2014-01-01

Multi-Objective Border Patrolling Optimization Using Game Theory And Evolutionary Algorithms

Franciso Oswaldo Aguirre
University of Texas at El Paso, faguirre@miners.utep.edu

MULTI-OBJECTIVE BORDER PATROLLING OPTIMIZATION USING GAME THEORY AND EVOLUTIONARY ALGORITHMS

OSWALDO AGUIRRE

Computational Science Program

APPROVED:

_____

Heidi A. Taboada, Ph.D., Chair

_____

Jose F. Espiritu, Ph.D.

_____

Vinod Kumar, Ph.D.

_____

Christopher Kiekintveld, Ph.D.

_____

Charles Ambler, Ph.D.
Dean of the Graduate School

To Liz

MULTI-OBJECTIVE BORDER PATROLLING OPTIMIZATION USING GAME THEORY AND EVOLUTIONARY ALGORITHMS

by

OSWALDO AGUIRRE

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

December 2014

ACKNOWLEDGEMENTS

I want to thank my parents, sisters, and advisor for all their unconditional support, and guidance that have led to the successful fulfillment of this great journey.

When I started this adventure, I never imagined what could happen or the way I would feel when I finished. Earning a doctoral degree is a very important accomplishment; it is not just another degree. Since the beginning of this doctoral journey, I realized that it would be something unique from the beginning to the end. At the beginning, I was alone in a new country with no friends, but that changed quickly. There were many people that gave me support, advices and guidance in this project. A pillar person that makes this degree a reality is my advisor Dr. Heidi A. Taboada. I will never thank her enough for all her continuous guidance and support.

Studying away, in a different city than your hometown, implies that you need to leave friends and family. This situation is not only hard to the person who leaves but also to the people that stayed behind. I am grateful to my parents that always supported me even if that meant to be away from them. I want to thank also my friends because when I needed their support they were always there for me with their words of encouragement giving me the strength that I needed to finish this project.

Now that I am at the end of this journey, I would like to recognize all the help that I received from my professors, they simply dedicate their lives to teach others. I also want to thank my friends and classmates of the Sustainability Engineering and Systems Optimization Lab that always were there when I needed them the most. Last, but not least, I want to thank God for showing me that with hard work and faith everything is possible.

ABSTRACT

Border security has evolved significantly since the days when no more than 75 Mounted Guards patrolled the U.S. border in the early 1900's. The border region includes thousands of miles of both land and maritime borders that must be controlled, as well as commercial transportation networks. One of the primary components of the Department of Homeland Security (DHS) is the U.S. Customs and Border Protection (CBP) agency. One of the most typical activities performed by CBP is patrolling the expansive open areas in between official points of entry (POEs) to prohibit illegal entry attempts. Roving patrols are routinely conducted by officers using many different modes of transportation depending on the terrain: foot, trucks, ATVs, boats, snowmobiles, etc. In addition to directly observing and banning illegal activity, CBP officers may conduct "sign cutting" operations which seek to identify recent signs of illegal entry (e.g., footprints or vehicle tracks). There are many factors that go into the patrolling policy for a particular area, including the type of terrain, the proximity to cities or major transportation arteries, the types of resources and infrastructure available in the area, the historical traffic pattern and apprehension rates, and so forth. The diversity and volume of illegal activity that must be controlled, and the variety of resources that can be deployed to secure the border make border security a complex mission. A key operational problem encountered by those charged with the task of border security is the scheduling and deployment of patrols. Given the complexity of the CBP patrolling problem (and other similar security/defense problems), it is clear that computational decision support tools have the potential to greatly improve resource allocation policies, and to reduce the burden on human schedulers. Therefore, this dissertation presents

new methods for generating solutions to large-scale patrolling optimization problems. Three different methods are presented which can be used for different scenarios. The first method presents a new framework for finding Pareto-optimal solutions to multi-objective patrolling problems. The objectives considered in the model are the minimization of worst idleness (amount of time that each location is without protection since last visit), the minimization of the infiltration ratio (success of the attacker to cross the border), and the minimization of the total patrolling cost. The second contribution is an extension to the first multiple objective patrolling problem. The second approach presents a hybrid model that combines a multiple objective evolutionary algorithm with game theory techniques to find an optimal patrolling strategy that provides the best objective values for the three objectives considered. For these two initial approaches, the solution methods are based on evolutionary optimization techniques in the interests of scalability and therefore, a variety of Pareto-optimal patrolling strategies are presented as the output to these problems. These Pareto-optimal solutions can then be presented to human analysts for decision making, visualization, or further post-Pareto analysis. Currently, the U.S. Border Patrol operates in three operational border enforcement environments: urban, rural, and remote. Models developed to protect urban areas do not necessarily perform well when applied to the protection of remote areas. Securing remote areas presents different challenges due to a wide range of terrain conditions, large areas to be covered, limited resources available, etc. It is in this environment that the U.S. Border Patrol does not have a persistent law enforcement presence and therefore seeks to increase its situational awareness without applying the same density of resources than resources allocated to urban and rural areas. The third

method presented in this dissertation describes the development of a decision support model to assists the U.S. Border Patrol with the deployment of the most appropriate CBP change detection resources available to effectively monitor the “remote” environment and to better respond to cross-border intrusions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Border Protection has changed significantly since the early 1900's when the patrolling task consisted of no more than 75 Mounted Guards patrolling the U.S. Border. Their main task was to intercept immigrants trying to enter the U.S. illegally and even then, success was intermittent.

Since the terrorist events of 9/11 border security became a priority for the new U.S Department of Homeland Security (DHS) created in 2003 as a direct response to the attack of 9/11. The interest in enforcing border security is reflected in the increased size of the Border Patrol from fewer than 3,000 agents to more than 21,000, the construction of nearly 700 miles of fencing along the southern border with Mexico, and deployed pilots drones, sensors, cameras and other new technology equipment to prevent illegal crossing at the land borders (Alden 2012). Besides wall enforcements and new equipment, new visa requirements were implemented. Foreigners can enter the United States legally in several ways. They can come temporarily as tourists, students or business travelers. They can immigrate permanently, either sponsored by a family member or on a basis of employment skills. A primary objective of border security is to ensure that only these legal channels are used to enter the country. Over the past years the CBP agency has increased border enforcement making crossing the border more difficult, more expensive, and more uncertain than ever before. Years ago illegal aliens preferred to cross urban areas, because they were harmless in comparison with the Arizona Desert. Recently this has changed; people choose remote areas because there is less surveillance. However, border control still remains imperfect. Most of Americans remain unconvinced that border security is improving; a Rasmussen poll

taken in May 2011 reveal that 2/3 of the public believes that the border with Mexico is not secure. Besides illegal immigration, there are other important threats faced by border security such as terrorism and drug smuggling. Following 9/11, preventing future terrorist attacks became the highest priority (All 19 hijackers entered the United States on legal visas). The attacks transformed border control from a public order issue into a national security topic. The Department of Homeland Security (DHS) was created largely by merging three border control agencies: The Customs Service, the Immigration and Naturalization Service, and the Coast Guard. Border immigration enforcement is the largest budget item for DHS, accounting more than half of the agency's budget (U.S Department of Homeland Security 2011:15)

Besides terrorist attacks, drug cartels are another real threat against border security. The border with Mexico is one of the largest drug smuggling corridors. Around 70% of all cocaine, up to 80 % of all marijuana, and 30% of the heroine entering the United States comes from Mexico (Pulat 2005). Today, border patrol agents encounter a much wider variety of illegal activity, and much more sophisticated criminal elements. An increasing trend is the presence of large, well-funded trans-national smuggling organizations that are involved in moving drugs, weapons, cash, and humans across international borders. As long as there is a demand of drugs, people, weapons or any other item that provides profit to illegal organizations, this activity will exist. For CBP to gain control over the borders, new methods and technologies need to be implemented to overcome these well prepared, organized and well financed illegal organizations.

One of the typical activities performed by CBP is the patrolling of open areas in between official points of entry (POEs) to interdict illegal entry attempts. Roving patrols

are routinely conducted by officers using many different modes of transportation depending on the terrain: foot, trucks, ATVs, boats, snowmobiles, etc. In addition to directly observing and banning illegal activity, CBP officers may conduct “sign cutting” operations which seek to identify recent signs of illegal entry (e.g., footprints or vehicle tracks). There are many factors that go into the patrolling policy for a particular area, including the type of terrain, the proximity to cities or major transportation arteries, the type of resources and infrastructure available in the area, the historical traffic pattern and apprehension rates, and so forth.

Given the complexity of the Border Patrol problem (and other similar security/defense problems), it is clear that computational decision support tools have the potential to greatly improve resource allocation policies, and to reduce the burden on human schedulers. Indeed, there is a large and growing literature on patrolling problems. However, there are some important features of the CBP domain that have been relatively neglected in the literature on patrolling. First, the considered problem is very large, both in the size of the physical domain and in the number of resources that need to be coordinated in the patrolling policy. Highly scalable algorithms are needed for cases with many patrolling resources. Second, the Border Patrol problem can be naturally perceived as a multi-objective optimization problem, due to the large variety of different types of illegal activity that CBP must consider in addition to costs and other factors. Most literatures on patrolling do not address this characteristic; rather they consider a single objective function making the model unrealistic. New methods are needed for generating solutions to large-scale, multi-objective patrolling problems considering all the constraints and unique characteristics that Border security requires.

Therefore, the problem to be solved becomes a multiple objective optimization problem. In order to solve such problem two approaches will be presented; the first one is based on an evolutionary algorithm and the second approach addresses the fact that adversaries can attain knowledge of agents' past routes to efficiently overcome agent strategies. In order to overcome this predictability, adversary reasoning is implemented in order to develop a more robust model.

Border enforcements in urban areas such as the model presented in this thesis have the side effect of making immigrants look for other areas with less surveillance. These areas are described by CBP agency as remote areas. The strategies used at these locations are different from the strategies used in urban areas. This thesis provides a model that can be used to efficiently allocate CBP resources at remote areas.

The thesis is divided into 9 chapters. Chapter 2 provides a brief background of how the problem has been addressed in the past as well as the methodologies and objectives considered in previous approaches. Chapter 3 describes the patrolling problem as well as some notations and objectives that will be optimized to solve the Border Patrol problem. Chapter 4 is a comprehensive review of the theory behind multiple-objective optimization and some of the most common methodologies that are used to solve these problems. Chapter 4 also includes a literature review of some post-Pareto methods, a vital part of any multiple objective optimization problem. The last part of Chapter 4 includes a review of different game theory techniques used to intelligently randomize patrolling strategies in order to overcome predictability, a main concern of any patrolling strategy. Chapter 5 is dedicated to describe the proposed methodology developed to

solve the problem as a multiple objective optimization problem. In this section, a customized multiple objective evolutionary algorithm (MOEA) is explained.

Chapter 6 presents the model introduced in section 5 modified to include game theory concepts in order to make the algorithm more robust and suitable for real world applications. Chapter 5 and 6 also presents a numerical example to show how the algorithm is implemented. Chapter 7 presents a detailed analysis of the results obtained in Chapter 6 and 7. Chapter 8 shows an additional patrolling model used to patrol remote areas. Finally, Chapter 9 presents some conclusions and discusses some important points that can be considering as part of future research.

CHAPTER 2: BACKGROUND & PREVIOUS WORK

Border Security is not a trivial activity. There is a lot of money, technological and human resources allocated to Border Protection. Due to the complexity and scale of the problem, it is not hard to find in the literature several journal publications presenting different approaches, models, and techniques to improve Border Security. The patrolling problem can be closely related to a type of network optimization problem due to the fact that many areas, in this case National Borders, can be modeled as networks. This area or network is patrolled by agents following different patrolling paths. Therefore, the nodes in the network are represented by check points or key location that needs to be secured. The roads used to travel from one location to another are the links in the network. There are many variations of path-finding problems on graphs that have been studied as challenging computational tasks, including the well-known Traveling Salesman Problem (Hahsler and Hornik 2007). Another well-known network problem related to security applications is the *Network Interdiction Problem* since this problem is aimed at detecting or interrupting flows in networks. For example, Wood (1993) addresses a specific security problem with the objective of reducing the flow of drugs in a river. He considers the problem in which an enemy attempts to maximize flow through a network while the defender (the interdictor) tries to minimize this flow by interdicting network arcs. This problem presents many similarities to the Border Patrol problem. The presented problem in Wood research is shown to be NP- complete even when the interdiction of an arc requires one unit of resources showing that even a relaxed instance is a very challenging problem. He addressed the interdiction problem using flexible integer programing formulation. In a similar manner (Cormican, Morton and

Wood 1998) based on Wood's work presented a stochastic version of the interdiction problem. A different approach that considers the network interdiction problem as a way to maximize the shortest path length in a network was also studied in the literature (Israeli and Wood 2002). All the solution methods used to solve network interdiction problems are different linear and integer formulations. An analysis of the maximum Flow Network interdiction integer programming formulations presented in Wood's work is analyzed by Altner *et al.* (2008). He proved that integrality gap of the LP relaxation of Wood's integer programming is not bounded by a constant factor. He provided an approximation-factor for the simpler R-Interdiction problem. Lim et al. (2011) presented a review of different algorithms used to solve different instances of the network interdiction problem. The models covered include bilinear programming, integer programming, and partitioning algorithms for discrete and continuous version of the problem. They stated the necessity of more effective algorithms for addressing the continuous version of the interdiction problem.

The similitudes or application of the max flow interdiction network problem with the Border protection patrolling problem presented in this thesis can be explained as follows. Consider an area that needs to be protected (in this case a border region) which is represented by a network. The attacker will try to maximize the amount of flow though this network (maximize illegal traffic) while the defender's (Border Patrol agents) mission is to interdict this traffic. The main objective of solving the patrolling problem is to find the best patrolling strategy that provides the most protection (coverage) to a given area. A patrolling strategy is defined by the actions that each agent will perform during the patrolling task.

Machado *et al.* (2002) explore the patrolling problem performing an empirical analysis. They describe the patrolling problem as a network problem where a network is visited by a defender in order to secure the specific area. They present different scenarios considering first the simplest case which is the “single agent patrolling problem”. In this case, the patrolling problem can be reduced to solve the traveling salesman problem. The natural evolution of the problem is to consider that more than one agent will perform the surveillance task simultaneously. As Machado’s team showed in their work, this addition raises the complexity of the problem. In each case the objective is to decide which routes or paths each agent has to follow in order to optimize the patrolling task. They called these routes patrolling strategies and are classified into different types such as cycle strategies and open strategies. Another important contribution of Machado’s work was the establishment of different quality measures to quantify the success of the patrolling strategy. One of these measures is the max idleness that represents the largest amount of time that a node in the network is without supervision since the last time visited. This measure is one of the most common measures used in other related works. For instance, Chevaleyre (2004) expanded Machado’s work by presenting a theoretical approach to the problem. In both studies, the complexity of the patrolling problem is clearly stated as well as the necessity of robust and advanced algorithms to solve such problems.

The multi-agent patrolling problem is considered as an NP-Hard (Non deterministic polynomial time hard) problem. As any NP-Hard problem, the patrolling problem is a very hard to solve using full enumeration or exact methods in order to obtain an optimal solution in a reasonable time. Instead of using exact methods,

heuristic methods represent a real option to solve an NP-Hard problem in a realistic time.

Glad *et al.* (2008) present a theoretical study that solves the problem of patrolling in an unknown environment using ant colony optimization. He solved the multi-agent patrolling problem considering a set of autonomous agents to visit all the places of an unknown environment as regularly as possible. Even though this work was not applied to real environments they showed that ant parading can be used to solve the patrolling problem obtaining stable cycle strategies. Lauri & Koukam (2008) introduced an approach that combines an evolutionary approach and ant colony optimization to solve the multi-agent patrolling problem. Lauri presents a very specific case of the problem considering two stages. At the beginning of the patrolling task all agents are at the same position and they need to leave the base and spread through the area. Then cycle routes are assigned to each agent once they are at the designed area. Each stage is evaluated in a different way using different algorithms. The multi agent patrolling problem shares many characteristics to the patrolling problem such as the fact that more than one agent patrols a specific area represented as a network. One important characteristic that is often omitted in literature is the fact that each agent can have different characteristics that can affect the performance of the patrolling task.

Another important area to analyze in different type of security applications is adversary reasoning. Many patrolling environments involve the interaction between two players, attacker and defender. In such scenarios the attacker has the ability to observe defender strategies in order to recollect knowledge, and come out with new attacker's strategies to overcome defender moves. This situation is real in the border patrol area

every day all attackers are more intelligent and sophisticated. The most used strategy used to overcome this predictability is to intelligent randomizing strategies in order to make agent's strategies difficult to predict. One example is the work of Paruchuri *et al*. (2004) which developed a framework for randomizing patrolling strategies that was deployed at the Los Angeles International Airport (LAX). Brown *et al.* (2006) presented a work that aims to defend critical structures against terrorist attacks. Gnanasambandam *et al.* (2004) described a work applied in military logistics for multi-agent systems considering the security of the agents, the stress of the battlefield, etc. Recently, Golalikhani *et al.* (2011) designed a model of optimal assignment of defensive resources between defender and attacker using a game theoretic approach. As can be observed from the previous works, game theoretical models are the most widely used strategy to deal with predictability. Being able to handle predictability is a key area in order to create a robust an efficient Border patrol model.

Besides human agents more automated systems have been introduced in Border security. There has also been work on developing strategies for automated patrolling by robots. Agmon *et al.* (2011) developed strategies for coordinating multiple patrolling robots, while Basilico *et al.* (2009) focuses on patrolling for arbitrary graph topologies, but with a limited number of patrollers. Work in this area has considered different types of objectives, ranging from detailed models of adversary reactions (as in the previous two papers), to simple frequency-based objectives that minimize the amount of idleness as in Elmaliach *et al.* (2009). Some of the most recent work focuses on very challenging cases where a patroller is trying to interdict mobile adversaries in a graph (Bosansky *et al.,* 2011).

The majority of the approaches to solve the patrolling problem found in the literature consider just one single objective function to be optimized. Adjoua(1996) presented a conflict analysis in a border area considering different objectives. Patrolling the Border area is a complex task that requires more than one objective function to provide a more realist quality measure to guarantee a good and efficient patrolling strategy. The primary novelty of our work is that we address patrolling problems with multiple objective functions – a very important case for real-world domains such as border security that has not been adequately studied in the literature. Next chapter will present in detail the patrolling problem to be considered.

.

CHAPTER 3: PROBLEM DESCRIPTION

This thesis introduces a formal model for the patrolling problem. Our motivating example is a border region between two countries, similar to the case of the U.S./Mexico border patrolled by CBP. The border region is divided into sectors, and within sectors into individual stations that are responsible for smaller regions of the border. Within these regions the number of resources for patrolling may vary. For our purposes we suppose that there are several agents available at any given time to conduct patrols to detect and interdict criminal activity. The main objective of the problem is to find a set of routes or strategies that each agent has to follow during the patrolling task in order to optimize Border security. In order to measure the quality of the strategies obtained three different objectives are considered.

We introduce three specific objectives below as a case study, which are motivated by the primary missions and patrolling activities of CBP. However, we note that our solution methods can easily be adapted to different sets of objectives, since our evolutionary algorithmic framework does not depend on the specific objectives used to evaluate the strategies. The three objectives we consider in this work are:

- **Idleness**: Minimization of maximum idleness, where idleness is defined by the time between visits for any node in the graph. This is denoted $\boldsymbol{WI}(\mathbf{x})$, where $\mathbf{x}$ is the defender strategy, which we describe in detail below. This is a frequency-based metric, and is suitable for activities like sign cutting where it may be desirable to minimize the time before detection to increase the probability of interdiction and reduce the deterioration of tracks and other signs of passage.

- **Infiltration**: Minimization of the infiltration ratio, which we define as the fraction of attackers that successfully transition from source to target nodes without being captured (denoted $IR(\mathbf{x})$). In this work, we estimate the infiltration ratio based on a fixed, probabilistic model of adversary behavior; in practice, this could be estimated based on data about previous apprehensions in different locations, or based on a more sophisticated behavioral model of how the adversary reacts to the patrolling strategy.
- **Cost**: Minimization of the total patrolling cost, defined by summing a cost function over each individual edge included in the patrolling strategy (denoted $TC(\mathbf{x})$). This can account for transportation costs, agent time, and other real costs of actually conducting patrols specified in the strategy **x**.

The Border patrolling problem is a complex and dynamic problem that has to be addressed as a multi-objective problem. Before defining the mathematical model of the problem some general notation is presented.

2.1 Notation & Assumptions

$G(V,E)$	Probabilistic graph
V	Number nodes or check points in the network
v	Node index, $v=1,2,3,\ldots, V$
E	Number of paths or links in the network
K	Total number of type of agents available
i	Agent index, $i=1,2,3,\ldots, K$
$\mathbf{s}_j(i)$	Patrolling strategy j performed by agent type i
$\mathbf{r}$	Attacker strategy
p	Number of single patrolling strategies performed
j	Index for a single strategy, $j=1,2,3, \ldots,p$

W	A matrix of size $\|V\|$ x $\|V\|$ that represents the distances to travel from node x to y
L_i	A matrix of size $\|V\|$ x $\|V\|$ that represents the time to travel from node x to y by agent i
x	Multi-agent strategy x ={$s_1(i)$, $s_2(i)$,…, $s_j(i)$,…, $s_p(i)$}
$WI(\mathbf{x})$	Worst idleness obtained by multi-agent strategy x
$\boldsymbol{c}$	Set of costs for each agent of type i
$\boldsymbol{\omega}$	Set of velocities for each agent of type i
$\tau_v(t)$	Idle time from previous visit to current visit at time t
T_v	Set of $\tau_v(t)$ values $T_v = \{\tau_1(t), \tau_2(t),\ldots, \tau_V(t)\}$ of node v
$\boldsymbol{\psi}$(x)	Set of T_v elements $\boldsymbol{\psi}(x) = \{\mathcal{I}(\tau_1), \mathcal{I}(\tau_2),\ldots, \mathcal{I}(\tau_V)\}$
TC (x)	Total cost of multi-agent strategy x
IR(x)	Infiltration ratio of multi-agent strategy x
A	Vector of which agent i will be selected for a specific patrolling strategy j
f_1(x)	Fitness metric 1: dominance count-based
f_2(x)	Fitness metric 2: distance-based
fa(x)	Aggregated fitness
psize	Population size
G	Number of generations
EL	Elitism parameter
cross	Crossover parameter
mut	Mutation parameter

2.2 Network Notation

The area to be patrolled is represented as a weighted graph $G(V,E)$, where V represents the set of vertices (nodes) and E represents the set of edges. Each vertex is an important location or waypoint that needs to be protected, or an intersection point where different paths can be selected. The edges represent the possible ways to travel between these locations. The weights on these edges represent the relative distance between nodes (including the travel conditions along the path), and serve as a metric for evaluating the cost of the patrolling strategy. We assume that the graph is

connected, so there is a path from any node to any other node in the network. This representation is a very general way to represent the physical topology of an area, and could be based on a simple grid system or a representation of roads or the transportation network in a real application.

We associate each vertex with a set of coordinates that are used for visualization purposes. We assume that the input graph including the coordinates and the path cost between each pair of nodes is provided by the user based on experience with the terrain and historical patrolling policies. An example of a simple network with 5 points is shown in Figure 1.

Point	x,y
1	1,1
2	7,1
3	7,9
4	4,13
5	1,9

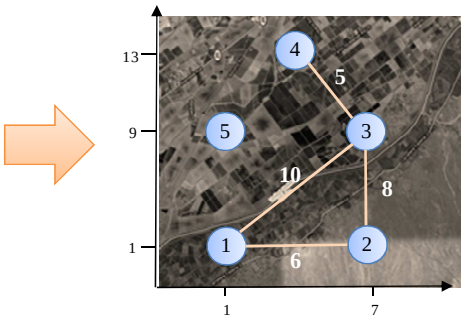


Figure 1: Network Example

Once a network is defined, we need to represent the possible policies that the security agents can use to move through this graph on patrols. The simplest case involves deploying a single agent. Even in this case, the problem becomes similar to the canonical Traveling Salesman Problem (TSP), in which a single agent must tour all of the nodes in a graph using the shortest possible path. The TSP is a well-known combinatorial optimization problem that belongs to the class of NP-complete problems; like many problems in graph theory, the primary difficulty is the exponential number of

possible paths that can be taken through the graph as the number of nodes and edges increases.

There are several algorithms that have been proposed to solve the TSP. Hahsler & Hornik (2007) present a recent review of the different algorithms to solve the TSP. Besides being formulated as a permutation problem, the TSP can be formulated as a graph theoretic problem with similarities to our patrolling problem. A graphical formulation to solve the TSP naturally leads to procedures involving minimum spanning trees for tour construction or edge exchanges to improve existing tours. Another alternative formulation is to model the problem as an integer linear programming problem as shown in Punnen (2002). Finding the exact solution to a TSP with n cities (nodes) requires checking $(n-1)!$ Possible tours. Evaluating all possible tours is infeasible for even small TSP instances, so heuristic methods that approximate the solution are needed. However, all the methods researched to solve the TSP consider that just one single agent travels through the cities.

In our problem description, a logical (and often necessary) way to improve performance on the patrolling task is to include more than one agent to patrol a given area. The present work considers the patrolling problem as a multi-agent patrolling model.

2.3 Patrolling Model

The first component of our model is the set of possible patrolling strategies. This is the set of paths in the graph that an individual patrolling agent can follow. We assume that these paths must be cyclic tours, so the agent must return to the starting location

(e.g., the agent's home station). Now, we allow for the possibility that there are multiple agents that can be assigned to perform patrols. The total number of patrols that will be performed is given by p. We also allow for the possibility that there are different types of agents available, which have different characteristics. The possible agent types are labeled by the indices $i=1,2,3,\ldots, K$. The patrolling strategy j performed by single agent of type i is denoted as $s_j(i)$. Therefore a multi-agent strategy can be defined as a set of patrolling strategies performed by p individual agents:

(1)

$$x = \{ s_1(i), s_2(i,),\ldots,s_j(i), \ldots,s_p(i)\}\ j=1,\ldots,p\ ;\ i \in \{1,\ldots,K\}$$

Where:

x multi-agent patrolling strategy

$s_j(i)$ patrolling strategy j performed by agent type i

j index of the single agent strategy $j=1,2,\ldots, p$

i Agent's type index $i=1,2,3,\ldots, K$

Each agent is assigned to follow a specific patrolling strategy j. However, not all agents are the same, so we define agent types to determine the different characteristics that affect the patrolling task. The agent characteristics we considered in this work are:

- Speed (velocity) of travel between nodes
- Cost of each agent

The speed or velocity of each agent determines the time to travel from one point to another. This is an important characteristic of real border patrolling problems, since agents may patrol on foot, ATVs, cars, or other vehicles. These values are represented

in vector **ω.** Another important factor to take into account is the cost for each type of agent. The cost is given in vector **c** and determines the cost associated with agent *i*. Table 1 provides an example of the information for vectors **ω** and **c** considering 10 different agents (*i*=1, 2, …, 10).

