Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

EVOLVABLE SIMULATIONS APPLIED TO AUTOMATED RED TEAMING: A PRELIMINARY STUDY

James Decraene
Mahinthan Chandramohan
Malcolm Yoke Hean Low

School of Computer Engineering
Nanyang Technological University
Singapore 609479

Chwee Seng Choo

DSO National Laboratories
20 Science Park Drive
Singapore 118230

ABSTRACT

We report preliminary studies on evolvable simulations applied to Automated Red Teaming (ART). ART is a vulnerability assessment tool in which agent-based models of simplified military scenarios are repeatedly and automatically generated, executed and varied. Nature-inspired heuristic techniques are utilized to drive the exploration of simulation models to exhibit desired system behaviors. To date, ART investigations have essentially addressed the evolution of a limited fixed set of parameters determining the agents' behavior. We propose to extend ART to widen the range of evolvable simulation model parameters. Using this "evolvable simulation" approach, we conduct experiments in which the agents' structure is evolved. Specifically, a maritime scenario is examined where the individual trajectories of belligerent vessels are evolved to break Blue. These experiments are conducted using a modular evolutionary framework coined CASE. The results present counter-intuitive outcomes and suggest that evolvable simulation is a promising technique to enhance ART.

1 INTRODUCTION

Automated Red Teaming is an agent-based simulation method which aims at identifying the critical weaknesses of military operational plans (Ilachinski 2004, Choo, Chua, and Tay 2007). In ART experiments, multiple simulation models are evaluated where two teams (a defensive "Blue" and belligerent "Red") are confronted against each other using different tactical plans. The modeling and analysis of these tactical plans are automated and are driven by search algorithms. The objectives of the search algorithms are, for instance, to generate Red tactical plans to best defeat Blue.

Through the analysis of simulation outcomes, we may thus identify Red tactical plans which may pose serious threats. Following on from this, defense analysts may first examine and validate the seriousness of the threats (this process may also assist in the correction or refinement of the base simulation model). Then it may subsequently be attempted to resolve the operational weaknesses revealed through ART.

To our knowledge, most ART studies have focused on the examination of the teams' behavior (e.g., aggressiveness, cohesiveness, determination, etc.), see (McDonald and Upton 2005), (Yang, Abbass, and Sarker 2006) and (Ilachinski 2009). Such properties were subjected to variation using for instance evolutionary computation techniques (Fogel 2006). In these studies, the set of "evolvable parameters" was fixed and commonly included less than 20 behavioral parameters.

We argue that such studies are limited when considering real life military operations. For instance, one may be interested in examining/generating critical complex courses of actions which cannot be expressed as a mere set of behavioral parameter values. More advanced techniques are required where additional simulation model properties are to be varied/evaluated.

To extend and enhance the ART methodology, we investigate the evolvable simulation approach presented in works such as (Upton et al. 2004) and (Xu, Low, and Choo 2009) in which additional simulation model properties (e.g., the model or distinct agent's structure) can be subjected to variation. To assist this research, we utilize a modular evolutionary framework coined CASE. Evolutionary computation techniques (e.g.,

evolutionary /swarm algorithms) are utilized as the search techniques due to their suggested ability to efficiently tackle complex optimization problems characterized by non-linear dynamics and considerable search spaces.

A brief literature review on Automated Red Teaming and involved technologies is first provided. Then we describe our framework CASE. Experiments, using CASE and the multi-agent system MANA, are then conducted to evaluate the application of evolvable simulations to ART. Finally, we conclude the paper and outline future research directions which may merit investigations to develop this work.

2 LITERATURE REVIEW

We first present the key technologies supporting Automated Red Teaming. Then a brief survey of ART approaches is provided.

