adequate regarding the expectations of the specialist. It sometimes is a matter of trade-off between objectives.

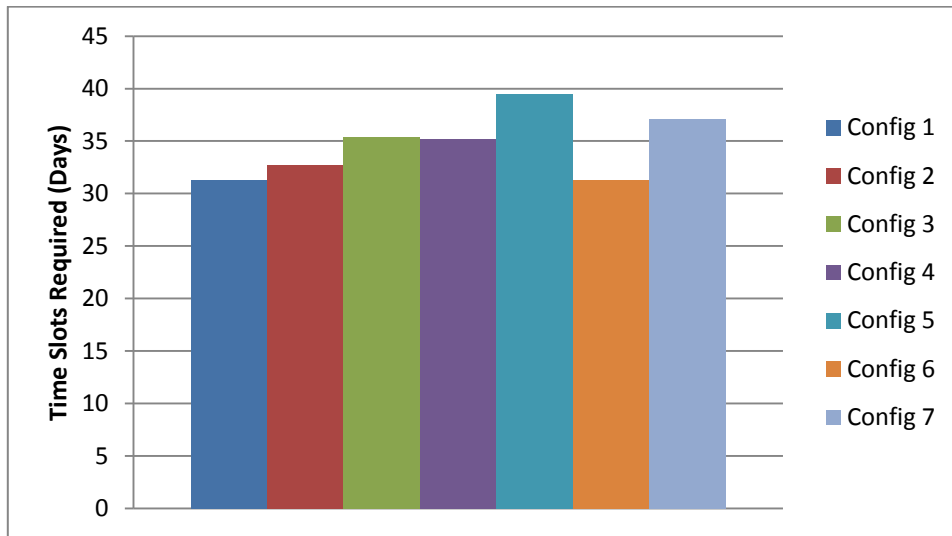


Fig. 21. Time slots required by configuration

Analyzing figure 22, we can see some variation in the values between configurations. If the main objective was to have the fewer casualties possible, the best configuration would be the seventh. On the other hand, if the objective was for the targets to desert instead of engaging in combat then the best would be the fourth. Again, as has been stated throughout this section it is always a trade-off between objectives which has to be made by specialist in this area with the aid of these figures and also the maps.

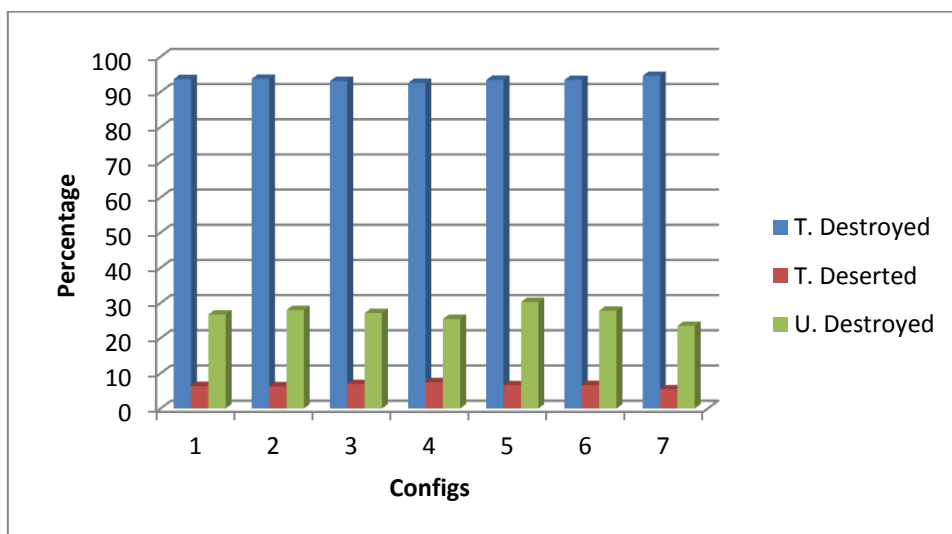


Fig. 22. Comparison between configurations regarding targets destroyed, targets deserted and units destroyed

In the figure 23 we can see the evolution of the occupation of oilfields. For example, the fifth configuration is the one that has an early spike, which means that most oilfields are occupied right in the beginning of the simulation. On the other hand, the seventh configuration has a late evolution, which means that the oilfields are occupied later in the simulation.

