1-1-2010

RedTNet: A network model for strategy games

Philip Hingston
Edith Cowan University

Mike Preuss

Daniel Spierling

10.1109/CEC.2010.5586505

RedTNet: A Network Model for Strategy Games

Philip Hingston, *Senior Member, IEEE* and Mike Preuss, *Member, IEEE* and Daniel Spierling, *Member, IEEE*

***Abstract*— In this work, we develop a simple, graph-based framework, RedTNet, for computational modeling of strategy games and simulations. The framework applies the concept of red teaming as a means by which to explore alternative strategies. We show how the model supports computer-based red teaming in several applications: realtime strategy games and critical infrastructure protection, using an evolutionary algorithm to automatically detect good and often surprising strategies.**

I. Introduction

In this paper, we are concerned with using computational models to investigate strategies for competing teams of agents. Whether it be in the context of a realtime strategy game or a real-world simulation, many aspects of the problem are the same, and a similar framework and similar methods can be used to explore the various strategies and tradeoffs available.

What is a strategy? Depending on who is asked, one may receive very different answers. In game theory, there is usually a fixed set of options available from which the strategy is chosen. For more complex games as e.g. realtime strategy games, the term may rather apply to a high-level plan which is employed as a general guideline followed until a faction either succeeds, or finds that its strategy becomes unsuitable due to unforeseen developments. In automated red teaming, which may be applied in military (e.g. wargaming) and civilian (e.g. *critical infrastructure protection* or CIP) scenarios, the first answer appears too restrictive as the setting can be too complex to investigate all possible options. So one may find oneself somewhere inbetween the two extremes, where a strategy could be a plan, but with much fewer degrees of freedom than in a full-fledged video game.

Classical military literature sees strategy as the way to set up a battle whereas the battle itself is dominated by tactics [1] (SunTzu). In a slight variation of this view, Clausewitz [2] demands that strategy must define the task of a battle, the reason why it is fought. Transferred to a CIP or RTS setting, it is up to the strategy to bring a specific set of units at a specific time to a point of interest, whereas it is up to tactics to win the resulting battle against the encountered enemy forces. Tactics thus cannot be modeled without detailed information about the setting, such as e.g. unit strengths and ranges or terrain properties. Attempts to do this possibly started with Reiswitz [3] and led to modern table top and computer war games and simulations. All these aspire to model reality as closely as possible. However, for strategy exploration, this degree of detail seems unsuitable and a much simpler setting is needed that carries only the most important aspects of the modeled situation.

In this work, we suggest RedTNet (for automated red teaming network game), a framework for exploring strategies in a simple, graph-based representation. We demonstrate that CIP as well as RTS settings can be mapped easily into RedTNet, and that optimization algorithms such as *evolutionary algorithms* (EA) are useful for learning good strategies to win the game. These strategies can then be analysed to obtain an improved understanding of the strategic situation.

The structure of the paper is as follows. In the next section, we provide background and review related work in automated red teaming, particularly in regards its application to two important application domains: critical infrastructure protection and realtime strategy games. In section III, we introduce our proposed framework, RedTNet. We then outline the requirements for agents to play RedTNet, and describe the player representation that we use for this study. Next, we describe a series of computational experiments that we carried out to test and validate our proposed framework. The results of these experiments are described and discussed, and in the last section, we review our findings and outline some likely areas for future work, based on what we learned in these experiments.

II. Background

A. Red Teaming

Red teaming is a method for assessing vulnerabilities in systems or structures. Two factions or teams – red and blue – are posited or formed. The red team is charged with attacking the system or structure being defended by the blue team. The role of the red team is to challenge the implicit assumptions in blue team defences. Although the idea originated in the context of military simulations or wargaming, it can be applied more broadly to, for example, civil defence scenarios, security assessment, business decision-making, and computer network vulnerability assessment. In the context of games, red teaming is a natural way to think about attack plans in realtime strategy games (RTS).

1) Automated Red Teaming: Traditionally, red teaming has been done manually, either in a manual simulation on a board or table, or in physical wargaming, or with real teams physically infiltrating a secure facility. More recently, computer-based simulations are also used, and have the advantage that many scenarios can be simulated and analysed.

The first to suggest using evolutionary algorithms and agent-based simulation for automated red teaming may have been Upton et al. [10]. Their application was testing proposed

Philip Hingston is with the School of Computer Science and Security, Edith Cowan University, Australia (phone: +61 8 9370 6427; email: p.hingston@ecu.edu.au).

Mike Preuss and Daniel Spierling are with TU Dortmund, Germany.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

security procedures. Details are sketchy, but an Evolutionary Programming algorithm was used to evolve the parameters of a red team strategy to defeat a fixed blue team strategy for defence of a fixed structure. This idea has been taken up and developed into the ART framework by researchers at Singapore's DSO and Nanyang Technological University [9], [11], [7]. The ART framework integrates an optimisation algorithm with an agent-based simulation. It currently supports particle swarm optimisation and a multi-objective evolutionary algorithm as the optimiser, and several simulations models, chiefly MANA[15]. ART has been used in a series of data farming workshops [12], [13], [14] for applications including urban operations, maritime defence and anchorage protection, and is claimed to be able to discover non-intuitive tactics that are superior to those obtained by manual red teaming.

Some work has also been done by Ang et al. [8] using a simple (1+1) Evolution Strategy algorithm, coupled with WISDOM, a low-resolution simulation model for military simulations. The aim of their study was to investigate the nature of the fitness landscape taking into account the personalities of the red and blue teams. The early part of the paper provides a useful survey of computational tools and techniques that are available for defence games.