Table 1: Agent's Characteristics

Agent (*i*)	1	2	3	4	5	6	7	8	9	10
Velocities,	53	8	23	5	65	17	27	32	45	13
Costs, **c**	50	10	30	8	75	25	37	40	43	22

A multi-agent strategy involves more than one agent to patrol a specific area. The next step is to determine how many routes are allowed in the complete multi-agent strategy. The example presented in Figure 2 allows three routes to be traveled simultaneously (*p*=3) Note that the number of patrols in our model may be different than the number of possible agents, consequently the same type of agent can be assigned to multiple patrols, and some types may not be assigned to any patrols. The choice of the optimal strategy also involves choosing the best agents for the specific patrols. The mapping of which agent follows which route is represented by vector **a.** For instance, **a**= [1,2,1] means that route 1, $s_1(1)$, is followed by agent type 1, route two, $s_2(2)$, is followed by agent type 2, and route 3, $s_3(1)$, is followed again by agent type 1.

he main decision variable is the set **x.** This set specifies the *p* routes that will be performed, and which type of agent will perform each route. The vector **a** specify separately which type of agent is assigned to each route, and can be used to determine the cost of the patrolling strategy. The size of the set **x** is determined by the number of simultaneous patrols, *p* and the number of vertices *V* (length of set x= *p***V*). Figure 2

shows one possible multi-agent strategy and its respective vector **a** considering 3 single patrolling strategies.

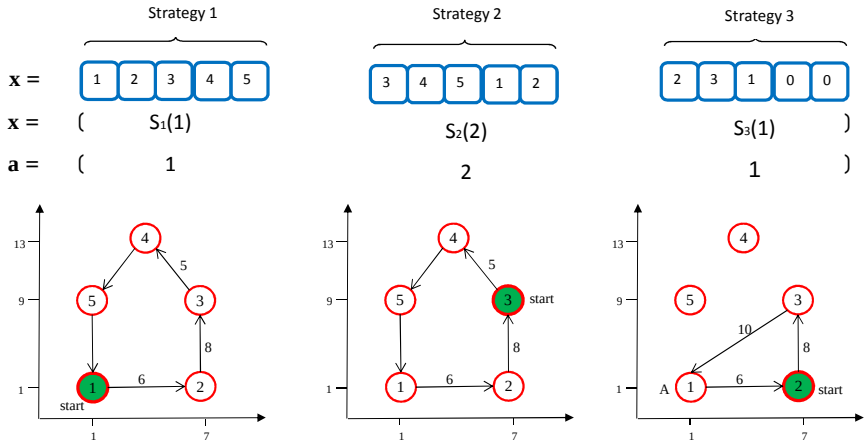


Figure 2: Complete Multi-agent strategy

Figure 2 show one example of a possible multi-agent patrolling strategy. This strategy shows the routes that each agent will follow during the patrolling task. The objective of this thesis is to find the patrolling strategy x that delivers the best objective values for the three objectives presented before. Therefore the mathematical formulation of the proposed multi-objective optimization problem is presented in Equation 2

(2)

$$Min\ \boldsymbol{WI}(\mathbf{x}) \quad , \quad \text{Min } \boldsymbol{IR}(\mathbf{x}) \quad , \quad \boldsymbol{TC}(\mathbf{x})$$
$$s.t.$$
$$\mathbf{x} \in \mathrm{X}$$

Where:

x: represents the decision variable (multi-agent patrolling strategy)

X: Represents the feasible region of solutions.

Once the problem is formulated the next step is to solve the multiple-objective optimization problem presented in Equation 2. Multiple-objective optimization problems

have specific characteristics that make them not very easy to solve. There are plenty of works and models that target this type of problems. An extensive review of multiple-objective optimization is presented in Chapter 4.

CHAPTER 4: MULTIPLE-OBJECTIVE OPTIMIZATION

Real life optimization problems usually involve different goals that are in conflict with one another, and taking the best decision is complicated. Humans often make decisions on the basis of intuition, common sense, experience, and a combination of all of the above. However, in areas such as engineering, decisions have to be supported by a scientific methodology. These types of problems that involve more than one objective or goal, and have to be optimized simultaneously are known as multi-objective optimization problems or multi-criteria decision making problems (Kaisa, 1999).

The multi-objective optimization field is so broad and has been intensively studied in literature. In a single objective optimization the best solution is selected based in the objective considered, in such problems the selection of the optimal solution is an easy task just one unique solution will deliver the best objective value. It is characteristic in multi-objective optimization problems that a unique optimal solution does not exist. The reason is the natural trade-off between conflicting objectives that makes impossible to achieve a solution that is optimal for all the objectives at the same time. The conflict between objectives makes the optimization of one objective decrease the optimal value for the other. This scenario makes impossible to select a unique solution with the best optimal values for all objective values. However, a set of mathematically equal good solutions can be obtained. This set of all feasible solutions whose vector of various objectives is not dominated by any other is known as Pareto set of optimal solutions (Diakonikolas, 2011) Pareto optimality is defined as a state in which it is impossible to make any objective better without making at least one objective worse off. Pareto

optimality in muti-objective optimization problems has been studied previously in literature. Censor (1977) extended the work presented by Milyu-Tin and produces the necessary conditions for local Pareto optima. Daruwala(2002) presented in his dissertation an analysis of computing the Pareto Optimal solution set in a large scale dynamic network.

Solving multi-objective optimization problems consists in optimizing several conflicting objectives and simultaneously finding a Pareto set of optimal solutions (Branke *et al.*, 2008).

The process of optimizing can be understood to choose the best element from some set of available alternatives for all the considered objectives.

The mathematical model for a multi-objective optimization problem is presented in Equation 3.

(3)

$$Optimize\ F(\vec{x})$$

$$subject\ to: \Omega = \{\vec{x}\ \in R^n |\ G(\vec{x}) \geq 0\}$$

Where $\vec{x}$ is a vector of decision variables $(x_1, \dots, x_n)$ and $F(\vec{x})$ is a vector of objective functions $\left(f_1(\vec{x}), \dots, f_K(\vec{x})\right)$. Here $f_1(\vec{x}), \dots f_K(\vec{x})$, are functions on R^n and Ω is a non-empty set in R^n. The vector $G(\vec{x})$ represents constraints that may be handled such as lower and upper bound on the variables.$f_1(\vec{x}), \dots f_K(\vec{x})$, represent the conflicting objectives to be optimized.

In multi-objective optimization the objective function $F(\vec{x})$ constitute a multidimensional space. For each solution $\vec{x}$ in the decision variable space R^n there is a point in the objective space Z denoted by $f(\vec{x}) = z = (z_1, \dots, z_K)^T$

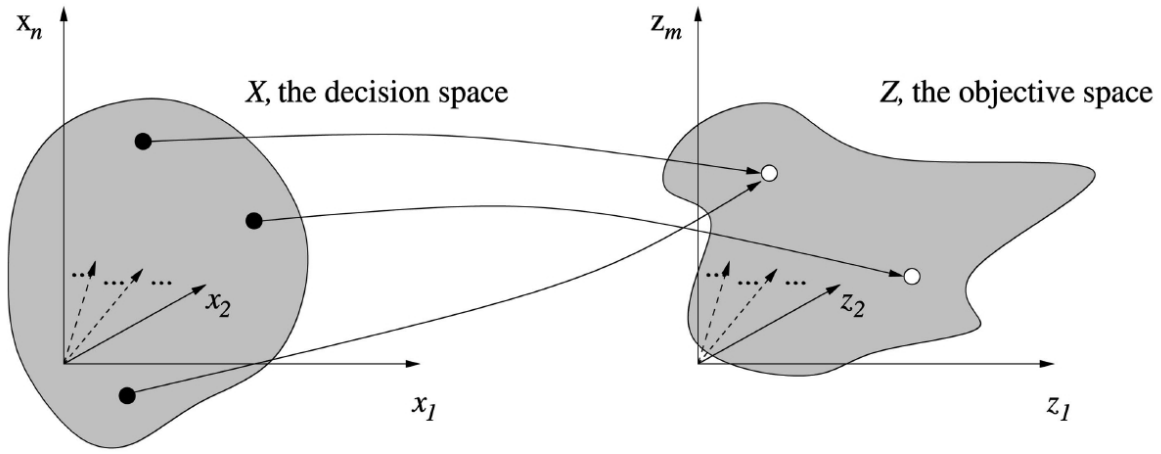


Figure 3: Two Dimensional Spaces

A multi-objective optimization problem is convex if all objective functions are convex and the feasible region is convex. Since a multi-objective optimization problem has two spaces, the convexity must be analyzed on both spaces. Therefore, although the search space can be non-convex, the Pareto optimal front can be convex.

In literature several methodologies have been developed to address optimization problems that involve multiple conflictive objectives. These methods can be divided in two big groups:

- Mathematical programming methods
    - Goal programming (Charnes, Cooper & Ferguson,1955)
    - Weighted sum method
    - Lexicographic method (Fishburn, 1974)
    - Utility functions

- Meta-heuristic methods
    - Tabu-search
    - Simulated annealing
    - Evolutionary algorithms
    - Ant colony
    - Swarm intelligence

4.1 Mathematical Methods

Mathematical methods transform the problem into a single objective problem and in most cases use linear programming to solve the problem.

4.1.1 Goal Programming

Goal programming (GP) was first used by Charnes, Cooper & Ferguson (1955). It is based on linear programming optimization, and it is one of the first methods developed specifically to solve Multiple Objective Optimization Problems (MOOP). At first this method was designed to solve Multi-Objective Linear Problems (MOLP).

GP aspiration levels (also called contribution coefficients) are assigned for each objective that is to be optimized, and then the deviations from these aspiration levels are minimized. In other words in the GP model the objective function is the summation of all the contribution coefficients and the objectives are constrains of the problem. The mathematical formulation for a GP is presented in equation 4.

$$Z = \sum_{i=1}^{n} c_j x_j \tag{4}$$

$$s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, for\ i = 1, \dots, m$$

$$x_j \geq 0, for\ j = 1, \dots, n$$

Where $x_1, x_2, \ldots, x_n$ are the decision variables, and $c_1, c_2, \ldots, c_n$ are the contribution coefficients that represent the contribution to Z for each objective function. $a_{i,j}$ are coefficients that represents the per unit usage by x_i of the right-hand side coefficient of b_j.

For this linear programming (LP) model some assumptions are considered:

- Each unit for each decision variable x_i contributes c_i units to the objective function and a_{ij} in the constraint.
- The contribution to the objective function and the coefficient in the constraints are independent of the values of x_i
- All parameters a_{ij}, b_j, and c_i must to be known

This method represents one of the most used methods to solve MOOP due to the simplicity of implementation. However, it does not optimize all of the objectives at the same time and in some complex problems a feasible solution cannot be obtained. Some modifications of GP have been proposed in literature such as the weighted sum method explained in the next section.

4.1.2 Weight Sum Method

This method is based in the GP method with the significant difference being that each objective function has a different weight or importance for the decision maker (DM). The general model for the weighted sum method is presented in equation 5

$$Z = \sum_{i=1}^{n} w_j\left(c_j x_j\right) \tag{5}$$

$$s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, for\ i = 1, \ldots, m$$

$$x_j \geq 0, for\ j = 1, \ldots, n$$

Where w_i represents the weight assigned for each objective.

This methodology is easy to implement for any problem that can be modeled as GP. The advantage of this method is that some weights of importance can be added for each objective. In application where one objective is more important than others this method can be used. However, the weighted sum method has the same disadvantages that GP has plus the fact that defines the weights for each objective can be ambiguous.

4.1.3 Lexicographic Method

This method assigns importance to each objective being optimized. However, in the lexicographic ordering (Fishburn, 1974) the more important objective is infinitely more important than the less important objective. At first the DM assign importance to the objectives$f_1(x), f_2(x), \dots, f_n\ (x)$. The most important objective is optimized without considering the other objectives Equation (6)

$$\min f_1(x) \tag{6}$$

$$s.t.\ x \epsilon X$$

Then the second objective function is optimized subject to the original constrain and a new one that guarantees the optimality of the first objective function. The procedure continues until the last objective function is optimized. This method is easy to implement, but choosing the order or of importance of the objectives can be an issue. Another disadvantage is that the method is very rough and often the process stops before less important objective functions are optimized, because the problem becomes infeasible before finishing to optimizing all the objectives.

In order to make the method more versatile or more suitable to practical problems some modifications have been proposed to the lexicographic method. The δ-lexicographic method tries to overcome some of the drawbacks of the lexicographic

method, allowing small increments of the first objective to be trade off with the decrement of the second objective.

Even though the lexicographic method has several disadvantages and does not optimize all the adjectives simultaneously, any lexicographic solution is efficient and can be proven to be Pareto optimal.

4.1.4 Utility Theory

Utility function, also known as a value function, is a mathematical expression that assigns a value to all possible choices. Utility function is a very good method used to solve multiple criteria problems when an explicit mathematical formulation for the value function is known. The problem can be defined by Equation 7

$$\begin{aligned} &maxi\ v\big(f(x)\big) \\ &s.t..x \in X \end{aligned} \tag{7}$$

Since the value function provides a complete ordering in the search space, a most fitting Pareto optimal solution is found with this method. Unfortunately, the real problem is to obtain this value function. Getting this value function can be very difficult, if not impossible, in multi-objective optimization problems. Even if the function is known it can be very difficult to optimize because of its complicated nature. However, if the value function can be defined this method becomes very effective and useful and can be applied to any kind of multiple criteria optimization problems (Maged *et al*., 2005).

All the methods mentioned before are mathematical methods that can be applied to several multiple objective optimization problems. However, they present important drawbacks that makes them impractical to apply in complex problems where the search space is very large or when the objective function is difficult to obtain analytically. Another alternative taken into account is represented by meta-heuristic methods.

Instead of obtaining a global optimal solution meta-heuristic methods obtain an estimated solution. These methods have proved to perform well in a variety of complex combinatorial problems getting good approximations to the true Pareto front with relatively low computational effort. These methods are explained next.

4.2 Meta Heuristic Methods

Heuristic methods are usually implemented to solve multi-objective optimization problems. Some examples include Tabu search, neural networks, and evolutionary algorithms among others. For instance Abbass, *et al.*(2001) proposes a partial differential evolution approach for multi-objective optimization problems. Zitzler, *et al* (2004) present a review of several evolutionary algorithms used to solve Multi-objective optimization problems.

4.2.1 Tabu-search

Tabu-search is a meta-heuristic search method presented first by Glover in 1986. Tabu search (TS) permits the incorporation of procedures for searching the solution space economically and effectively. The memory system is one of the most important aspects of the TS method and the main difference with other meta-heuristic methods that are memory-less such as genetic search and simulated annealing approaches. While other exploration methods keep in memory the objective value $f(x)$ of the best solution visited so far, TS also keeps information of the characteristics of the last solution visited. This information is used to move from one solution to another. Memory can be used to identify elements that are common to good solutions or to paths that lead to such solutions. The flexibility of these memory structures allows the search to perform well in a multi-objective environment (Hertz *et al.2002*).

Tabu search begins in the same way as ordinary local or neighborhood search, proceeding iteratively from one solution to another until a chosen termination criterion is satisfied. For instance consider a simple optimization. The pseudo code in Figure 4 presents the outline for the TS method.

1. Chose $x \in \mathrm{X}$
2. Find $x' \in \mathrm{N}(x)$ such that $f(x') < f(x)$
3. If no such x' can be found, x is the local optimum and the method stops
4. Otherwise, designate x' to be the next new x and return to step 2

Figure 4: TS Structure

TS has been implemented in many optimization problems in different fields and have proved to be a good meta-heuristic algorithm. Some of the more important features of TS are: easy implementation for almost any optimization problem and that can be easily modified to enhance the algorithm performance.

4.2.2 Simulated annealing

This algorithm was proposed by Metropolis *et al.*as an algorithm that simulates the evolution of a solid to thermal equilibrium (Laarhoven & Aarts, 1987). The name comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. By analogy with this physical process, each step of the simulated annealing (SA) algorithm replaces the current solution by a random "nearby" solution.

Given a neighborhood structure, simulated annealing continuously attempts to transform the current configuration into one of its neighbors. The acceptance of this neighbor is given by a probability presented in equation 8

$$p = exp\,(-\delta f(x) / T) \tag{8}$$

Where δ represents the increase in $f(x)$ and T is an important control parameter in simulated annealing which by analogy represent temperature. Some of the advantages of the method include:

- Performs well with nonlinear models, chaotic and noisy data and many constraints.
- Flexibility and ability to approach global optimal solution and escape for a local optimal.
- Easy implementation

SA is not a perfect optimization method; some of its disadvantages are stated next:

- Since SA is a meta-heuristic, a lot of choices are required to turn it into an actual algorithm and the optimal solution is not guaranteed.
- There is a clear tradeoff between the quality of the solutions and the time required to compute them because the search is random.
- The tailoring work required to account for different classes of constraints and to fine-tune the parameters of the algorithm can complicated.

4.2.3 Ant Colony Optimization.

The original ant colony optimization algorithm is known as Ant System and was proposed by Dorigo *et al.,* (1996). The underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained results. The collective behavior emerging from the interaction of the different search threads has proved effective in solving optimization problems.

Artificial ants also have some extra features that real ants do not. They have memory of their previous actions, and they can use that data to improve the quality of

computed paths. In many cases, pheromone is updated only after having constructed a complete path and the amount of pheromone deposited is usually a function of the quality of the complete path.

The process of the algorithm is described next.

1. The first ant finds the food source (F), then returns to the nest (N), leaving behind a trail pheromone (b)
2. Ants indiscriminately follow possible ways, but the strengthening of the runway makes it more attractive as the shortest route.
3. Ants take the shortest path, long portions of other ways lose their trail pheromones.

ACO was first applied to the traveling salesman problem searching for the shortest path. Recently the method has been applied to a variety of engineering problems. And also some methods have been applied to solve MOOP. Ant colony optimization has also very common choice in patrolling problems due to the similarity to the TSP

4.2.4 Evolutionary Algorithms.

Evolutionary algorithms (EA) are based in evolutionary computation. The principle of these methods mimics how biological evolution works. It was first presented by J. H. Holland. EAs are population-based meta-heuristic optimization algorithms and there are several kind of these methods implemented for specific situations but all of them follow the structure presented in Figure 5.

Figure 5: EA Structure

In the first stage a first population of possible solution is generated in most cases randomly. The next step is selection. In this stage the strongest individuals are chosen to be parents of the next generation. The chosen parents undergo to reproduction and a new population is generated. The algorithm continues until a stopping criterion is met. It can be observed in figure 3 that an EA is divided in four main stages: initialization, evaluation, selection, and reproduction. The success of an EA in searching for a good solution depends on how these stages are performed in the algorithm.

EAs have proved to perform well in several kind of MOOP. One of the more important advantages is the flexibility that EAs have to escape from a local optimal to a global optimal exploring more efficient the search space. The next section explains the most important stages and parameters for a general evolutionary algorithm.

4.2.4.1 Encoded Strategies

The first step in an EA is initialization or generation of the first population. A First population is formed for several proposed solutions. These solutions are called individuals or chromosomes. A chromosome can be defined as string of components

that have a meaning or correlation to a real problem. Each component of the chromosome is called gen and contains important information of the generated solution. Some of the most common encoded strategies in EAs are presented below:

- Binary coded: it is one of the most used due to its simplicity. Binary encoded refers to the values that gene can contain, 0 or 1. These values usually represent active or disconnected components. An example of this encoding is presented in Figure 6.

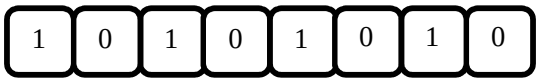


Figure 6: Binary Encoding

- Permutation encoding: this encoding strategy allows integer numbers in the chromosome. These values usually represent a position in a sequence or a number of components in certain position. This encoded normally does not allow repeated values in the string. A sample chromosome using this encoding strategy is presented in Figure 7.

Figure 7: Integer Encoding

- Value encoding: this kind of encoding is more difficult to implement in an EA. Value coding allows any type of number: integers and real numbers and also letters. An example is presented in Figure 8.

Figure 8: Value Encoding

4.2.4.2 Selection Strategies

An important point in EAs is selecting the better individual for a population. Normally a fitness function or the objective value is used to make this selection. How

this fitness function is obtained is discussed later. Some of the well-known strategies used in EAs are:

- Roulette wheel selection: this method works as roulette. All the solutions are grouped respecting to its fitness value. The solutions that have better fitness values has more probability to be selected
- Tournament selection: two solutions are chosen randomly and the solution with the higher fitness is selected
- Rank selection: all the solutions are ranked from higher fitness to lower and these solutions that are better ranked are selected.

Selection is a very important step in EAs. However how recombination or offspring are generated is equally important. Some strategies used are presented in next section

4.2.4.3 Crossover Strategies

The main purpose for a crossover is to generate better individuals getting the best genetic material from each parent. To achieve that, several crossover operators have been proposed in literature. Some of approaches can be seen the work of Li *et al.* (2005) and Zhang *et al.* (2007). The main concept behind the improvement of a crossover operator is trying to distinguish between bad genes and good genes. Zhang *et al.* (2007) used hill-climbing search to find good blocks in a chromosome. Another similar approach is presented by Li *et al.* (2005) who uses the rough set theory to identify good genes in a specific individual. Another important aspect of a crossover operator is the ability to create diversity of children in order to efficient exploration of the search space. Therefore, it is important to select a crossover operator that can create

good offspring and a good diversity for the network reliability problem. Some of the most well-known crossover strategies are presented next.

- Single point crossover: this is the most common crossover operator. Its behavior is presented in Figure 9 The point that states where the parent is divided is chosen randomly

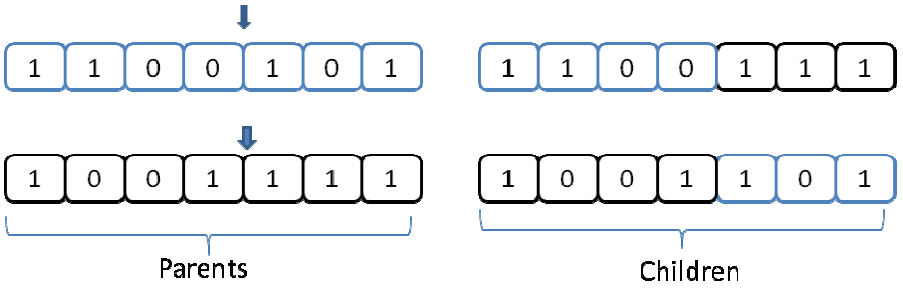


Figure 9:Single Point Crossover

- Double point crossover: this strategy works in similar way than the previous one, but instead of select one point to points are selected randomly. Figure 10 presents this strategy.

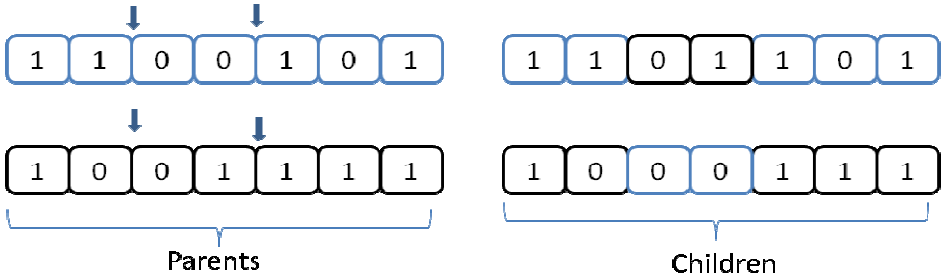


Figure 10: Double Point Crossover

These strategies are the most common used in EAs but it doesn't mean that these crossovers strategies are the more efficient or the only ones. Some variation of the double point that propose multiple points have been proposed.

In order to solve multi-objective optimization several EAs have been proposed in literature these algorithms are called multi-objective optimization evolutionary algorithms (MOEAs) and are explained in the next section.

4.2.5 Multi-objective Evolutionary Algorithms MOEAs

Some of the more common MOEAs reviewed in literature are:

- Strength Pareto Evolutionary Algorithm (SPEA) by Zitzler and Thiele (1999)
- Strength Pareto Evolutionary Algorithm (SPEA 2) by Zitzler (2001)
- Pareto Archived Evolution Strategy (PAES) by Knowles (1999)
- Non-dominated Sorting Genetic Algorithm (NSGA II). by K. Deb (2002)
- Adaptive Pareto Algorithm (APA). Dumitrescu (2001)

All of the previous mentioned algorithms are MOEAs and all of them use evaluation, selection, and reproduction to obtain a new generation. The differences are how they assess fitness evaluation and reproduction. Some of them used mutation as a reproduction procedure others combine elitism and crossover to generate a new population and others use tournament selection to select the best children for the new population. All of those algorithms have advantages and disadvantages. A brief description of each algorithm is presented in the next.

- Strength Pareto Evolutionary Algorithm (SPEA): It was developed by Zitzler and Thiele. The algorithm maintains an external population at every generation storing all nondominated solutions obtained so far. At each generation both populations current and external are mixed. All nondominated solutions in the mixed population are assigned fitness based on the number of solutions they dominate. Giving better scores to solutions that dominate more solutions. At the end deterministic clustering method is used to ensure diversity among nondominated solutions.
- Strength Pareto Evolutionary Algorithm (SPEA 2): Zitzler, Laumanns and Thiele proposed a variant of SPEA. In SPEA2 after fitness evaluation, all nondominated solutions from current population and from external population are passed into the

next population. If the number of these solutions is less than the population size then the next population is filled with dominates individuals from both population. The main difference between SPEA and SPEA2 is how fitness functions are calculated. SPEA2 uses a fine-grained fitness assignment that incorporates density information that identify individual with same fitness value.

- Pareto Archived Evolution Strategy (PAES): It was developed by Knowles and Corne. In this algorithm the crossover operator is different. A parent generates one offspring by mutation. If the offspring dominates the parent, the offspring is accepted as the next parent and the iteration continues. If the parent dominates the offspring, the offspring is discarded and the new offspring is generated. If the offspring and the parent do not dominate each other, a comparison set of previously nondominated individuals is used. For maintaining diversity along Pareto front, an archive of nondominated solutions is archived. A new offspring is compared with the archive to verify if it dominates any member of the archive. If so, then the offspring enters the archive and is accepted as a new parent. The dominated solutions are eliminated from the archive. If the offspring does not dominate any member of the archive, both parent and offspring are checked for their nearness with the solution of the archive. If the offspring resides in the least crowded region in the parameter space among the members of the archive, it is accepted as a parent and a copy is added to the archive.
- Non-dominated Sorting Genetic Algorithm (NSGA): It was developed by K. Deb and his students. Initially a random population, which is sorted based on the no domination criterion, is created. Each solution is assigned fitness equal to its no

domination level. Binary tournament selection, recombination and mutation are used to create a children population. A combined population is formed from the parent and offspring population using elitism criteria. The population is sorted according to the no domination relation. The new parent population is formed by adding the solutions from the first front and the followings until it exceeds the population size. Crowding comparison procedure is used during the population reduction phase and in the tournament selection for deciding the winner.

- Adaptive Pareto Algorithm (APA): This algorithm uses a new technique called Adaptive Representation Evolutionary Algorithm. The main idea of this technique is to allow each solution be encoded over a different alphabet. Moreover, the representation of a particular solution is not fixed. Representation is adaptive and may be changed during the search process as effect of mutation operator. The algorithm uses a single population of individuals. Initial population is randomly generated. Each individual is selected for mutation, which is the unique variation operator. The offspring and parent are compared. Dominance relation guides the survival.