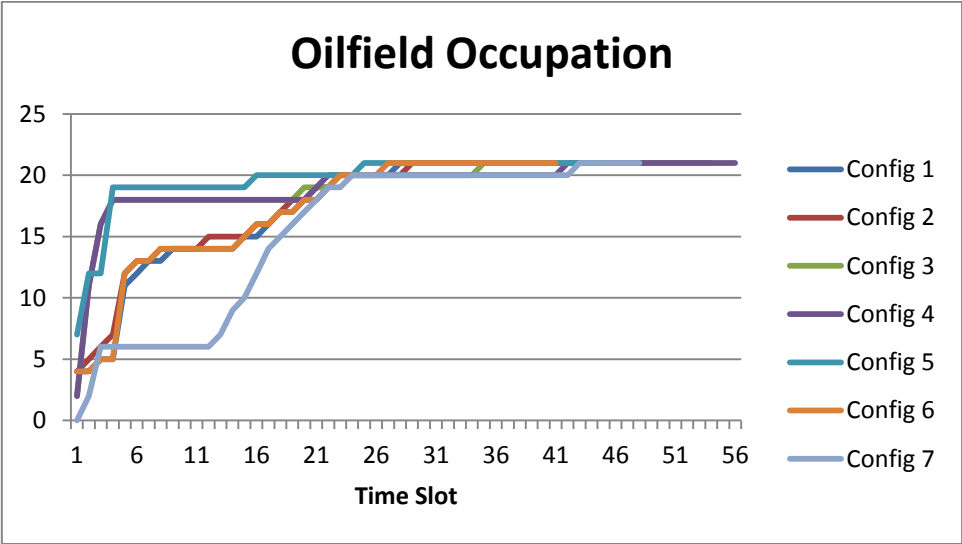


Fig. 23. Comparison between configurations regarding oilfield occupation

In the figure 24 we can see the evolution of the occupation of occupation zones. For example, the seventh configuration is the one that rises early, which means that occupation zones are being occupied right in the beginning of the simulation, however it is also the one with lowest final occupation zone fitness, that can mean that although it is the first to have occupation zones occupied it is also the last one to occupy all of them. On the other hand, the sixth configuration has a late evolution, which means that the occupation zones are occupied later in the simulation. The fourth configuration has a sharp decline due to one or more groups which were attacking a target were destroyed in battle and no other units were available. So, some of the units that were occupying an occupation zone were obliged to engage those targets.

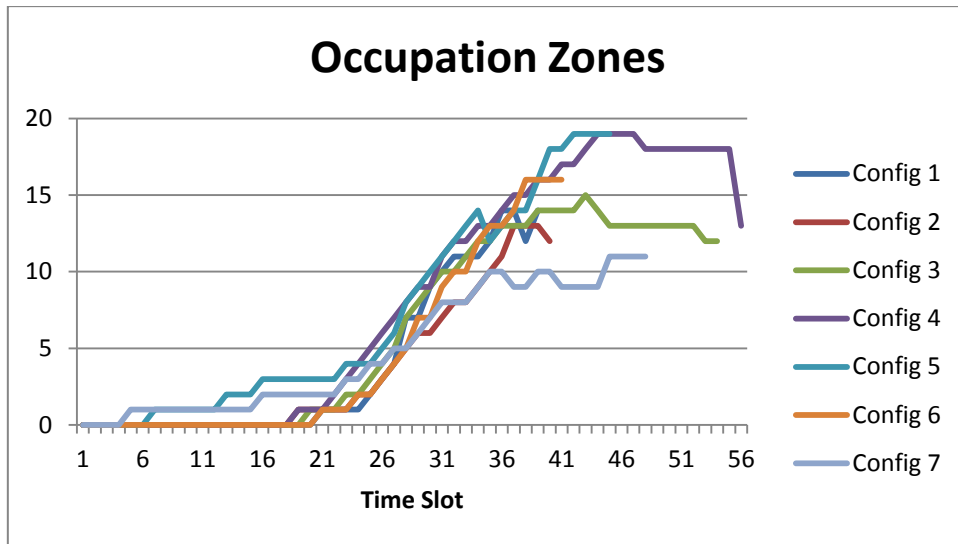


Fig. 24. Comparison between configurations regarding occupation zone

8.2. Operation Analysis Model

This model compares fitness values that show the effectiveness of the model in terms of operation, in order to verify if the units are following the right path to target and are going in the right direction to reach the primary and secondary military objectives.

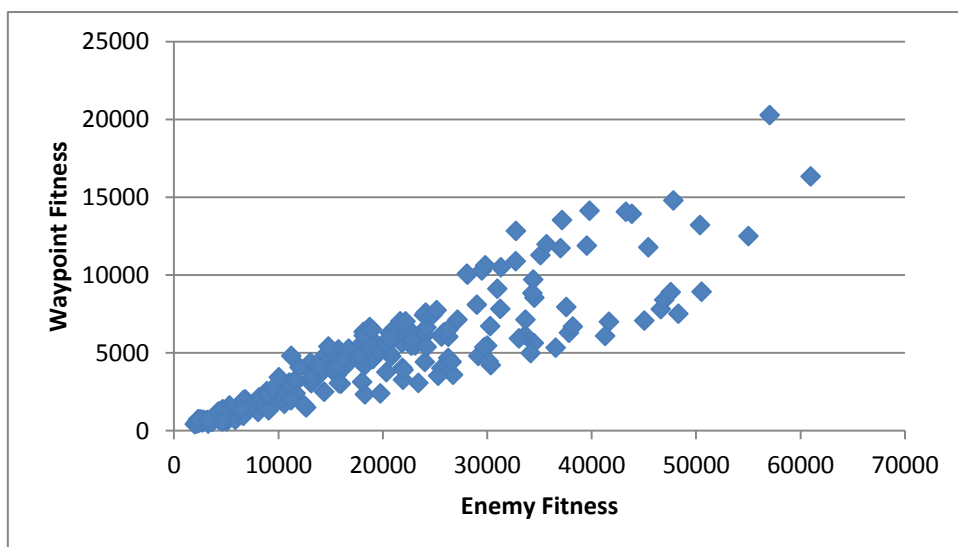


Fig. 25. Comparison between waypoint fitness and enemy fitness

In the figure 25 we can take many conclusions. We can correlate the enemy fitness (distance to the target) and the waypoint fitness (distance to the nearest waypoint in the path to the target). If a unit group is very distant to the target the enemy fitness will be higher, but if the waypoint fitness is high means that the group might be too distant from the best path to the target. If the enemy and waypoint fitness are low, that means that the group is near the target and following the best path to it.

