Dynamic Games to Assess Network Value and Performance

Gregory Calbert, Peter Smet, Jason Scholz, and Hing-Wah Kwok

Command and Control Division, Defence Science and Technology Organisation
Edinburgh, South Australia, 5011
Greg.Calbert@dsto.defence.gov.au

Abstract: This paper looks at the analysis of network effectiveness and vulnerability through dynamic games. Each opposing side consists of a collection of vertices (pieces) connected in a network. Only vertices in the largest sub-graph exhibit mobility. We use the mobility and piece removal rules in the game checkers and modify the evaluation function to include sub-graph size balance. With this modified evaluation function, win-lose results of richly versus sparsely connected and centralised versus decentralized topologies are analysed. The results are compared with the current vulnerability studies in networks of varying topology. Finally we use temporal difference learning to calculate advisor weights for richly or sparsely connected vertices.

1 Introduction

The study of networks and their vulnerability to disruption or deliberate attack is a topic of increasing importance for society [2]. Many complex systems may be described in terms of networks where vertices represent the agents of the system and the edges represent interactions between agents [1]. With this view of networked agents, broad statistical properties of the system as a whole may be described. For example, the level of interaction within the system can be inferred from the edge richness within the network. Once a topologically dependent model of network function is found, critical agents within the system may be identified [2].

A wide ranging class of naturally occurring networks have similar statistical properties, in that the degree of such networks follows a power law [1]. Such networks include chemical interactions in animal metabolism, sexual partner networks, routing connections in the world-wide-web and power generation grids, to describe a few [1]. These networks, termed scale-free are characterised by a small proportion of highly connected vertices. In contrast, random networks, where each vertex has equal probability of being connected with any other vertex, do not have these highly connected vertices to any statistical significance [2].

Of importance is the performance of such scale free networks compared to randomly connected networks. Performance is defined either through the average diameter of the network or the size of largest connected sub-graph [2]. For the later

T.D. Gedeon and L.C.C. Fung (Eds.): AI 2003, LNAI 2903, pp. 1038-1050, 2003.

performance measure, as vertices are removed, the proportion of vertices in the largest connected subgraph drops to zero. For scale free networks, it has been shown that when random vertices, independent of their degree are removed or are no longer functional, the largest subgraph size decreases slowly [2]. However the largest subgraph size decreases drastically under network attack, when vertices with the highest degree are preferentially removed [6]. In contrast, random networks, having no central vertices of high degree show no difference in functionality between random vertex error or deliberate attack [2, 6].

Though these models of network performance are highly insightful, the results are generated assuming that under network attack, all vertices may be accessed with equal probability. There is no underlying model of the how an attacking system could manage to either sequentially or simultaneously disable vertices of high degree. In order to understand the true vulnerabilities of a networked system, one must model the dynamics of the attacking system at the outset. For example, if considering a computer virus disabling router nodes in a complex network, one must model the dynamics of transmission and replication through the network. When considering a police force attempting to disable a criminal network, one should simultaneously model the functionality of the criminal network as lynch-pin criminals are removed, as well as the communications/interactions of the legal/police network. When viewed this way, we have interactions between networks which we call a *network game*. Within each network, agents cooperate to achieve a non-cooperative goal against the opposing network.

In this paper we discuss non-cooperative dynamic games *between* two opposing networks. By this, we mean that within each network a cooperative game is played, in which a strategy is chosen by an agent controlling connected vertices to maximise some cost to the other network. In turn, an agent controlling the opposing network selects a strategy to minimize some cost function imposed by the original network. Vertices in these games are not only purely characterised by links with other vertices. In general, vertices will be characterised by a number of states, such as internal states (alive or removed from game) and possibly spatial states (coordinates of each vertex). Opposing network games are distinct from other *within* network non-cooperative games, in which each agent in the network adopts a strategy (such as choosing a packet routing path, or acceptance/rejection of market contract bids) to maximise its utility when competing against its direct neighbouring agents [11]. Such games have been extensively studied with communications or economics applications in mind.