To date, most work in automated red teaming has been in defence related application, but there are many potential applications in civilian settings. One important application where automated red teaming could have great benefits is in critical infrastructure protection.

B. Critical Infrastructure Protection

Critical infrastructure protection is an increasingly important security issue in modern society, with threats from natural disasters and terrorist groups. Critical infrastucture is "The array of physical assets, processes and organizations across which [essential goods and services] move" [5]. Examples include power plants and distribution networks, transportation systems, computer-based control systems, computer networks and so on. One of the aims of this work is to develop a red teaming framework that can be applied in the domain of critical infrastructure protection, as well as in RTS games and other applications.

When analysing requirements for critical infrastructure protection, it is important to take into account interdependencies between infrastructure. An example is described in [4] – In July 2001 a freight train derailed in a tunnel Baltimore. This started a fire, which broke a water main, causing local flooding which cut the electricity supply. Fibre optic cables were also damaged, affecting telecommunications, and so on. These interdependecies can be modelled using a layered network of nodes and connections, as in Figure 1. Nodes in each layer represent infrastructure assets, such as a water main, or a police station, or a telephone exchange. Not only are there connections within layers (the dotted lines), between, say, a pump and a water main, but there are connections between layers (the solid lines with arrows),

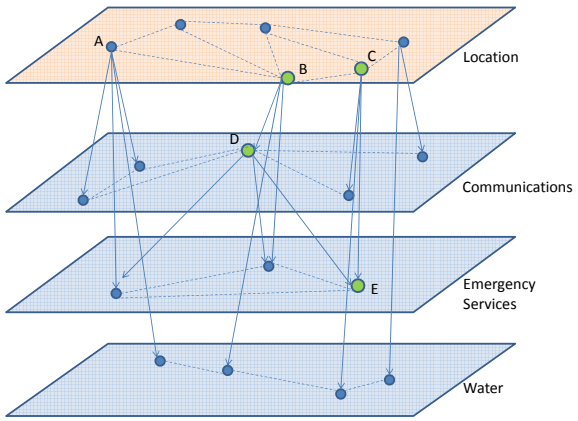

Fig. 1. An example of an infrastructure dependency network with a location layer added. Each infrastructure node is connected to a location node indicating that control of the location gives control of the infrastructure. Critical subnetworks of the infrastructure network (shown using large green nodes) thus induce critical subnetworks of the location network. RedTNet is played on the location network.

for example a police station may depend on a telephone exchange for communications.

Suppose that a dissident or terrorist group wanted to disrupt an infrastructure network by physically attacking and destroying, damaging or taking control of nodes in the network. To model this kind of scenario, we propose to add an additional "location" layer. Nodes in this layer represent physical locations from which infrastructure assets can be controlled. Connections within this layer represent pathways for movement between locations. Every infrastructure asset node is connected to a physical location node, and a single physical node may be connected to several infrastructure nodes, either in the same or different layers, or both (for example, node A in the figure). Some physical nodes may be connected to no infrastructure nodes, but represent "waypoints" that must be passed through to move between locations.

From the network of interdependencies, it is possible to determine certain "critical subnetworks", such that a failure of all the nodes in a subnetwork would cause a catastrophic failure of the whole system. For example, a failure of both the police station (E) and the communications hub (D) in Figure 1 might prevent an effective police response. This critical subset induces a corresponding critical subnetwork in the location network (nodes B and C).

In order to apply automated red teaming to critical infrastructure protection, a framework is needed that includes support for modeling interdependencies, including critical subnetworks. We introduce such a framework in Section III below.

We are not aware of any existing automated red teaming work in the CIP domain, however, Permann [6] proposes to

use agent-based simulation to model critical infrastructure networks, along with genetic algorithms to optimise decisions on which assets to protect and restore in the event of an attack or other disaster.

C. Realtime Strategy Games

If one breaks down a concrete RTS game into the strategical challenges it offers the player (and the game AI), the situation is not much different from the CIP context. Most RTS games have very obvious critical points: The home bases and the satelite bases possibly built during the game. Losing them usually leads to instant defeat (e.g. in Starcraft a faction without buildings cannot exist). Some recent games explicitly utilize more critical points. Conquering these makes more ressources available (e.g. *Dawn of War*, *Battle for Middleearth*) or triggers an explicit termination criterion (*Dawn of War*).

A 'strategy machine' such as the RedTNet framework introduced in Section III could be a very valuable tool for the game designer as well as for online adaptation of the game towards the gamer's needs. Instead of test playing a designed level to see if it is doable, one may try to automatically obtain a strategy that beats a defending strategy. If the AI strategy employed in the real game is transferred to the framework, obtaining a good opponent strategy can be formulated as an optimization problem. Depending on the hardness of this problem, which may be detected experimentally just by applying an optimization algorithm, one may judge if the AI strategy is good enough to cope with many different player strategies. Of course it is still possible that a strategy exists that beats the AI, but many of the strategies the optimization algorithm is allowed to try are automatically tested this way. One may even set up a scenario in a way that it provides a predefined approximate level of difficulty for the opponent, thus balancing the chances of AI and player via changes in the setup (more/less units, more/less critical points etc).