2.1 Agent-based Simulations

ABSs (Bonabeau 2002) are computational methods which, on the contrary to the Lanchester equations (Lanchester 1916), can model the intricate and non-linear dynamics of warfare. Combat is here conceptually regarded as a complex adaptive system (Holland 1992). The agents' computational methods may include stochastic processes resulting in a stochastic behavior at the system level. Examples of ABS applied to Military Decision Making include: CROCADILE (Easton and Barlow 2002), ISAAC/EINSTein (Ilachinski 1997, Ilachinski 2009), MANA (Lauren and Stephen 2002), Pythagoras (Bitinas et al. 2003) and WISDOM (Yang, Abbass, and Sarker 2004). A review of ABS applied to various military applications is provided by (Cioppa, Lucas, and Sanchez 2004).

These systems have been specifically devised to simulate defense related scenarios in which the properties of the environment and the Red/Blue teams may be specified (Lucas et al. 2007). The level of representation/abstraction (e.g., number of spatial dimensions, range of agents' properties, type of vehicles, etc.) varies among these ABS systems. Although the level of accuracy in representing real world environments/individuals may not faithfully reflect reality, it is argued that such ABS models account for the key features (e.g., local interactions between agents) necessary to exhibit complex emerging phenomena/behavior at the system level which are typical of real battlefields (Ilachinski 2004, Yang, Abbass, and Sarker 2006). Thus, these systems can expose the emerging phenomena of interest without the burden of modeling and simulating unnecessary complex features (e.g., gravity, wind, detailed physics of distinct simulated agents/weapons/vehicles, etc.).

2.2 Evolutionary Computation

Evolutionary computation (EC) techniques are non-deterministic search algorithms inspired by real phenomena occurring in nature (Fogel 2006). These algorithms can be classified into two main categories: Evolutionary Algorithms (EAs) and Swarm intelligence based Algorithms (SAs). These techniques differ from each other on the specification and implementation of common system properties: problem representation, variation and selection of candidate solutions.

EAs simulate natural evolution through the variation (i.e., chromosomal recombinations and gene mutations) of genetic material and selection of fittest (from a phenotypic viewpoint) candidate solutions. Examples of EAs include: Genetic Algorithms, Genetic Programming and Evolution Strategy. SAs exploit the collective intelligence emerging from the crowd behavior of social entities such as fish schools, bird flocks and insect colonies. Examples of SAs include: Particle Swarm Optimization and Ant Colony Optimization.

Both families of methods have successfully been applied to a wide range of both numerical and combinatorial optimization problems. Specifically, EC techniques have proven to be highly efficient when applied to optimization problems characterized by non-linear interactions and vast multi-dimensional search spaces. Finally, these techniques have been extended to address explicitly (in contrast with linear combination or weighted sum of objectives methods) Pareto-based multi-objective optimization problems (Deb 2001). This is relevant for military operations as they are often characterized with multi-dimensional constraints which are often conflicting with each other.

2.3 Automated Red Teaming

Automated Red Teaming combines the agent-based simulation and evolutionary computation techniques as follows. ART exploits EC techniques to evolve simulation models to exhibit pre-specified/desirable output behaviors (i.e., when Red defeats Blue). Note that the term "Automated Red Teaming" was coined by Choo et Al. in (Choo, Chua, and Tay 2007), nevertheless the methodology was originally proposed by Ilachinski (Ilachinski 2004). We here utilize the ART term as, we believe, it best captures the "automated" approach to

red teaming, in contrast with the traditional "manual" red teaming method (Defense 2003). Two main classes of ART approaches can be distinguished:

- *One-sided parametric ART*: Only the parameter values (e.g., troop clustering/cohesion, response to injured teammates, aggressiveness, stealthiness, etc.), defining the *behavior* or *personality* of the Red team, are evolved to optimize its efficiency (e.g., maximize damage to target facilities) against the Blue team. Example studies include the works (Ilachinski 1997, Low, Chandramohan, and Choo 2009, Yang, Abbass, and Sarker 2006, Choo, Chua, and Tay 2007, Upton, Johnson, and McDonald 2004). These studies demonstrated the promising potential of ART systems to automatically identify the Blue team's weaknesses. Nevertheless the analytical work necessary to resolving these weaknesses or making the Blue team more robust still have to be researched through time-consuming manual means. In the next section, we examine further techniques which aim at automating this analytical process by coevolving the Blue team to counteract the adaptive Red team.
- *Coevolutionary parametric ART*: The set of behavioral parameter values of both teams are coevolved. This arms race approach complements the previous one by automating the analysis required to improve the Blue team's defense operational plan against the adaptive Red team. Example coevolutionary ART studies can be found in (Kewley and Embrechts 2002, McDonald and Upton 2005, Choo, Chua, Low, and Ong 2009). These approaches presented significant improvements over their one-sided predecessors. A major benefit of these approaches is the suggested ability of coevolution to resolve the local optima issue of EC techniques. This benefit enables one to generate operational tactics that are more efficient and robust against a larger range of scenarios. Nevertheless a trade-off exists in terms of robustness over efficiency according to the range of confronted Red tactics (i.e., the evolved tactics only yield average performances against multiple Red tactics).

Note that the extension of one-sided to coevolutionary ART significantly increases the search spaces allowing for the exploration of more diverse simulation models. As the diversity of evaluated simulation models is increased, a wider range of potentially critical scenarios may be identified. Exploring more diverse scenarios enables one to devise more robust and effective defensive strategies against potential threats and adaptive adversaries. Nevertheless, expanding this search is associated with a dramatic increase in computational cost which would commonly require the use of High Performance Computing techniques.

Similarly, evolvable simulations extend the coevolutionary ART approach. We intend to evolve additional simulation model properties to identify more intricate conditions exposing the weaknesses of operational plans. It is thus expected that computational requirements will also increase. Future work will address these budget computing issues through examining the cloud computing paradigm (Buyya et al. 2009). Nevertheless this issue is not further discussed here as it is beyond the scope of this paper.

3 THE CASE FRAMEWORK

A detailed description of the CASE framework is provided in this section.

3.1 Overview

CASE was implemented in a modular manner (using the Ruby programming language presented in the work (Flanagan and Matsumoto 2008)) to accommodate with ease the user's specific requirements (e.g., use of different simulation engines or evolutionary algorithms, etc.). This framework was inspired by the Automated Red Teaming framework (Chua et al. 2008) which was developed by the DSO National Laboratories of Singapore. In contrast with DSO's system (which was dedicated to examining military simulation models), we aim at providing a flexible and platform-independent system capable of evolving simulation models for a wider variety of application domains.

3.2 Architecture

CASE is composed of three main components which are distinguished as follows:

1. *The model generator*: This component takes as inputs a base simulation model specified in the eXtended Markup Language and a set of model specification text files. According to these inputs, new XML simulation models are generated and sent to the simulation engine for evaluation. Thus, as currently devised, CASE only supports simulation models specified in XML. Moreover, the model generator may consider constraints over the evolvable parameters (this feature is *optional*). These

constraints are specified in a text file by the user. These constraints (due for instance to interactions between evolvable simulation parameters) aim at increasing the plausibility of generated simulation models (e.g., through introducing cost trade-off for specific parameter values).

2. *The simulation engine*: The set of XML simulation models is received and executed by the stochastic simulation engine. Each simulation model is replicated a number of times to account for statistical fluctuations. A set of result files detailing the outcomes of the simulations (in the form of numerical values for instance) are generated. These measurements are used to evaluate the generated models, i.e., these figures are the fitness (or "cost") values utilized by the evolutionary algorithm (EA) to direct the search.
3. *The evolutionary algorithm*: The set of simulation results and associated model specification files are received by the evolutionary algorithm, which in turns, processes the results and produce a new "generation" of model specification files. The generation of these new model specifications is driven by the user-specified (multi)objectives (e.g., maximize/minimize some quantitative values capturing the target system behavior). The algorithm iteratively generates models which would incrementally, through the evolutionary search, best exhibit the desired outcome behavior. The model specification files are sent back to the model generator; this completes the search iteration. This component is the key module responsible for the automated analysis and modeling of simulations.

The above components are depicted in Figure 1 which presents the flowchart of a CASE experiment.