Following this Introduction, we discuss formally modelling games between opposing networks in Section 2. We then discuss the network checkers game in Section 3. We next look at some results on the vulnerability of networks with differing topologies in Section 4 and look at valuing the network through reinforcement learning in Section 5. Discussion and Conclusion follow in Section 6.

2 Modelling Opposing Network Games

Formally, in order to specify the a game between two networks, we define at each time $t = 0,1,2,3,\cdots$ a sequence of graph pairs. Each graph pair corresponds to the new

vertex-edge set of each side, following the application of some strategy by the opposing network, such as the removal of an edge or the removal of a vertex. Thus, as the game progresses, we define the sequence of graph pairs as

$$(G_1(0), G_2(0)), (G_1(1), G_2(1)), ..., (G_1(t), G_2(t)), ... \quad (1)$$

where

$$G_i(t) = (V_i(t), E_i(t)) \quad (2)$$

specifies the number of vertices or elements and the topology of edges between them at the time t. The strategy used by a network will depend on the vertex states and the connectivity between them. In general, the strategy will be a map

$$S : G_1 \times G_2 \to G_j, j = 1,2. \quad (3)$$

In particular, let $S(v_j)$ be the strategy set for vertex $v_j \in V_i(t)$ at some time of the game t. Now suppose that vertices $\{w_1, w_2, ..., w_k\} \in V_i(t)$ comprise a connected sub-graph of the network. Then at time t, the sub-graph has strategy set

$$S(t) = S(w_1) \oplus S(w_2) \oplus .... \oplus S(w_k). \quad (4)$$

When coding an agent to play such network games, strategies can be evaluated through the use of heuristics or the more sophisticated game playing techniques. In particular, our approach is to use a linear evaluation function of weighted game features in combination with either min-max decision tree search or the use of temporal difference learning to find appropriate weights for the features chosen. Furthermore, we assume that the strategy set is found from the maximal subgraph, which is the connected graph with the largest number of vertices. The largest sub-graph may be found by applying Djistraka's algorithm on the current network [9].

3 Networked Checkers

We chose the game of checkers as our template game, in which the pieces become vertices for an underlying network [7]. There are several principal reasons for this choice. First the checkers pieces exhibit spatial manoeuvre. Our aim is to model manoeuvre based conflict between opposing sides, as seen in human sporting games and warfare, thus the game of checkers is seen as an important start. Coupled with manoeuvre is the attrition of forces in the game, again a feature of warfare. Finally, checkers with a well defined end-set of states[1] has had considerable research into the agent based play. This means we are able to conduct Monte Carlo simulations across many games in order to assess the performance of a particular network topology. There are three end states in the game of checkers- a win, loss or draw. Our measure

[1] Here, we assume the game ends either with the loss of all pieces or no mobility from one side or no pieces taken after 40 moves.

of network performance is the observed frequency of wins plus half the draws across games. If we define, for each end-state graph pair (G_1, G_2) the function

$$\varphi(G_1, G_2) = \begin{cases} 1 \text{ for win,} \\ 1/2 \text{ for loss,} \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

then we estimate the expectation $\mathrm{E}(\varphi(G_1, G_2))$. If $(G_1, G_2)^1, (G_1, G_2)^2, ..., (G_1, G_2)^n$ are independent and identically distributed random outcomes of the end-states of the network game, then our unbiased estimate of the expectation is

$$p = \frac{1}{n} \sum_{i=0}^{n} \varphi\left((G_1, G_2)^i\right). \tag{6}$$

Assuming we have a binomial model, then the unbiased estimate of the variance is

$$\mathrm{var}(p) = \frac{p(1-p)}{n}. \tag{7}$$

Network topologies are specified at the commencement of the game and only alter due to the removal of a vertex by the opposing network. Topologies are specified by the number of links between vertices at the instigation and the network growth rule. The maximum number of vertex to vertex connections is $\binom{12}{2}$ links.