All the heuristic methods presented as well as all the evolutionary approaches focuses on obtaining a set of nondominated solutions. After the Pareto set is obtained the mission of the decision making is to select one among the Pareto set of solutions to be implemented in the field. This is not a trivial task due to the large amount of solutions that can be obtained and the different preferences the decision maker can have regarding the importance of all the objectives considered. The next section covers a

review of a very important of the Multi-objective optimization process that is called post Pareto optimality analysis.

4.3 Post Pareto Analysis

Solving a multiple-objective optimization problem involves two stages; the optimization stage and the post-Pareto analysis stage. The first stage consists in finding a set of nondominated solutions that represents the solution of the problem. However, one question remains: which solution from the set should be selected? Most of the work in multiple-objective optimization found in the literature spends a lot effort in developing algorithms and models in order to find a true Pareto set of nondominated solutions. However, the decision making stage is as important as the first one. Selecting one solution over others or at least reducing the number of alternatives to choose from is not a simple task since the Pareto-optimal set can potentially contain a very large number of solutions

Literature shows some approaches related to Post-Pareto analysis. For instance Venkat *et al,* (2004) address post-optimality analysis introducing the Greedy Reduction (GR). This method works by obtaining subsets of the Pareto front based on maximizing a scalarizing function. Venkat uses a rank to define their functions. Another approach was presented by Padhye *et al,* (2009). Their work proposes a mutation driven hill climbing local search using achievement scalarizing function to refine the solutions from the Pareto set. Kacem, *et al,* (2003) develop a hybrid between Fuzzy Logic and evolutionary algorithms (EAs) in order to construct a satisfactory set of solutions. All the approaches presented before deal with the problem by reducing the number of solutions

in the Pareto set by eliminating not satisfactory solutions. Well defined method to select which solutions are considered good solutions still ambiguous.

Another approach consists in groping the solutions into subgroups or clusters that contain similar solutions. For each cluster a reference solutions is selected (closer solution to the cluster centroid). Using this approach the number of solutions is reduced to the number of clusters. In other words the reduced Pareto set contains as many solutions as clusters were defined. There are several clustering methods that have been developed to classify data. This approach has been explored by Taboada *et al*, (2005) using k- means method. Another similar approach was presented by Runkler (2006) using a fuzzy c- means clustering model that is also based in the k-means model. These models have been criticized for two main reason; the necessity for validation of the solution (different cluster can be found depending of initial guess) and the fact the number of clusters has to be defined before implement the method. Taboada *et al*. (2005) suggested the use of silhouettes values based on work presented by Rousseeuw (1987) to define what number of clusters is appropriate for a specific set of data. This validation procedure evaluate different amount of cluster and based in a silhouette value. The number of clusters that delivers the best value is selected. However this is an additional procedure that has to be performed before the clustering method. Therefore, a method that automatically defines the number of clusters is desired. The main idea of these approaches is group the data into clusters. Each group is formed by similar data (solutions from the Pareto set), after finding the clusters one solution from each cluster can be selected as a representative solution for each cluster. Figure 11 shows the basic flow of how a clustering procedure works.

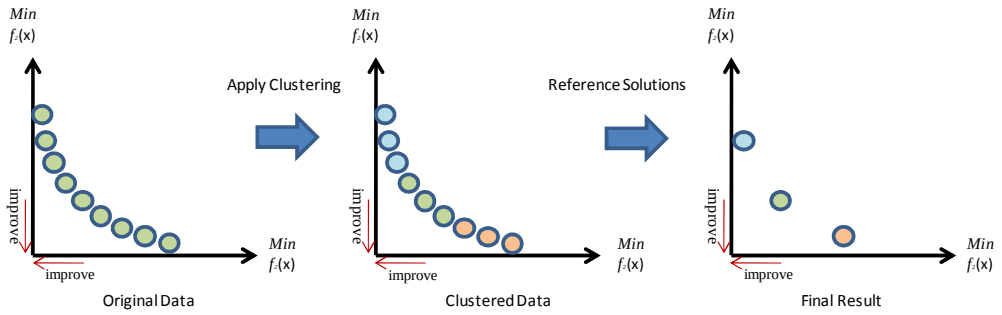


Figure 11: Clustering Procedure

The post Pareto optimality problem can be represented as a classification problem where a specific amount of data (solutions) has to be classified into clusters.

4.3.1 Clustering

Cluster analysis can be defined as the groping of a set of data into subsets (called clusters) so that data (solutions) in the same cluster are similar in some sense. Cluster analysis does not refer to a specific method or algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. The most common notions of clusters used include groups with low distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. The appropriate clustering algorithm and parameter settings depend mainly on the individual data set and intended use of the results. The two main branches of clustering algorithms are partitional and hierarchical methods.

4.3.2 Partitioning Clustering

The k-means algorithm (partitional) is probably the most widely applied partitional clustering technique (Kaufman & Rousseuw, 1990). The grouping is done by calculating

the centroid for each group, and assigning each observation to the group with the closest centroid Figure 12 show an example of this procedure.

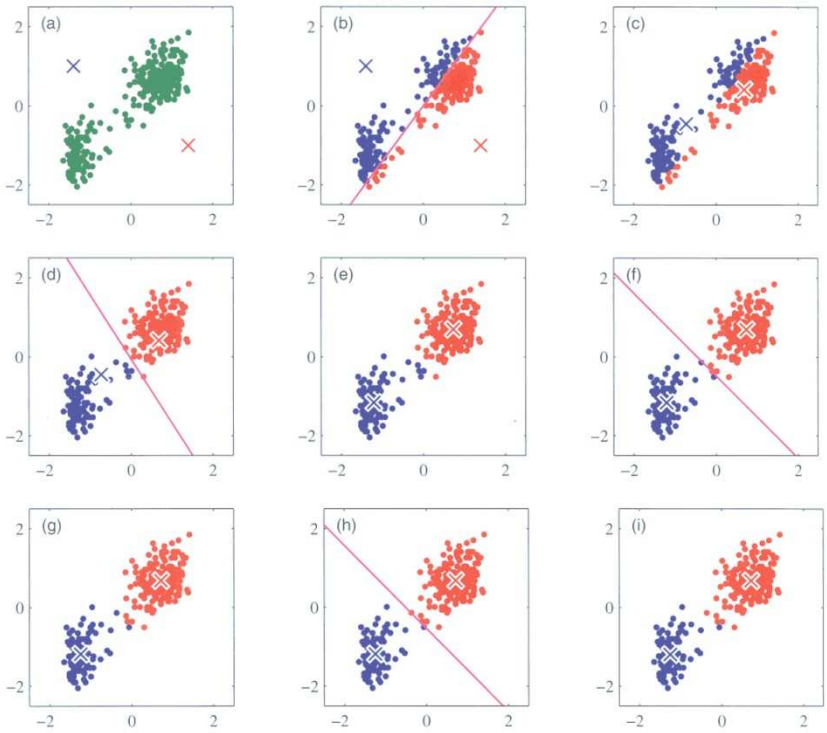


Figure 12: K-means Algorithm

4.3.3 Hierarchical Clustering

Hierarchical algorithms find successive clusters using previously established clusters. These algorithms usually are either agglomerative ("bottom-up") or divisive ("top-down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Figure 13 show an example of this procedure. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

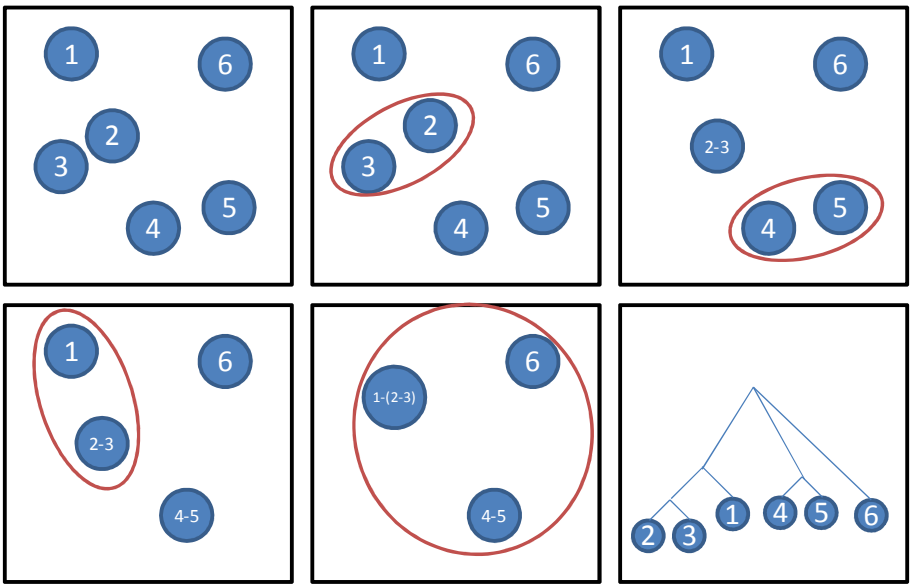


Figure 13: Hierarchical Bottom-up Algorithm

Each clustering method has its advantages and disadvantage and the selection of the method depends basically in the application. Some of the most important disadvantages of k- means and hierarchical methods are presented in Table 2

Table 2: Clustering Methods Comparison

K-means	**Hierarchical clustering**
Number of k cluster has to be defined at the prior algorithm execution	Doesn't give discrete clusters Need to define clusters with cut-offs
Fixed number of clusters can make it difficult to predict what K should be.	Cannot return to previous hierarchical level to reassign misclustered data
Different initial partitions can result in different final clusters. It is helpful to rerun the program using the same as well as different K values, to compare the results achieved.	Selection of merge or split points is critical as once a group of objects is merged or split, it will operate on the newly generated clusters and will not undo what was done previously.

Solving the Patrolling Problem as a multiple objective problem involves obtaining a set of nondomitaded solutions. Therefore a method that can help to reduce the amount of solutions obtained is needed.

Beside the inherent complexity of obtain a Pareto set of solutions that optimize the three different objectives patrolling problems face a very challenging issue predictability. The border area allows the intruders to observe defender movements and recollect data in order to overcome agent's strategies. Therefore, even the efficiency of patrolling strategies is high repeating the same routine over and over again is a critical disadvantage. There are several strategies that have been explored in literature in order to overcome predictability. The next section will explain game-theory techniques one of the most used frameworks to analyze defender attacker scenarios.

4.4 Game Theory

Game theory refers to branch of mathematic that models interactive decision-making between multiple individuals of groups that have different interests (Gura & Maschler, 2008). The concept of game theory was first presented by John von Neumann and Oskar Morgenstern in the publication of *Theory of Games an Economics Behavior.* They present a theory of decision making under risk called the theory of utility. This theory of utility has become a common used tool in many human science and economics (Bacharach, 1997) .

The basic idea of game theory is to predict what the other player will do and take advantage of that knowledge to make the right decision that offers the best reward. Game theory defines this situations or scenarios where players have to make decisions for instance which routes to patrol. A game can model conflicts of interest between two

players. (Cross the border and stop the intruder). Game theory consists of ways of analyzing these problems in order to help the player to make the right decision at the right time (Thomas, 1984).

The main reason of Neumann's work was the Second World War, because many military problems can be modeled as games. However, there are plenty of scenarios where game theory can be applied. Wherever people interact, wherever there are strategies to adopt and outcomes or prizes to win, a game is played. Game theory can be applied in many different fields such as economics, evolutionary biology, social interaction, military operations, traditional games such as poker, and many others situation that involve decision making between two or more players (Binmore, 2007).

4.4.1 Game Characteristics

4.4.1.1 Utility

Game theorists study rational decision making, describing satisfaction for a decision made by means of utility. Consider a person who likes the taste of candies but dislikes vegetables. He might be said to associate higher utility with states of the world in which, all else being equal, he consumes more candies and fewer vegetables than with states in which he consumes more candies and fewer vegetables. This example suggests that 'utility' denotes a measure of subjective psychological fulfillment. The economist Paul Samuelson (1947) therefore set out to define utility in such a way that it becomes a purely technical concept. Since Samuelson's re-definition became standard in the 1950s, when we say that an agent acts so as to maximize his utility, it is meaning by 'utility' the level of reward derived from a specific player decision. Theorists who follow Samuelson intend the statement 'agents act so as to maximize their utility' as a

tautology, where an ‘(economic) agent’ is any entity that can be accurately described as acting to maximize a utility function, an ‘action’ is any utility-maximizing selection from a set of possible alternatives, and a ‘utility function’ is what an economic agent maximizes.

4.4.1.2 Games information

A crucial aspect of the specification of a game involves the information that players have when they choose strategies. An extensive-form game represents the sequencing of players' possible moves, their choices at every decision point, information each player has about the other player's moves when he makes a decision, and his payoffs (utility) for all possible game outcomes. There are three different cases considered in game theory regarding the information available.

4.4.1.2.1 Perfect and complete information

Complete information describes a game in which knowledge about other players is available to all participants. Every player knows the payoffs and strategies available to other players. Perfect information describes the situation when a player has available the same information to determine all of the possible games (all combinations of legal moves) as would be available at the end of the game.

Although similar, complete and perfect information are not identical. Complete information refers to a state of knowledge about the structure of the game and the objective functions of the players, while not necessarily having knowledge of actions inside the game.

The game presented in Figure 14 present a game with two players and perfect information. The numbers by every non-terminal node indicate to which player that

decision node belongs. The numbers by every terminal node represent the payoffs to the players. The labels by every edge of the graph are the name of the action that that edge represents.

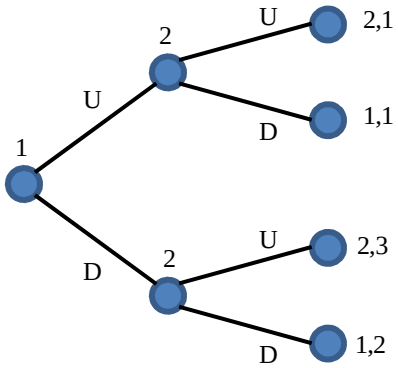


Figure 14: Extensive Form with Perfect Information

4.4.1.2.2 Imperfect information

In this type of games one player does not always observe the choice for the other player. In extensive form, an information set is indicated by a dotted line connecting all nodes in that set. In the game presented in Figure 15 it is clear that player 2 does not know the move that player 1 did.

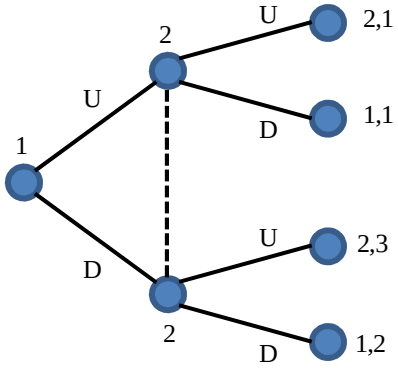


Figure 15: Imperfect information Game

4.4.1.2.3 Incomplete information

If a player does not know exactly what the payoffs of the game are or of what type his opponents are it is considered a game with incomplete information. Figure 16 shows a game with complete information (all players knows the payoff), but imperfect information.

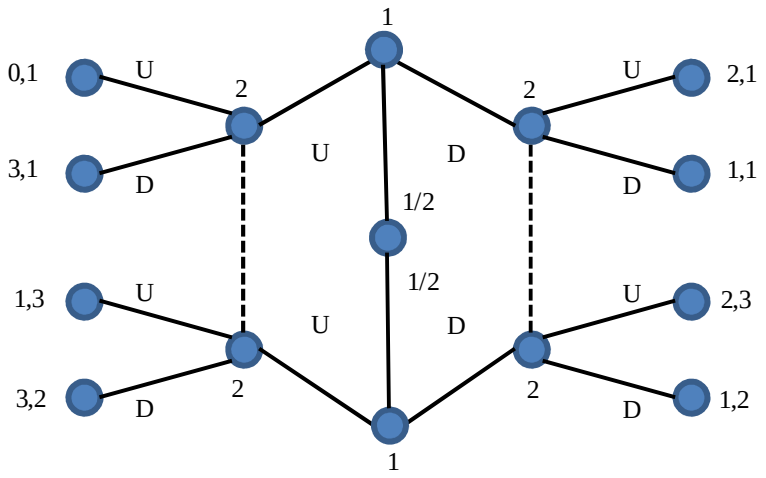


Figure 16: Imperfect Information Game

4.4.2 Game Representation

In order to represent games, two different representations are used. One is the extensive form where the game is presented as a tree, as in the examples were presented in Figures 14-15. The other form is called normal form. In this case instead of a tree a matrix is used to represents the strategies to each player and the values inside the matrix represent the payoffs. Table 3 shows the normal form of the prisoners dilemma, game a very well-known game.

Table 3: Normal Representation

Player1/Player 2	Cooperate	Defects
Cooperate	1,1	1,0
Defects	0,1	.5,.5

4.4.3 Game Solution

A solution for a game is a formal rule for predicting how the game will be played. These solutions can describe which strategies will be adopted by players, therefore predicting the result of the game. Solutions may involve pure strategies, in which each player uses only one play or mixed strategies, in which actions are selected randomly according to some probability distribution. Four techniques that can be used to characterize the game solution are described.

4.4.3.1 Nash Equilibrium

Nash equilibrium is a strategy profile in which every strategy is a best response to every other strategy played. If there is no other strategy that could be played that would yield a higher pay-off in any situation in which the other player's strategy is played. In other words Nash equilibrium is a decision that independently which strategy the opponent selects will lead a better pay-off or reward for both players.

4.4.3.2 Backward induction

There are games that have multiple Nash equilibrium, some of which are unrealistic. In the case of dynamic games, unrealistic Nash equilibrium might be eliminated by applying backward induction, which assumes that future play will be rational. It therefore eliminates non credible threats because such threats would be irrational to carry out if a player was ever called upon to do so.

4.4.3.3 Forward induction

Forward induction is so called because just as backward induction assumes future play will be rational, forward induction assumes past play was rational. Where a

player does not know what *type* another player is (i.e. there is imperfect and asymmetric information), that player may form a belief of what type that player is by observing that player's past actions. Hence the belief formed by that player of what the probability of the opponent being a certain type is based on the past play of that opponent being rational. A player may elect to signal his type through his actions.

4.4.3.4 Iterated elimination of dominated strategies

The iterated elimination (or deletion) of dominated strategies is one common technique for solving games that involves iteratively removing dominated strategies. In the first step, at most one dominated strategy is removed from the strategy space of each of the players since no rational player would ever play these strategies. This results in a new, smaller game. Some strategies that were not dominated before may be dominated in the smaller game. The first step is repeated, creating a new even smaller game, and so on. It is possible that in any step 0 strategies may be deleted for some players. The process stops when in any round 0 strategies are deleted. This process is valid since it is assumed that rationality among players is common knowledge, that is, each player knows that the rest of the players are rational.

There are two versions of this process. One version involves only eliminating strictly dominated strategies. If, after completing this process, there is only one strategy for each player remaining, that strategy set is the unique Nash equilibrium.

As can be observed there are many similitudes of the problems analyzed by game theory and the Border patrolling problem. This chapter covered all the methodology needed to solve the presented optimization problem. As can be inferred the problem is very complex in nature that make the necessity to implement different

type of strategies to obtain a robust methodology that can be applied to real world scenarios.

The presented problem in Chapter 3 will be solved in two different scenarios the easiest scenario will consider that problem without apply any game theory techniques. This is important understand deeply all the characteristics of the problem and define which areas are more predictable. Chapter 6 will present a improve version of the methodology presented in Chapter 5.

CHAPTER 5: MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

The multi-agent patrolling problem was presented in chapter 3 as a multi-objective optimization problem with three different objectives to be optimized simultaneously. These objectives are conflicting objectives. This means that improving one objective decreases the value of another objective. This is a classic scenario faced in any multi-objective optimization problem. As in any multi-objective optimization problem the solution of the problem is not represented as a single objective. Instead a set of solutions is obtained. This set is called the Pareto set. This set is formed by all nondominated solutions founded in the optimization process. The concept of Pareto optimality and the strategies used to solve these problems was presented in Chapter 4.

The main objective of the multi-agent patrolling problem is to find the best way to patrol. The problem arises when it is optimized to find the routes or paths that each agent has to follow in order to increase Border security. The concept of security can be very subjective. Therefore, in order to measure the performance of a specific multi-agent patrolling three different objectives were considered. These three objectives are explained in detail in this chapter.

5.1 Objective functions

5.1.1 Objective Function 1: Minimization of the Maximum Idleness, (WI(x)):

The first objective considered is the minimization of the worst idleness $\boldsymbol{WI}(\mathbf{x})$ This value represents the amount of time elapsed since a specific node has received a visit of an agent This objective was deeply explained by Machado *et al* (2002) and

Chevaleyre. (2004) and is one of the most common quality measures used in patrolling problems. To illustrate the concept of worst idleness consider two different locations that are patrolled by a single agent. If the agent uses the following strategy (cyclic route) [1,2,1,2,1] and the speed is constant the max idleness can inferred from Figure 17. This figure shows the position of the agent at different points in time during the patrolling task. At time t=0 the agents starts in node 1 and the Idleness for each node is zero. At time 1 the agents reaches node 2. At this point node one is one unit of time without supervision, so the idleness for node 1 at time 1 is equal to one. At time two the agent reaches node one again. Just a moment before agent reaches node one the idleness of node 1 is 2 units of time. Therefore, the WI for this specific example is equal to two units of time.

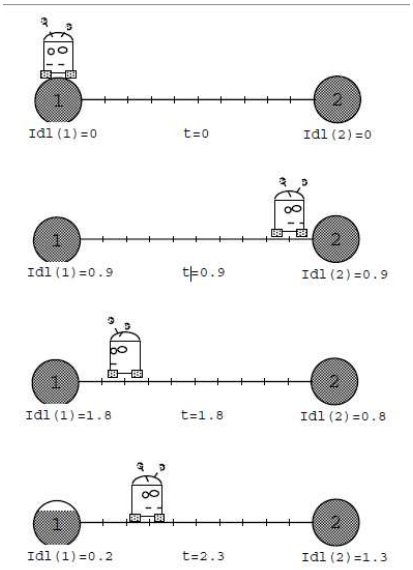


Figure 17: Worst idleness

The minimization of $WI(x)$ makes sense because reducing the time that a point in the network is without protection reduces the probability that an attacker can reach that

point. Reducing the WI= 0 will be the ideal scenario meaning that all point are covered all the time. This situation is typically not achieved in real world applications

The network $G(V,E)$ to be covered is represented by a set of coordinates. Therefore, we can define a matrix **W** of size |V| x |V| in which each element $W_{x,y}$ represents the distance to travel from node *x* to node *y*. Considering the distances in **W** and the velocities given for each agent given by vector **ω,** The matrix L_i of size |V| x |V| is defined, in which each element $L_{i(x,y)}$ represents the time needed to travel from node *x* to node *y* by agent *i.* This matrix represents the per-agent travel time. Additionally, let τ_v (*t*) be the idle time since the last visit of a specific node or vertex *v* at time *t, w*here *t* is the time at which point *v* is visited for any agent involved in the patrolling task.

For each specific multi-agent strategy **x**={s_1, …, s_p} and a specific interval of time t_0 (starting time) to t_f (final time) [t_0, t_f] there are *n* number of τ_v(t) values where *n* is the number of visits that vertex *v* is visited for all the agents during the complete patrolling task (from t_0 to t_f) Therefore, $\mathbf{T}_v$ is the set of all the idle times for specific vertex *v* from t_0 to t_f. For instance, consider the multi-agent strategy **x**={[1,3,4,5,0], [3,5,1,2,0], [2,3,1,0,0]} and assume that the velocity for all agents is equal to one. The values of τ_1 (idle times for node 1) from t_0=0 to t_f = 70 are presented in Table 4.

Table 4: The values of τ_1 (idle times for node 1) from t_0=0 to t_f = 70

Node 1	$\mathbf{T_1}$	$\tau_1(0)$	$\tau_1(14)$	$\tau_1(18)$	$\tau_1(28)$	$\tau_1(36)$	$\tau_1(42)$	$\tau_1(54)$	$\tau_1(56)$	$\tau_1(70)$
		0	14	4	10	8	6	12	2	14

Let **ψ(x)** be the set **ψ**(**x**) ={$\mathcal{I}(\tau_1), \mathcal{I}(\tau_2), \ldots, \mathcal{I}(\tau_V)$} that represents all the recorded idle times for all the nodes for a specific multi-agent strategy **x.** Therefore, the worst idleness is given by Equation 9.

$$WI(\mathbf{x}) = \text{Max}\,[\boldsymbol{\psi}(\mathbf{x}) = \{\mathcal{I}(\tau_1), \mathcal{I}(\tau_2), \ldots, \mathcal{I}(\tau_V)\}] \quad (9)$$

By considering the minimization of the worst idleness we are reducing the amount of time that each point or node in the network is without protection and at the same time we also are minimizing the chances the attacker has to surpass a specific checkpoint without detection (e.g., though sign cutting).

Reducing Max idleness really improves the patrolling task quality. It is a static measure that does not depend on attacker movements which means that it does not matter what the attacker does. If the attacker crosses to a specific location and the time the location is unprotected is very low, then the probability of the agent to stop illegal traffic is very high. However, vast areas do not allow small enough values that deliver an acceptable detention probability. The second objective is focused in the attacker movements in order to minimize the success rate of the attacker to reach specific points in the network.

5.1.2 Objective Function 2: Minimization of the Infiltration Ratio, (IR(x)):

The main objective of the adversary is to cross the border without being interdicted by a patrolling agent. If the attacker reaches a specific point in the network, then the attacker has succeeded. Specific nodes in the network are defined as a starting points and target points. In a real application, these points might represent

staging areas on one side of the border, and safe houses or major cities on the other size of the border.

Figure 18 show an example that represents a classic patrolling scenario. Agents perform different patrolling strategies and attackers tries to cross the border using their own routes or strategies.