Of course, a similar method may also be used for obtaining a good attacking strategy. If, for a fixed scenario, the size and distribution of forces of the defending faction is known at least approximately, one may try to simulate different strategies (or do that automatically via an optimization algorithm) and take the best one available. What 'best' actually means is a matter of the concrete tasks, which should be modeled after the game winning conditions.

III. RedTNet Framework: the Red Teaming Network game

A. Rules of the Game

As suggested in Figure 1, RedTNet is played on a single network layer, consisting of a set of nodes G and edges E. A number r of red (attacking) agents is placed on a specific red home base node, and b blue (defending) agents is put onto a specific blue home base node. It is also possible to place agents on other nodes at the start and to define multiple home base nodes. Figures 2 and 3 show two example networks—these are described in more detail below. We assume that all nodes and edges have infinite capacity and that edges are bidirectional, so that any number of agents can travel from any node to any other neighboring node. The only nodes which cannot be visited by the opposing faction are the opponent's home base nodes. The home base nodes resemble locations outside the game – their edges characterize where the agents of a faction can enter the network.

Some of the nodes are marked as belonging to critical subsets. There may be one or several critical subsets with one or more nodes each. Possession of these nodes decides the game: If the red faction manages to conquer all nodes of any critical subset and concurrently holds them, it wins. If this does not take place before a predefined time is over, blue wins (it managed to defend the critical subsets). We denote each critical subset by C_i, i being the number of the subset.

The game is played in rounds, but as e.g. in the Diplomacy game (see section VI), factions move simultaneously. Thus, the rounds may be interpreted as planning breaks in which the targetted moves can be set up. Each round consists of three phases:

1) Either faction moves any number of its own agents to their neighboring nodes or lets them stay where they are.
2) For all nodes, the network is updated simultaneously. Movements of either side are executed tentatively and then for every node, conflicts are resolved in the following way: The faction with the majority of agents wins the node, and the agents of the opposing faction are removed from the game. In case the forces of both factions are equally strong, blue wins the node and the red forces are eliminated.
3) If red owns all nodes of any critical subset, it wins. If this is not the case and the predefined number of rounds is over, blue wins. In any other case, the game continues with phase 1.

For the time being, we assume perfect information concerning the current status of each node, which is node possession and the number of agents on the node. However, the opponent cannot see the planned moves of the other faction before the network is updated. Note that the game is asymmetric: The start and winning conditions for red and blue are not identical. A strategy working well for red may completely fail for blue and vice versa.

B. Example scenario 1: DoubleCrit - a simple CIP-inspired network

The first example (Figure 2) resembles a critical infrastructure protection scenario. There is only one critical subset with two nodes, I and H. Blue starts with 35 agents distributed over three nodes (10 on F, 10 on I, and 15 on J), and does not have a home base, red starts with 20 agents on the red home base and 10 on node E.

Blue does not have sufficient forces to defend both critical nodes at the same time against a full force red attack, so he cannot simply sit and wait. On the other hand, red does not have sufficient forces to overwhelm a critical node

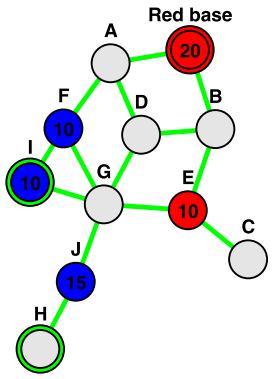

Fig. 2. The network definition for Scenario 1, DoubleCrit. Nodes occupied by any faction are shown in the appropriate color. Nodes with a surrounding green circle belong to a critical subset (here, we have only one critical subset, with two nodes). Other nodes with a surrounding circle indicate a home base node.

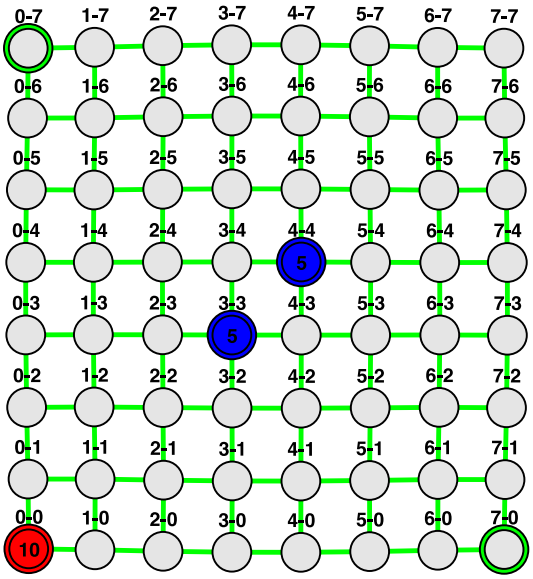

Fig. 3. The network definition for Scenario 2, an 8x8 RTS-style grid.

defended with more than 20 agents. Thus, neither player has an obviously trivial winning strategy.

C. Example scenario 2: RTS-inspired grid

The second example (Figure 3) has some resemblance to the first example, but models an RTS game map with its typical rectangular grid. We employ it in two sizes, 8x8 and 10x10. Again, there is only one critical subset with two nodes, 0-7 and 7-0, and 0-9 and 9-0, respectively. Red and blue both have 10 agents, red starting with all agents its home base 0-0, blue starting in two middle locations 3-3 and 4-4, and 4-4 and 5-5, respectively, having 5 agents on each starting point. Note that red does not have sufficient agents to attack the full force of blue agents at once.

