AN EVOLUTIONARY AGENT APPROACH TO DOTS-AND-BOXES

Terry Bossomaier
School of Information Technology,
Charles Sturt University and
Visiting Fellow, Centre for the Mind
email: tbossomaier@csu.edu.au

Anthony Knittel
Centre for the Mind
Main Quadrangle (A14)
University of Sydney
NSW 2006
Australia
Phone: +61 2 9351 5816
Fax: +61 2 9351 8534
email: anthony@centreforthemind.com

ABSTRACT
We describe the evolution of intelligent agent teams for the combinatorial difficult game of Dots and Boxes. Agents operate within an artificial economy deriving reward only from won games, with no other kinds of feedback. By providing representations of component graph structures, upon which agents can evolve rules, the agent teams achieve significant performance against existing AI players based around established mathematical frameworks.

KEY WORDS
evolutionary economic multi-agent systems dots boxes

1 Introduction

The evolution of teams of intelligent agents has widespread applications in business, in the management of web services and in all kinds of game playing scenarios. Ideally such agent teams need to evolve or learn with very minimal feedback and to avoid structured training in what is generally referred to as reinforcement learning [1]. As such they are easily transferred to other application areas as required.

In building such agent teams there are two prime considerations. The first is how the agents themselves represent their actions. They may be rule based, from simple predicates to fuzzy logic, or they may use connectionist representations. Reinforcement learning has employed a variety of such representations, such as Tesauro's world class backgammnon program using neural networks in the TDλ framework [2].

The second consideration is how the teams are themselves constructed. One approach is through formal methods of construction [3], in which communication protocols and other polices are defined upfront. The alternative is to allow evolution to determine team structure.

As with human societies and natural ecologies, the Tragedy of the Commons easily results in the development of agent teams. Social systems will consume a resource to total depletion because any given member will lose out if they do not exploit a free common resource. The end outcome is often total resource depletion, such as the disappearance of cod from waters around the United Kingdom and the global decline in world fisheries.

In other words, the optimisation by an agent of its own performance may impact on global performance. Of the several ways to avoid this problem, three powerful strategies are:

1. Wolpert's COIN framework, in which each agent optimises its own gains subject to constraints that any actions it takes should not impede the global gain. [4]

2. Boltzmann learning of agent rules in a reinforcement paradigm.

3. Baum's artificial economies in which various fiscal constraints are imposed.

In Harré and Bossomaier [5] we compare the performance of Boltzmann and Baum on the game of Dots and Boxes, using no prior knowledge or structuring of the game space. In the present paper we develop further the artificial economy model, but inject some structural information about the game space. Our overarching goal in a forthcoming paper, is to blend these two approaches in which agent populations learn *both* improved representations *and* rules of action based on the augmented set of representations.

The use of structural descriptors falls into the domain of "coarse graining" [6] of the game or problem space. This idea, recently demonstrated for 1 dimensional cellular automata, has profound implications for machine learning. In essence a problem which appears chaotic or otherwise intractable at one level, may become qualitatively different at an aggregated level.

2 Artificial Economies

In the artificial economy (AE) model, agents bid for ownership of a world, which represents some problem domain. When they make the highest bid and take up ownership, they may then apply actions to the world, increasing its wealth, and subsequently on-sell it to other agents. This model, introduced by Baum and Durdanovic [7], turned

out to be a powerful method of reinforcement learning, achieving better results on Blocks Worlds problems and some progress on extremely hard problems such as Rubik's Cube.

The Blocks World problem is a difficult one for AI, but it can be solved with a relatively small number of agents. Even so, in the course of learning the number of agents grows significantly. It is thus interesting to know how AEs scale to problems of increased complexity, where the number of agents required might be large and the number of steps to a solution also much larger.

Games provide a good testbed since they range from static strategies to the most complex scenarios where strategy depends on other agents. The game of Dots and Boxes (section 3) is an example of very difficult [8] static strategy. At any given point in the game, the best move is independent of the opponent and independent of previous game history. However, as we demonstrate herein, it has the potential for coarse graining – building new concepts, essentially special patterns, allowing for hierarchical collaborative agent behaviour.

The AE now proceeds along the following lines for a given game.

1. Seeding with some initial agents (or carry over of agents from a previous game)
2. Allowing agents to bid for and operate on the world if the bid is successful. In this case operation on the world consists of adding an edge somewhere. After adding an edge, the agent puts the world up again for auction. The next agent move may occur immediately if the current agent has completed a box, or it may occur after the opponent has moved
3. An agent which has won the auction pays the current owner and takes control
4. Agents update their economic state, and if sufficiently wealthy spawn new agents with mutation, as described in section 4.2
5. At the end of the game, a new game is started, with the current agent team carrying on to the next round.