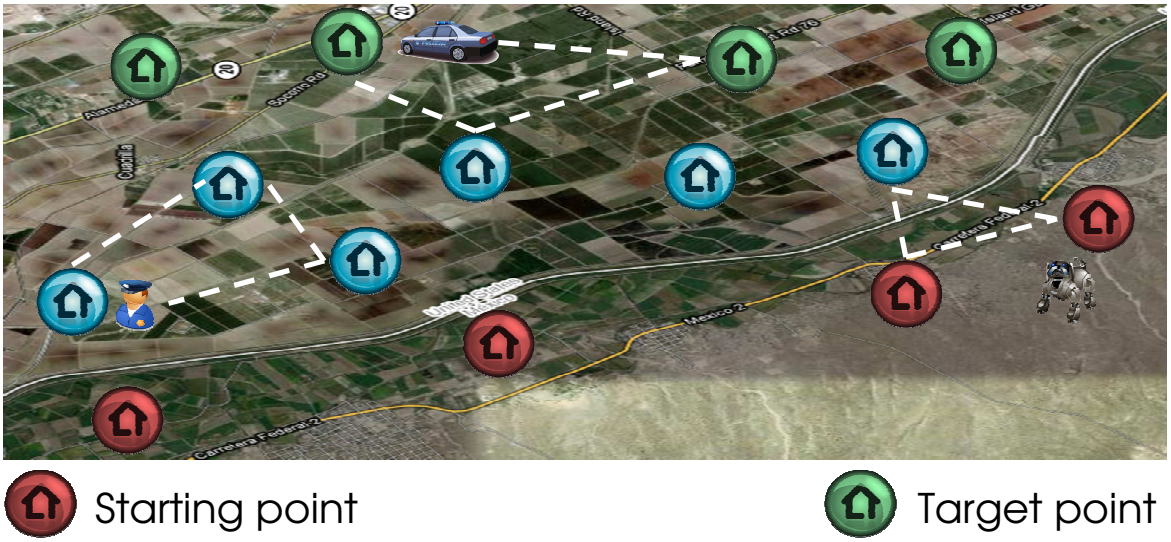


Figure 18: Patrolling Scenario

The second objective we consider in this section is to minimize the attacker's infiltration ratio, which is a measure of how often the attacker is able to cross the border from a source to a target without being caught. We assume that every patrolling agent is 100% efficient at capturing adversaries if they are in the same location (this could be relaxed fairly easily in the model).

The attacker strategy is defined by vector **r.** Vector **r** represents the path (sequence of nodes in the travelled route) that the attacker uses to cross the border**.** All attacker strategies must start in a designated source node and end in a designated

target node. Figure 19 shows an example of an attacker strategy in which the attacker starts in node 1 and ends in node 4, **r** = [1,2,3,4].

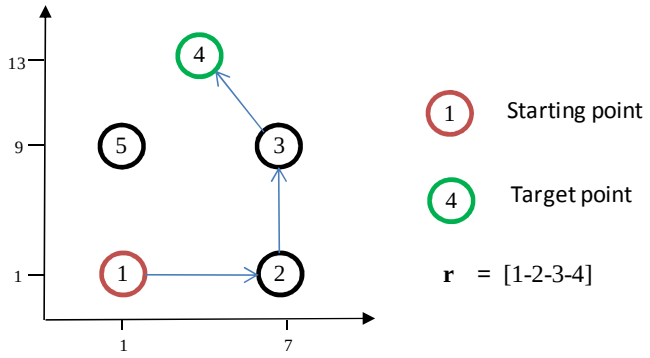


Figure 19: Attacker Strategy

Due to the large number of routes that the attacker can use to cross though a large network, we estimate the infiltration ratio using Monte Carlo simulation for a particular multi-agent patrolling policy. We assume a fixed random distribution of attacker paths (so the attacker's strategy does not depend on the patrolling policy). The simulation generates several attacker paths which start at different time steps during the simulation. It then calculates the probability that there is a patrolling agent in the same location as the attacker. This calculation depends on the specific patrolling policy. If the defender fails to catch the attacker, then the attacker succeeds. The infiltration ratio $\boldsymbol{IR}(\mathbf{x})$ is given by Equation 10:

$$\boldsymbol{IR}(\boldsymbol{x}) = \frac{Q}{\#\ of\ iterations} \tag{10}$$

The pseudo-code to evaluate the infiltration ratio is presented next:

```
1. Initialize Q=0 (successes)
2. Select a multi-agent patrolling strategy.
3. Randomly generate an attacker strategy (route and starting time t=0
```

4. Check if the attacker reaches the target point without being captured by the defender, if that occurs update the value of Q with Q=Q+1.
5. Repeat steps 3-4 for a large number of *iterations,* (i.e., 10,000)
6. The infiltration ratio is obtained by *IR*(x)= Q/*iterations*

From Figure 18 it can be concluded that focusing on critical nodes (starting and target nodes) in the likely paths of attackers the patrollers may be able to increase the interdiction rate. However doing so may increase the time that others nodes in the network are without protection. This situation harms the first objective (idleness), so these two objectives are potentially in conflict.

5.1.3 Objective Function 3: Minimization of the Total Patrolling Cost, (TC(x)):

The third objective function we consider in this paper is to minimize the total cost of the patrolling task. Even though achieving security is the most important aspect, achieving high security at the lowest cost is also a very important consideration in real world patrolling tasks. Naturally, reducing the cost of patrolling is typically in conflict with improving security. To evaluate this objective we sum the costs over each agent involved in the patrolling task, as shown in Equation 11:

$$TC(\mathrm{x}) = \sum_{l=1}^{p} c[a(l)] \tag{11}$$

Where:

c **=** Set of costs for each agent

a **=** Vector that defines which type of agent is selected for a specific patrolling task **x**

p **=** Total number strategies involved in the patrolling task **x**

Now that we have introduced the individual objectives, we can state the full multiple-objective formulation of the patrolling problem as:

$$\text{Min } \boldsymbol{WI}(\mathbf{x}) \quad , \quad \text{Min } \boldsymbol{IF}(\mathbf{x}) \quad , \quad \boldsymbol{TC}(\mathbf{x}) \qquad (12)$$

$$s.t.$$

$$\mathbf{x} \in \mathrm{X}$$

Where:

x: represents the decision variable (multi-agent patrolling strategy)

X: Represents the feasible region of solutions

5.2 MOEA

The methodology proposed in this thesis to solve the multi-objective patrolling problem in equation 12 is a multi-objective evolutionary algorithm presented by Taboada *et al* (2007) and modified to be applied to the patrolling problem. The proposed algorithm is based in other MOEAs presented in Chapter 4 the main structure of the algorithm is presented in Figure 20.

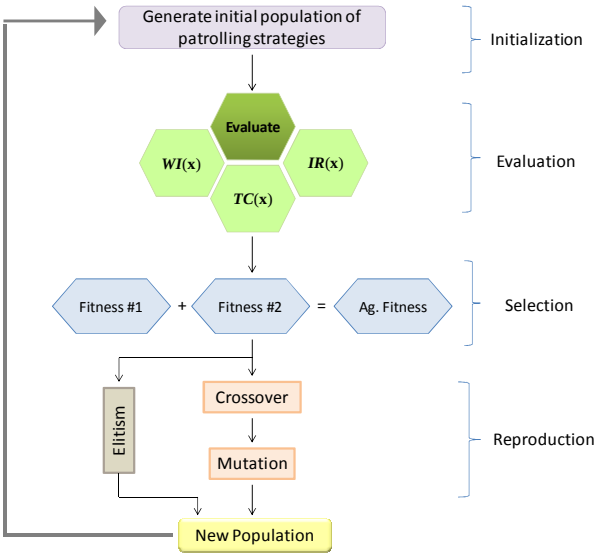


Figure 20: MOEA Flowchart

5.2.1 Initialization

The first step of the algorithm is to generate the initial generation of possible solutions. This step is of key importance because all the next generations will be created from the first one. The way many proposed evolutionary algorithms deal with this issue is to generate the first generation randomly to obtain a diverse set of solutions. The way in which the random generation affects the performance in terms of the number of iterations it takes to find the approximate solution of the algorithm has been the topic of different previous works. For instance, Maaranen *et al.*, (2004) presented and studied of how Quasi-random initial population affects GAs. On the other hand Karci (2004) proposed a method of generating the initial population by using upper and lower bounds of variables instead of a pseudo-random sequence. Park *et, al* (2003) considers also that generating a good initial population is very important and they used G&T algorithm to generate the initial population for a scheduling problem.

In the presented algorithm, a specific number of patrolling strategies (**x** vectors) are randomly generated. Each multi-agent patrolling strategy represents one possible solution to the patrolling problem. The purpose of generating the initial population randomly is to obtain a variety of genetic material in order to cover the search space of the problem efficiently. Otherwise if there is not enough genetic diversity a good solution will be very difficult or almost impossible to be reached. Figure 21 shows an example of initial population for the patrolling problem.

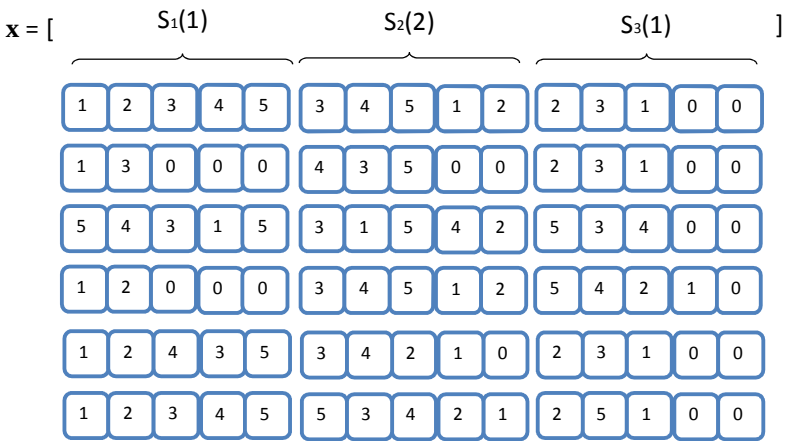


Figure 21: Initial Population

5.2.2 Evaluation

The evaluation of the objectives considered for the patrolling problem was presented earlier in Chapter 5. After all objectives are evaluated for all of the solutions, dominated solutions are removed from the set and just nondominated solutions survive to the next stage of the algorithm. Figure 22 shows all the solutions for the initial generation with their respective objective values and the nondominated solutions of the set.

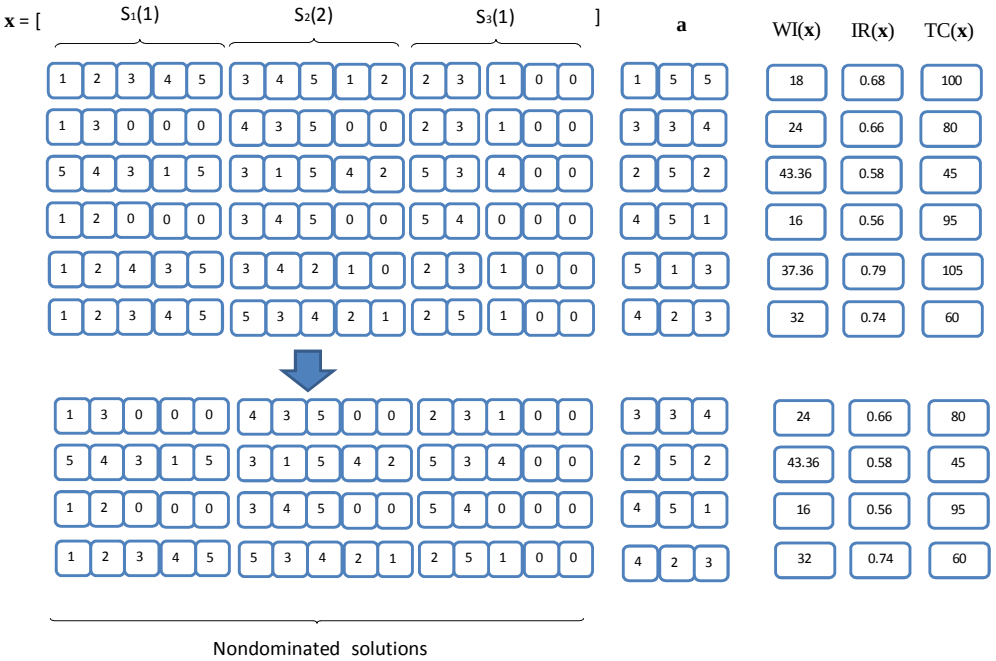


Figure 22: Evaluated and Nondominated Solutions

5.2.3 Selection

The best solutions have to be selected in order to become parents of future generations. It is believed that strong parents will generate strong children. Therefore the selection process gains importance in any evolutionary algorithm. In order to select the best solutions among the set of nondominated solutions, two fitness functions are considered (Taboada *et al.*, 2008). The main idea for a fitness function in evolutionary algorithms is to measure the *quality* of the represented solution. For instance Sano & Kita (2000) propose a GA for optimization of continuous fitness functions with observation noise utilizing history of search so as to reduce number of fitness evaluation. Kuncheva(1997) proposed an editing technique for the *k*-nearest neighbor (k-NN). Another approach treats the problem using a vectorizing fitness function. In order to obtain a quality solution the proposed MOEA considers two different fitness functions. The first fitness metric, $f_1(x)$, is a dominance count-based metric. It aims to select individuals which are more dominating (intended to achieve proximity). While the second fitness metric, $f_2(x)$, is distance-based. This metric is intended to maintain population diversity. Solutions that are farther away in respect to other solutions (Euclidian distance) have better fitness values. Both metrics will be explained in detail in next sub sections.

5.2.3.1 Fitness metric # 1 (f_1)

This fitness metric intends to obtain solutions that are close to the true Pareto front. The true Pareto front is defined as the set of nondominated solutions that are obtained when all the possible solutions in the problem are calculated. In other words the true Pareto is the set of global Pareto optimal solutions. However, the true Pareto

front is never known. Hence, selecting solutions that are close to the true Pareto front is not straightforward. This fitness metric is dominance count based. The main idea behind this metric is that solutions that are close to the true Pareto front tends to dominate more solutions than solutions that are far away from the true Pareto front (see Figure 23). Notice that the solution that is closest to the true Pareto front dominates more solutions than other solutions.

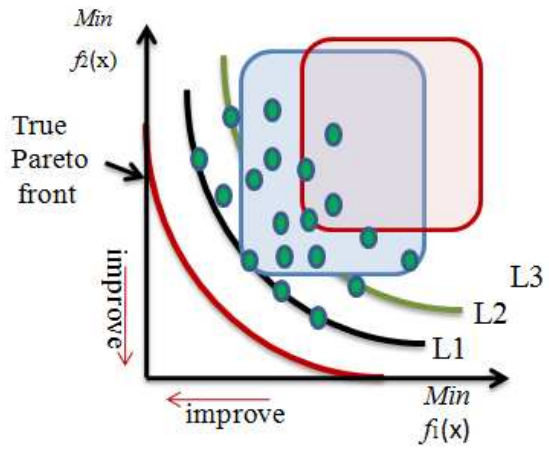


Figure 23: Dominance Count Based Fitness Concept

The purpose of this metric is to give a better fitness value to solutions that dominate more solutions, and provide a lower value. Let us consider the population presented previously in Figure 22. Figure 24 shows the dominance count for each solution and the given fitness value for each solution.

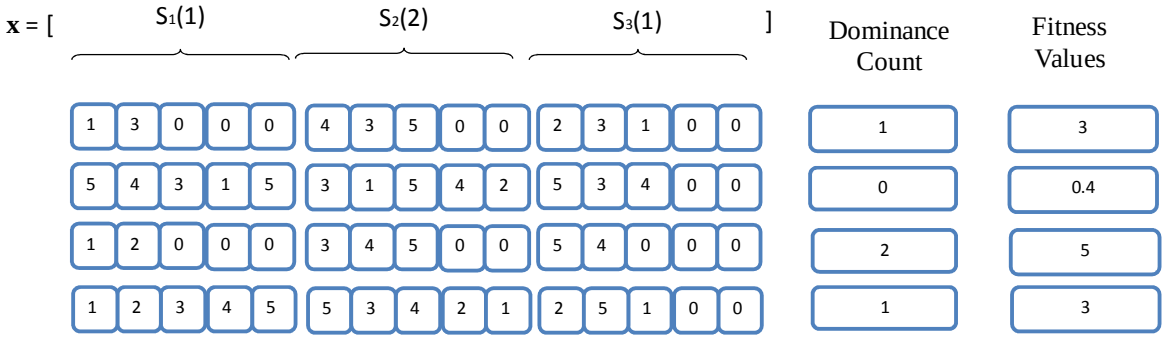


Figure 24: Fitness #1 Evaluation

The fitness values are calculated as follows: Let us consider the solutions in Figure 24, the maximum dominance count is 2. Then consider 5 intervals and divide the maximum dominance count by the number of intervals. Then give a fitness value depending in the interval that each solution belongs. The fitness values for each interval are presented in Table 5. The number of intervals can be defined by the designer.

Table 5: Fitness Values for Intervals

Fitness value	*Intervals*
1	0 ≤ Dominance count < 0.4
2	0.4 ≤ Dominance count <0.8
3	0.8 ≤ Dominance count < 1.2
4	1.2 ≤ Dominance count < 1.6
5	1.6 ≤ Dominance count ≤ 2.0

5.2.3.2 Fitness metric #2 (f_2)

This fitness metric intends to achieve diversity of solutions. Sometimes generated solutions cover just one part of the search space and these solutions normally generate solutions near to them. This situation is not desirable because there is a risk to stay in a local optimal and not find a global optimal solution. In order to prevent this situation fitness #2 (distance based metric) is used. This metric gives a better value to solutions that are far away from other solutions. Doing so assumes that solutions that are far away from one another will also generate solutions far away from them. Figure 25 shows a scenario where similar solutions are arranged into clusters. The objective of this fitness metric is to select solutions from different areas instead of selecting solutions from the same cluster.

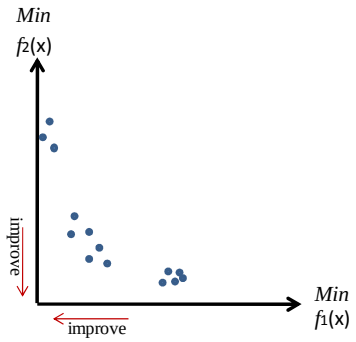


Figure 25: Solution in Clusters

Fitness metric #2 is distance based and is evaluated as follows: Consider again nondominated solutions from Figure 13. The first step is to normalize all the objectives in order to have all the objectives in the same scale (see Table 6).

Table 6: Normalized Objectives

Non dominated Solution	WI(**x**)	IR(**x**)	TC(**x**)
1	0.2923	0.5556	0.7000
2	1	0.1111	0
3	0	0	1
4	0.5846	1	0.3000

The next step is to compute the Euclidean distance from each solution to the others: the sum of the distances from each solution to the rest of the solutions is obtained, and the maximum and minimum value of all the sums is calculated (see Table 7).

Table 7: Euclidean Distance

Individual	*1*	*2*	*3*	*4*
1	0	1.0901	0.6958	0.6656
2	1.0901	0	1.4186	1.0260
3	0.6958	1.4186	0	1.3534
4	0.6656	1.0260	1.3534	0
Sum	2.4514	3.5347	3.4678	3.0450
Min	2.4514			
Max	3.5347			

The ranges of values are considered in the same manner as fitness metric #1. The values for each solutions for 5 ranges is presented in Table 8.

Table 8: Intervals for Fitness Metric 2

Fitness value	*Intervals*
1	2.4514 ≤ Dominance count < 2.6680
2	2.6680≤ Dominance count <2.8847
3	2.8847≤ Dominance count < 3.1013
4	3.2012≤ Dominance count < 3.31804
5	3.31804≤ Dominance count ≤ 3.5347

Then each solution obtains a specific fitness metric depending of its distance to other solutions in the set. Figure 26 shows nondominated solutions with their respective fitness values.

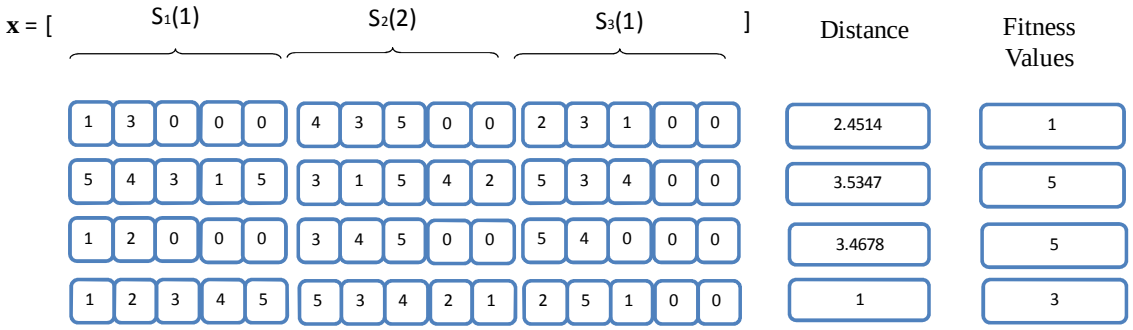


Figure 26: Fitness Metric 2 Calculations

5.2.3.3 Aggregated Fitness (f_a)

The third fitness metric used is the aggregated fitness metric, $f_a(x)$. The aggregated fitness metric is the result of the sum of fitness metrics 1 and 2: $f_a(x) = f_1(x) + f_2(x)$. It weighs both metrics equally. Then, the nondominated solutions are ranked based on this aggregated fitness metric. At this point it is assumed that solutions with better aggregated fitness value are solutions that are closest to the true Pareto front and also far away from other solutions. Figure 27 shows nondominated solutions ranked based on their respective aggregated fitness values

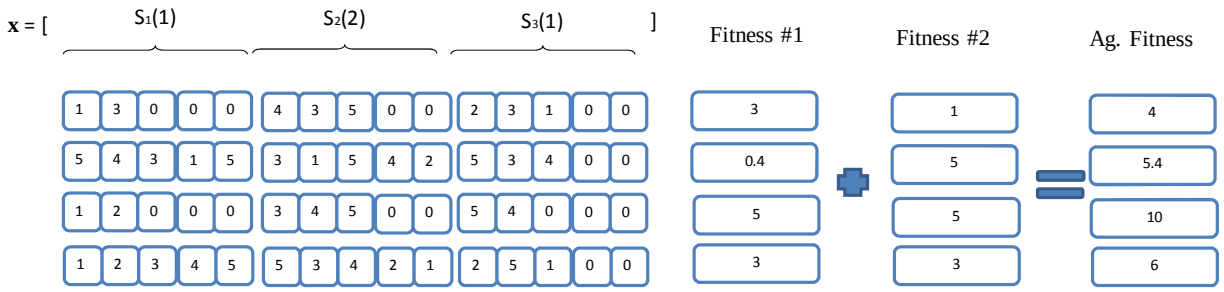


Figure 27: Aggregated Fitness Metric

Solutions that have higher aggregated fitness values are close to the true Pareto front and far away to other solutions. Such strong solutions are solutions that are more likely to undergo crossover and generate new solutions for future generations.

5.2.4. Reproduction

In this section the ranked solutions from the selection step generate new individuals that will be part of the next generation. Three important parameters of the reproduction are considered in next subsections.

5.2.4.1 Elitism

Elitism is used to prevent losing the best solutions from each generation. At this step, a percentage of the nondominated solutions with the best aggregated fitness are selected to survive into the next generation. Figure 28 shows the set of nondominated solution and the part selected by elitism and crossover by considering the 25% of elitism

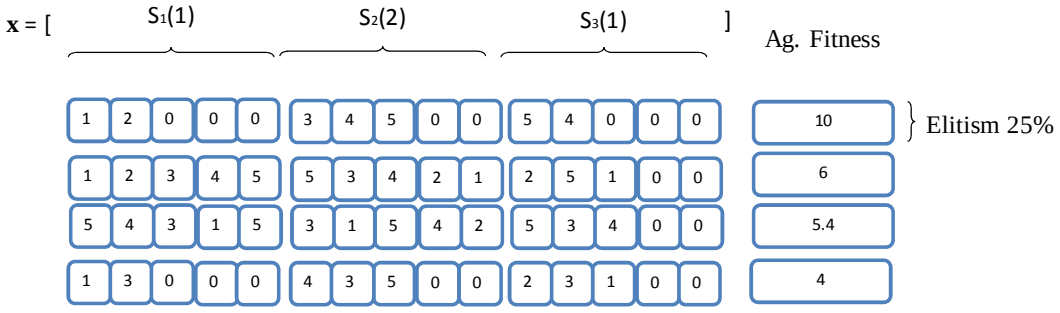


Figure 28: Solutions Selected for Elitism and Crossover

5.2.4.2 Crossover

Some of the most common crossover operators used in EAs such as one-point or multiple points are likely to destroy the information obtained previously because of their random choices of crossover points (Li *et al.*, 2005 & Zhang *et al.* 2007). Another important aspect of a crossover operator is the ability to create diversity of children in order to explore efficiently the search space. Therefore is important to select a crossover operator that can create good offspring and diversity for the patrolling problem. The crossover method used in this study due to the characteristics of the problem is the subsystem rotation crossover (SURC) presented by Taboata & Coit (2008). This method produces a larger number of children in the mating pool, providing a large number of diverse solutions to choose from.

The subsystem rotation crossover divides the chromosome in several subsystems. The first subsystem is rotated in order to generate new solutions and repeat the procedure until the subsystem returns to its initial accommodation. Then the process is repeated with all the other subsystems. Figure 29 shows the procedure.

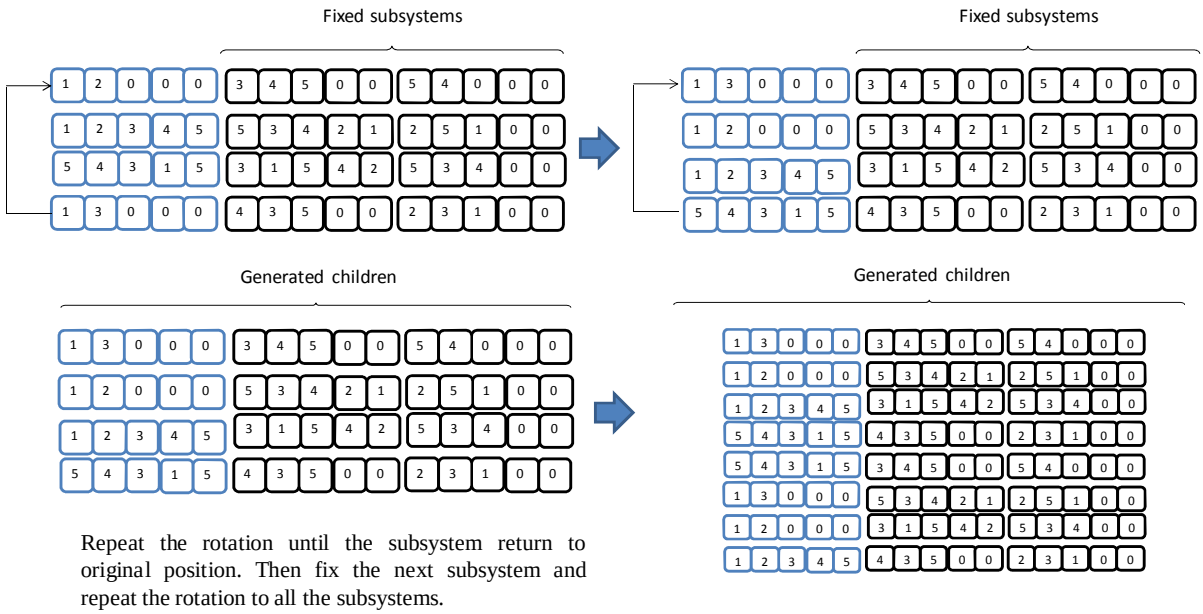


Figure 29: Subsystem Rotation Crossover

5.2.4.3 Mutation

The mutation selected for this algorithm is a two point mutation. The mutation probability selected was p_{mut}= 0.01. For each child selected, a random number is generated between 0 and 1, if this number is smaller than 0.01 then the child is selected to undergo mutation. Once the child is selected for mutation two random point in the chromosome are generated and the values are switched (see Figure 30)

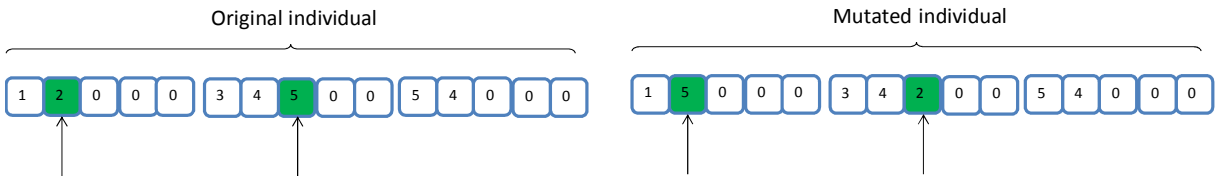


Figure 30: Two Point Mutation

5.2.4.4 New Population

The new generation is formed by elite parents and new children. 25% of the new population is form by the elite parents and 75% of children are selected randomly from the mating pool. Once the next generation is completed, the algorithm returns to the

evaluation stage and repeat the procedure until the specified number of generations is reached. Figure 31 presents how the new generation is formed.