IV. Players and strategies for RedTNet

In order to detect vulnerabilities in defence strategies, the behaviour of the red team has to be unpredictable, preventing

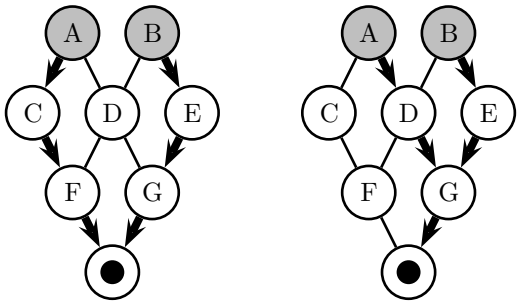

Fig. 4. Two possible attacks for a player that attacks directly.

an optimisation of the blue team towards one special offense tactic. For this reason, we developed several different player types. This also allows us to test the behavior of the blue team in extreme cases. Each of these player types operates using a similar principle:

- for each move, a random set of agent movements is generated by iterating over all nodes occupied by this player's agents, selecting a fraction to be moved, and selecting a valid adjacent node to move them to. Note that for some nodes, the fraction may be zero;
- these sets of movements are analysed with respect to some criteria and a "quality" function value is calculated;
- the best set of movements is selected, based on this quality function value.

This procedure provides unpredictability in the selected moves of all player types. Different kinds of quality function produce different player types.

A. Critical Weight Player

A common strategy to attack an opponent is to focus on the target directly. In this process all the agents concentrate on achieving their subtargets together. In general a path with the shortest distance is chosen to realise this intention. Unpredictability with this strategy comes from the selection of random fractions of agents, as well as from random selection between equal length paths. Figure 4 shows different attack paths for a red team starting from two different nodes. The combination of concentrated power and unpredictability make it difficult to defend against this kind of opponent. This strategy is realised in the Critical Weight Player (CWplayer).

The quality function for the CWplayer $fit_{CW}(m)$ calculates quality by evaluating the positions of all its own squads $s_i \in S_{own}$ as in Equation 1. Positions are rated using the minimum and average values of the distances $d_c(s_i)$ of each squad to the next critical node. These values are multiplied with two weights w_{min} and w_{avg}, which are initialised with 1 and adjusted during the optimisation.

$$Q_{CW}(m) = w_{avg} \times \frac{\sum_{s_i \in S_{own}} d_c(s_i)}{\sum_{s_i \in S_{own}} |s_i|} + w_{min} \times min(\{d(s_i) | s_i \in S_{own}\}) \quad (1)$$

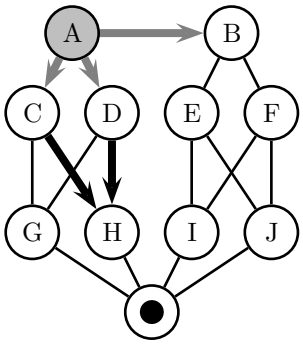

Fig. 5. Splitting and merging can be used to spread danger and then concentrate it again.

B. Local Weight Player

In contrast to the straightforward CWplayer, a more complicated strategy is to split the attacking force into several squads. This approach spreads the danger for the defender over more locations. Rather than attacking head-on, detours can be made without losing sight of the general intention. Figure 5 gives such an example. Three squads are formed, two of which merge again later, after passing through nodes that might not be considered otherwise.This combination of spreading and consolidation makes defense very difficult.

Two player types, the Local Weight Player (LWplayer) and Target Weight Player (TWplayer) are designed to be able to use these kinds of stategies. The difference between them is that TWplayer concentrates on few nodes whether LWplayer considers the full network.

The quality function for the LWplayer $Q_{LW}(m)$ is shown in Equation 2. All nodes $n_j \in N$ are assigned two weights, representing how important it is to own them – one weight for oneself $w_{own}(n_j)$ and one for the opponent $w_{opp}(n_j)$. Here, $o(s)$ denotes the node which is occupied by squad s.

$$\begin{aligned} Q_{LW}(m) &= Q_{CW}(m)+ \\ &\quad \textstyle\sum_{n\in\{o(s)|s\in S_{own}\}} w_{own}(n)+ \\ &\quad \textstyle\sum_{n\in\{o(s)|s\in S_{opp}\}} w_{opp}(n) \end{aligned} \tag{2}$$

C. Target Weight Player

The quality function for the TWplayer (see equation 3) takes account of the cohesiveness of the squads by calculating the distance to the next friendly node $d_{own}(s_i)$ in relationship with the longest path of the network $lp(N)$. The influence of the distance to the next critical node is multiplied by an adjustable weight w_{d_c} and $lp(N)$. The quality calculation includes a term for every node the player occupies, allowing the player to respond to the game situation. In addition, a term representing the distance to the enemy $d_{opp}(n)$ in relationship with the longest path in the network, is multiplied with a "threat" weight $w_{threat}(n)$ for the opponent owning the node.

$$\begin{aligned} Q_{TW}(m) &= \textstyle\sum_{s_i\in S_{own}} \frac{lp(N)-d_{own}(s_i)}{lp(N)} \times w_{own}(o(s_i))+ \\ &\quad \textstyle\sum_{s_i\in S_{own}} w_{d_c} \times lp(N) \times d_c(s_i) \\ &\quad \textstyle\sum_{s_i\in S_{own}} \frac{lp(N)-d_{opp}(s_i)}{lp(N)} \times w_{threat}(o(s_i)) \end{aligned} \tag{3}$$

D. Blue team strategies

All the above players can also be used to play blue team strategies. The CWplayer attempts to controls the critical nodes by concentrating its forces at them. Both other players attempt to dominate the regions around critical nodes.