Communications between the three components are conducted *via* text files for simplicity and flexibility (enabling the use of PISA evolutionary algorithm modules (Bleuler et al. 2003)[1]). Note that the flexible nature of CASE allows one to develop and integrate different simulation platforms (using models specified in XML), and evolutionary algorithms. Moreover CASE may exploit multi-core CPUs through simultaneously evaluating (using multi-threads) multiple simulation models.

The input files are specified as follows:

- *Evolvable parameters setting*: This text file specifies the list of simulation model parameters which are subjected to the evolutionary process. For each parameter, the XPath, name and numerical values ranges (min,max) have to be provided. In addition to (real) numerical values, it is possible to include parameter values in the form of enumerable sets (e.g., low, medium, high, etc.) to address model properties that cannot be expressed as numerical values.
- *Constraints setting*: Optional constraints may be devised to introduce specific considerations when evolving particular parameter values. For instance, the user may devise Interactions between parameters which would occur according to some pre-defined threshold values. The specification of such constraints may be carried out through the definition of a mapping table. The XPath, name and threshold values of interacting parameters have to be provided. According to the value of the "master" parameter, the values of the "slave" parameters are adjusted according to the associated mapping table. Constraints are applied over the model specifications prior to the generation of XML models. A Ruby script manages the application of the constraints. Note that this feature is not utilized in the experiments presented in this paper.
- *Experiment setting*: Here, the overall experimental run settings are specified. This includes:
  - The selected simulation engine.
  - The selected evolutionary algorithm and associated setting (e.g., population size, number of search iterations, mutation probability, set of objectives, etc.).
  - The number of simulation execution replications (ABSs are stochastic systems which require repetitions of the simulation model executions to account for statistical fluctuation).
  - The number of CASE run replications (similarly to ABSs, evolutionary algorithms are stochastic processes, replications of the experimental runs may also be necessary).
  - The number of available CPUs (for multi-threading purposes).

In the next section, we report our series of experiments using the above framework.

[1]PISA is a platform and programming language independent interface for search algorithms which aims at facilitating the evaluation of evolutionary algorithms. A number of search algorithms and benchmark problems have been implemented and are interfaced via text-files. More details can be found at http://www.tik.ethz.ch/sop/pisa/.