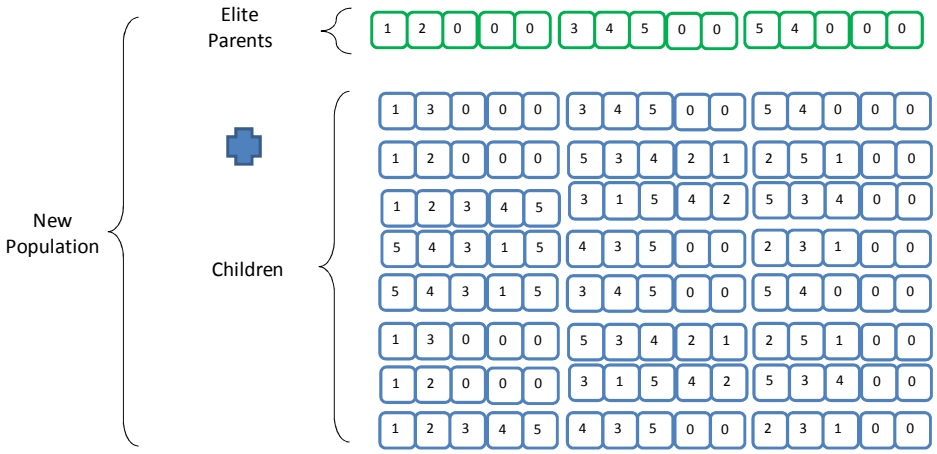


Figure 31: New Population

After the desired number of generations is reached, the algorithm stops and the nondominated solutions for the last generations represent the solution of the problem. This is the first approach presented in this thesis to solve the multiple objective patrolling problems. The next section of this chapter will show a numerical example

5.3 Numerical Example

The algorithm presented in chapter 5 was coded using the MATLAB language. All the functions and subroutines were created in the MATLAB environment as well as an intuitive user interface that allows the user to easily input data for any patrolling problem. Besides data input the graphical user interface (GUI) allows the user to visualize the routes that each agent will follow during the patrolling task. This section will present two different theoretical network problems. The Hardware used to perform the following test is a desktop computer Intel Core 2 Duo CPU with a 2GB DDR2 Memory.

5.3.1 MOEA Example #1

The first problem describes an area that needs to be patrolled. The area is represented by a network consisting of 10 nodes. There are 5 types of agents to patrol the area with different characteristics and 5 strategies are executed simultaneously. The data used in the numerical example is presented in Table 3.

Table 9: Checkpoints Location and Agents' Characteristics Data

Coordinates of nodes

NODE #	X	Y
1	5	8
2	7	5
3	7	6
4	9	3
5	6	2
6	7	4
7	8	9
8	9	6
9	4	2
10	4	6

Agents' characteristics

Agent #	Cost (dll)	Vel. (m/s)
1	12	3
2	35	4
3	18	2
4	27	3
5	15	1

This multiple objective problem was solved using the MOEA described in the previous section with the following parameters:

- Population size = 100
- Generations= 50
- Elitism = 0.25
- Cross =0.75
- Mutation= 0.01

The result of this multiple objective optimization problem is a set of nondominated solutions. The Pareto set of solutions, as well as the user interface of the developed program are shown in Figure 32

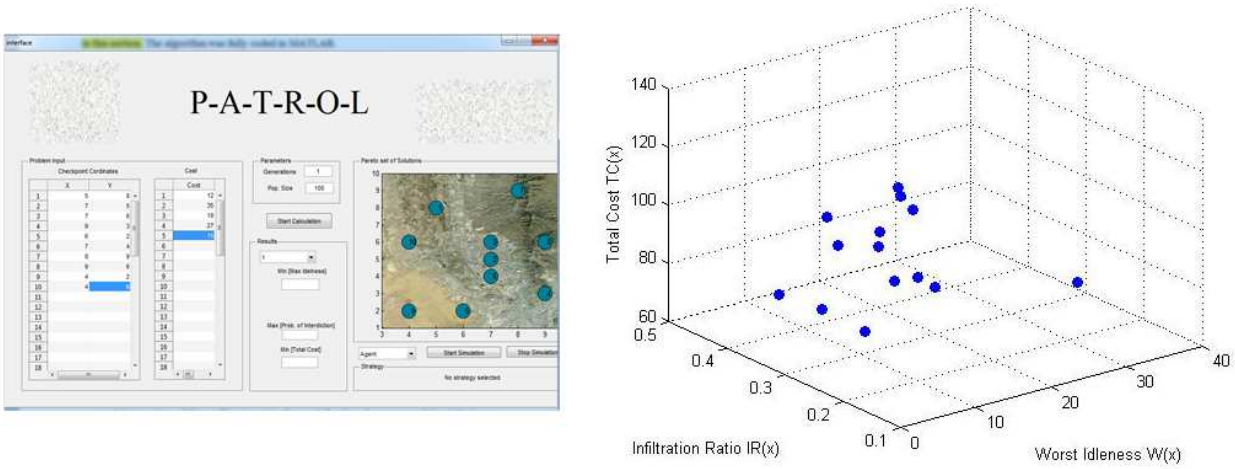


Figure 32: User Interface and Pareto Set of Solutions

Each point in the nondominated set represents one solution to the problem (a multi-agent patrolling strategy), and all of these solutions are considered to be (approximate) Pareto-optimal solutions. A very important part of any multiple objective optimization problem is the decision making process that consist of selecting one solution among the Pareto set. This process is called post-Pareto optimality analysis and it is not considered in this work, but standard techniques could be applied.

To illustrate one solution, the closest solution to the ideal point [0,0,0] in a normalized space was selected. The objective values for the closest solution to the ideal point are: [6.74, 0.37, 78] and the types of agents selected to perform each strategy were **a**=[3,1,3,1,3]. Figure 33 shows the routes to be chosen in the patrolling task and all the strategies performed simultaneously. The first five images in Figure 33 shows the route that each agent will follow during the patrolling task. All routes are cyclic and fixed during the time that the patrolling task is conducted. It is important to note that not just the best routes are selected but also the best type of agents to perform these routes. In this case two agents of type 1 and three agent of type 3 were selected. The last picture

in Figure 33 shows all the routes or strategies that the selected agents will follow during the patrolling task at the same time to minimize the selected objectives

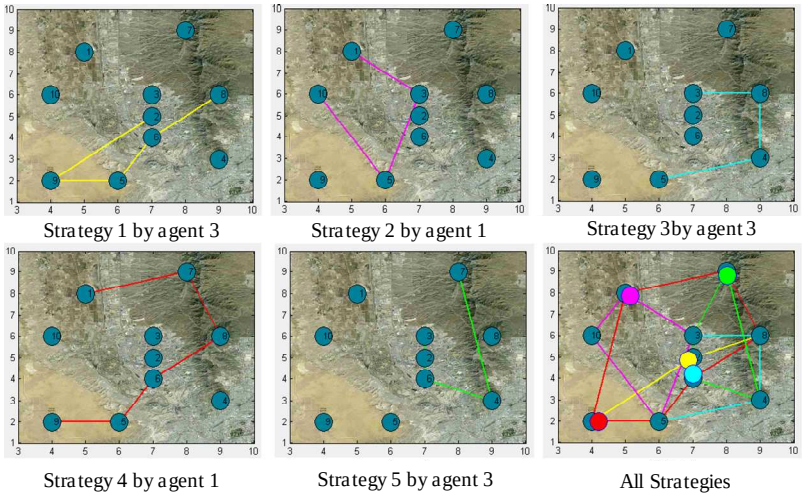


Figure 33: Pareto-optimal Patrolling Strategy for each Agent and Overall Strategy

5.3.2 MOEA Example #2

The second problem describes a network with 20 nodes distributed through the area to be covered. There are 10 available agents to patrol the area with different characteristics, with 10 strategies performed simultaneously. The data used in the numerical example is presented in Table 4. The parameters of the algorithm are the same that the previous example.

Table 10: Checkpoints Location and Agents' Characteristics Data

Coordinates of Nodes

NODE #	X	Y
1	12	13
2	13	8
3	9	5
4	10	7
5	11	4
6	7	10
7	9	9
8	11	11
9	6	5
10	12	6
11	1	8
12	15	3
13	14	1
14	2	12
15	13	18
16	5	11
17	16	7
18	9	18
19	6	2
20	17	13

Agent's Characteristics

Agent #	Cost	Vel.
1	3	6
2	5	8
3	9	11
4	12	12
5	15	13
6	17	15
7	20	17
8	25	19
9	30	22
10	50	25

The 28 Pareto solutions found, as well as the user interface of the program are shown in Figure 34:

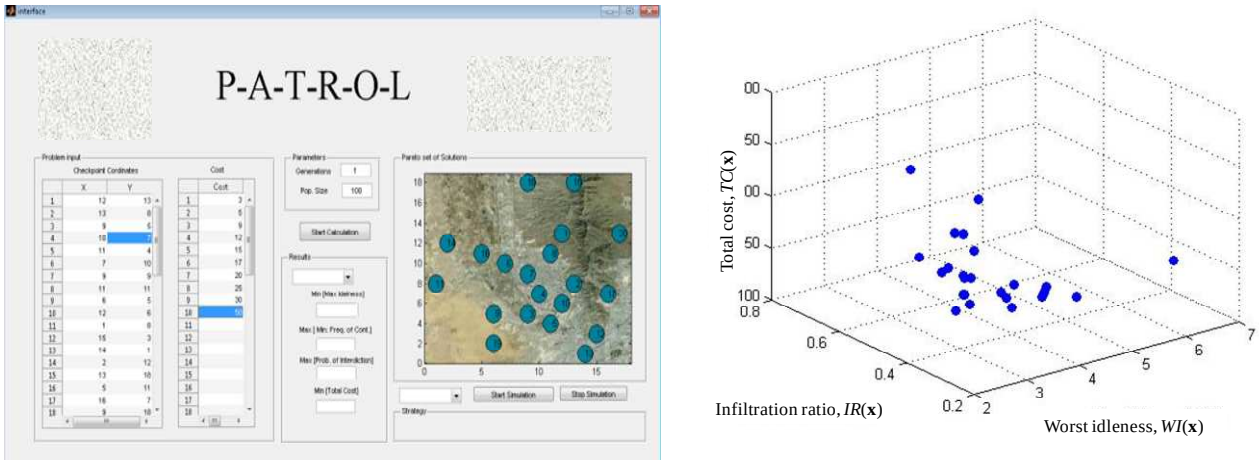


Figure 34: User Interface and Pareto Set of Solutions

The two dimensional views of the last Pareto set are presented in Figure 35:

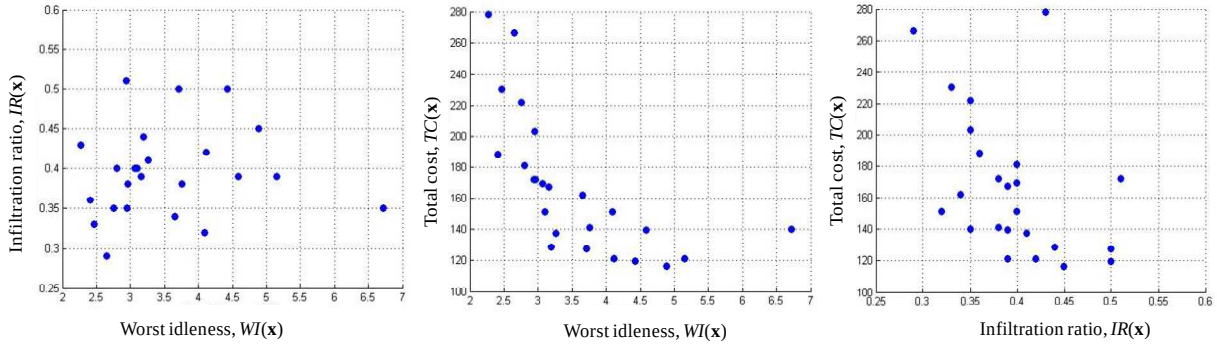
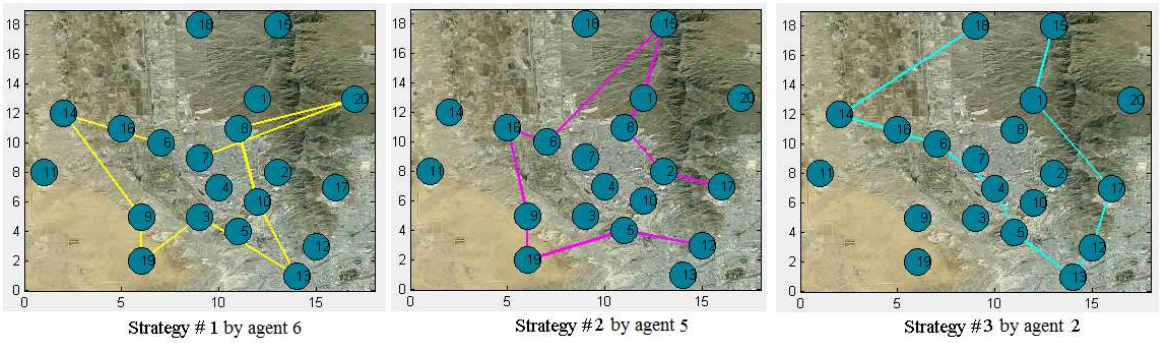


Figure 35: Two Dimensional Views

Each solution in the Pareto frontier is considered to be a Pareto-optimal solution. By definition, a Pareto-optimal solution is not dominated by any other solution in the solution space. To choose one solution to represent our approach, the closest solution to the ideal point [0,0,0] in a normalized space is presented in Figure 36. The objective values for the selected solution are [3.194, 0.44, 128], and the agents patrolling are: [6,5,2,5,7,4,5,5,2,3], that is strategy one is followed by agent 6, strategy 2 is followed by agent 5, and so on.



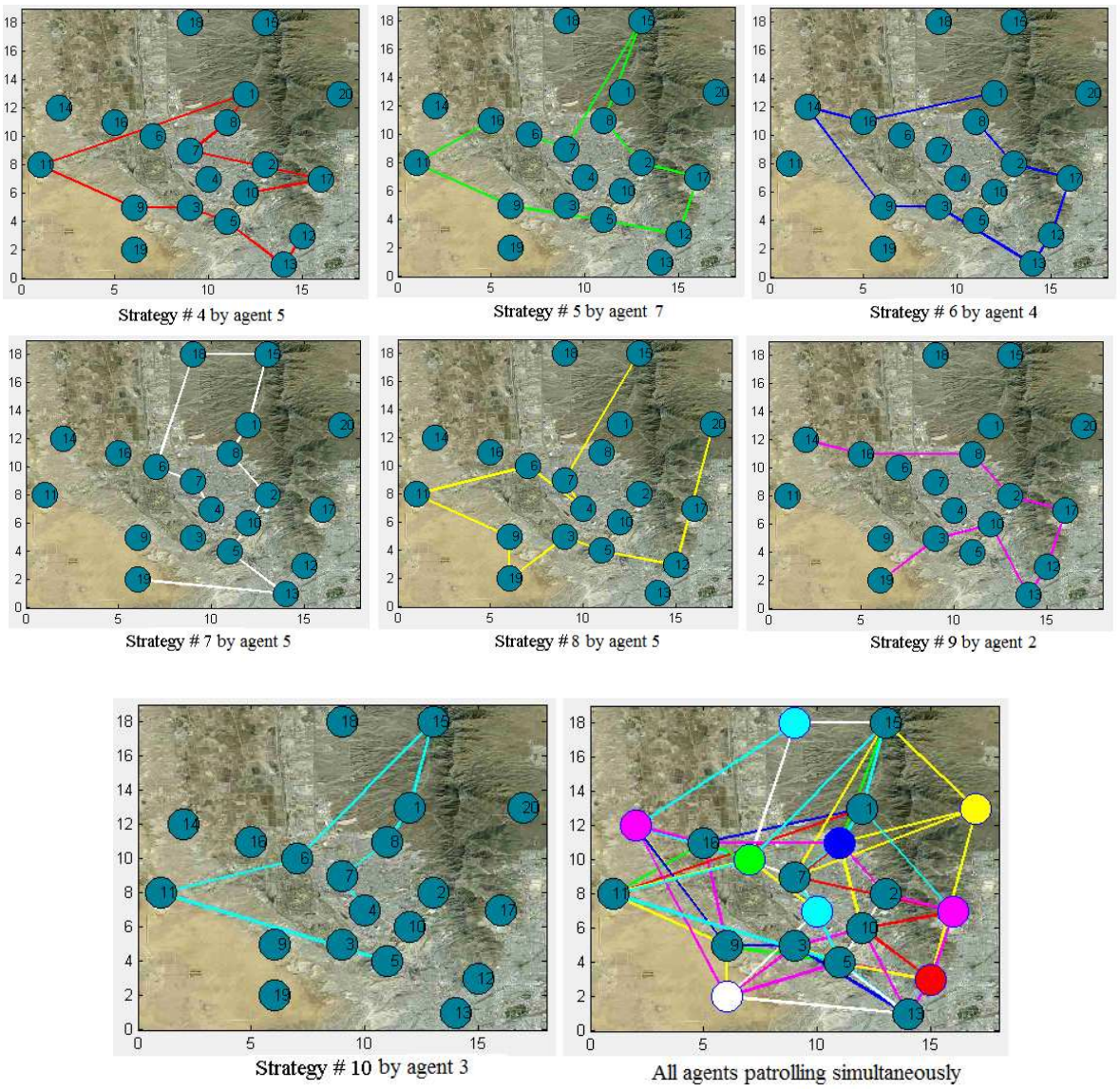


Figure 36: Pareto-optimal Patrolling Strategy for Each Agent and Overall Strategy

5.3.3 Results Analysis

In order to show the importance or relevance of the presented approach against other approaches a direct comparison between the presented work and previous work would be useful. However, one of the main contributions of this work is the way that the model has been formulated in order to address the border protection patrolling problem. The main difference between this method and other methods presented in literature is

that previous studies consider just one objective to be optimized instead of several objectives. Therefore a direct comparison between a single objective optimization method and multiple objective optimization approach is not possible. Another issue that makes a direct comparison difficult is the lack of the data. Researchers normally don't publish their data used in their publications due to in the majority of the cases it is sensitive data.

In the literature the problem has been solved using heuristic approaches such as ant colony optimization but considering just one objective. For comparison purposes our developed algorithm was modified to solve the problem as a single objective optimization problem. Doing so will give a rough idea of the results obtained by other heuristic methods. This modification implies to eliminate the fitness evaluation used in the presented multi-objective evolutionary algorithm and replace it with a single fitness function that gives a better value to solutions that has the better objective value that is being evaluated.

The parameters used to solve the problem are the same that the ones used in the multiple objective case:

- Population size = 100
- Generations= 50
- Elitism = 0.25
- Cross =0.75
- Mutation= 0.01

Tables 11 to 13 present the result of solving the problem for each objective independently that's mean that the problem was solved three different times. The tables also show the objectives values obtained for the proposed multiple objective

evolutionary algorithms. The data used for the problem is the same data used in the first example presented in this work. It can be observed that in two cases the best objective obtained for the single objective approach outperforms the best value obtained for the multiple objective algorithm. This behavior was expected due to the tradeoff between objectives presented in the considered multiple objective optimization problem..

Table 11: Comparison Between Single and Multiple Objective Algorithms Considering WI(x)

Single Objective			Multiple Objective		
WI(x)	**IR(x)**	**TC(x)**	**WI(x)**	**IR(x)**	**TC(x)**
7.2671	0.32	132	6.3107	0.31	109
7.28538	0.35	141	6.43577	0.22	112
7.82534	0.31	108	6.74063	0.37	78
8.22585	0.32	113	7.86841	0.31	98
8.32573	0.39	107	7.88226	0.24	104
8.38465	0.27	144	8.05773	0.21	127
8.42407	0.29	112	8.27545	0.27	72
8.76094	0.23	130	8.41194	0.21	124
8.76786	0.25	144	9.22158	0.23	92
8.91361	0.19	120	10.0952	0.21	118
8.9344	0.37	115	10.3507	0.37	69
9.09554	0.31	98	11.5016	0.19	92
9.15612	0.33	121	14.6484	0.26	87
9.36181	0.34	101	36.568	0.27	69
9.87831	0.36	127			
9.939162	0.26	147			
9.968409	0.3	125			
10.04869	0.33	113			
10.14721	0.3	135			
10.88166	0.44	113			
10.99754	0.35	127			
11.53615	0.38	115			
11.84769	0.3	125			
12.4211	0.35	75			

It can be observed from Table 11 that the best objective value obtained by the single objective approach is worse than the best WI(x) obtained by the MOEA. This behavior was not expected and can be attributed to the most robust fitness metrics used

in the proposed approach against the more general fitness used in the single objective approach. These results showing that the proposed algorithm not only optimizes the objectives simultaneously but also obtains good individual objectives values.

Table 12: Comparison Between Single and Multiple Objective Algorithms Considering IR(x)

Single Objective			**Multiple Objective**		
WI(x)	**IR(x)**	TC(**x**)	WI(x)	**IR(x)**	TC(**x**)
10.26054	**0.18**	98	11.50164	**0.19**	92
13.8764	**0.22**	133	8.41194	**0.21**	124
17.24787	**0.23**	115	8.057733	**0.21**	127
7.97611	**0.25**	113	10.09519	**0.21**	118
8.810815	**0.26**	152	6.435766	**0.22**	112
10.52151	**0.26**	131	9.22158	**0.23**	92
9.833568	**0.27**	135	7.882262	**0.24**	104
14.08012	**0.27**	92	14.64835	**0.26**	87
9.45801	**0.28**	113	36.56799	**0.27**	69
7.46956	**0.28**	135	8.27545	**0.27**	72
14.73337	**0.29**	93	7.868409	**0.31**	98
8.68649	**0.29**	136	6.3107	**0.31**	109
11.32533	**0.29**	144	10.35074	**0.37**	69
10.07024	**0.29**	122	6.74063	**0.37**	78
14.01629	**0.3**	116			
10.06833	**0.3**	66			
13.18719	**0.31**	72			
11.79068	**0.31**	101			
19.16351	**0.32**	84			
24.81838	**0.32**	101			
9.400351	**0.33**	107			
12.44907	**0.33**	84			
11.26971	**0.34**	127			
12.06759	**0.35**	106			

Table 13: Comparison between single and multiple objective algorithms considering TC(x)

Single Objective			Multiple Objective		
WI(x)	IR(x)	**TC(x)**	WI(x)	IR(x)	**TC(x)**
28.51693	0.28	**63**	36.56799	0.27	**69**

11.16271	0.31	**63**	10.35074	0.37	**69**
24.09739	0.35	**66**	8.27545	0.27	**72**
13.11626	0.44	**69**	6.74063	0.37	**78**
26.73821	0.31	**72**	14.64835	0.26	**87**
17.66351	0.31	**72**	9.22158	0.23	**92**
20.63109	0.35	**75**	11.50164	0.19	**92**
16.2946	0.31	**75**	7.868409	0.31	**98**
23.13289	0.36	**78**	7.882262	0.24	**104**
14.44498	0.26	**78**	6.3107	0.31	**109**
35.15039	0.32	**78**	6.435766	0.22	**112**
33.14271	0.47	**78**	10.09519	0.21	**118**
29.1155	0.31	**78**	8.41194	0.21	**124**
13.07241	0.24	**81**	8.057733	0.21	**127**
10.83722	0.33	**84**			
23.10472	0.38	**84**			
32.89129	0.41	**84**			
12.61993	0.35	**87**			
12.65083	0.26	**89**			
12.24264	0.39	**89**			
10.05551	0.32	**89**			
12.23408	0.35	**89**			
11.15432	0.27	**90**			
10.146	0.39	**93**			

The expected behavior is repeated with the TC(x) where the best value is obtained by the single objective approach. The results presented in tables 11 to 13 shows that even compared to a single objective optimization methods the proposed algorithm do a good job obtained a good objective values for each objective function.

Once that the problem has been solve as a single objective optimization problem for each objective all solutions can be collected and then obtain all nondominated solutions in order to build a Pareto set of solutions. This set can be compared against the one obtained by our proposed approach Figure 37 shows the Pareto set for both approaches. The multiple objective approach obtained 14 nondominated solutions against 17 nondominated solutions obtained by solving the problem three times for each

objective.

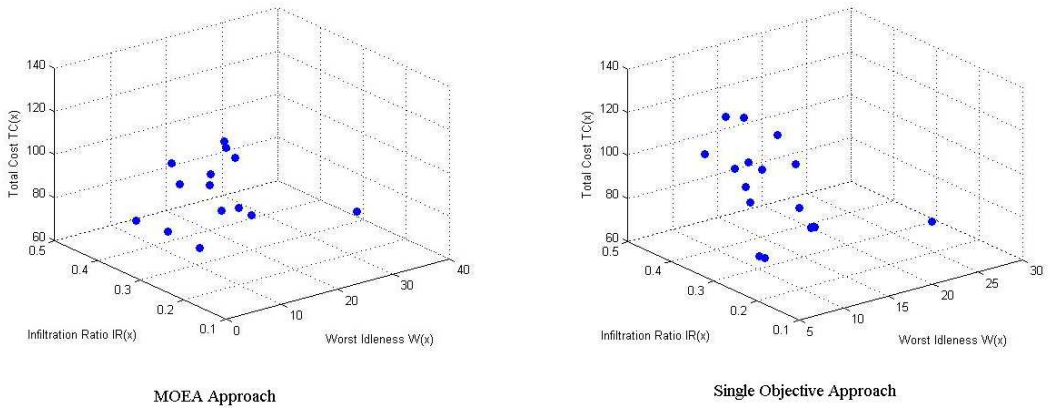


Figure 37: Pareto Sets

In order to visualize in a more easy way the differences or advantages between one method and another all solution for both approaches will be collected, and then a new Pareto set will be built from the nondominated solutions from both approaches. Figure 38 shows the solutions that survived from both methods. The blue points correspond to the single objective approach and the red points correspond to the MOEA approach. The final set is composed of 20 solutions 10 for each approach. Seven solutions were eliminated from the single objective approach and 4 solutions were eliminated of the MOEA approach

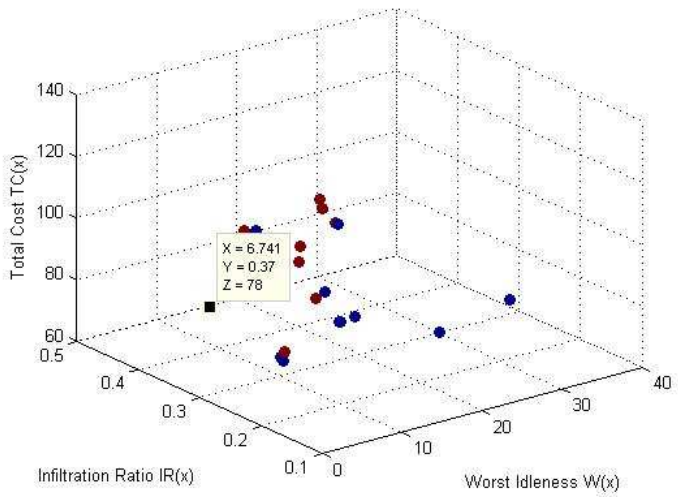


Figure 38: Combined Pareto Set

Solving the problem for each objective and the construct a Pareto set is a good alternative to solve a multiple objective optimization problem. However it requires more time to obtain results because you solve the problem several times. The proposed algorithm proves to obtain good results optimizing the considered objectives simultaneously.