V. Experiments

To demonstrate the usefulness of our framework as a development, search and testing engine for strategies, we undertake two experiments, one on a CIP-type network, and one on an RTS-type network. As our players use real-valued representations, we chose a CMA-ES [19] to learn a good red team strategy. The general learning scheme tests every generated strategy in 100 games and computes the fitness (to be minimized) as the fraction of blue wins. See sect. IV for a description of the representation. We are generally interested in the possibility of automatically learning good red team strategies against different opponents and regard it as success if the red team beats the blue team in more than 50% of the cases. However, the theoretical limits for each combination of players under the given agent numbers and time limits are not known, so that 50% could be unreachable. In these cases, we at least require a visible improvement over the initial (random initialization) success rate.

Experiment: Can we evolve a strategy beating a nearly arbitrary static opponent in a CIP-network via an EA?

Pre-experimental planning. During initial experimentation, we tried different simple networks and decided to use the DoubleCrit network with one critical subset of two nodes (Figure 2) as it poses a certain strategic challenge without being too complex. The CMA-ES was run with different amounts of fitness evaluations, finding a run length of 500 evaluations sufficient to obtain reasonable progress. Attempts to find a good value for the number of targets in a red TWplayer led to a compromise value of 5, which seems to work well for many different networks.

Task. We test the hypothesis that by learning, we can obtain a red player variant that reliably beats all static blue players yet defined. Therefore, we require that the finally obtained players are a) statistically different from the best of the randomly initialized players in the starting population, and b) play significantly better than 50% success rates. Statistical difference shall be measured by Wilcoxon-ranksum tests (U-tests).

Setup. We run 21 repeats of CMA-ES runs of the following player combinations (red-blue): TWplayer vs. CWplayer, TWplayer vs. TWplayer, LWplayer vs. TWplayer, and LWplayer vs. CWplayer. Each individual is allowed 20 test moves in each iteration and is tested with 100 games. The

maximum number of iterations of each game is 20. The CMA-ES parameters are set to default, except the population size ($\mu = 20$) and the initial stepsize $\sigma = 0.2$. The blue players are run in default configuration, which is weights of $\{0.5,0.5\}$ for the CWplayer, and one target for each critical node for the TWplayer. The number of targets for the red TWplayer is set to 5.

Results/Visualization. Table I reports the median and best fitness values of the initial populations (of 20) and the final best fitness, all averages of 21 runs. P-values of significance tests between initial best and final best are all around or below 10^{-4}, and the same holds true for the tests of the final best against the winning rate of 50%, with the exception of LWplayer against TWplayer, which is significantly worse than 50%. Additionally, figure 6 visualizes one of the best TWplayers against a CWplayer (first combination). The learned weights of the TWplayer are depicted in table II.

TABLE I

BLUE WINNING RATES (=FITNESS) OF DIFFERENT PLAYER COMBINATIONS PRIOR TO AND AFTER OPTIMIZATION.

Red player	blue player	median init	best init	best final
TWplayer	CWplayer	0.992	0.509	0.123
TWplayer	TWplayer	0.984	0.735	0.276
LWplayer	TWplayer	0.995	0.803	0.677
LWplayer	CWplayer	0.998	0.259	0.083

TABLE II

LEARNED WEIGHTS OF ONE OF THE BEST TWPLAYERS FROM GAMES AGAINST THE CWPLAYER.

	targets	t 1	t 2	t 3	t 4	t 5
Target nodes		I	C	D	E	B
posession weights		-0.084	0.422	0.054	-0.003	0.753
threat weights		0.627	1.0	0.624	0.456	0.619

Observations. The median initial fitnesses of all four combinations are near 1, meaning that a random individual usually cannot beat the blue adversary. Furthermore, the ordering of the combinations induced by the best initial fitnesses is consistent with the one induced by the final best fitnesses. The TWplayer seems to be harder to beat than the CWplayer (both final fitness values are worse for the TW).

Discussion. We assert that by learning, the red team strategy is improved considerably in all cases, beating the blue strategy with probability $p > 0.5$, except for the LWplayer against the TWplayer. As the largest conceptual difference of CWplayer and TWplayer is that the latter takes enemy movements into account to a certain extent, which are ignored by the CWplayer, it may be this property that makes the TWplayer stronger and harder to beat. It should also be remembered that the LWplayer needs to adapt a larger set of variables ($2 + 2 \cdot \#\text{nodes} = 24$) than the TWplayer