Given a number of edges between vertices, the structural aspects of the topology are governed by a number of rules. In particular, the baseline topology is the *random network.* Here, two vertices are selected at random and an edge placed between them if not connected. This process continues till the link budget is exhausted. In this paper, we compare the performance of random networks with that of *hub-spoke networks.* As the name implies, the graph consists of one or number of centralised hubs, connecting to the spokes of the network. With a link budget of 11, the network is fully connected, with one central hub. Link budgets greater than 11 have additional hubs, with a new hub at link budgets of 12, 23, 33, 42, 50, 57 and 63 edges.

We have chosen the random and hub-spoke topologies to compare performance with communication network results [2]. In particular, we will compare the performance of centralised (hub-spoke) networks under network destruction, as the game is played, with that of decentralized random networks.

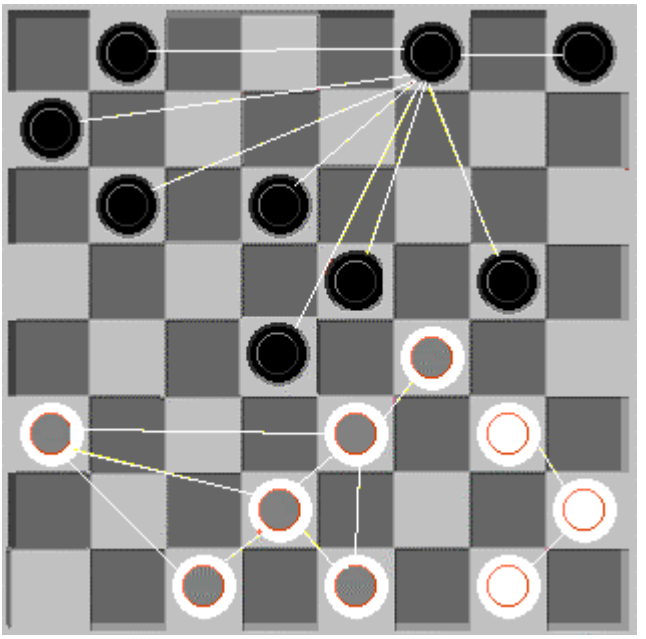

Fig. 1. Example of the network checkers game. Here the white side is connected with a random network topology, the red with a hub-spoke topology. Pieces on the white side which exhibit maneuver, the largest connected subgraph, are shown in shading

The results of Monte Carlo simulations will be dependent on the evaluation function used in the game. Indeed any performance results obtained must be coupled with the structure of the evaluation function [8]. At this stage of the research, we chose an simple as possible evaluation function that highlighted the features of materiel balance, value of more mobile kings as opposed to unkinged pieces, the relative rank balance[2] of pieces (to encourage king making) and the balance of the largest subgraph sizes, this being a measure of the strategy set and the network viability. Thus our evaluation function takes on the form

$$V = 100(V_1 - V_2) + 250(KV_1 - KV_2) + R_1^2 - R_2^2 + 100(N_1 - N_2), \tag{8}$$

where V_i, KV_i and N_i $i = 1,2$ are the number of unkinged, king and largest subgraph pieces respectively. R_i, $i = 1,2$ is the rank of each side.

It should be noted that the values for coefficients for the materiel terms in the evaluation have been taken from, a commonly played Java checkers web-site [7]. The value of 100 for the coefficient in the largest subgraph balance term was chosen as the minimal value congruent with materiel and mobility. There is however no a-priori experience for setting this value and machine learning techniques such as supervised and reinforcement learning are appropriate, as seen in Section 5.

[2] The rank is the sum of the distances of unkinged pieces from its own baseline.

4 Experiments with Network Checkers.

Monte Carlo simulations were conducted in the network checkers game. The focus of our research was on the frequency of performance success (wins plus half draws) as a function of different opposing network characteristics. In one set of trials, we considered the performance success as a function of the ratio of vertices to edges. As our starting number of edges is small, rather than considering this ratio, we only considered the absolute number of edges up to the maximum of 66. The commencing network topology in this set of experiments was random for each side.