2.1 Production Rules

The agents need some way of making decisions about the move they would make. They do this via Production Rules as follows. Each rule has a pattern which matches some area of the game space and an operation that can be carried out if the pattern matches (e.g. a move is made at some specified point). Both the patterns and the place to move are under evolutionary control.

3 Dots and Boxes

The game of Dots and Boxes is surprisingly difficult, although often played by children. A rectangular grid of dots is drawn and players take it in turns to join up adjacent dots on the grid, ultimately completing a grid of squares (boxes).

The possible patterns which can be created are numerous. For a 3x3 set of boxes (24 edges), there are 2^{24} or 16 million patterns which can arise! Yet the rectangular symmetry means that many of these patterns are effectively the same from a strategical perspective. Furthermore, common structures arise, which can assume various positons and shapes on the grid, yet require the same strategy. For this reason we adopt an alternative network representation, which we describe next in section 3.1.

3.1 The Graph Dual

If we now imagine replacing each square of dots by a token and connecting it to adjacent tokens where there is an empty edge (two dots but no line) between the two squares we create a square lattice of connected tokens. Adding an edge on the board corresponds to cutting a link between two tokens.

It is immediately clear how this simplifies the representation of the game, since the lattice rapidly becomes a collection of branching chains. The strategies for these substructures may be quite difficult to learn without any direct feedback. Figure 1 shows a *dipper with earmuffs* [9]. Any move at this point will concede tokens to the opponent. However, breaking any of the chain links, denoted by **l2** will concede three tokens. Breaking either ring (**l0** and **l4**) will concede a lot more.

Thus we conjecture that a graphical representation of this kind will allow more "high-level" rules to be discovered and lead to advanced play.

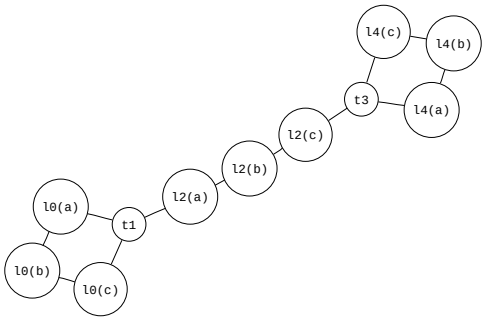

Figure 1. The dipper with earmuffs

3.2 Reduction of Computational Complexity

The problem of matching a subgraph (from a rule) to the game is non-trivial. If there are N tokens in the game and p tokens in the subgraph, then in the worst case there are $\frac{N!}{p!}$ possible ways of choosing a set of tokens to determine if the edge connectivity matches.

To reduce the costs, we label each node in terms of its connectivity (quarternary (q), tertiary (t), chain link (l) or box (b) with one side open) and then order the connections, including in addition the edges of the grid (n). Thus a node has a descriptor such as (q:qtlb). The process of matching rule graphs and game graphs is described as follows.

4 Implementation

Complete game states in dots and boxes are captured using a set of graphs, where each node represents a square on the board and edges connecting these nodes are unfilled lines on the board. The symbols used in this representation are based on the degree of each node, with additional symbols to capture features such as properties of chains. The representation used for rule graphs is based on the game graph representations, with additional freedoms for defining properties of chains. Using this language the agent system understands *a priori* that a series of nodes of degree 2 is a feature worth being aware of, and evolution of strategies follows from this principle.

Each rule carried by an agent is simply a graph where each vertex is a symbol from the set $\{x, l, t, q, n, w\}$, and an action, defined as an edge between two nodes in the graph. Each symbol represents nodes with 1-4 connections respectively, and 'null nodes', used to represent unfilled lines at the edge of the board. Wildcard symbols w are used to allow a rule to match as a subgraph of a larger graph. A wildcard symbol can match any class of vertex in the game state graph without the requirement of subsequent symbols needing to be specified. Chain symbols are also given a property representing the length of the chain, and a flag to indicate the chain should match all chains longer than its specified length. Where actions refer to chains (l-symbols), an additional value is given to specify the position along the chain. Symbols are numbered to allow referencing. Textual representations of these rules can be produced using the format: [symbol][neighbours] : [symbol][neighbours] => [action symbol] - [action symbol]

For example $x0n => x0 - n$ defines a rule to complete a single box connected to an edge, and the following rule fills the line one distance along the chain l2, from the end connected to t1:

$$l0t1t1 : t1l0l0l2 : t3l2l4l4 : l4t3t3 => l2[1] - t1$$

This language is not sufficient to capture the more sophisticated strategies described by Berlekamp [8], however should be sufficient to produce a recognisable strategy and allow fairly effective play.