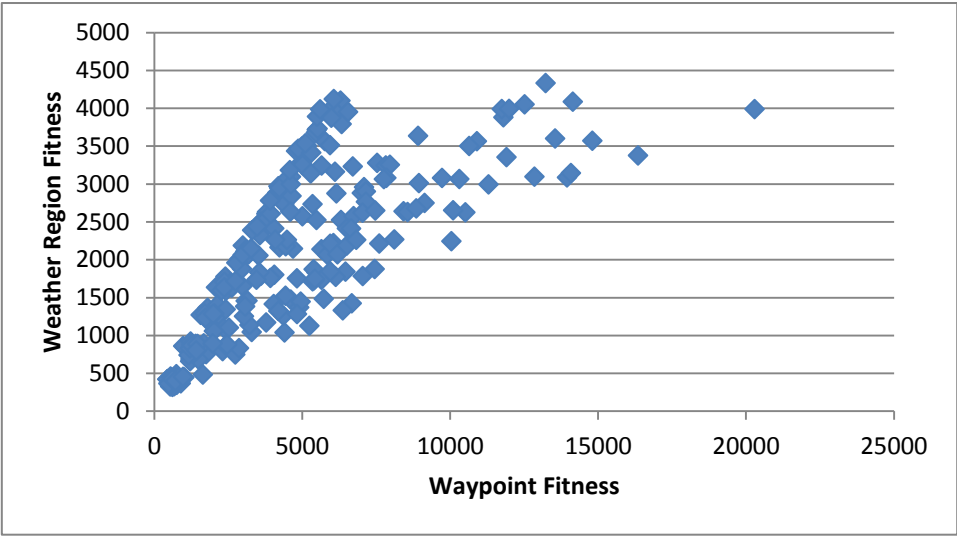


Fig. 26. Comparison between weather region fitness and waypoint fitness

In the figure 26 we can correlate the weather region fitness and the waypoint fitness. The waypoint fitness can be higher if a unit group encounters a weather zone, this means that the group must, in some cases, go around the weather zone to reach the target and that makes the group deviate from the best path to the target. A lower waypoint and weather region fitness means that the group did not find any or few weather regions throughout the simulation which made it deviate from the best path to target.

8.3. *Logistic Analysis Model*

This model helps the analyst to gather data to assert to what level the supplies and ammo being deployed in battle are being used in the most effective way. It can also be used to see how the model makes use of the supplies and ammo throughout the time slots and between control parameters, for fine tuning.

➤ *Without control parameters*

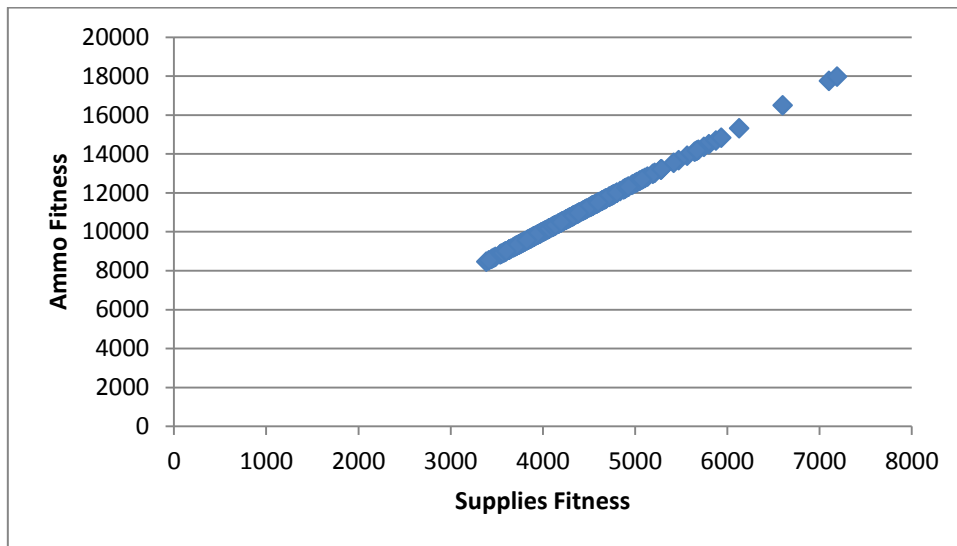


Fig. 27. Comparison between ammo fitness and supplies fitness

In figure 27 we can correlate the supplies fitness with the ammo fitness. When the ammo fitness is higher logically the supplies fitness is also higher because if a unit is attacking, it is also using supplies. With this graph we can also verify the effectiveness of the model, because we can reach the conclusion that the units are generally engaging in combat while consuming supplies, and are not by any means only consuming supplies without engaging in combat.

➤ *With control parameters (fine tuning)*

In the figure 28 we can see the consumption of supplies throughout the time slots of the simulation; we can see when supplies are most and less used. This data can be used by a specialist to optimize supplies consumption and deployment.