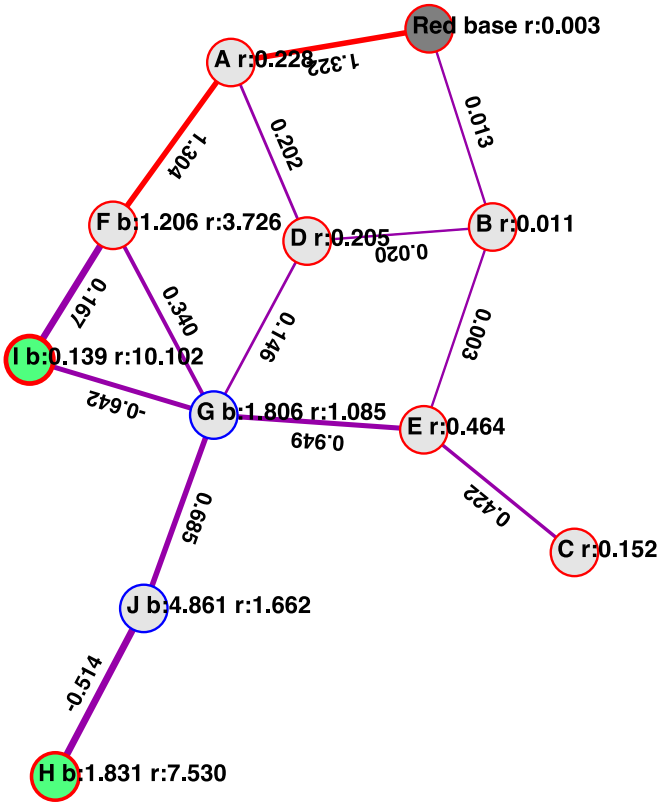

Fig. 6. Accumulated games of one of the best TWplayers against a CWplayer on the DoubleCrit network. The nodes hold the average number of agents (b=blue, r=red) when the game was finished, the edges hold two different pieces of movement information. Widths reflect the number of (all factions) agents moving through an edge per game, log10-scaled. The numbers indicate the faction that accumulated more moves, positive for red and negative for blue (absolute values are also log10-scaled. Recall that blue starts at F (10 agents), I (15), and J (15), red at the red home node (20) and E (10). The two critical nodes are I and H

$(3 \cdot \#\text{targets} = 15)$ so that learning a good strategy may take more time. Interestingly, the strategy optimization problem is seemingly not deceptive — one may get a first impression of its difficulty from a small random sample.

To get an impression of how a successful player looks after learning, we discuss the TWplayer of table II, who is able to beat a default value CWplayer in about 90% of the cases. The CWplayer largely concentrates on protecting the critical subset nodes H and I and does not react to red agents moving towards it. In the board values used for the internal move composition of the TWplayer, small values are better. Thus, low possession weights mean that a node is interesting, whatsoever other agent may be on, and high values mean that it is to be avoided. The threat weights mean that a node is to be avoided if the value is high and there are some enemy agents on (and is attractive if enemy agents are on and the value is negative). For the given network, the TWplayer is largely built to avoid nodes instead of moving to them. Only the built-in reward for minimizing the distance to critical nodes induces a direction to move to. By means of the weights, the TWplayer learns a way by learning where NOT to go. It avoids B and C at all cost, and additionally I,

TABLE III

BLUE WINNING RATES (=FITNESS) OF DIFFERENT PLAYER COMBINATIONS, 8X8 AND 10X10 RTS NETWORK

Network	red player	blue player	med.init	best init	best final
8x8	TWplayer	CWplayer	0.999	0.823	0.293
8x8	TWplayer	TWplayer	0.996	0.695	0.305
10x10	TWplayer	CWplayer	1.0	0.941	0.293
10x10	TWplayer	TWplayer	1.0	0.906	0.684

TABLE IV

LEARNED WEIGHTS OF ONE OF THE BEST TWPLAYERS FROM GAMES AGAINST THE CWPLAYER IN THE RTS SCENARIO.

	targets	t 1	t 2	t 3	t 4	t 5
Target nodes		5-9	3-2	4-2	0-2	6-4
posession weights		0.159	0.376	-0.477	1.0	0.936
threat weights		0.518	0.981	0.559	1.0	0.135

D, and (weaker) E if enemies are on it. We can assume that it moves down on the left side, avoiding I (moving through F) and meets the other group of agents starting from E at G or J, and then attacks H with a clear majority. Then it moves all agents back to attack I and wins.

The results show that with the combination of the RedTNet and an evolutionary optimization algorithm, one may explore and test the 'strategy space' of a red team player in this CIP-like network.

Experiment: Does strategy learning scale well to larger networks?

Pre-experimental planning. Initial experiments on larger networks as used in this experiment revealed that learning of strategies is still far from completed after 500 evaluations, thus the number of evaluations was increased to 2000.

Task. The tested hypotheses are the same as in experiment 1. Additionally, we investigate the strategy learning difficulty related to map size and test the results of otherwise equal situations of a smaller map against a larger map. Even if the difference is statistically significant (U-tests), it should not be too large. A difference of $\approx$10-20% in winning fractions of the (averaged) best red strategies would be tolerable.

Setup. We largely employ the same setup as for experiment 1, with the following changes. To resemble an RTS-like scenario, we use an 8x8 network (Figure 3) and an enlarged, but structurally identical network of size 10x10. The CMA-ES is allowed 2000 function evaluations, and the TWplayer is not used as it would require excessively large strategy representations of 130 and 202 variables, respectively.

Results/Visualization. We report the learning results in table III, again as means of 21 runs, median initial, best initial, and best final fitness. P-values of the tests between best initial and best final are all below 10^{-3}, and except TWplayer against TWplayer on the 10x10 network, all final best values are significantly better than 0.5 (p-values below 10^{-2}). Comparing the final best of row 1 and 3, and 2 and 4, respectively, also results in statistical significance (results are not equal). In case of rows 1 and 3, this is due to high variance, despite identical mean values. Figure 7 visualizes one of the best TWplayers against a CWplayer on the 10x10 map, the learned weights of the TWplayer are given in table IV.

Observations. Learning strategies against the CWplayer appears to be more difficult in the beginning for both network sizes, however final results are also much better. Interestingly, learning a good anti-TWplayer for the large network is much harder than for the small network whereas there is little difference when learning against the CWplayer.