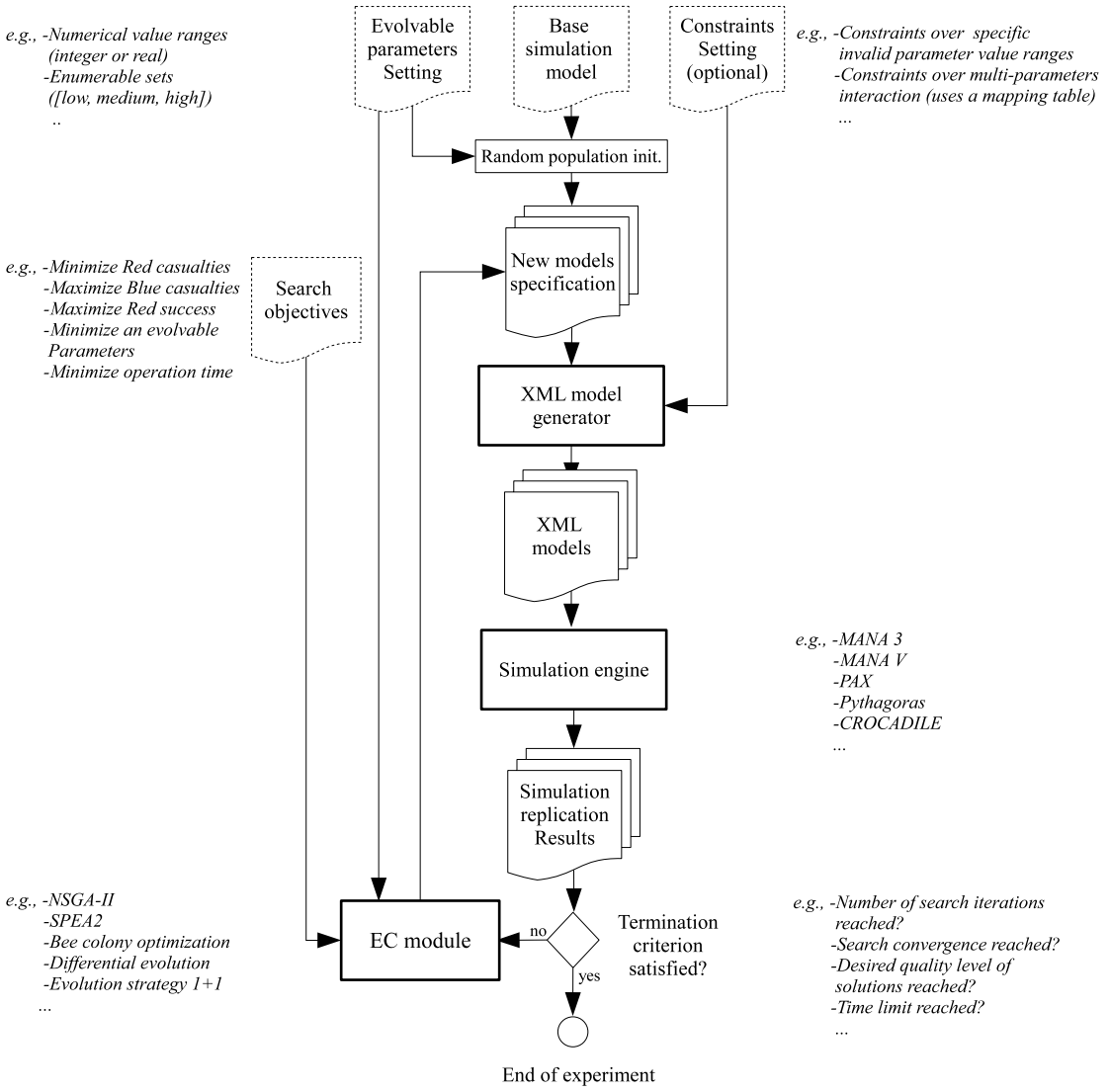

Figure 1: Flowchart of a CASE experiment. The dashed documents distinguish the user inputs. Using the base XML model, a population of randomly generated model variants is first created. The initial parameter values are randomly generated using a uniform distribution and are bounded by the evolvable parameters setting file provided by the user. Three core modules (the simulation engine, evolutionary computation and XML generator modules) are implemented as Ruby scripts. Both the simulation engine and evolutionary computation module call external libraries and/or binaries. The XML model generator employs the Libxml library (`http://libxml.rubyforge.org`) to parse and generate XML models. The constraint setting file is utilized by the XML model generator to apply user-defined constraints over the evolvable parameters. The ease of development provided by the Ruby language and script based nature of CASE enable one to rapidly introduce/implement new termination criteria, use additional simulations/search algorithms or devise new constraints over evolvable parameters.

4 EXPERIMENTS

We report a series of experiments using the CASE framework and the agent-based simulation platform MANA. In these studies the agents' structure, specifically the number of pathway points and associated coordinates determining the trajectories of the agents, are subjected to the evolutionary process.

4.1 The Scenario

We consider a maritime anchorage protection scenario which was originally proposed in (Wong et al. 2007) and later further developed in (Xu et al. 2009, Low et al. 2009). In this scenario, a Blue Team (composed of 7 vessels) conducts patrols to protect an anchorage (in which 10 Green commercial vessels are anchored) against threats. Red forces (5 vessels) attempt to break Blues defense tactics and inflict damages to anchored vessels. The aim of the study is to discover Reds strategies that are able to breach through Blues defensive tactic. Figure 2 depicts the scenario which was modeled using the ABS platform MANA.