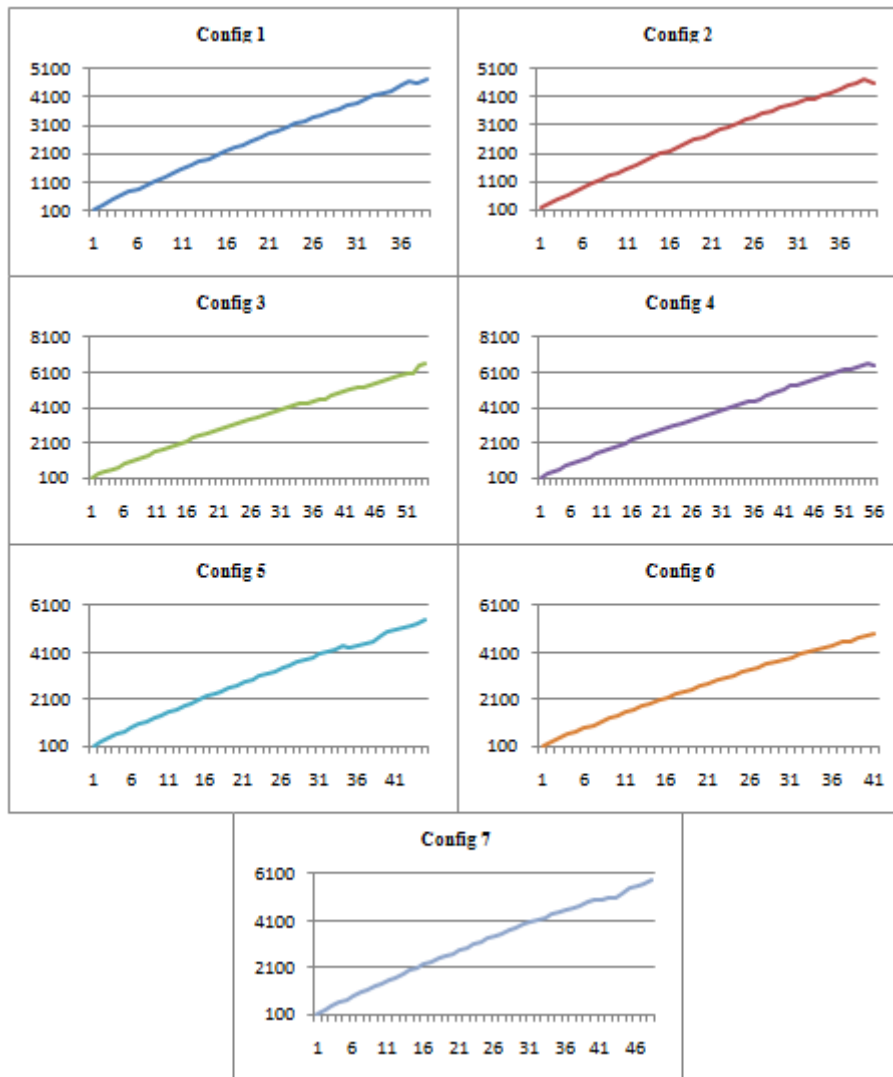


Fig. 28. Comparison between configurations regarding supplies spent per time slot (days)

In the figure 29 we can see the consumption of ammo throughout the time slots of the simulation; we can see when ammo is most and less used. This data can be used by a specialist to optimize ammo consumption and deployment. He can also assert when there are the most units attacking.