CHAPTER 6: EVOLUTIONARY GAME THEORY ALGORITHM

In this chapter the MOEA presented in Chapter 5 will be modified in order to include game theory techniques that make the patrolling task to be performed in a more efficient way. In the previous approach the defender assume no intelligence for the attacker and considers just random attackers strategies. However, the attacker has the opportunity to learn from previous failures.

The problem of decision making presented in the patrolling problem where the defender has to choose a specific patrolling strategy to face the action of the attacker can be modeled as a game. In this patrolling game two players are involved: the defender and the attacker. In this case the attacker has complete information of the movement of the other player (the defender). However, the defender has no information about the strategy choice of the attacker. In order to define which patrolling strategies will reduce the success of the attacker a game will be played. The strategy used in this algorithm is the iterative elimination of dominated solutions that assumes that the attacker always plays trying to increase their reward.

Therefore in this approach not just the defender will evolve during the algorithm, but the attacker strategies will also evolve as well at each generation of the algorithm. The main change of the algorithm will be made in the evaluation of the second objective the IR(**x**). The new evolutionary game theory algorithm is composed by the four main sections of the MOEA with significant modifications in the evaluation stage. The flowchart of the algorithm is presented in Figure 39.

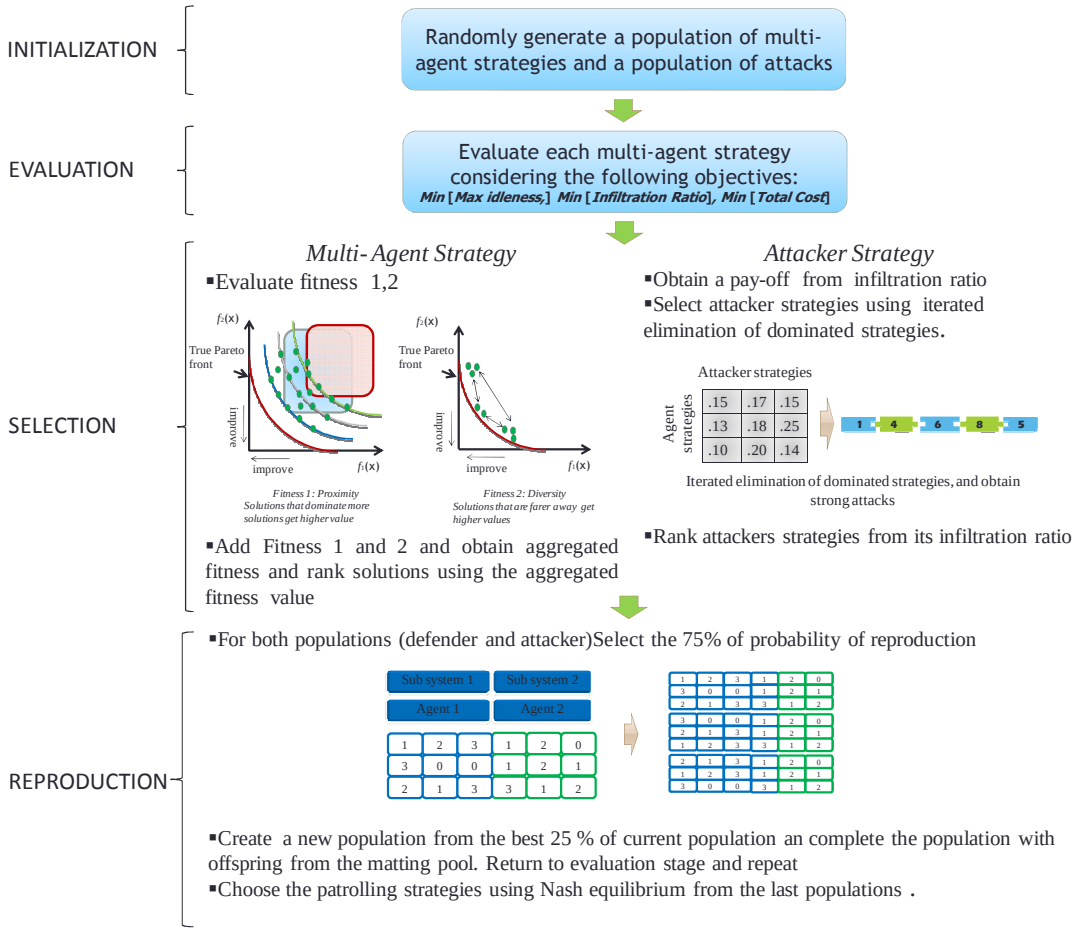


Figure 39: Flow Chart of the Evolutionary Game Theory Algorithm

6.1 Initialization

The first step of the algorithm is to generate randomly the initial generation of possible solutions for both player attacker and defender. The population sizes for both populations can be different. Figure 40 shows an example of an initial population.

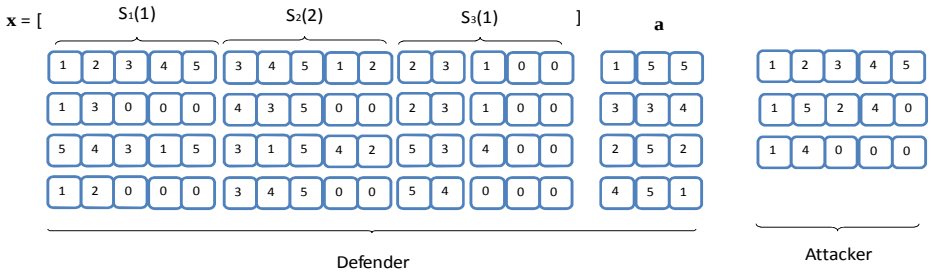


Figure 40: Generation of the First Generation

6.2 Evaluation

In this stage, the three objectives considered will be evaluated for each generated patrolling strategy. The evaluation of the three objectives was described in Chapter 5. Those objectives are: max idleness WI (**x**), Infiltration ratio IR(x), and total cost TC(**x)**. However there is a significant variation of how the IR(**x**) . In the previous approach presented in chapter 5 a random patrolling strategies were generated at a random time. The attacker strategies were evaluated using Monte Carlo simulation. However, these strategies were mere random paths. This situation does not represent the reality in real life applications. In reality the attacker learn from defender strategies and adapts to overcome agents strategies. Instead of generating random attacker strategies a population of the strong attacker strategies will be selected each generation. The Evaluation of the IR(x) uses these strategies in the Monte Carlo Simulation instead of generating random attacker strategies at each generation. The pseudo code of this procedure is presented next

1. Initialize Q=0 (successes)
2. Select a possible multi-agent patrolling strategy and one possible attacker strategy
3. Generate a random attacking starting time for the selected attacker strategy
4. Check if the attacker reaches the target point without being captured by the defender, if that occurs update the value of Q with Q=Q+1.
5. Repeat steps 3-4 for a large number of *iterations,* (i.e., 10,000).The infiltration ratio is obtained by $IR(\boldsymbol{x})= Q/iterations$

At the end of the evaluation a matrix of size n x m is generated with the IR(**x**) values. These values represent the payoffs of both players. While the attacker wants to increase the IR(**x**) the defender wants to reduce this value Figure 41 shows an example of a pay-off matrix. The total $\boldsymbol{IR}(\mathbf{x})$ will be obtained by the average of each strategy.

Attacker

Defender	Strategies	1	2	3
	1	.48	.36	.52
	2	.67	.39	.26
	3	.70	.58	.55
	4	.56	.53	.54

Figure 41: Payoffs Matrix

6.3 Selection

This step is intended to select the strongest parents for both players to undergo crossover and generate strong children to be part of the next generation

The procedure presented in Chapter 5 for the defender remains the same. The selection for the strongest attacker strategies is performing using an iterative elimination of dominated strategies. The purpose of this evaluation is to face agent strategies against strong and adapted attacker strategies.

This technique has its foundation in the idea that if a strategy is strictly dominated, then it makes sense to rule it out as a possibility for a rational player. Why would a player choose that strategy when there is another strategy that is always better? The elimination proceeds in rounds: First eliminate all dominated strategies from player 1, and then eliminate all dominated strategies for player 2 until no more strategies can be eliminated. The solutions or strategies that remain after the process is complete are elected to undergo crossover. This process is presented in Figure 42.

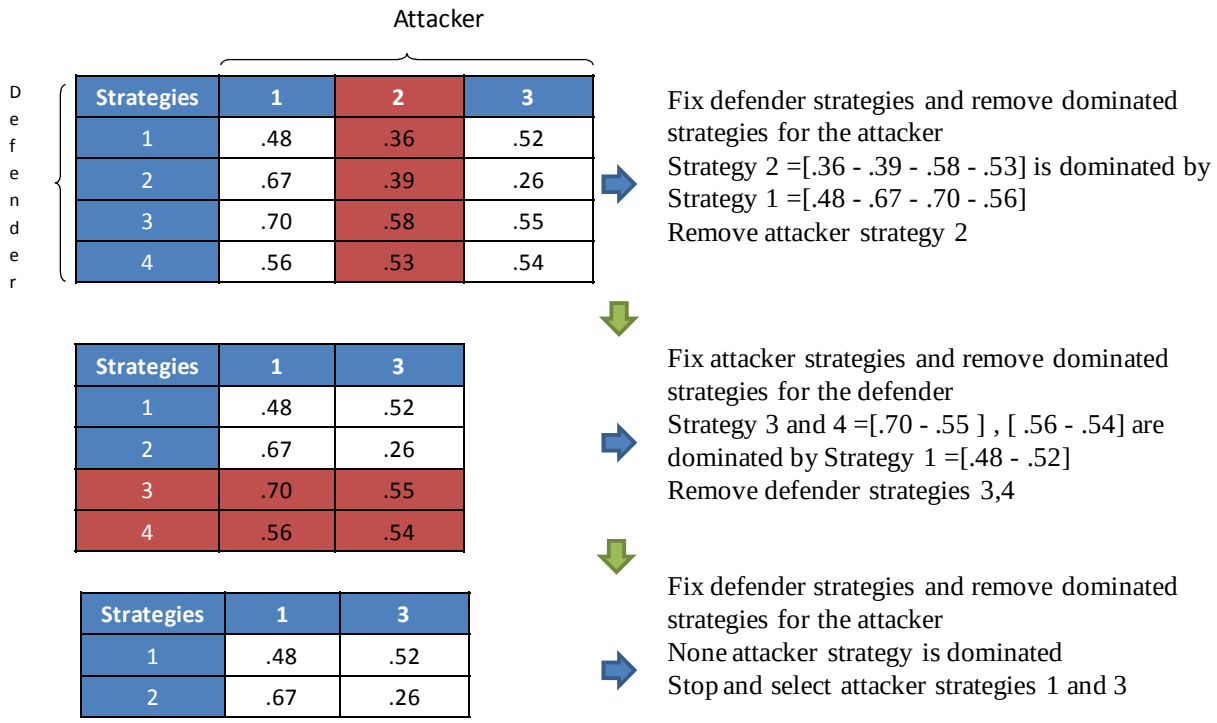


Figure 42: Selection of Strong Attacker Strategies

6.4 Reproduction

In this section the current ranked solutions from the previous step (defender and attacker strategies) will generate the new individuals that will be part of the next generation. The recombination method used is called sub-system rotation crossover (SURC). This procedure was explained in Chapter 5. In this process the chromosomes are divided into subsystems and then the subsystems are rotated in order to obtain children. The new generation is formed by elite parents and new children. Once the next generation is completed, the algorithm returns to the evaluation stage and repeats the procedure until the specified number of generation is needed.

6.5 Numerical Examples

Similar to the examples presented in Chapter 5 two different networks will be evaluated in order to observe the performance of the developed evolutionary game theory algorithm presented previous section.

6.5.1 Examples #1

The first example is a network of 10 nodes. The parameter related to the algorithm is the same used in previous section. There are 5 different agents to choose from and 5 patrolling strategies are allowed to be performed simultaneously. The positions of the check points of the network (nodes) and the cost and velocities for each agent are presented in Table 14 . The cost for each agent are expressed in dollars and the velocities are expressed in m/s

Table 14: Example Data

Coordinates of Nodes

NODE #	X	Y
1	1	3
2	3	6
3	2	5
4	9	6
5	4	5
6	6	2
7	8	4
8	4	2
9	6	4
10	8	1

Agent's Characteristics

Agent #	Cost	Vel.
1	10	6
2	17	8
3	23	11
4	35	12
5	40	13

After running the program for 50 generations a Pareto set of nondominated solution is obtained. All the solutions in the Pareto set are considered mathematical optimal and are presented in Figure 43 as the solution of the multiple objective patrolling problem

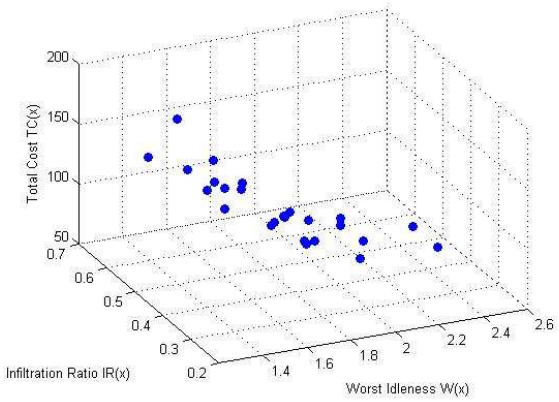


Figure 43: Pareto Set of Solutions

Each point in the problem represents one multi-agent patrolling strategy, each one with different objective values. Even though a post-Pareto optimality is not performed it is important to select one solution among the Pareto set. The ideal point will be a solution with [WI(**x**)=0,IR(**x**), TC(**x**)=0]. The objectives for the closest solution to the ideal point are the following: WI(**x**)=1.854, IR(**x**)=40, and TC(**x**)=89. Figure 44 shows the 5 strategies performed in the patrolling task and all the patrolling strategies at the same time covering a specific area.

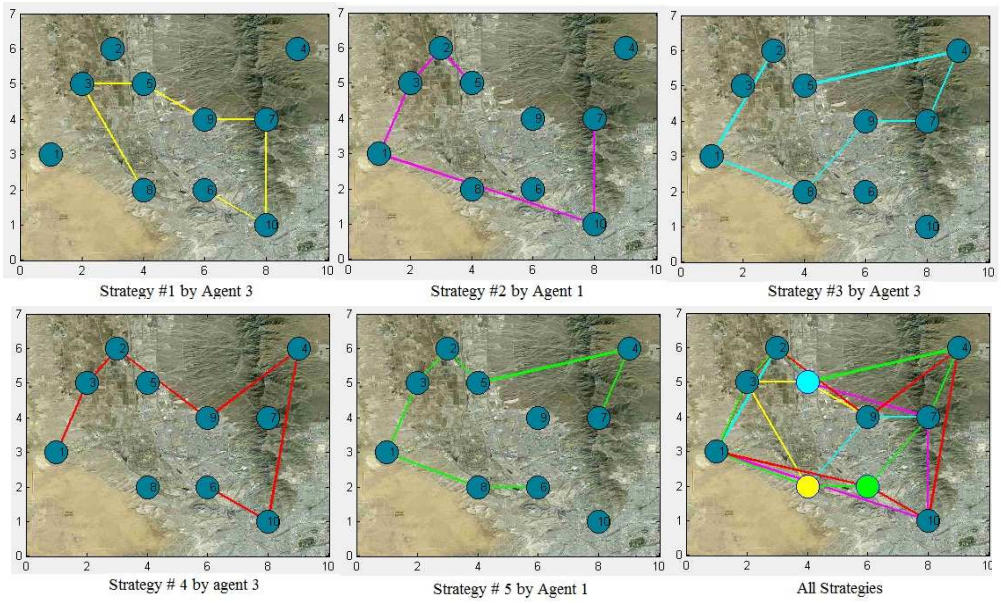


Figure 44: Patrolling Strategies.

6.5.2 Examples #2

The network to be covered considered in the second example is formed by 25 nodes and 10 agents, 10 different strategies are allowed to perform simultaneously. The characteristics of each agent and position of each point are presented in Table 15

Table 15: Example Data.

Coordinates of Nodes

Node #	X	Y
1	3	1
2	7	4
3	5	2
4	6	9
5	5	4
6	2	6
7	4	8
8	2	4
9	4	6
10	1	8
11	1	2
12	8	7
13	9	1
14	10	5
15	6	13
16	4	11
17	10	12
18	1	12
19	2	15
20	8	10
21	10	14
22	6	15
23	12	9
24	12	2
25	9	16

Agent's Characteristics

Agent #	Cost	Vel.
1	3	6
2	5	8
3	9	11
4	12	12
5	15	13
6	17	15
7	20	17
8	25	19
9	30	22
10	50	25

The solutions obtained by the algorithm are presented in Figure 45.

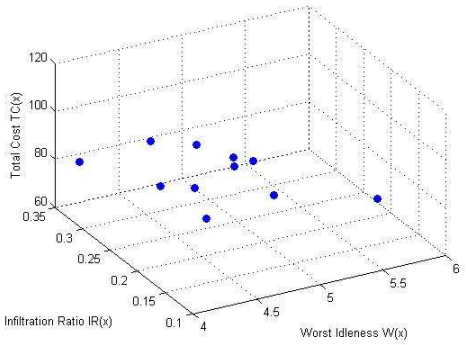
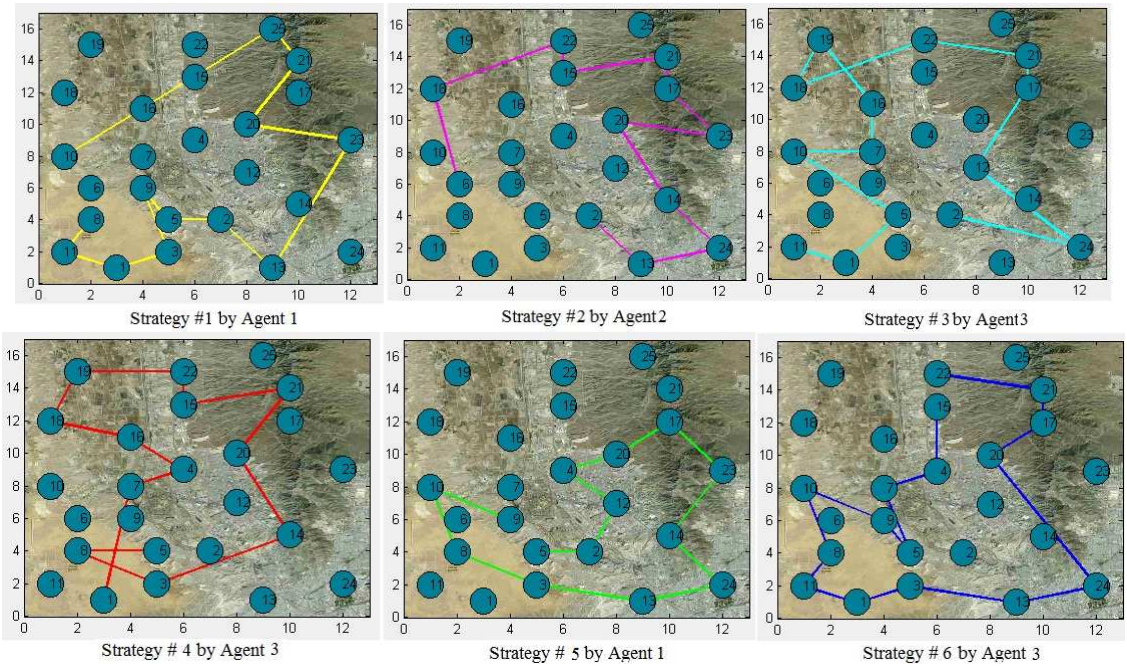


Figure 45: Pareto Set of Solutions

The objectives for the selected solution (closest to the ideal point) are the following: WI(**x**)=4.45, IR(**x**) =18%, TC(**x**) =62.

One important observation for the results is that the IR(**x**)=18% is relative small even though there are significantly more points than the number of patrolling strategies. These results show some of the advantages of the hybrid algorithm to find efficient patrolling strategies where the number of resources is limited. The WI(**x**)=4.45 means that the maximum time that any node is without supervision is 4.45 units of time. TC(**x**)=62 represents the cost associated with the agents used in the patrolling task.

Figure 46 shows the 10 strategies performed in the patrolling task and all the patrolling strategies at the same time covering a specific area.



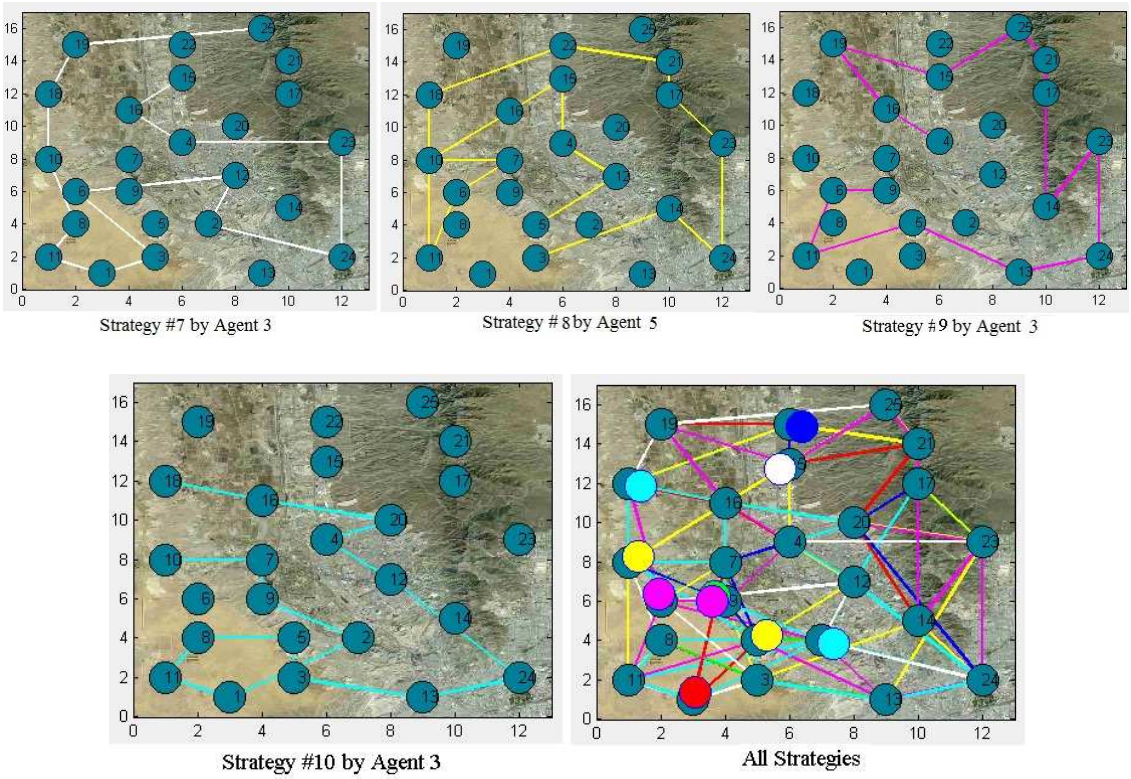


Figure 46: Patrolling Strategy

CHAPTER 7: RESULTS DATA ANALYSIS

In this section the approaches presented in Chapters 5 and 6 are compared. The main difference between the evolutionary algorithm and the game-theory algorithm is the lack of consideration that the attacker is an intelligent individual that can learn from previous data. The objective function that is affected by this modification (game theory implementation) is the Infiltration Ratio ***IR***(**x**). This objective is closely related to the quality of attacker strategies selected by the intruder.

As presented in Chapter 5, the attacker strategy is defined by vector **r.** This vector represents the route that the attacker will use to travel through the network in order to cross the border. This attacker strategy or route has some characteristics:

- Starting and ending nodes for the route are specified as starting and target points
- The velocity of the attacker is fixed
- The length of the attacker strategy is limited by the number of nodes in the network

Therefore, in order to cross the border the attacker has to reach the target point or complete its attacker path without being stopped by the defender. The attacker is considered to be stopped if the attacker is within the vision area of the defender. The evaluation of the ***IR***(**x**) is performed by evaluating each patrolling strategy against different attacker strategies. The ***IR***(**x**) measures the lack of efficiency of the defender strategy to stop different attacker's strategies. The way that the evaluation was performed in both cases (evolutionary and game theory approach) was using Monte Carlo simulation. However there are key differences between both approaches. A

description of how the evaluation of this objective function was performed in each of the different approaches is presented next:

Evaluation of IR(x) using the Evolutionary approach: The evaluation of the ***IR***(**x**) is performed using Monte Carlo simulation. A specific patrolling strategy is compared against different attacker strategies at each generation of the evolutionary algorithm. All the attacker strategies are generated randomly. At each generation of the algorithm a specific patrolling strategy is evaluated against 10,000 randomly generated attacker strategies and none information of these attacker's strategies is stored for further generations. Therefore no intelligent behavior is presented in the attacker strategy because each generation the attacker's strategies are generated randomly allowing the defender to face poor attacker strategies instead of stronger attacker strategies.

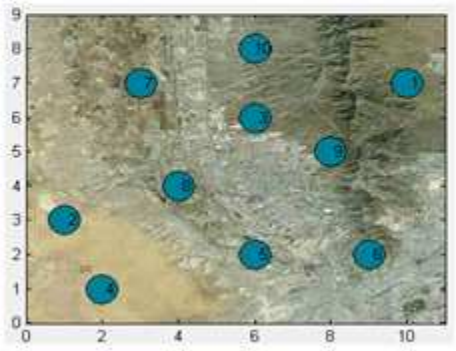
Evaluation of IR(x) using the Evolutionary game theory approach: The evaluation of the ***IR***(**x**) is conducted in the same manner as in the evolutionary approach. However, the attacker's strategies are generated randomly only at the first generation, and then in further generations the strongest attacker's strategies are selected to undergo crossover and reproduction. The selection step is the main difference between both approaches. The selection of best attacker strategies is performed by using a game theory technique called Iterated Elimination of Dominated Strategies (IEDS).