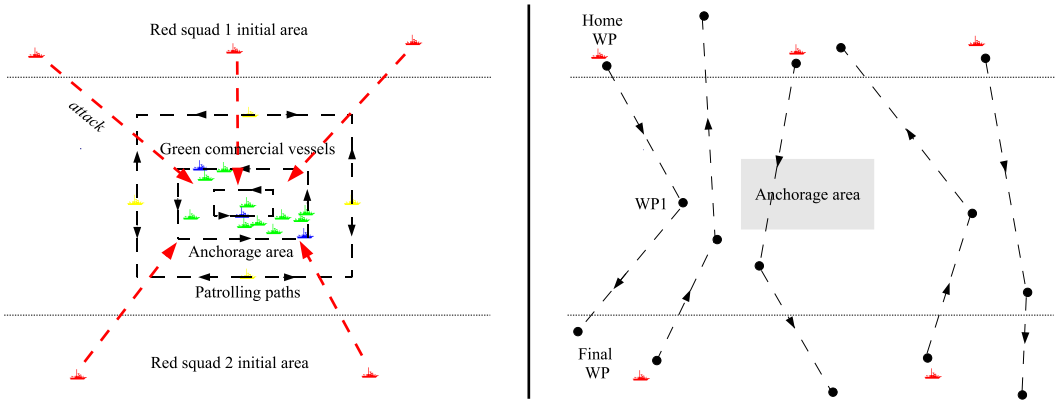

Figure 2: MANA model of the maritime anchorage protection scenario. The map covers an area of 100 by 50 nautical miles (1 nm = 1.852km). **Left**: The dashed lines depict the patrolling paths of the different Blue vessels. The Blue patrolling strategy is composed of two layers: an outer (with respect to the anchorage area, 30 by 10 nm) and inner patrol. The outer patrol consists of four smaller but faster boats. They provide the first layer of defense whereas the larger and heavily armored ships inside the anchorage are the second defensive layer. Three of the Red crafts (Team 1) were set up to initiate their attack from the north while the remaining two attack (Team 2) from the south. This allows Red to perform multi-directional attacks at the anchorage. The initial positions of Blue vessels are fixed. In contrast, the Green commerical vessels' initial positions are randomly generated within the anchorage area at each MANA execution. **Right**: Example Red trajectories. Home waypoints (Home WP) are constrained to the distinct agent's initial areas. Similarly, the final waypoints are to be located in the opposite area. Intermediate waypoints occur in the remaining middle area. Note that in the below experiments, we dynamically evolve the number of intermediate waypoints. In addition the coordinates of all waypoints, including the home and final ones, are subjected to evolution.

In (Xu, Low, and Choo 2009), the measures of effectiveness only considered the number of Green casualties. In other words, Red was evolved to maximize the number of commercial vessels casualties regardless of Red casualties. In this study, the number of trajectory waypoints was incrementally increased (manually) in a series of experiments. It was found that for a particular number of waypoints (3)[2], ART was able to generate Red tactical plans that were more efficient and complex than other models involving differing number of waypoints.

Low, Chandramohan, and Choo (2009) proposed a multi-objective approach to this model where the number of Red casualties was also considered. Trade-offs in Green vs Red casualties were considered, however, the number of waypoints was fixed limiting the comparison of strategies resulting from the two studies.

Here we study further this simulation model and address both multi-objective optimization and the automated evolution of Red trajectories.

4.2 Experimental Setting

In CASE, each candidate solution (a distinct simulation model) is represented by a vector of real values defining the different evolvable Red behavioral parameters (Table 1(c)). As the number of decision variables increases, the search space becomes significantly larger. According to the number of evolvable properties and

[2]We conducted further experiments to examine the reasons underlying these optimal tactical plans involving 3 waypoint trajectories. It was found out that the maximal number of MANA running time steps was potentially set too low. Indeed, the simulations would actually terminate before Red vessels could visit the fourth and following waypoints. This time limit constraint may have prevented the evolution of more complex and efficient trajectories involving more waypoints.

associated ranges given for this experiment, the search space contains 1.007×10^{28} distinct candidate solutions (i.e., variants of the original simulation model).