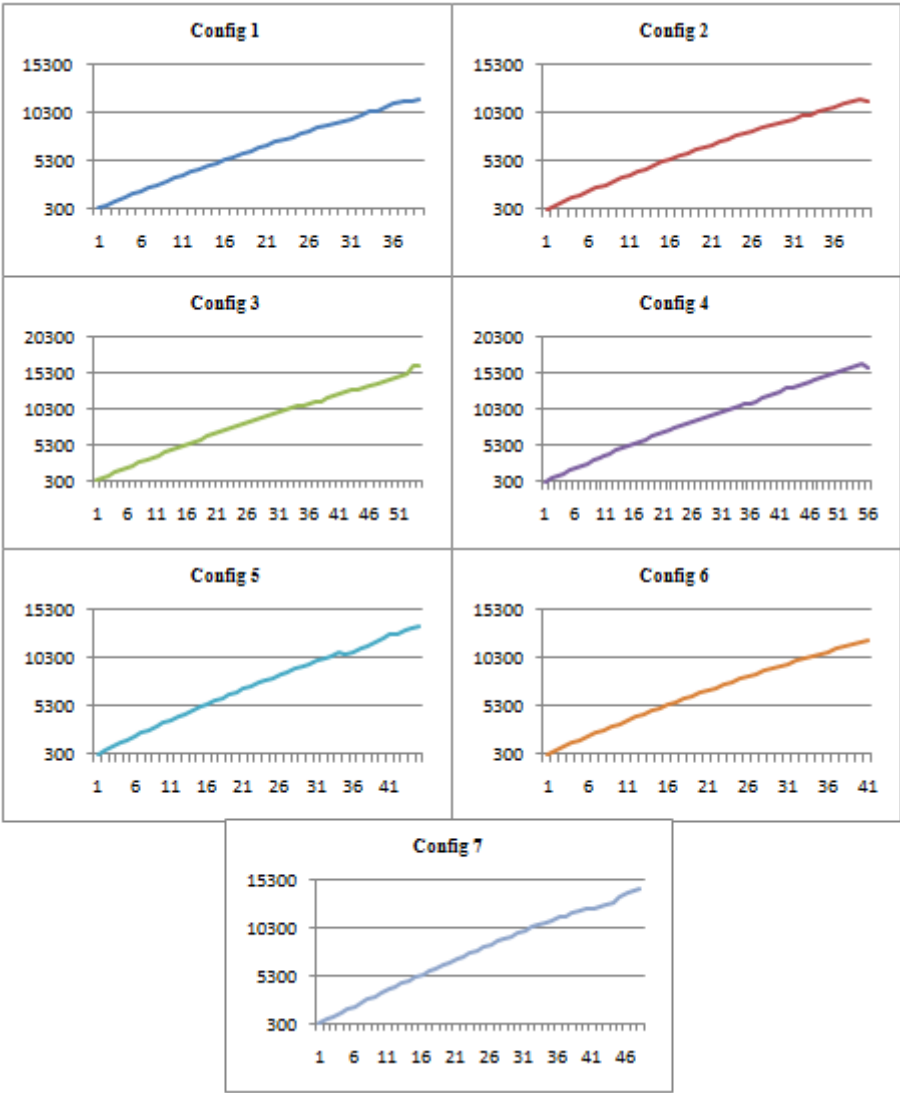


Fig. 29. Comparison between configurations regarding ammo spent per time slot (days)

8.4. *General Objective Model*

This model compares primary and secondary military objectives between themselves, for example, to assert if they are correlated or not. This is, to check if one influences the other.

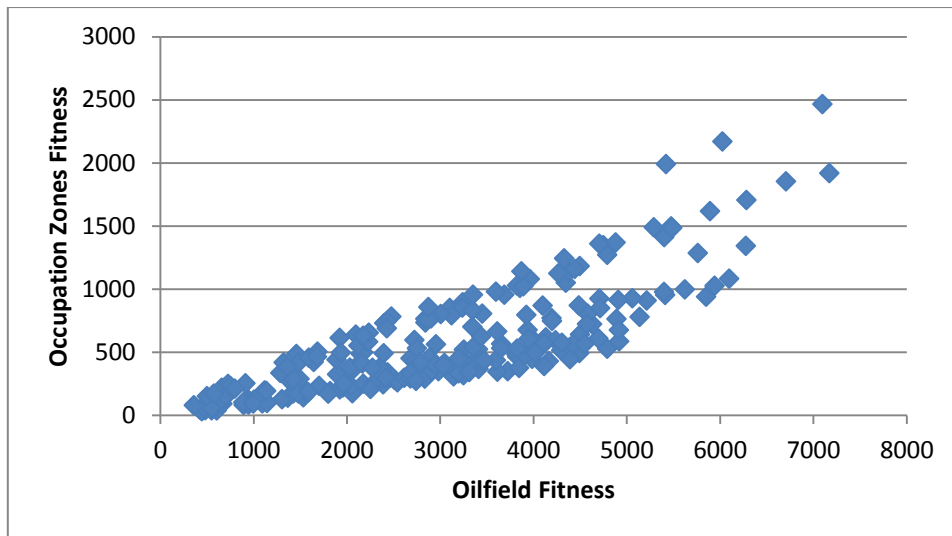


Fig. 30. Comparison between occupation zone fitness and oilfield fitness

In figure 30 we can reach the conclusion that the occupation zones fitness is generally higher when the oilfield fitness is too. That means that the occupation zones become occupied after the oilfields become occupied. A higher oilfield fitness means that the oilfields were occupied early in the simulation, a lower one can mean that the oilfields were occupied later in the simulation or weren't even occupied depending on the value achieved.

8.5. *Detailed Objective Analysis Tables*

In the tables 14 - 24 it is presented the maximum, the minimum, the mean and the standard deviation for each objective per configuration. These results must be analyzed by the specialist in conjunction with the maps of the simulations and the graphs in figures 19 -30, because the data presented might not mean anything without the proper background. A higher

standard deviation value means that the solutions achieved have more variation than the ones with a lower value.

Time Slots required

Table 14. Time slots required per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	28	39	31,257143	2,7706921
2	27	40	32,72222	3,396167
3	28	54	35,41667	6,038649
4	26	56	35,22222	7,775
5	31	48	39,48276	4,680264
6	25	41	31,30556	3,340155
7	28	48	37,125	4,601291

Example analysis for table 14: Regarding the Minimum value, we can verify that configuration 6 is the best one, because we want to minimize the time slots required by the simulation. But when comparing the maximum value, the best configuration is number 1. So, to reach the right option we have to compare the mean and standard deviation of these two configurations. Configuration 1 has a lower mean compared with configuration 6, which means that the majority of the results of this configuration are lower than those of configuration 6. This can also be verified by the standard variation values. If a configuration has a lower standard deviation, this means that most values are near to the mean value, so there is not much variation in the values. In conclusion, if our sole objective was to minimize the time slots required, the best solution would be number 1.

The analysis made for table 14 can be made for tables 15 – 24, having in consideration if the objective is to minimize or maximize the fitness values.

Enemy Fitness

Table 15. Enemy fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	1981	21034	11432,54	5858,338
2	2420	25847	13474,25	7077,388
3	4338	48296	24679,17	12400,88
4	3087	60970	22529,47	13617,09
5	4822	57040	26126,1	15333,48
6	2241	21063	11422,47	5873,089
7	3204	50373	22255,59	12299,37

Waypoint Fitness

Table 16. Waypoint fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	432	6286	3167	1692,441
2	529	6325	3303,028	1723,988
3	611	7956	4314,028	2226,595
4	470	16343	5822,611	3622,506
5	680	20285	6224,966	4826,864
6	481	6542	3225,639	1720,118
7	604	14139	7037,406	4193,981