Next we conducted a series of experiments to look at the performance of two different opposing topologies. In order to draw conclusions purely from the topological structure, the number of links was kept constant for each side during each set of simulations. The experiments focused on the random topology against the hub-spoke topology. The relative performance of the random topology was simulated, both as the level of edge connectivity increased and for differing strategy search depths of 0 (random legal moves played) to 3- ply search.

4.1 Results with Differing Edge Number

Simulations show that having an increased number of edges than the opponent does confer an advantage. This is not surprising as increased network connectivity implies both a larger maximum subgraph size and robustness over the game.

Of note is the performance of the white network against a red network either with 36 edges or 60 edges – nearly fully connected. Performance gains are not substantially greater, highlighting diminishing returns as edge numbers increase.

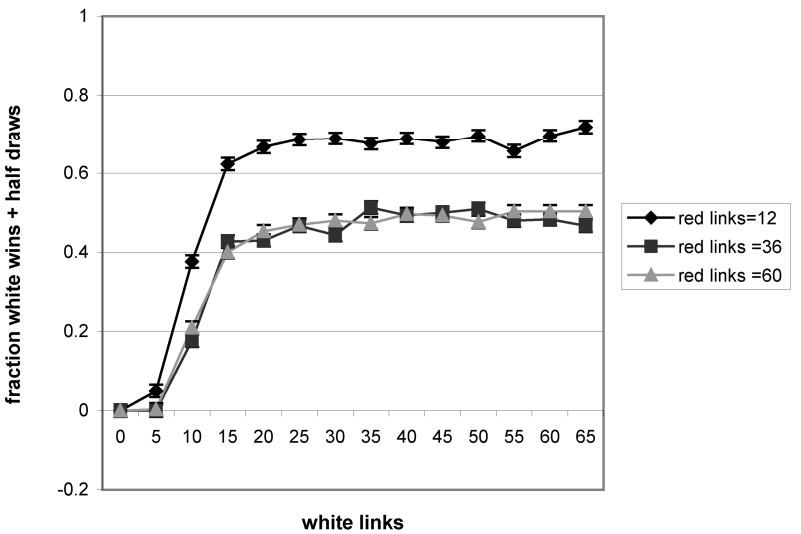

Fig. 2. Graph of the performance measure as a function the number of white edges

4.2 Results Comparing Different Topologies

Network performance and vulnerability models compare differing topologies to highlight the relative strengths and weaknesses of one structure over another. In these models, the network is eroded either by the removal of vertices or links and measures of performance, such as throughput or largest subgraph size are compared [1]. There are no assumptions on the network structure of the opposing agents. We call these models net versus no-net models. Here, we take the approach of comparing topologies by direct competition, a net versus net model. In this case, the removal of vertices is dependent on the performance of the adversary network, hence our model of performance may be seen as richer, as it does not make poor assumptions about the behaviour of the collection of agents attempting to destroy or disrupt a network.

In order to make a direct comparison between net versus no-net and net versus net results, suppose we considered the performance of a hub-spoke network and a random network. In a net versus no-net model, the hub-spoke architecture is more vulnerable either in measures of subgraph size or diameter [2]. If the hub vertex is removed both subgraph size drops to zero and diameter goes to infinity. In a random network, there is a non-zero probability that the sub-graph size will still be greater than one after removal of the vertex with highest degree. For example, in a random graph with 4 vertices with 3 edges, if the topology is not hub-spoke (which occurs with probability $4/6^3$), then removing the vertex with highest degree results in a sub-graph size of two. Hence the expected subgraph size after the removal of the vertex with highest degree is

$$\mathrm{E}(\text{subgraph size}) = 2\left(1 - 4/6^3\right) \approx 1.96. \tag{9}$$

Hence the net versus no-net model considers the random network less vulnerable.