Discussion. It seems that the difficulty of learning an opponent for the CWplayer does not depend very much on network size, but it does for learning against the TWplayer. The reason for this could be that the CWplayer ignores the location of its opponents squads, focusessing only on possessing the critical nodes or a subset of these. However, the TWplayer can react to what its opponent does and then has more move alternatives on the large map, possibly avoiding the attacking red squads. Again, we have a look at one exceptionally good TWplayer from games against the CWplayer, on the 10x10 network, depicted in fig. 7. The driving factors of its behavior are mainly that 0-2 is to be avoided (whether or not possessed by blue), and that there is a weak attraction towards 4-2, so that the TWplayer choses to go down towards the lower critical node. Somewhere on the way to it or at the critical node, it seemingly meets not yet fully assembled groups of blue agents (note that blue takes all possible routes to the lower critical node, although that does not necessarily mean that this happens in one game, but it is still realistic to think of two parties of blue moving separately). So what the red player actually learned is to take out the blue agent groups one after the other, which results in all blue agents being killed, which is the alternative winning condition. Note that getting into possession of both critical nodes would be much more challenging. It requires a well timed distribution of forces as red needs a majority of agents to beat the blue ones near the lower critical node (both forces start with 10 agents).

VI. CONCLUSION AND FUTURE WORK

We have presented and experimentally evaluated a simple network-based framework which is suited well for evolving and evaluating strategies. Application to CIP and RTS based scenarios are straightforward. Inspection of single strategies via visualization has led to several surprises: Automatically developed player strategies are successful, but often chose paths a human player would probably not prefer, which makes our framework a viable help for detecting and evaluating strategies and counter-strategies in several contexts. As expected, we observed that reactive strategies are harder to beat than static ones, although our results in this direction are rather preliminary.

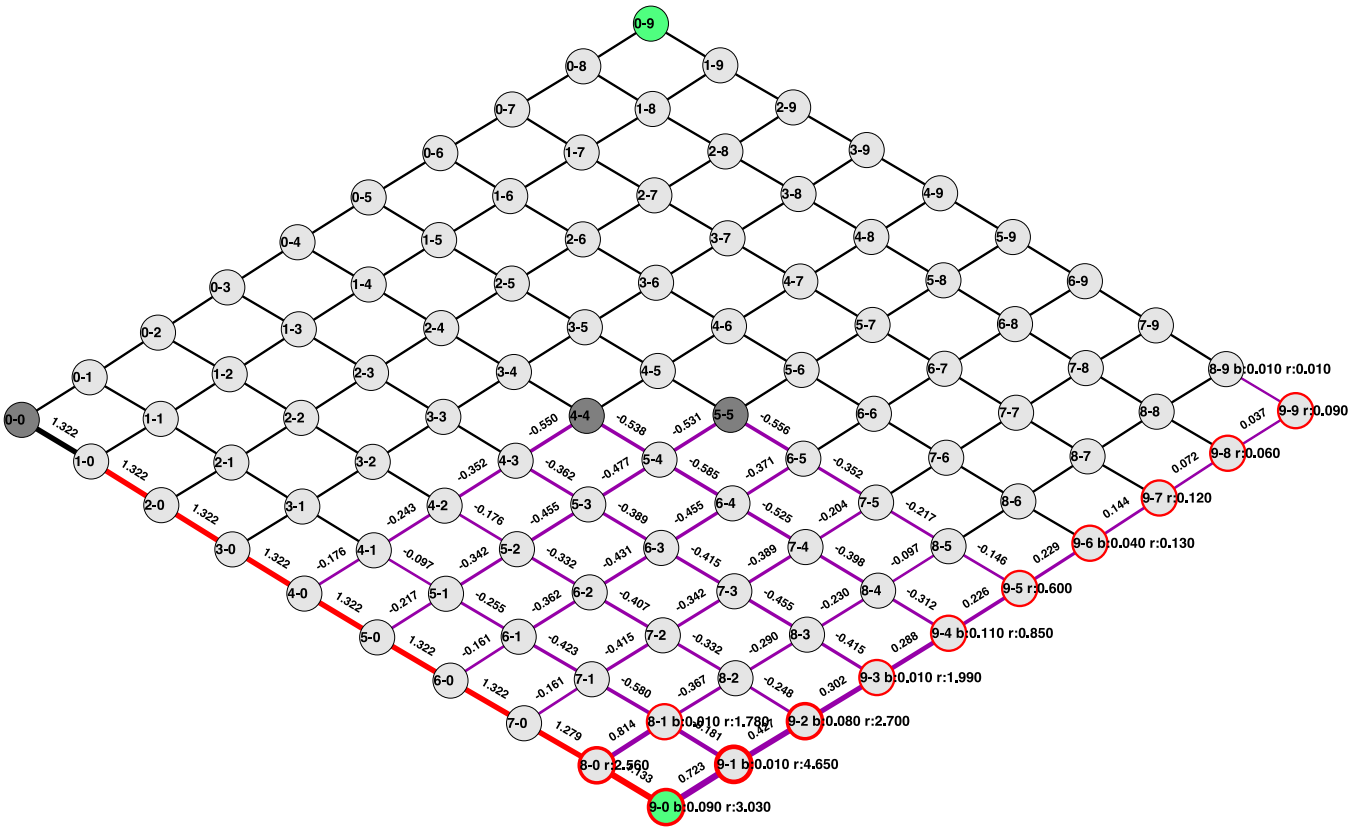

Fig. 7. Accumulated games of one of the best TWplayers against a CWplayer on the 10x10 network. Data presentation is similar to the one of fig. 6. Blue starts at 4-4 (5 agents) and 5-5 (5), red starts at 0-0 (10). The two critical nodes are 0-9 and 9-0