Weather Region Fitness

Table 17. Weather Region fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	423	4102	2172,714	1080,241
2	328	4123	2155,333	1134,605
3	347	3280	1813,472	934,5971
4	322	3378	1799,083	931,5269
5	418	4053	1971,448	1106,419
6	363	3987	2113,778	1105,438
7	400	4334	2236,875	1216,486

Supplies Fitness

Table 18. Supplies fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	3386	4754	3965,257	327,2832
2	3484	4926	4122,833	314,6799
3	3582	6600	4470,222	625,455
4	3464	7190	4490,389	852,7628
5	3830	6128	4751,931	508,4531
6	3414	4898	3955,889	343,2485
7	3602	5934	4728,125	486,0272

Ammo Fitness

Table 19. Ammo fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	8465	11885	9913,143	818,208
2	8710	12315	10307,08	786,6997
3	8955	16500	11175,56	1563,637
4	8660	17975	11225,97	2131,907
5	9575	15320	11879,83	1271,133
6	8535	12245	9889,722	858,1213
7	9005	14835	11820,31	1215,068

Oilfield Occupation Fitness

Table 20. Oilfield occupation fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	447	4626	2501,6	1236,983
2	420	4916	2582,639	1376,286
3	602	6274	3334,583	1683,001
4	510	7095	3261,083	1749,742
5	656	7169	3336,034	1917,496
6	442	4693	2447,167	1294,874
7	357	5485	2687,188	1450,745

Occupation Zone Fitness

Table 21. Occupation Zone fitness per configuration

Config	Minimum	Maximum	Mean	Standard deviation (σ)
1	41	723	303,0286	163,6146
2	40	764	319,9444	191,3889
3	39	1343	636,9167	315,8748
4	63	2466	691,9444	544,515
5	132	1919	895,5172	530,0706
6	33	610	299,9722	175,4687
7	80	1485	748,2813	392,8262

Targets Destroyed

Table 22. Targets destroyed per configuration

Config	Minimum (%)	Maximum (%)	Mean (%)	Standard deviation (σ)
1	85,71429	98,57143	93,63265	3,521576
2	85,71429	100	93,73016	3,388619
3	85,71429	100	93,05556	2,957614
4	85,71429	97,14286	92,53968	2,975926
5	87,14286	98,57143	93,39901	3,125654
6	85,71429	100	93,37302	3,54998
7	87,14286	100	94,55357	2,911043

Targets Deserted

Table 23. Targets deserted per configuration

Config	Minimum (%)	Maximum (%)	Mean (%)	Standard deviation (σ)
1	1,428571	14,28571	6,367347	3,521576
2	0	14,28571	6,269841	3,388619
3	0	14,28571	6,944444	2,957614
4	2,857143	14,28571	7,460317	2,975926
5	1,428571	12,85714	6,600985	3,125654
6	0	14,28571	6,626984	3,54998
7	0	12,85714	5,446429	2,911043

Units Destroyed

Table 24. Units destroyed per configuration

Config	Minimum (%)	Maximum (%)	Mean (%)	Standard deviation (σ)
1	10,52632	42,10526	26,69173	7,400383
2	13,15789	40,78947	27,99708	6,806101
3	14,47368	44,73684	27,11988	7,046799
4	10,52632	48,68421	25,40205	7,972446
5	5,263158	47,36842	30,26316	7,999904
6	9,210526	42,10526	27,77778	7,941975
7	14,47368	42,10526	23,47862	5,829977

In conclusion, we can conclude that, with the numbers of tables returned, the analysis for this problem is quite complex. This complexity is an important issue in many-objective optimization problems, because it can lead to a difficult analysis of the results.

Conclusions and future work

9.1. Conclusions

Warfare simulations are widely recognized to be a fundamental part of a country's modern and future warfare arsenal and tend to become more realistic in years to come. In this thesis we made an overview of an approach using multi-objective optimization and EAs to solve this problem. As research proof no one before took this approach to the field of warfare simulation, to much of our knowledge.

The results are promising for the use of these techniques in this field, although it still misses a lot of important warfare situations and tactical procedures, it is a starting point for those who are interested in this area.

As discussed earlier, the problem depicted in this thesis is a many-objective optimization problem, which makes it very difficult to use with global search methods (i.e. Genetic Algorithms), as these would generate a lot of invalid solutions.

Also, this problem instance requires that the solutions are presented in a very restrict temporal sequence, which contains many dependencies and restrictions, due to the primary and secondary military objectives.

In conclusion, the main contributions of this thesis are:

- Multi-objective meta-heuristic proposal, focused on this specific problem (Warfare), although it can be used in other problems with a similar mathematical model;
- An analysis framework, this problem is a many-objective optimization problem which has many objectives. So, an analysis framework is of extreme importance to ease the task of analyzing the solutions achieved.

9.2. *Future work*

As future work, the following procedures would help the performance of the approach of this thesis:

- Backwards path relinking

In addition to the path relinking method used, it would be great to create all backward movements of one group in order to create more solutions and generate more diversity in the solutions.

- Multi-core/Multi-processor optimization

With the use of threading, the heuristics would become much more efficient and the execute time would decrease drastically.

- GPU/FPGA acceleration

Following the multi-core or multi-processor optimization, it would be great if the meta-heuristic proposed could be executed by GPU's and possibly applied to games.

- Logistics network

The introduction of a logistics network would bring more realistic results, assigning an ammo and supply value to each unit, and spending them during the simulation would create the necessity of special units to supply these units or supply points. Together with the necessity of a better scheduling plan for engaging in combats. For example, if a target is 5 days away but only has 2 days of supply it could not engage this target.