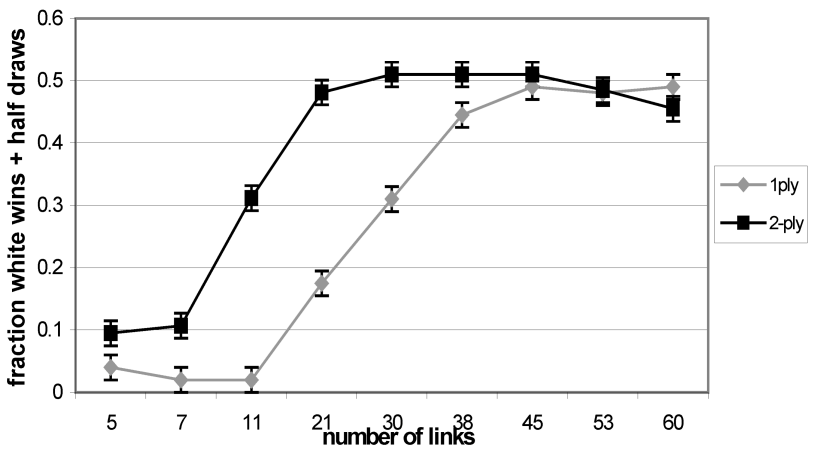

Fig. 3. Performance of a random against a hub-spoke topologies at different edge numbers and for search depth 1-2 ply. Our measure is the fraction of random net wins plus half draws

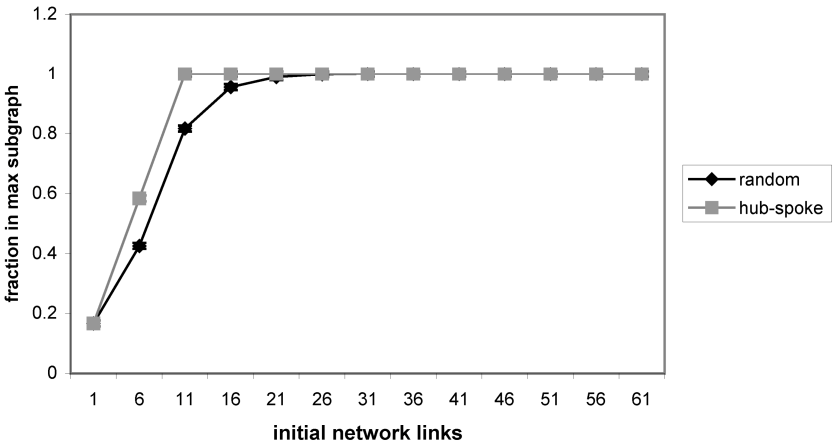

Fig. 4. Fraction of vertices in the largest subgraph, for random and hub-spoke topologies, assuming different edge budgets up to a maximum of 66

The figure on the previous page shows the performance of a random against a hub-spoke topology in the net versus net model. In contrast to the net versus no-net model, the hub-spoke topology shows higher performance than that of the random network, each with the same number of links. This is explained by considering the largest sub-graph size for the hub-spoke and the random networks given a fixed edge budget, prior to the removal of any vertices. Here, the hub-spoke network has the largest sub-graph size with the smallest number of edges. For a hub-spoke network, the addition of an edge always increases the sub-graph size by one if this sub-graph size is less than the number of vertices. This property is not assured in the random graph. The following simulation compares largest subgraph sizes in hub-spoke and random networks with 12 vertices at differing edge budgets.

The result that the hub-spoke topology performs better than the random topology shows that the dynamics of the interacting networks must taken into account. At the search depth of 2-ply, [8] the opponents response to a move is taken into account. Typically in observations of these games, the hub vertex is not placed in the position as one of the front pieces and is thus protected.

5 Learning the Value Function

We chose the value of 100 as our coefficient for the balance in sub-graph sizes arbitrarily. In general, we cannot assume that the coefficients for materiel balance and mobility will remain the same in the network game either. Finding the relative weights for these game features, termed advisors, can be done through reinforcement learning. Here, we briefly review the approach taken.