Table 1: (a): Fixed Blue parameters. Note that value pairs are specified for the determination and aggressiveness properties. In this model, Blue changes its behavior upon detecting Red, i.e., Blue "targets" Red, with aggressiveness being increased, when the latter is within Blue's detection range. (b): Fixed Red parameters. The behavioral parameters are not specified as these parameters are subjected to evolution. (c): Evolvable Red parameters: As mentioned earlier, the home and final positions together with the n intermediate waypoint(s) define the trajectory of each distinct Red vessel. In addition, the final positions of the Red crafts are constrained to the opposite region (with respect to initial area) to simulate escapes from the anchorage following successful attacks.

(a) Fixed Blue parameters

Parameter	Value
Detection range (nm)	24
# hits to be killed	2
Weapon hit prob.	0.8
# patrolling agents	7
Speed (unit)	100
Weapon range (nm)	8
Determination	50∨0
Aggressiveness	0∨100
Cohesiveness	0

(b) Fixed Red parameters

Parameter	Value
Detection range (nm)	8
# hits to be killed	1
Weapon hit prob.	0.8
# agents	5
Speed (unit)	100
Weapon range (nm)	5

(c) Evolvable Red paramaters

Parameter	Min	Max
Squad 1 home (x,y) $\times 3$	(0,0)	(399,39)
Squad 2 home (x,y) $\times 2$	(0,160)	(399,199)
Number of inter. WPs n	1	5
Intermediate WPs (x,y) $\times n$	(0,40)	(399,159)
Squad 1 final (x,y) $\times 3$	(0,160)	(399,199)
Squad 2 final (x,y) $\times 2$	(0,0)	(399,39)
Determination	20	100
Aggressiveness	-100	100
Cohesiveness	-100	100

Behavioral or "psychological" elements are included in the decision variables. The aggressiveness determines the reaction of individual vessels upon detecting an adversary. Cohesiveness influences the propensity of vessels to maneuver as a group or not, whereas determination stands for the agent's willingness to follow the defined trajectories (go to next waypoint). The Red vessels' aggressiveness against the Blue patrolling force are varied from unaggressive (-100) to very aggressive (100). Likewise, the cohesiveness of the Red crafts are varied from independent (-100) to very cohesive (100). Finally, a minimum value of 20 is set for determination to prevent inaction from occurring.

In contrast with (Xu, Low, and Choo 2009), the waypoints composing the Red vessels' trajectories are here evolved automatically through the ART process. The Non-dominated Sorting Algorithm II (NSGA-II) is employed to conduct the evolutionary search. This algorithm is executed using the following parameters: population size = 100, number of search iterations = 200, mutation probability = 0.1, mutation index = 20, crossover rate = 0.9 and crossover index = 20. These parameter values are commonly used, as reported in the literature, when MSGA-II is applied to two-objective optimization problems. The population size and number of search iterations indicate that 20,000 distinct MANA simulation models are generated and evaluated for each experimental run. Each individual simulation model is executed/replicated 30 times to account for statistical fluctuations.

The efficiency of the search is measured by the number of Green casualties with respect to the number of Red casualties. In other words, the NSGA-II objectives are:

- To minimize the number of Green (commercial) vessels "alive".
- To minimize the number of Red casualties.

Considering the current scenario, these objectives are thus conflicting. In the next section we report our series of experiment using the above model.

4.3 Case Study 1

As outlined earlier, the simulation model time limit may potential affect the outcomes of the evolutionary search. Therefore it is here increased (from 250 up to 2000 time steps) to allow for complex trajectories to emerge and potentially ameliorate the Red tactical plans. Given the results reported in (Xu, Low, and Choo 2009), it was moreover predicted/expected that the ART process would result in generating more efficient Red tactical plans through exploiting more complex trajectories (for instance some Red vessels could divert Blue patrols whilst the remaining Reds cover the anchorage area).

Six independent CASE runs were conducted. Figure 3 depicts the final set of best solutions resulting from these experimental runs. The dynamics of the hypervolume indicator provide information with regards to the improvement of the Pareto optimal front over time. This approach considers the hypervolume of the dominated portion of the objective space as a measure for the quality of Pareto set approximations. This