- Simulate with military systems in order to find areas of improvement

Use one or more solutions to simulate on a real military simulator and get a military specialist so that he can challenge the solution achieved. Then use the results of this exercise to find areas of improvement in the meta-heuristic.

- Study analysis tools for direct comparizon of more than two objectives

The analysis models depicted in this thesis only make use of two objectives simultaneously, although they can be used for more than two objectives. So, if we could find a tool to analyze more than two objectives, ideally many as there is, we could reach many other conclusions with the results achieved that we could not reach with only two objectives.

From this thesis resulted a scientific paper intitlued “Realistic ground warfare simulation analysis framework based on evolutionary multi-objective meta-heuristic techniques”.

Bibliography

- [1] Garey M, Johnson. Computers and Intractability: A guide to the theory of NP-Completeness, 1998.
- [2] Alsuwaiyel MH. Algorithms: Design Techniques and Analysis. USA, World Scientific Publishing Company, 1998
- [3] Zitzler E, Deb K, Thiele L. Comparison of multi-objective evolutionary algorithms: Empirical results. *Evol Comput* 2000, vol. 8, issue 2, pp.173-95.
- [4] Jones DF, Mirrazavi SK, Tamiz M. Multi-objective meta-heuristics: an overview of the current state-of-the-art. *Eur J Oper Res* 2002, vol. 137, issue 1, pp.1-9.
- [5] Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: *Proceedings of the international conference on genetic algorithm and their applications*, 1985.
- [6] Fonseca CM, Fleming PJ. Multi-objective genetic algorithms. In: *IEE colloquium on 'Genetic Algorithms for Control Systems Engineering' (Digest No. 1993/130)*, 28 May 1993. London, UK: IEE; 1993.
- [7] Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm for multi-objective optimization. In: *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, 27–29 June, 1994. Orlando, FL, USA: IEEE; 1994.*
- [8] Hajela P, Lin C-y. Genetic search strategies in multicriterion optimal design. *Struct Optimization* 1992.

- [9] Murata T, Ishibuchi H. MOGA: multi-objective genetic algorithms. In: Proceedings of the 1995 IEEE international conference on evolutionary computation, 29 November–1 December, 1995. Perth, WA, Australia: IEEE; 1995.
- [10] Srinivas N, Deb K. Multi-objective optimization using non-dominated sorting in genetic algorithms. *J Evol Comput* 1994, vol. 2, issue 3, pp. 221-48.
- [11] Zitzler E, Thiele L. Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans Evol Comput* 1999, vol. 3, issue 4, pp.257-71.
- [12] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Technology: Zurich, Switzerland; 2001.
- [13] Knowles JD, Corne DW. Approximating the non-dominated front using the Pareto archived evolution strategy. *Evol Comput* 2000, vol. 8, issue 2, pp.149-72.
- [14] Corne DW, Knowles JD, Oates MJ. The Pareto envelope-based selection algorithm for multi-objective optimization. In: Proceedings of sixth international conference on parallel problem solving from Nature, 18–20 September, 2000. Paris, France: Springer; 2000.
- [15] Corne D, Jerram NR, Knowles J, Oates J. PESA-II: region-based selection in evolutionary multi-objective optimization. In: Proceedings of the genetic and evolutionary computation conference (GECCO- 2001), San Francisco, CA, 2001.
- [16] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput* 2002, vol. 6, issue 2, pp.182-97.
- [17] Sarker R, Liang K-H, Newton C. A new multi-objective evolutionary algorithm. *Eur J Oper Res* 2002, vol. 140, issue 1, pp.12-23.
- [18] Coello CAC, Pulido GT. A micro-genetic algorithm for multi-objective optimization. In: Evolutionary multi-criterion optimization. First international conference, EMO 2001, 7–9 March, 2001. Zurich, Switzerland: Springer; 2001.

- [19] Lu H, Yen GG. Rank-density-based multi-objective genetic algorithm and benchmark test function study. *IEEE Trans Evol Comput* 2003, vol. 7, issue 4, pp.325-43.
- [20] Yen GG, Lu H. Dynamic multi-objective evolutionary algorithm: adaptive cell-based rank and density estimation. *IEEE Trans Evol Comput* 2003, vol. 7, issue 3, pp.253-74.
- [21] Coello CAC. A comprehensive survey of evolutionary-based multi-objective optimization techniques. *Knowl Inform Syst* 1999, vol. 1, issue 3, pp.269-308.
- [22] Coello CAC. An updated survey of evolutionary multi-objective optimization techniques: state of the art and future trends. In: *Proceedings of the 1999 congress on evolutionary computation-CEC99*, 6–9 July 1999. Washington, DC, USA: IEEE.
- [23] Coello CAC. An updated survey of GA-based multi-objective optimization techniques. *ACM Comput Surv* 2000, vol. 32, issue 2, pp.109-43.
- [24] Fonseca CM, Fleming PJ. Genetic algorithms for multi-objective optimization: formulation, discussion and generalization. In: *Proceeding of the ICGA-93: fifth international conference on genetic algorithms*, 17–22 July 1993. Urbana-Champaign, IL, USA: Morgan Kaufmann; 1993.
- [25] Fonseca CM, Fleming PJ. Multi-objective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Trans Syst Man Cybern A* 1998, vol. 28, issue 1, pp.26-37.
- [26] Jensen MT. Reducing the run-time complexity of multi-objective EAs: The NSGA-II and other algorithms. *IEEE Trans Evol Comput* 2003, vol. 7, issue 5, pp.503-15.
- [27] Xiujuan L, Zhongke S. Overview of multi-objective optimization methods. *J Syst Eng Electron* 2004, vo. 15, issue 2, pp.142-6.
- [28] Knowles J, Corne D. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multi-objective optimisation. In: *Proceedings of the 1999 congress on evolutionary computation- CEC99*, 6–9 July 1999. Washington, DC, USA: IEEE; 1999.