Given the board is in state s, we specify a action-value function $Q^{\pi}(s,a)$. This action-value function is the expected reward from winning the game given that the

policy π determines what action to take at each particular state. At a terminal state s_T we define the action-value function (reward) to be

$$Q^{\pi}(s_{T,}\bullet)=\begin{cases}1\text{ for a win,}\\1/2\text{ for a draw,}\\0\text{ for a loss.}\end{cases} \tag{10}$$

For a particular policy π, this action-value function $Q^{\pi}(s,a)$ is therefore the probability that a win is achieved, given we start from state s and adopt action a. Furthermore, in network checkers, as there are no intermediate rewards or discounting, the action-value function satisfies the Bellman equation [3] for optimal return

$$Q^{*}(s_t,a_t)=\mathrm{E}^{\pi^*}\left\{\max_a Q^{*}(s_{t+1},a_{t+1})\right\}, \tag{11}$$

where the expectation is taken over the optimal policy π^*. Indeed for the optimal policy π^* the expected difference between successive state action-value pairs will be zero [3],

$$\mathrm{E}^{\pi^*}(Q^{*}(s_t,a_t)-Q^{*}(s_{t+1},a_{t+1})\,|\,s_t,a_t)=0. \tag{12}$$

It is the observation of equation (12) that is the basis of the temporal-difference learning approach, in the sense that the difference between successive action-value functions should be close to zero according to Bellman's equation and learning is achieved by "shifting" the action value function $Q(s_t,a_t)$ towards $Q(s_{t+1},a_{t+1})$ [3].

In the network checkers game, we approximate the action-value function by some function $\widetilde{Q}:S\times\Re^n\to\Re$. We adopt the non-linear approximation of mapping the linear evaluation function onto the standard sigmoidal function, meaning that

$$\widetilde{Q}(s,a)=\left(1+\exp\left(-\beta V_n(s,\vec{\mathrm{w}})\right)\right)^{-1} \tag{13}$$

where $V_n(s,\vec{\mathrm{w}})$ is the principal value taken in a max-min search of depth n from state s and $\vec{\mathrm{w}}$ is the vector of weights (advisors) we wish to calculate [5]. Adopting the sigmoidal function has the advantage that for the set of states and actions, $S\times A(s)$,

$$\widetilde{Q}:S\times A(s)\to(0,1). \tag{14}$$

When the evaluation function is large, $\widetilde{Q}$ is close to one, indicating a win, large and negative, $\widetilde{Q}\approx 0$ indicating a loss. The constant β determines the sensitivity of $\widetilde{Q}$ to adjustments of weights in the evaluation function.

Weight changes are determined through gradient descent [10] and are done on-line, meaning that weights of the currently applied action-value function are changed

during the course of the game [3]. In particular, as a pair of states are visited during the game, an increment in the weight vector is given by

$$\Delta\vec{\mathrm{w}} = \alpha\left(\widetilde{\mathrm{Q}}(\mathrm{s}_{t+1}, a_{t+1}) - \widetilde{Q}(s_t, a_t)\right)\nabla_{\vec{\mathrm{w}}}\widetilde{Q}(s_t, a_t), \tag{15}$$

where α is the learning rate and ∇ is the gradient operator. Here,

$$\nabla_{\vec{\mathrm{w}}}\widetilde{Q}(s_t, a_t) = \beta\widetilde{Q}(1-\widetilde{Q})\nabla_{\vec{w}}V_n(s, \vec{\mathrm{w}}). \tag{16}$$

The learning rate is taken to satisfy stochastic convergence conditions given in [3]. To approximate the optimal policy, we determine the action for states s_t, s_{t+1} through an ε-greedy policy of choosing action $\arg\max_a \widetilde{Q}(s,a)$ with probability $(1-\varepsilon)$ and a random action with probability ε. This approach directs the policy towards the optimal one, as specified by Bellman's equation, whilst assuring the exploration of novel states through the random action [3].