It is certain to consider that in real life the attacker will study the defender moves in order to overcome the defender's strategy. Therefore, it is a must to consider an intelligent adversary in order to model the patrolling problem. The hybrid approach considers the intelligence and knowledge for the attacker. Modeling the problem as a game ensures that the defender can react against intelligent opponent. This game

theory implantation that is the main difference between both approaches represents a key part for the algorithm in order to solve a more complex and realistic patrolling problem.

In order to show a comparison between both approaches the same test problem will be solved using both approaches. The test problem considers an area represented by a network consisting of 10 nodes. There are 4 types of agents to patrol the area and 5 patrolling strategies are performed simultaneously. The data used in the numerical example is presented in Figure 47.

Agent's characteristics				
Agent	1	2	3	4
Cost	12	15	34	50
Vel.	5	13	15	20



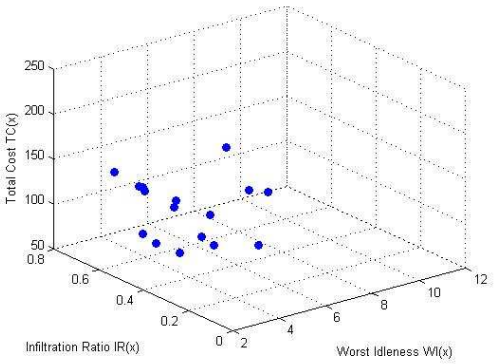
Network Representation

Point's location		
Points	X points	Y points
1	10	7
2	1	3
3	6	6
4	2	1
5	6	2
6	9	2
7	3	7
8	4	4
9	8	5
10	6	8

Figure 47: Example Data

The problem was solved using the algorithms presented in Chapter 5 & 6. Each approach will generate a set of nondominated solutions. Both Pareto sets and solutions are presented in Figure 48

Game theory approach



Evolutionary approach

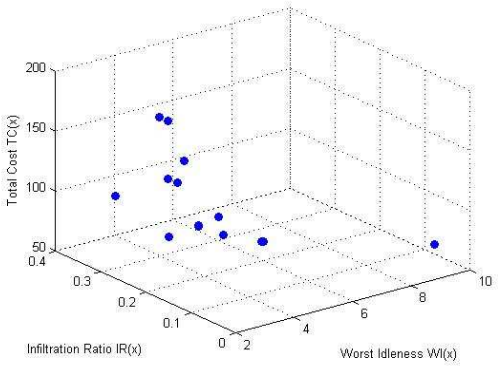


Figure 48: Pareto set of solution of both algorithms

One thing that can be observed from Figure 50 is that even though the two algorithms were configured with the same parameters the number of solutions obtained for each algorithm is different (28 for the game theory approach and 14 for the evolutionary approach). Table 16 shows objectives function values obtained from both algorithms:

Table 16: Game theory approach vs evolutionary approach

G-MOEA			**MOEA**		
***WI*(x)**	***IR*(x)**	***TC*(x)**	***WI*(x)**	***IR*(x)**	***TC*(x)**
2.726361	0.215667	126	2.14981	0.12	199
2.054582	0.035667	250	2.485524	0.17	145
6.72801	0.376	69	6.157811	0.19	63
2.287378	0.432333	107	2.957608	0.08	88
2.778806	0.425667	94	2.404643	0.29	167
2.200913	0.120667	164	2.640052	0.15	129
2.665501	0.332333	145	2.404643	0.18	167
Infinity	0.364333	66	2.609239	0.08	142
10.57354	0.738333	60	2.966413	0.16	110
2.14932	0.435333	161	2.812557	0.22	104
5.553637	0.453	69	3.045417	0.13	107
2.257437	0.281333	161	5.959508	0.22	129
2.146306	0.547333	164	2.726277	0.16	113
3.175791	0.359667	88	9.684371	0.16	88

2.14932	0.419333	161			
2.175218	0.412	158			
2.200913	0.120667	164			
2.665501	0.332333	145			
infinity	0.364333	66			
10.57354	0.738333	60			
2.14932	0.435333	161			
5.553637	0.453	69			
2.257437	0.281333	161			
2.146306	0.547333	164			
3.175791	0.359667	88			
2.14932	0.419333	161			
2.175218	0.412	158			
Average	**Average**	**Average**	**Average**	**Average**	**Average**
3.466764	0.385654	127.4074	3.643134	0.1650	125.07
Min	**Min**	**Min**	**Min**	**Min**	**Min**
2.054582	0.035667	60	2.14981	0.08	63

The averages for each objective are very similar for both methods. However, a significant difference can be observed from the ***IR***(**x**). In fact the ***IR***(**x**) is better in the MOEA approach than in the game theory approach (minimization of ***IR***(**x**)). The main reason for that result is that all the patrolling strategies obtained by the MOEA method are compared against weak attacker's strategies (random strategies).

In order to have a more realistic comparison between solutions, the solutions obtained by the G-MOEA approach will be evaluated using the same method used in the MOEA. In other words, the solutions obtained by the G-MOEA will be compared against random attacker's strategies (just for the ***IR***(**x**)). The comparison of both methods is presented in Table 17.

Table 17: G-MOEA vs. G-MOEA

IR(**x**)		
G-MOEA Original	**G-MOEA Random attacks**	**MOEA Random attacks**
0.215667	0.12	0.19
0.035667	0.09	0.17

0.376	0.14	0.19
0.432333	0.16	0.08
0.425667	0.15	0.29
0.120667	0.14	0.15
0.332333	0.08	0.18
0.364333	0.11	0.08
0.738333	0.09	0.16
0.435333	0.22	0.22
0.453	0.17	0.13
0.281333	0.14	0.22
0.547333	0.19	0.16
0.359667	0.22	0.16
0.419333	0.23	
0.412	0.18	
0.120667	0.11	
0.332333	0.09	
0.364333	0.12	
0.738333	0.16	
0.435333	0.17	
0.453	0.12	
0.281333	0.09	
0.547333	0.14	
0.359667	0.16	
0.419333	0.15	
0.412	0.14	
Average	**Average**	**Average**
0.385654	**.1466**	**.1650**

It can be observed form previous table that the quality of attacker's strategies really affects the value of the objective function ***IR***(**x**). Another aspect to highlight from the previous tables is that the patrolling strategies obtained by the G-MOEA algorithm are not only effective against intelligent attacks but also against non-intelligent attacks.

Table 18 shows results from another type of comparison of the solutions obtained. In this comparison, the patrolling strategies obtained by the MOEA were tested against the

intelligent attacker's strategies obtained by the G-MOEA. The results are presented in Table 18.

Table 18: MOEA vs. G-MOEA

***IR*(x)**		
MOEA Original	**MOEA strong attacks**	**G-MOEA Original**
0.12	0.443167	0.21
0.17	0.430333	0.03
0.19	0.544333	0.37
0.08	0.226333	0.43
0.29	0.624917	0.42
0.15	0.23175	0.12
0.18	0.587667	0.33
0.08	0.457	0.36
0.16	0.171	0.73
0.22	0.568667	0.43
0.13	0.563917	0.45
0.22	0.572333	0.28
0.16	0.589417	0.54
0.16	0.177333	0.35
		0.41
		0.41
		0.12
		0.33
		0.36
		0.73
		0.43
		0.45
		0.28
		0.54
		0.35
		0.41
		0.41
Average	**Average**	**Average**
0.165	**0.4422**	**0.3852**

From Table 18, it can be observed that solutions obtained using the MOEA algorithm perform 6% worse than the G-MOEA in average. A similar comparison between both

methods was performed using a larger network. The information of the network and agents is presented in Figure 49.

point	x-points	y-points
1	27	20
2	20	13
3	15	14
4	4	1
5	4	2
6	3	4
7	28	21
8	21	24
9	17	6
10	19	2
11	7	17
12	27	18
13	17	28
14	16	3
15	4	8
16	29	28
17	3	30
18	6	20
19	19	13
20	3	26
21	6	12
22	2	8
23	26	4
24	25	30
25	24	10
26	15	19
27	21	10
28	8	19
29	21	16
30	27	14

Agents	Velocity	Cost
1	4	50
2	26	45
3	13	14
4	2	16
5	18	39
6	4	35
7	5	7
8	16	15
9	24	1
10	23	12
11	28	13
12	7	48
13	12	3
14	5	41
15	7	41
16	12	16
17	8	18
18	18	22
19	16	42
20	10	33
21	7	9
22	14	42
23	3	44
24	14	7
25	19	6
26	6	35
27	29	3
28	13	20
29	27	14
30	22	24

Figure 49: Network and agents characteristics

In this example 30 nodes are considered and 30 types of agents are available, and 10 strategies are considered to be performed simultaneously. In this scenario the number of nodes to be patrolled is 3 times the number of agents that will be used to patrol the area. The comparison is presented in the same manner than in the previous analyses. Table 15 shows solutions obtained by both approaches.

Table 19: G-MOEA vs MOEA

G-MOEA			MOEA		
***WI*(x)**	***IR*(x)**	***TC*(x)**	***WI*(x)**	***IR*(x)**	***TC*(x)**
16.03472	0.092955	159	15.0515	0.195484	222
16.26769	0.476463	113	18.90656	0.082908	223
14.04356	0.44993	231	26.00144	0.115485	136
22.38662	0.180671	102	14.44789	0.338104	159
21.10932	0.0914	261	14.35147	0.200635	245
30.16835	0.090387	95	16.47827	0.128341	173
16.24756	0.124274	121	15.94729	0.128755	193
19.29893	0.05152	234	16.93042	0.093816	227
16.52771	0.048879	158	16.55181	0.104671	170
27.92114	0.200204	101	38.25334	0.081424	226
14.30919	0.153507	164	19.25594	0.084492	203
16.9656	0.755685	107	17.80553	0.241809	112
18.4238	0.485693	104	15.6014	0.33932	122
17.87822	0.040777	118	18.37649	0.088629	159
14.90826	0.5777	137	23.06456	0.08162	235
17.67684	0.76	85	25.49954	0.231217	133
15.25939	0.286436	147	20.8909	0.083398	138
15.16797	0.351342	141	30.28431	0.403691	88
17.33745	0.548791	88	15.34699	0.152858	306
17.30987	0.3478	154	20.70522	0.080627	241
16.85257	0.1458	221	17.93467	0.094863	179
15.61671	0.056213	199	16.5274	0.145508	139
19.43854	0.274313	93	15.82508	0.088042	273
58.60433	0.074083	112	16.08611	0.132207	178
15.17221	0.3647	109			
17.44702	0.1748	124			
15.2016	0.12587	222			
26.20293	0.07478	276			
14.7381	0.703734	118			
16.15385	0.988203	102			
15.90154	0.138218	187			
26.49501	0.0685	327			
15.61429	0.238109	144			
Average	**Average**	**Average**	**Average**	**Average**	**Average**
19.3539	0.2891	153.1515	19.4218	0.154913	186.6667
Min	**Min**	**Min**	**Min**	**Min**	**Min**
14.0435	0.040777	85	14.35147	0.080627	88

In this second example it can be observed a similar behavior tothe previous example. The average for the *WI*(x) objective is quite similar in both methods and there is a difference in the *IR*(x) objective. However in this case there is also a difference in the *TC*(x) objective being better for the game theory approach. In a similar way than in first example, the number of nondominated solutions obtained by the game theory approach is larger than the solutions obtained by the MOEA approach (33 vs. 24 solutions).

***IR*(x)** values are analyzed in Table 20 as it was done in the previous example. However, in this case only averages are presented instead of all the values obtained by both algorithms in both directions. (G-MOEA vs. MOEA and MOEA vs. G-MOEA)

Table 20: Comparison of IR(x)

G-MOEA vs MOEA		
***IR*(x)**		
G-MOEA Original	**G-MOEA Random attacks**	**MOEA Random attacks**
0.2891	0.1158	0.1549
MOEA vs. -GMOEA		
***IR*(x)**		
MOEA Original	**MOEA strong attacks**	**G-MOEA Original**
0.1549	0.38415	0.2891

Table 20 shows that even if the problem is of moderate size the algorithm performance is capable. It is important to clarify that in this example the number of nodes to be protected is three times the number of agents used to patrol the area. For the comparison between both approaches it can be concluded that the hybrid approach performs better in terms of ***IR*(x)** than just using the evolutionary approach.

In order to test the scalability of both algorithms both methods were tested using a large network configuration. In this case, a network with 100 nodes was used. 30 agents were available to choose from and 25 patrolling strategies were used simultaneously. The data for this test problem is presented in Figure 50.

Point	x-point	y-point	Point	x-point	y-point
1	43	65	**51**	49	70
2	4	1	**52**	53	61
3	40	17	**53**	9	31
4	15	34	**54**	48	6
5	55	7	**55**	24	29
6	43	17	**56**	36	8
7	30	10	**57**	58	63
8	18	30	**58**	1	13
9	18	50	**59**	23	58
10	36	54	**60**	26	64
11	52	69	**61**	4	32
12	28	36	**62**	35	65
13	6	52	**63**	56	16
14	49	43	**64**	66	11
15	59	60	**65**	60	43
16	17	26	**66**	16	22
17	43	56	**67**	70	61
18	10	39	**68**	18	10
19	35	14	**69**	29	35
20	62	21	**70**	19	61
21	60	41	**71**	61	49
22	37	1	**72**	7	31
23	6	42	**73**	39	55
24	14	3	**74**	9	5
25	8	62	**75**	67	13
26	34	9	**76**	16	30
27	39	43	**77**	61	36
28	18	31	**78**	12	4
29	37	19	**79**	7	36
30	61	62	**80**	28	29
31	58	36	**81**	12	50
32	18	28	**82**	70	30
33	11	21	**83**	55	59
34	70	49	**84**	19	39
35	56	41	**85**	19	34
36	55	23	**86**	23	16
37	39	69	**87**	8	45
38	66	40	**88**	1	33
39	50	22	**89**	64	26
40	44	14	**90**	39	32
41	34	35	**91**	17	45
42	63	21	**92**	55	18

Agents	Velocity	Cost
1	4	50
2	26	45
3	13	14
4	2	16
5	18	39
6	4	35
7	5	7
8	16	15
9	24	1
10	23	12
11	28	13
12	7	48
13	12	3
14	5	41
15	7	41
16	12	16
17	8	18
18	18	22
19	16	42
20	10	33
21	7	9
22	14	42
23	3	44
24	14	7
25	19	6
26	6	35
27	29	3
28	13	20
29	27	14
30	22	24

Figure 50: Example Data

The parameters used for this problem were: population sizes of 200 and both methods were run for 100 generations. Due to the large amount of nodes, a larger population was needed. After solving the problem using both algorithms two different populations were obtained. The averages for each objective function value, for each method are presented in next Table.

Table 21: Averages values for both approaches

G-MOEA			**MOEA**		
***WI*(x)**	***IR*(x)**	***TC*(x)**	***WI*(x)**	***IR*(x)**	***TC*(x)**
101.521	0.230227	483.2222	97.80214	0.148107	495.2353
Min	**Min**	**Min**	**Min**	**Min**	**Min**
81.80069	0.0294	344	81.11526	0.13329	378

In order to analyze both methods it is important to analyze the ***IR*(x)** objective. This analysis is presented in Table 22.

Table 22: Comparison between both approaches

G-MOEA vs MOEA		
***IR*(x)**		
G-MOEA Original	**G-MOEA Random attacks**	**MOEA Random attacks**
0.230227	0.10385	0.148107
MOEA vs. -GMOEA		
***IR*(x)**		
MOEA Original	**MOEA strong attacks**	**G-MOEA Original**
0.148107	0.35887	0.230227

For the three different size problems, the proposed Evolutionary Game Theory approach proved to give better solutions in all the objective function values. However, the most important attribute for the hybrid approach is to be able to face more intelligent attacks as well as non-intelligent attacks making the hybrid approach a more robust

approach than the MOEA approach. Figure 51 shows a plot that summarizes the results obtained in all the cases presented before.

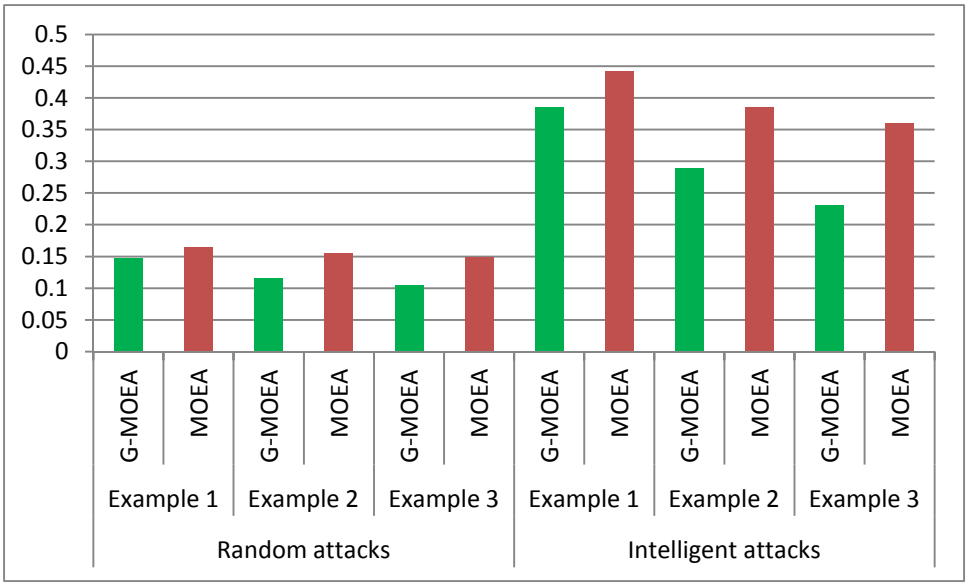


Figure 51: G-MOEA and MOEA comparison

A very interesting behavior between both approaches is that the Evolutionary Game Theory approach seems to provide good results and even improves results as the size of the network increases, contrary to the results obtained by the MOEA in which the results are almost the same for the three examples presented. Infiltration ratio is also dependable of specific network configuration. However, testing the algorithm in different conditions shows a wide idea of the general performance of the proposed algorithm. The proposed algorithm provides a fixed set of strategies that has the best performance against well-educated attacker strategies.

This thesis proposes a general model that can be applied to a variety of patrolling problems. Virtually any region can be analyzed as long as the locations of the nodes are provided and the characteristic of the agents is known.

All regions represent different challenges that are difficult to address to a general approach. This scenario required to personalized model in order to be applied to specific regions.

One region that is difficult to efficiently allocate available sources due to the terrain conditions and the really limited amount of resources allocated is remote areas. Remote areas are considered low risk areas because hazard environments make harder to cross than urban areas. However, this situation has been changed recently. Law enforcement and new patrolling models such as the one presented in this work is making illegal traffic to move to remote areas.

In collaboration with CPB agency a development of a decision support tool was created to help CBP agents to efficiently allocate limited available resources. The details of this project are presented in Chapter 8

.

CHAPTER 8: REMOTE PATROLLING

8.1. Introduction to Remote Patrolling

There are many factors that make the patrolling task a complex optimization problem since many considerations need to be taken into account go into for particular area that needs to be patrolled such as the type of terrain, the proximity to cities or major transportation arteries, the types of resources and infrastructure available in the area, historical traffic patterns and apprehension rates, and so forth. Recent reinforcement in infrastructure and technology deployed in urban and rural areas have improved security in the urban and rural areas forcing illegal activity to move into remote areas. In remote areas, the U.S. Border Patrol does not have a persistent law enforcement presence and therefore seeks to increase its situational awareness without applying the same density of recourses used in urban and rural areas to respond to this need, the CBP agency has recently introduced a new national strategy. This risk-based strategy emphasizes information, integration and rapid response.

The U.S. Border Patrol's national strategy emerged as a key element for the improvement of border security, between the ports of entry. The strategy's overarching premise is that the U.S. Border Patrol would require the appropriate mix of personnel, technology, and border infrastructure -operational pillars- to gain, maintain, and expand operational control of the border. The appropriate mix would be determined by one of the different operational enforcement environments that exist along the southwest border. As can be seen in Figures 52-54, currently, the U.S. Border Patrol operates in three operational border enforcement environments (urban, rural, and remote) that are delineated by the time an Agent has to detect the entry, classify the type of entry,

respond to the cross-border intrusion, and bring the event to the appropriate law enforcement resolution. The “urban” environment requires the U.S. Border Patrol to maintain a persistent presence of law enforcement resources (personnel and technology) in a manner which affords the agency a constant situational awareness of the area.

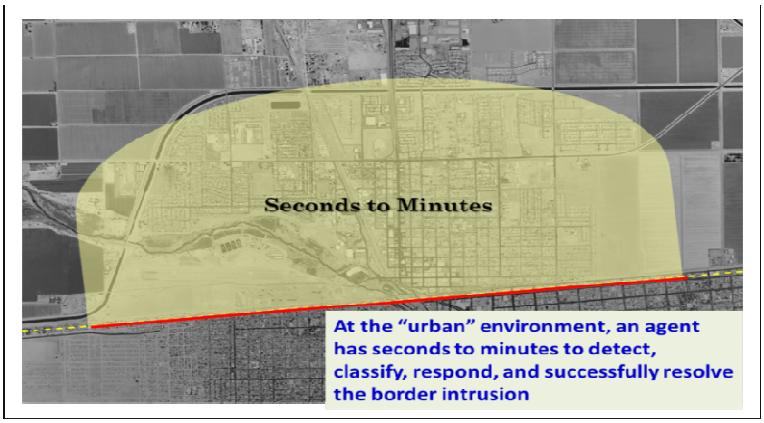


Figure 52: The "urban" environment

The “rural” environment provides an agent minutes to hours to detect, classify, respond, and successfully resolve the intrusion. The “rural” environment requires a persistent presence of resources, but these resources need not be as dense as with an “urban” environment in order to provide an effective level of situational awareness.

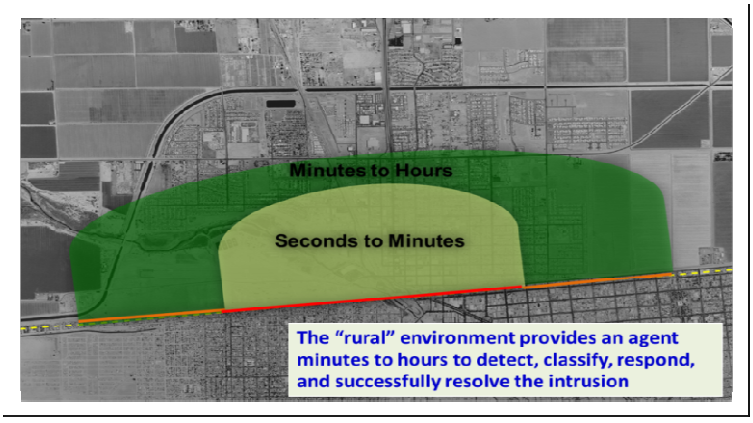


Figure 53: The “rural” environment

The third operational domain is called the “remote” environment where agents may range from hours to days to detect, classify, respond, and bring the cross-border intrusion to an appropriate resolution before the intrusion reaches a ‘vanishing point’ where the chance of achieving an appropriate resolution significantly diminishes. It is in this environment and in many cases low-risk areas, that the U.S. Border Patrol does not have a persistent law enforcement presence and therefore seeks to increase its situational awareness without applying the same density of resources.

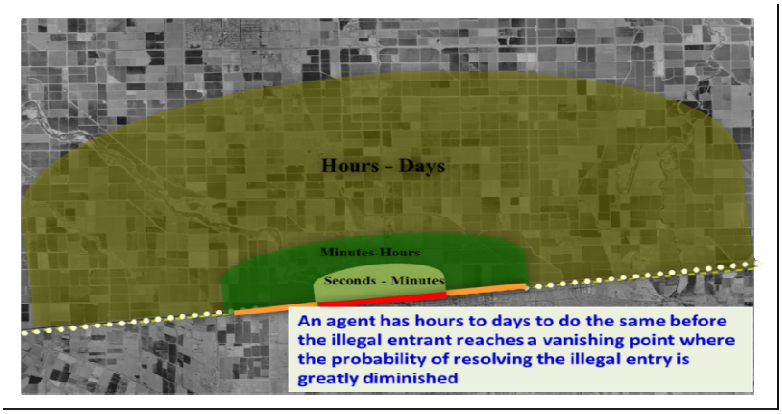


Figure 54: The “remote” environment

The risk posed by any potential illegal traffic in remote areas can be managed through the implementation and utilization of change detection capabilities. Therefore, change detection has emerged as a necessary strategy in the protection of America’s national security. Change detection is considered a strategy in which various techniques are used to gather information and intelligence in, what was previously considered, low risk areas. Rensink (2002) presents an overview of a change detection system. This work gives a clear notion of the basics of change detection systems as well as basic definitions and describes the differences between change and motion, dynamic vs. complete change. He also presented some empirical approaches with different

applications. Kennedy *et al*. (2009) proposed a work related to remote sensing in remote areas using change detection systems. The CBP agency currently has deployed a change detection system in some remote areas in order to keep track of illegal activity. Coulter *et al*. (2012) presented a work consisting in the detection of people and vehicles in natural environments using high temporal resolution airborne remote sensing using manned and unmanned aerial vehicles. Another interesting work done by Coulter and his team was a near real-time change detection system for border monitoring. In their work, they used aerial platforms, imaging sensors, imaging processing/computing, and geo-positioning systems making near real-time detection possible to track objects on the field.

Given the complexity of the CBP patrolling problem, it is clear that computational decision support tools have the potential to greatly improve resource allocation policies, and therefore reduce the burden on human schedulers. Moreover, it can be recognized that models developed to protect urban areas do not necessarily perform well when applied to the protection of remote areas. Securing remote areas presents different challenges due to a wide range of terrain conditions, large areas to be covered, limited resources available, etc. This work presents the development of a decision support model that could potentially assist the U.S. Border Patrol with the deployment of the most appropriate CBP change detection resources available to effectively monitor the "remote" environment and to better respond to cross-border intrusions.

8.2 Patrolling in a Remote Area

As introduced before, the CBP agency classifies the border in three different operational border enforcement environments: urban, rural, and remotes areas. The

main difference between these areas is the time that the CBP agency has to react against illegal traffic. Urban areas are the most critical areas to be protected. Urban areas are represented by cities and populated areas. In these areas, the border patrol response has to be as quick as possible (second to minutes) because the intruder can hide very easy in populated areas and get out of the reach of border protection agents. Rural areas are areas near to urban areas where it is not that easy for the intruder to vanish from the agent. Remote areas are areas located far away to rural and urban areas. These areas are represented by mountains, deserts and other hostile terrain conditions. Remote areas are far from populated areas. Due to this condition, border patrol agents have hours to days to detect, classify, respond, and bring the cross border intrusion to an appropriate resolution before the intruder reaches a vanishing point.