- [29] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of sixth international conference on parallel problem solving from nature, 18–20 September, 2000. Paris, France: Springer; 2000.
- [30] XTR Corp. Command Magazine: Desert Storm, Issue 13, November-December, 1991.
- [31] Wikipedia. Gulf War, http://en.wikipedia.org/wiki/Gulf_War, retrieved 10 September 2011.
- [32] In Depth Info. The Gulf War, <http://www.indepthinfo.com/iraq/>, retrieved 10 September 2011
- [33] The History Channel. Persian Gulf War, <http://www.history.com/topics/persian-gulf-war>, retrieved 10 September 2011
- [34] Chadwick Frank. Gulf War Fact Book. USA, GDW Inc., 1991.
- [35] Mesko Jim. Ground War Desert Storm. USA, Squadron/Signal Publications Inc., 1991.
- [36] Finlan Alastair. Essential Histories The Gulf War 1991. England, Osprey Publishing Ltd., 2003.
- [37] Lies.com. US Deaths in Vietnam and Iraq by Month, <http://www.lies.com/wp/2003/10/20/us-deaths-in-vietnam-and-iraq-by-month/>, retrieved 10 September 2011.
- [38] Çayirci Erdal, Marincic Dusan. Computer Assisted Exercises and Training. USA, John Wiley & Sons Inc., 2009
- [39] Carter Lee F., Komer Chad. Interactive Networked Battlefield Simulation Training Technologies. L-3 Communications Systems West.
- [40] Allender Eric, Loui Michael, Regan Kenneth. Complexity Classes.
- [41] Stroppa Nicolas. Algorithms & Complexity – Complexity Classes. Dublin, Dublin City University, 2006.

- [42] Rothe Jörg, Roos Magnus. Introduction to Computational Complexity. Germany, Institut für Informatik – Heinrich-Heine-Universität Düsseldorf, 2010.
- [43] Bridge Derek. Theory of Computation – Lecture 33: Np-Hard and NP-Complete Problems. England, University College Cork, 2003.
- [44] Zitzler Eckart, Laumanns Marco, Bleuler Stefan. A tutorial on evolutionary multi-objective optimization. Switzerland, Swiss Federal Institute of Technology.
- [45] CUED Divison A. Multi-objective Optimization. England, University of Cambridge, 2011.
- [46] Eiben A. E., Smith J.E.. Introduction to Evolutionary Computing. USA, Springer – Natural Computing Series, 2007.
- [47] Alba Enrique, Cotta Carlos. Evolutionary Algorithms. Spain, Universidad de Málaga, 2004.
- [48] Jones Gareth. Genetic and Evolutionary Algorithms. UK, University of Sheffield.
- [49] Konak Abdullah, Coit David W., Smith Alice E.. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety, 2006.
- [50] Zitzler Eckart, Thiele Lothar. Multi-objective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, vol. 3, issue 4, 1999.
- [51] Zitzler Eckart, Thiele Lothar, Deb Kalyanmoy. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. USA, Massachusetts Institute of Technology, 2000.
- [52] Leslie Martin. The eight queens problem – a neural network approach. USA, The university of Arizona, 2009.
- [53] Chekuri Chandra. The Knapsack Problem. USA, University of Illinois, 2009.
- [54] Electronic Systems Group. Lecture 13: The Knapsack Problem. Netherlands, Technische Universiteit Eindhoven, 2011.

- [55] A.E. Eiben, E.H.L. Aarts, K.M. Vanttee. Global Convergence of Genetic Algorithms: a Markov chain analysis. In: Schwefel, Männer, pp.4-12.
- [56] A.E. Eiben. Multiparent Recombination. In: Bäck et Al., Chap. 33.7., pp.289-307.
- [57] D.H. Wolpert, W.G. Macready. No free lunch theorems for optimization. IEEE Transactions on evolutionary computation, vol. 1, issue 1, pp.67-82, 1997.
- [58] D.E. Goldberg. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989.
- [59] E.H.L. Aarts, J. Korst. Simulated Annealing and Boltzmann Machines. Wiley, Chichester, UK, 1989.
- [60] S. Kirk Patrick, C. Gelatt, M. Vecchi. Optimization by simulated annealing. Science, 220, pp.671-680, 1983.
- [61] F. Glover. Tabu search and adaptive memory programming – advances, applications and challenges. In: R.S. Barr, R.V. Helgason, J.L. Kennington, Eds., Interfaces in computer science and operations research. Kluwer Academic Publishers, Norwell, MA, 1996, pp.1-75.
- [62] Matteo Matteucci. *A Tutorial on Clustering Algorithms - K-Means Clustering*, http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html, retrieved 10 September 2011.
- [63] Ishibuchi Hisao, Tsukamoto Noritaka, Nojima Yusuke. Evolutionary Many-Objective Optimization. USA, 3rd International Workshop on Genetic and Evolving Systems, March issue, pp.1-6, 2008.