Some possibilities for the extension of our framework have already been mentioned: finite capacity edges and information hiding. Another possible modification would be a non-deterministic rule to determine the outcome when both teams attempt to occupy a node simultaneously. The current majority rule has the advantage of simplicity, but a stochastic rule might be more realistic. It is relatively easy to *emulate* related strategy games with RedTNet. With a capacity of one for every edge and slightly changed update rules, one obtains rules that resemble the board game Diplomacy very closely. This game was invented by Calhamer in 1959, also see [16], and has attracted some interest already in the CI in games community [17], [18] due to its strategic variability despite simple rules.

In the future, we also plan to investigate more complex strategy spaces, and more intelligent agents, on different networks, and using different optimization methods. For example, we will use coevolution of red and blue team strategies.

References

[1] SunTzu, Giles, L.(translator): The Art of War, El Paso Norte Press (2005), *original Chinese version from 600 BC*

[2] Clausewitz, C. von: Vom Kriege, Dümmlers Verlag, Berlin, (1832), *in German*

[3] Reiswitz, G.H.R. von: Anleitung zur Darstellung militairischer Manöver mit dem Apparat des Kriegs-Spieles. Berlin, (1824), *in German*

[4] Pederson, P., Dudenhoeffer, D., Hartley, S., Permann, M: Critical Infrastucture Dependency Modeling: A Survey of U.S. and International Research. `https://www.pcsforum.org/library/files/1159904563-TSWG_INL_CIP_Tool_Survey_final.pdf`, INL/EXT-06-11464, Idaho Falls ID, August 2006. (2006)

[5] Congressional Research Service Report for Congress. 2002 Critical Infrastructures: Background, Policy and Implementation. `http://www.iwar.org.uk/cip/resources/pdd63/crs-report.pdf` (2002)

[6] Permann, M. R.: Genetic algorithms for agent-based infrastructure interdependency modeling and analysis. In: Proceedings of the 2007 Spring Simulation Multiconference - Volume 2 (Norfolk, Virginia, March 25 - 29, 2007). Spring Simulation Multiconference. Society for Computer Simulation International, San Diego, CA, 169-177 (2007)

[7] Xu, Y.L., Low, M., Choo, C.S.: Enhancing automated red teaming with evolvable simulation. In: GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp.687–694, ACM, New York, NY, USA (2009)

[8] Ang Yang, Abbass, H.A., Sarker, R.: Characterizing warfare in red teaming. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , vol.36, no.2, pp.268-285, April 2006 `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1605376&isnumber=33735` (2006)

[9] Choo, C.S., Chua, C.L., Tay, V: Automated red teaming: a proposed framework for military application. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp 1936–1942, ACM, New York, NY, USA (2007)

[10] Upton, S.C., Johnson, S.K., McDonald, M. J.: Breaking Blue:

Automated Red Teaming Using Evolvable Simulations, Workshop on Military and Security Applications of Evolutionary Computation, GECCO 2004.

[11] Chua, C.L., Sim, W.C., Choo, C.S., Tay, V: Automated Red Teaming: An objective-based Data Farming approach for Red Teaming. In: Simulation Conference, 2008. WSC 2008. Winter , vol., no., pp.1456-1462, 7-10 Dec. 2008 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4736224&isnumber=4736042 (2008)

[12] Lee, M., Ang, D., Hung L.F.: Applying automated red teaming in an urban ops scenario. In Scythe 1: Proceedings and Bulletin of the International Data Farming Community, pages 2430, Montery, CA, USA, Naval Postgraduate School (2006)

[13] Sim, W.C., Choo, C.S., M-Tiburcio, F., Lin, K., Shee, M.: Applying automated red teaming in a maritime scenario. In Scythe 2: Proceedings and Bulletin of the International Data Farming Community, pages 2629, Montery, CA, USA, Naval Postgraduate School (2006)

[14] Wong, A. C. H., Chua, C. L., Lim, Y. K., Kang, S. C., Teo, C. L. J., Lampe, T., Hingston, P., Abbott, B.: Applying automated red teaming in a maritime scenario. In Scythe 3: Proceedings and Bulletin of the International Data Farming Community, pages 35, Montery, CA, USA, Naval Postgraduate School (2007)

[15] Lauren, M., Stephen, R.: Mana: Map-aware nonuniform automata. A New Zealand approach to scenario modeling. J. Battlefield Technol., vol. 5, no. 1, pp. 27-31 (2002)

[16] Calhamer, A.B., Calhamer on Diplomacy: The Boardgame Diplomacy and Diplomatic History, Authorhouse (1999)

[17] Johansson, S.J., Håård, F.: Tactical coordination in no-press diplomacy. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 423–430. ACM, New York (2005)

[18] Kemmerling, M., Ackermann, N., Beume, N., Preuss, M., Uellenbeck, S., Walz, W.: Is Human-like and Well Playing Contradictory for Diplomacy Bots? In: CIG'09: IEEE Symposium on Computational Intelligence and Games, pp. 209–216, IEEE Press, New York (2009)

[19] Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. In: IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 159-195 (2001)