Patrolling and securing remote border areas bring different and specific challenges that have to be addressed. Remote areas are characterized by vast territories. This makes it very hard to efficiently cover the area with a limited amount of resources. Besides the extension of the terrain, another important characteristic which makes remote areas very difficult to secure is the hostile terrain conditions found in these areas. This makes accessibility limited to just some types of agents, vehicles, etc. However, if some locations are inaccessible or very unsecure for an agent to supervise that does not mean that an intruder will avoid these locations; in fact these locations will be more likely to be used by the intruder. Another important point to be acknowledged is the fact that illegal traffic in remote areas has been increased recently due to the improvement in border security implemented in urban and remote areas.

In order to secure remote areas, it is important to keep track of any possible threat. The CBP agency has employed different types of change detection equipment such as sensors, and cameras in order to detect possible illegal traffic across the border. Some examples of the equipment used are presented in Figure 55. A description of this equipment is listed below:

- UGS: Unattended ground sensors: These sensors are located across the border in a specific strategic position. These sensors can detect the vibration in the ground an alert the agent that something is there. The main disadvantage of these sensors is the number of false alarms because the sensor can go on/off due to an animal, a falling tree, wind, or illegal activity.
- RVSS: Remote video surveillance system: This equipment consists of a truck equipped with a camera and radar. This resource is a very high end technology device that can have a good range of vision in a day time and at night. However, this equipment is fixed and once that a location is defined, it cannot move to other location.
- MVSS: Mobile video surveillance systems: This equipment is very similar to RVSS with the difference that the range of vision is limited; it does not have radar, but provides the advantage that an MVSS can be moved to different locations.

Figure 55: Examples of detection systems

Besides the detection technology presented above, the CBP agency also uses fixed patrolling routes performed by CBP agents. Figure 56 shows an example of agents patrolling remote areas.

Figure 56: Example of CBP agents patrolling remote areas

Fixed routes are performed by CBP agencies on a daily basis. Fixed routes are selected because of terrain conditions. Some areas are very difficult to access or too dangerous for the agents to supervise. However, fixed patrolling routes have several disadvantages. Currently, Border protection in remote areas work in the following manner: change detection systems (sensors and cameras) monitor remotes areas. Once that a change is detected, a coordinate station gather this information and send the most capable agent(s) to the area in order to identify, classify, and manage the threat. How this decision is made is based only on the personal experience of the

coordinator and it is not based in any scientific method. The communication between the station and the agents is by radio communicators. There are two main challenges or problems to overcome in the current scenario, the large amount of false alarms and the loss of radio signal of agents due to the terrain conditions.

8.3 Automated Decision Support Model for Remote Patrolling

The objective of the current research work is to develop a software application as a decision tool to help CBP agency to efficiently allocate human resources for border protection once a change in a remote area is detected. The decision support tool has to be flexible enough to model or consider different remote locations; different terrains conditions, and accept a variety of change detection resources and patrolling agents. These characteristics give the program the flexibility to be applied to a large variety of remote areas.

The ***"Automated Decision Support Model for Remote Patrolling"*** software developed has two main components: the MATLAB component and the Google earth component. The MATLAB component is the part of the program dedicated to analyze all data introduced by the user and needed to monitor a specific area. This component also provides the user with the ability to select different available change detection resources that need to be allocated to secure the analyzed remote area.

The second part of the tool is the Google earth program. This part of the program is used to visualize all the geographical information processed by the MATLAB program. That means that in order to visualize data, the Google earth program has to be installed in the computer. This software can be downloaded for free at

https://www.google.com/earth/. Figure 57 shows how the main interface of the software looks.

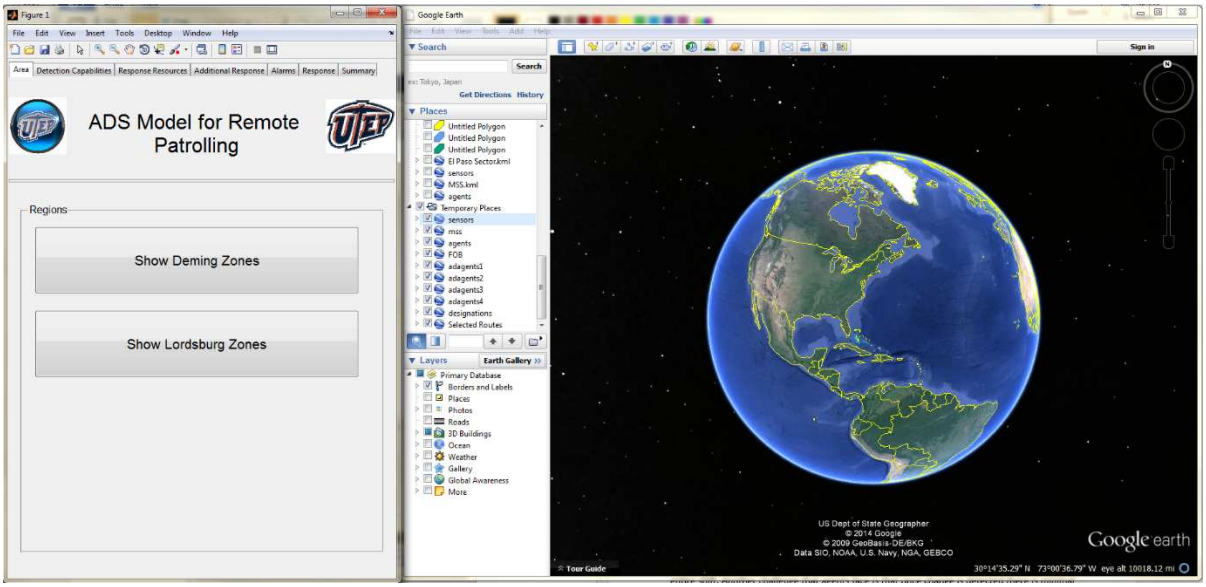


Figure 57: Main interface: the software consist of two different components

The main objective of this software is to provide the U.S. Border Patrol with an intuitive decision support tool to assists them in the optimal deployment of CBP resources to help them make a better use of their change detection capabilities to monitor change in the physical environment in remote areas of the border to efficiently and effectively respond when change is detected.

This support tool can potentially help CBP supervisors to analyze graphically all the change detection systems implemented at the Border area. Besides change detection systems, the user has the ability to allocate different types of response resources such as regular agents; horses patrol agents, ATVs vehicles, among other types of resources. Figure 58 shows an example of the visualization of one possible scenario.

Figure 58: Remote Border Patrolling Scenario

As it was mentioned before, the main purpose of this tool is to help CBP agents to better allocate resources once that a change has been detected. The software allows the user to define which alarms, sensors, or cameras have detected a change. The software determines which agent(s) is(are) close to the threat and suggest the most suitable answer to the user.

8.4 Additional Information

The *"Automated Decision Support Model for Remote Patrolling"* software is currently being protected by Invention Disclosure Agreement (Taboada, *et al.,* 2014-D2015-0011) and by Provisional Patent Application (Taboada *et al.,* 2014-1000-3329).

CHAPTER 9: CONCLUSIONS & FUTURE WORK

9.1 Conclusions

Border Security has evolved over the years. Sixty years ago, border protection consisted basically in patrolling the border with a few agents and the main threat was illegal immigrants trying to enter the country illegally. These immigrants were mainly poor people trying to get a better life in a new country. Nowadays, illegal traffic has evolved dramatically. Due to different factors, a large number of people adventure to cross the border with the promise of a better life. Illegal organizations take advantage of this necessity to make money. Besides trafficking with people, drugs and arms smuggling represent a large amount of money for the drug cartels. These cartels are well funded and highly organized organizations. Therefore, CBP agency has the necessity to improve border security, and optimization security models based on scientific methods have the potential of being a great aid to CBP agents.

As presented in Chapter 1, patrolling the national borders is a very complex task that requires the implementation of different techniques in order to develop reliable and robust patrolling models that can be applied in real-life applications. Several works in the literature have addressed the problem in different forms. This dissertation presents three different models that can be applied to secure National Borders. The first model consists in a multi-objective evolutionary algorithm used to find more efficient patrolling strategies (patrolling routes) to secure the border. The second approach represents a natural evolution of the first approach. In this approach a game theory technique is used in order to overcome predictability. The third approach represents a decision support tool developed with the objective to efficiently allocate available resources in remote

locations. Remote patrolling adds additional changes to the patrolling task that are considered in this approach.

The first approach considers 3 different objective functions: the minimization of the worst idleness (WI(x)), the minimization of the infiltration ratio (IR(x)) and the total cost of the system (TC(x)). These three objectives efficiently measure the quality of a specific multi-agent patrolling strategy. The worst idleness measures the time that a node is without protection (most basic measure used in literature), but the second objective which is the minimization of the infiltration ratio measures the success of the defender to stop the attacker when trying to cross the border. This objective is evaluated performing Monte Carlo simulation. This method evaluates thousands of possible routes that the attacker can use in order to cross the border and compare them against a potential multi-agent patrolling strategy. This simulation gives an accurate measure of the quality of a specific patrolling strategy to stop a variety of attacker's strategies. The third objective addresses the cost of the patrolling task. It is obvious that using a large amount of advanced resources will lead to a better patrolling strategy but will also increase the cost to impractical levels. Therefore, the minimization of the cost is also an important objective to consider in the model.

The evolutionary game-theory approach presented in Chapter 5 was tested in two different sample networks. The presented algorithm is flexible enough to be applied into a variety of networks (in size and configuration) considering different types of agents with different characteristics. The examples presented in section 5.4 show the ability of the algorithm to obtain a diverse set of nondominated solutions. An additional analysis was presented as a comparison of the proposed multi-objective technique

against solving the problem as a single objective problem for each of the considered objectives. The results obtained showed two main advantages. The first one is the reduction in time that represents to run the algorithm just one run instead of several runs. The algorithm also shows to obtain a better quality of solutions than the single objective approach.

The infiltration ratio, *IR*(x), objective is very dependent of the quality of the attacker's strategies used in the Monte Carlo simulation. In the first approach, the agent's strategies are compared against random generated attacker's strategies. Weaker attacker's strategies provide a misleading high *IR*(x) value. The first approach addressed this issue by generating a very large amount of attacker's strategies. However this issue can be addressed in a better way using strong attacker's strategies. The second approach uses a game theory technique called iterated elimination of dominated solutions to eliminate weak attacker's strategies. In addition of this selection, attacker's strategies evolve in the same manner that the agents 'strategies. This guarantees that at each generation not only the strongest agent's strategies survives but also the strongest attacker's strategies. This game-theory addition provides reliability to the evaluation of the *IR*(x) value.

In order to support the necessity and advantages of the game-theory approach, three different networks were evaluated using both algorithms facing random attacker's strategies and intelligent attacker's strategies. The results show that the evolutionary-game theory approach performs better in both cases in all tested networks even if random or intelligent strategies were used. These results prove the efficiency of the proposed methodology.

The third approach addresses a very specific case of the Border Protection problem: the remote patrolling case. The recent security enhancements in populated areas have made illegal immigration move to remote areas. Patrolling remote areas represents new challenges due to vast extensions of terrain to be covered. Remote areas are monitored using a variety of resources to detect illegal traffic. The proposed decision support tool provides the user the optimal resource allocation given that a change is detected in the field. The proposed tool is able to display and analyze two different sectors (Deming and Lordsburg sectors), and additional sectors can be added in the future. The tool also permits to allocate a variety of change detection resources. Once that a change is detected, the system will analyze, all available data and determine which resource is the most capable to be assigned to that alarm.

9.2 Future Research

The complexity and scale of Border Security make it very difficult to develop a perfect model that can solve all problems at a time. Even though the case studies proposed in this dissertation are robust models that can be applied directly to real life applications there is still room for improvement. This section will cover several aspects that are considered as future research.

At the beginning of the project, the patrolling problem was relaxed omitting some elements in order to make the problem easy to handle. As the project matured, more elements were added to the model. However, there are still some constraints and elements that still have to be added to the model in order to have a better representation of the real problem. For instance, the model considers different types of agents, but the model does not consider that there are some limitations on the number

of agents that have to be used. Another limitation of the model is that all agents can follow the same routes. In reality, different resources patrol the area in different ways (vehicles use terrain routes, airplanes does not). Environmental conditions also play an important role. For instance: wind can affect agents' visibility, rain can block some routes making impossible to use some types of vehicles. Therefore, in order to have a better representation of real conditions all these aspects need to be included in the future.

Another improvement proposed as a future research is related to the optimal location of the nodes in the network. At this point, the model considers that the network is already defined and all nodes (check points) and links (roads) are already established. However, the optimality of the location of nodes can also be included in the optimization model in order to define how many nodes and where these have to be located in order to optimize the patrolling task. Similar to the nodes location, another parameter that is not being optimized at this moment is the number of patrolling strategies used during the patrolling task. Currently, this information is given by the user, but it is recommended that the optimization model evaluates the optimal number of patrolling routes to be used while patrolling a specific area

Regarding the evolutionary algorithm a sensitivity analysis is proposed. This analysis is related to the algorithm parameters such as population size, number of generations, crossover and mutation probabilities, etc., in order to identify which values perform better.

In order to find the set of optimal patrolling strategies needed to patrol a specific area, the algorithm evaluates a large number of strategies. This operation is computationally expensive making solving large scale problems (100-500 nodes) impractical in regular computers. Future research includes the implementation of high performance computing to be able to solve large scale problems in a realistic period of time.

Remote Patrolling is a very interesting and challenging instance of the patrolling problem, the vast terrain that has to be covered and the variety of terrain conditions such as dessert, mountains, etc., make very difficult to efficiently allocate limited resources to remote areas. The presented decision support tool is a robust tool that can be used to model different sectors and visualize in a realistic way all the detection capability resources and the detention resources available to respond once a change is detected. Future additions to the model may include: Additional sectors: Using geographical information, more sectors can be added using KML (Keyhole Markup Language) files or manually introducing latitude and longitude coordinates. Routes: Routes in remote areas are dynamic and can change depending on the season of the year or environment phenomena. This information can be added to the model allowing the user to be able to see the routes available at the moment. Routes characteristics also play an important role because these determine if specific agents can be used to patrol these routes. Another implementation to the model includes the ability to specify different sectors and define the resources that are available for each sector. Collaboration: the model can analyze different sectors individually, but at the same time allowing collaboration between sectors.

At this moment, the best response is computed based just on the distance from the alarm and the closest available agent, and the agent is assigned directly to the alarm location. A future approach includes a more intelligent analysis. This analysis can be supported using historical data that will allow the model to predict attacker's behavior allocating detention resources more efficiently. Instead of sending the agent directly to the alarm location, allocating the agent(s) to an area that is more likely that the intruder would be.

Moving from reactive to predictive is the next step of the presented remote patrolling model. Applying the techniques used in the two previous models such as evolutionary algorithms and game theory techniques can improve the probability of success of the detention resources to stop illegal traffic in remote areas. Remote areas can be also be modeled as a network with nodes and links to travel from one node to another, using this information the model presented in Chapter 6 can be used to define the routes that each agent will follow during the patrolling task. This approach has additional advantages. Pre-defined multi-agent patrolling strategies allow to efficiently defining agent's location at any time of the patrolling task. This information can be sued to decide which agent (resource) is the best option to respond to the alarm once a change is detected. Definitely, addition of game theory techniques could play a vital role in remote patrolling allowing the agent to introduce some randomness that make more difficult to the attacker to predict the movements performed by the agents.

REFERENCES

1. Abbass, H., Sarker, R., & Newton, C. (2001). PDE: a Pareto-Frontier Differential Evolution Approach for Multi-Objective Optimization Problems. *Evolutionary Computation, 2001. Procedings 2001.*
2. Adjoua, A. (1996). Conflict analysis under climatic uncerrtanties: The upper Rio grande Basin". *Dissertation, The University of Arizona Graduate College*.
3. Agmon, N., Kraus, S., & Kaminka, G. (2009). Uncertainties in Adversarial Patrol. *Conference on Autonomous Agents and Multiagent Systems* (pp. 1267-1268). International Foundation for Autonomous Agents and Multiagent Systems.
4. Alden, E. (2012). Immigration and Border Control. *CATO Journal; Winter 2012*, Vol 32(1),107.
5. Altner, D., Ergun, O., & Uhan, N. (2010). The Maximum Flow Network Interdiction Problem: Valid inequalities, integrality gaps, and approximability. *Operations Research Letters* , 33-38.
6. Bacharach, M. (1997). Economics and the Game Theory of Games. *West View Press.*
7. Basilico, N., Gatt, N., & Amigoni, F. (2009). Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. *International Foundation for Autonomous Agents and Multiagent Systems*, 978.
8. Binmore, K. (2007). Game theory:a very short introduction.
9. Bosansky, B., Lisy, V., Jakob, M., & Pechoucek, M. (2011). Computing Time-Dependent Policies for Patrolling Games with Mobile Targets. *10th International Conference on Autonomous Agents and Multiagent Systems.*
10. Branke, J., Deb, K., Miettinen, K., & Slowinski, R. (2008). *"Multiobjective Optimization Interactive and Evolutionary Approaches".* Springer.
11. Brown, G., Carlyle, M., Salmeron, J., & Wood, K. (2006). Defending critical infrastructures. *Interfaces*, 36(6):530–544.
12. Censor, Y. (1977). Pareto Optimality in Multiobjective Problems. *Applied Mathematics and Optimization*, 41-59.
13. Charnes, A., Cooper, W., & Ferguson, R. (1955). Optimal estimation of executive compensation by linear programming. *Management Science*, 138-151.
14. Chevaleyre, Y. (2004). Theoretical analysis of the multi-agent patrolling problem. *IEEE/WIC/AMC International Conference on Intelligent Agent Technology*, 302-308.
15. Cormican, K., Morton, D., & Wood, K. (1998). Stochastic network interdiction. *Operation Research*, 46(2).
16. Coulter, L., Stow, D., Lippitt, C., & McCreight, R. (2012). Near Real-Time Change Detection for Border monitoring. *ASPRS 2011 annual Conference*.
17. Daruwala, R. (2002). On Computing the Pareto Optimal Solution Set in a Large Scale Dynamic Network. *New York University*.
18. Deb, K., Pratap, A., & Agarwal, M. (2002). A Fast en Elitist Multiobjective Genetic Algorithm: NSGAII. *IEEE Transactions on Evolutionary Computation*.
19. Diakonikolas, I. (2011). Approximation of Multiple Optimiztion Problems. *Columbia University*.

20. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1):29-41.
21. Dumitrescu, D, Groşan, C., & Oltean. (2001). M., A new evolutionary adaptive representation paradigm. *Studia Universitas Babes--Bolyai, Seria Informatica*, 15-30.
22. Elmaliach, Y., Shiloni, A., & Kaminka, g. (2008). A realistic model of frequency-based multi-robot polyline patrolling. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* (p. 978). International Foundation for Autonomous Agents and Multiagent Systems.
23. Fang, F., Xin Jiang, A., & Tambe, M. (2013). Optimal patrol strategy for protecting moving targets with multiple mobile resources. *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* , (pp. 957-964). International Foundation for Autonomous Agents and Multiagent Systems.
24. Fishburn, P. (1974). Axioms for Lexicographic Preferences. *Oxford University Press*, 415-419.
25. Glad, A., Simonin, O., Buffet, O., & Charpillet, F. (2008). Theoretical Study of Ant-based Algorithms for Multi-Agent Patrolling. *ECAI 2008: 18th European Conference on Artificial Intelligence*, (pp. 626-630).
26. Glover, M., Lagua, M., & Marti, R. (1997). Tabu Search.
27. Gnanasambandam, N., Lee, S., Gautam, N., & Kumara, S. (2004). Reliable MAS Performance Prediction Using Queing Modelss. *IEEE First Symposium on Multi-Agent Security and Survivability*, 55-64.
28. Golalikhani, M., & Zhuang, J. (2011). Modeling arbitrary layers of continuous-level defenses in facing with strategic attackers. *Risk Analysis*, 533-547.
29. Gura, E.-Y., & Maschler, M. (2008). Insights into Game Theory: An alternative Mathematical Experience. *Cambridge University Press.*
30. Hahsler, M., & Hornik, K. (2007). TSP - Infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 1-21.
31. Hertz, A., Taillard, E., & Werra, D. (2002). A Tutorial on Tabu Search. *Department of mathematics University of Montreal.*
32. Israeli, E., & Wood, k. (2002). Shortest-path network interdiction. *Networks*, 97-111.
33. Kacem, I., Hammadi, S., & Borne, P. (2003). Fuzzy Evolutionary Approach for Multiobjective Combinatorial Optimization: Application to Scheduling Problems. *Fuzzy Sets Based Heuristics for Optimization Studies in Fuzziness and Soft Computing*, 197-219.
34. Kaisa, M. (1999). *Nonlinear Multiobjective Optimization.*
35. Karci, A. (2004). Novelty in the generation of initial population for genetic algorithms. *Knowledge-Based Intelligent Information and Engineering Systems,*, 3214.
36. Kaufman, L., & Rousseuw, P. J. (1990). Finding Groups in Data. *New York: John Wiley and Sons, Inc*.
37. Kennedy, R., Townsend, P., Gross, J., Cohen, W., Bolstad, P., Wang, Y., et al. (2009). Remote sensing change detection tools for natural resource managers: Understending concepts and tradeoffs in the design of landscape monitoring projects. *ELSEVIER, Remote sensing environment*, 1382-1396.

38. Knowles, J. C. (1999). The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation. *In Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, 98-105.
39. Kuncheva, L. (1997). Fitness functions in editing k-NN reference set by genetic algorithms. *Pattern Recognition*, 30(6), 1041-1049.
40. Laarhoven, P., & Aarts, E. (1987). Simulated Annealing: Theory and Applications. *Kluwer Academic Publishers*.
41. Lauri, F., & Koukam, A. (2008). A two-step evolutionary and ACO approach for solving the multi-agent patrolling problem. *IEEE Congress on Evolutionary Computation*, 861-868.
42. Li, F., Liu, Q., Min, F., & Yang, G. (2005). A New Crossover Operator Based on the Rough Ret Theory for Genetic Algorithms. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, (pp. 18-21).
43. Lim, C., & Smith, J. (2007). *Algorithms for Discrete and Continuous Multicomodity Flow Network Interdiction Problems.* 15-26: IIE Transactions.
44. Maaranen, H., Miettinen, K., & Makela, M. (2004). Quasi-random initial population for genetic algorithms. *Department of Mathematical Information Technology*.
45. Machado, A., Ramalho, A., Zucker, j., & Drogoul, a. (2002). Multi-Agent Patrolling:An Empirical Analysis of Alternative Architectures. *Third International Workshop on Multi-Agent Based Simulation*.
46. Maged, E., Chang, L.-M., & Zhang, L. (2005). Utility-Function model for engineering performance assessment. *J. Constr. Engrg. and Mgm*, (131):558-568.
47. Marier, J.-S., Besse, C., & Chaib-draa, B. (2010). Solving the Continuous Time Multiagent Patrol Problem. *Conference on Robotics and Automation (ICRA), IEEE International,*, 941-946.
48. Padhye, N., & Deb, K. (2009). Multi-Objective Optimization and Multi-criteria Decision Making For FDM Using Evolutionary Approaches. *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, 219-247.
49. Park, B., Choi, H., & Kim, H. (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering*, 45(4):597-613.
50. Paruchuri, P., Pearce, J., & Kraus, S. (2007). . An Efficient Heuristic for Security Against Multiple Adversaries in Stackelberg Games. *American Association for Artificial Intelligence*.
51. Pulat, H. (2005). A two-sided optimization of border patrol interdiction. *Monterey, California : Naval Postgraduate School, 2005*.
52. Punnen, A. (2002). *The Traveling Salesman Problem: Applications, Formulations and Variations.* In Gutin and Punnen.
53. Rensink, R. (2002). Change Detection. *Annual Reviewers Psycology*, 53:245-272.
54. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 53-65.
55. Runkler, T., & Katz, C. (2006). Fuzzy Clustering by Particle Swarm Optimization. *IEEE International Conference on Fuzzy Systems, Canada, pp*, 601-608.
56. Samuelson, P. (1947). *Foundations of Economics Analysis.*

57. Sano, Y., & Kita, H. (2000). ."Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of search. *Parallel Problem Solving from Nature VI*, 571-580.
58. Taboada, H., & Coit, D. (2005). Post-Pareto Optimality Analysis to Efficiently Identify Promising Solutions For Multi-Objective Problems. *Rutgers University ISE Working Paper* , 05-15.
59. Taboada, H., Baheranwala, F., Coit, D. W., & Wattanapongsakorn, N. (2007). Practical Solutions of Multi-Objective Optimization: An Application to System Reliability Design Problems. . *Reliability Engineering & System Safety*, 92(3):314-322.
60. Thomas, L. (1984). Games, Theory and Applications. *Halsted Press*.
61. Venkat, V., Sheldon, H., & Stori, J. (2004). A Post-Pareto Analysis Algrithm for Multi-Objective Optimization. *Computational Optimization and Applications*, 357-372.
62. Wood, K. (1993). Deterministic network interdiction. *Mathematical and Computer Modeling*, 17:1-18.
63. Zhang, M., Gao, X., & Lou, W. (2007). A new crossover operator in genetic programming for object classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 1332-1343.
64. Zitzler, E., & Thiele, L. (1999). Multiobjective Evolutionay Algorithms: Acomparative Case Ctudy and the Sthregth Pareto Approach. *EEE Transactions on Evolutionary Computation*, 257-271.
65. Zitzler, E., Laumanns, & Bleuler, S. (n.d.). A Tutorial on Evolutionary Multiobkective Optimization. *Metaheuristics for Multiobjective Optimisation Lecture Notes in Economic and Mathematical Systems*, 3-37.
66. Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the Sthrength Pareto Evolutionary Algorithm. *In K. Giannakoglou,D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, Evolutionary Methods for Design, Optimisation, and Control*, 19-26.
67. Zoroa, N., Fernandez-Saez, M., & Zoroa, P. (2012). Decision support patrolling aPerimeter. *European Journal of Operational Research.* , 222, 571-582.

VITA

Oswaldo Aguirre was born in Chihuahua, Chih., Mexico. The first of three offspring of Ruben Aguirre and Teresa Ortega. Oswaldo earned his Bachelor of Engineering degree in Electrical & Mechanical Engineering from Insituto Tecnologico de Chihuahua, Mexico in 2005. He received a Master of Science degree in Industrial Engineering in 2009 and a Master of Science degree in Computational Science in 2012, both from The University of Texas at El Paso.

While pursuing his Ph.D. studies, Oswaldo worked as an assistant instructor and as a research assistant under the Department of Industrial, Manufacturing and Systems Engineering. Oswaldo's research focuses on the development of models to optimize patrolling strategies for border protection. His research work was partially supported by the U.S. Department of Homeland Security (DHS). Oswaldo has made over 20 presentations at local, state, national, and international conferences including the Institute for Operations Research and the Management Sciences (INFORMS) conference, the Industrial & Systems Engineering Research conference (ISERC), and the Complex Adaptive Systems Conference, among others. His research work has been submitted for publication at prestigious journals such as the Computers & Industrial Engineering Journal, and the Engineering Optimization journal.

As a student research innovator and, as part of the integral education received at The University of Texas at El Paso, Oswaldo's research work has led to invention disclosures and a provisional patent application. Oswaldo's dissertation, Multi-Objective Border Patrolling Optimization Using Game Theory and Evolutionary Algorithms, was supervised by Dr. Heidi A. Taboada.