5.1 Results of Evaluation Function Learning

We applied the learning method described, referred to as SARSA(0) [10], to the network checkers game, in order to explore values for the subgraph balance advisor. In applying TD(0), the opponent's advisor weights were fixed at values of 1.0, 2.5, 1,0, 0.01 for the unkinged, kinged, subgraph and rank balance weights respectively. The initial weights chosen for learning were 1.0, 1.0, 1.0 and 0.01. Min-max search was conducted at a depth of 3-ply and the value of β was 0.5. We ran the SARSA(0) algorithm over 2500 games.

Our initial trials showed that altering the rank balance advisor value is generally troublesome to learning the advisor values of the other three variables. As games progressed, the rank balance fervently swung from positive to negative values. In our current game formulation, kings have a rank of zero and we believe that this is the cause of the rank balance instability. A side that is close to victory may have either a positive rank balance (many unkinged pieces versus few unkinged pieces) or negative rank balance (many kinged pieces against a few unkinged pieces). For this reason, we chose to set the rank balance advisor weight at a constant value of 0.01.

First, we ran the SARSA(0) method when the network was fully connected, with 66 edges. Next we ran the algorithm with a random topology connected 20 edges.

Our results show that a fully connected network, after 2500 games, the subgraph advisor weight is slightly lower than that of kinged piece (advisor values 0.29 and 0.31 respectively). Not surprisingly, the kinged pieces have the highest value, approximately double that of an unkinged piece (value 0.15). As kings have the highest mobility, in checkers they are considered to have the highest value [8].

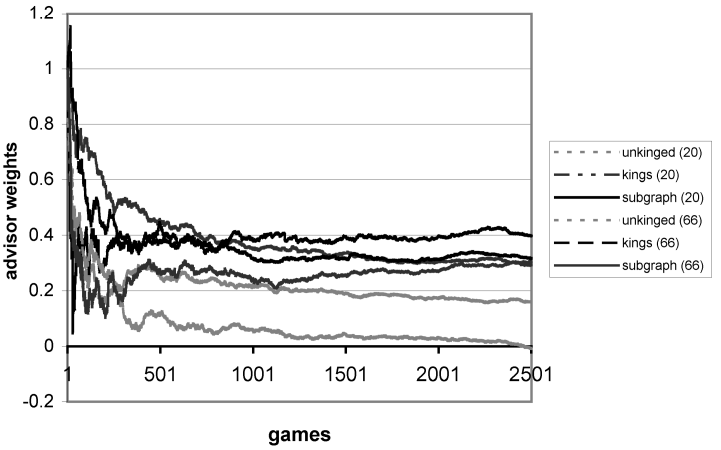

Fig. 5. Graph of the unkinged, kinged and subgraph advisor weights for 2500 games

With a sparse network, maintaining the subgraph size is crucial, as the loss of a single vertex with high degree may fragment the network into isolated small subgraphs or single vertices [2]. As we assume that only the largest subgraph exhibits mobility, a loss of a single piece may cause an incommensurate loss in mobility. The subgraph balance advisor weight reflects this, as it has the highest value in the learned advisor weights for a sparsely connected network (20 links, value 0.39).

At this stage of our research, the advisor weights are only taken as an indicator of the importance of unkinged, kinged and subgraph balance for networks either abundantly or sparsely connected. It is suggested that when attempting to find the exact optimal values for the advisor weights, one should apply the $SARSA(\lambda)$ algorithm for λ values close to one initially, then refine the values by choosing λ close to zero [4]. This is the next stage of our research.

6 Discussion and Conclusions

Our research goals are to understand the impact of element connectivity and communications in conflicts/games where two or more groups of agents cooperate with one another against an opposing group of elements. Several directions need to be taken for greater understanding of the role of networked gaming vertices. First, we assumed in our simulations that once an vertex is isolated from the main subgraph, it no longer exhibits any maneuver. This type of assumption may be termed *control by direction* in the sense that each vertex requires direction or guidance from peers to exhibit any mobility. Any truly autonomous act distinct from the main group is excluded.

The results presented in this paper may be contrast with agent models, where control is done by *veto*. Here, agents only recognize peers connected in their own subgraph and will act autonomously unless vetoed with other connected agents. Since control is by veto, this means that agents not connected will not recognize each other

as peers, but will still act autonomously. Agents of the same group may eliminate one another. With this new assumption, the same set of simulations may be run, for example, comparing outcomes with differing connectivity to agent ratios (vertex to edge) and comparing outcomes with centralised versus de-centralised connectivity.

In real-world situations agents such as humans naturally establish or cut links between other agents because of either environmental or adaptive reasons [12]. This means a dynamic network is formed, depending on the state of each agent. In this sense, network formation may be seen as a strategic decision along with the maneuver of agents. Such adaptive dynamic network games are worth modelling, however this generates a cost in the problem domain. Consider a game with an approximate movement branching factor of 8 (as is checkers) and 20 edges to be placed amongst 12 agents. The total branching factor will be the product of the movement branching factor and the number of ways to place 20 edges amongst 12 agents, $8\binom{20}{12} \approx 800000.$ This simple analysis implies that games, of dynamic network formation are of complexity vastly exceeding games such as go or backgammon.

In conclusion, we examined games in which mobility of vertices was determined by a network structure connecting vertices. The complexity of strategy was also dependent on the size of the largest subgraph connecting those vertices. Mobility was determined through the rules of checkers.

Through Monte-Carlo simulation, we examined game outcomes where one side had a differing edge to vertex ratio to the opponent. Larger edge to vertex ratios increased the probability of winning.

We also compared game outcomes opposing differing topologies with the same edge number on both sides. In this respect only the topological properties of the network were considered. We found through simulation that centralised networks with a hub vertex fare better against decentralized networks for two reasons. Firstly a centralized hub network connects the maximum number of vertices together with the minimum number of links. Second, once a hub vertex is protected from attack through manoeuvre, the network shows the greatest resilience to attack, as the loss of a vertex only decreases the subgraph size by one.

Finally, we conducted a series of machine learning experiments, implementing reinforcement/temporal difference methods in the play of the game. Through implementing gradient based $SARSA(0)$, we determined the values of the specific advisors for the importance of materiel and subgraph balance. For fully connected networks, the king balance advisor had the greatest value. With a sparse network, the subgraph balance advisor had the greatest value, due to the sensitivity of subgraph size as vertices as lost.

References

[1] Albert, R. and Barabasi, A. L.: The statistical mechanics of complex networks. Reviews of Modern Physics. Vol. 47, (2002), 47-94.

[2] Albert, R. *et al.*: Error and attack tolerance of complex networks. Nature. Vol. 206 (2000), 378-381.

[3] Barto, A.G. and Sutton R. S.: Reinforcement Learning: An Introduction. MIT Press, (1998).
[4] Baxter, J. *et. al.:* Experiments in Parameter Learning Using Temporal Differences. ICCA Journal, Vol. 211(2), (1998), 84-99.
[5] Beal, D. F. and Smith, M.C.: Learning Piece Values Using Temporal Differences. ICCA Journal, Vol. 20(3), (1997), 147-151.
[6] Holme, P. *et. al.:* Attack vulnerability of complex networks. Physical Review E. Vol. 65, (2002), Section on Complex Networks.
[7] Huang, V. and Huh, S.H. `http://www.cs.caltech.edu/~vhuang/cs20/c/applet/s11.html`.
[8] Levy, D.: Computer Gamesmanship: Elements of Intelligent Game Design. Simon and Schuster, New York, (1984).
[9] Lynch, N.: Distributed Algorithms. Morgan Kaufmann, 1997.
[10] Sutton, R. S.: Learning to Predict by the Methods of Temporal Differences. Machine Learning. Vol. 3, (1988), 9-44.
[11] Washburn, A. and Wood, K.: Two Person Zero Sum Games for Network Interdiction. Operations Research, Vol. 43(2), (1995), 243-251.
[12] Watts, A:. A Dynamic Model of Network Formation. Games and Economic Behaviour. Vol.34, (2001), 331-341.