4.1 Matching of Subgraphs in Rule Testing

Rules are recorded as a graph structure, using an adjacency table to record links between objects. A graph is produced to represent each board state, and each rule matches a board state if there is a substitution for every node in the rule graph for every node in the state graph, such that all the associations between nodes in the rule graph are preserved. This testing process allows a rule to match a game state if the rule graph is a subset of the state graph, allowing features in the state graph to be captured without requiring the entire state to be defined.

In order to test the substitution of nodes between the two graphs a translation map is produced for every possible combination of nodes of the same degree. At worst case this is an expensive operation on large games, however this approach is sufficient for the game states found on small games. To further limit computational problems, a random sampling procedure is used when a graph substition is being assessed that has a large number of possible substitutions, and the graph is determined to not match if no match is found under the fixed number of samples taken.

4.2 Mutating Rules

New rules are initially generated by copying the world state and selecting a random action, or by randomly generating a rule graph. Subsequent variations to rules are generated by mutations. During the mutation process nodes are added or removed from the graph at random, and links between existing nodes are created or removed. If a chain node is chosen for mutation, the length and multiplicity properties of the chain are also randomly altered. Repeated mutations occur with a given probability (0.5), the probability of the various mutations chosen is evenly distributed.

New graphs resulting from mutations are assessed for validity, and invalid mutations are revoked. Every node in the rule graph must be connected to the correct number of other nodes according to the designated degree of the node, and nodes must not be connected to themselves. To allow a rule to match as a subgraph of a larger state graph, wildcard symbols are necessary. For example, the rule $x0w1 => x0 - w1$ can match the state graph $x0t2 : t2x0x3x4 : x3t2 : x4t2$, where the wildcard $w1$ matches the symbol $t2$. This allows the symbol $x0$ to be connected to the correct number of partner nodes, 1, satisfying the requirements for valid rules.

4.3 Auction Model

The agent player needs to determine a move for each game state it encounters. The auctioning process follows the method described by Baum [10], each agent is presented the world state and offers a bid if it matches. The agent with the highest bid pays its wealth to the previous owner and performs its action on the world. It subsequently sells its ownership of the world to the next agent, and if the agent has placed the state of the world in a state valued higher by another agent it collects a profit and is able to be maintained.

Determination of bid values is done using a fixed value which remains a constant property of the agent. New

agents entering the system must bid higher than any existing bids in order to be relevant, and the bid prices stabilise at the highest sustainable level. The original motivation behind the bid values is an analogy with the price of goods in an economy, that agents need to be able to offer a high enough price for goods to be able to acquire them, and be efficient enough to be able to make a profit according to the price they sell the goods on at. The bid value can also be considered in practical terms as a prioritisation, if there are existing agents with a given priority any new agents must enter with a higher priority in order to be used by the system. The set of bid values in a collection of agents indicates the priority of rules that define the policy of the rule set, where each rule is bid according to the agent it is held by.

During play, the game state is presented to the player and it selects the highest bidding agent to act on the given game state. If there are no agents with matching rules a random move is played. This is performed independently of the agent system, so the next player to offer to move will pay the last agent that moved as it remains the owner. Transfer of ownership of the world is used to avoid the principle of Tragedy of the Commons, as described in section 1.

4.4 Simulation Framework

The software platform used has been developed using C++ in a gnu environment. Reusable libraries have been developed for the underlying agent framework independent of the implementation of the dots and boxes game. The simulations are run on a Macintosh Quad G5 desktop computer.

The agent system is evolved by playing games against an artificial player based on fixed rules. The artificial player used is Nonie's Dots and Boxes [11], and was selected as it is easy to understand, runs fast and shows understanding of strategies such as leaving the last two boxes of a long chain to force the other player to lead.

Training is done by starting the agent system in a relatively simple environment and increasing the difficulty as the agent progresses. Two independent variables are used to set the difficulty, firstly the actions of the artificial player are mixed with noise such that according to a given probability the player performs a random action instead of a policy directed action. Secondly games are played that are mostly completed so the agent can learn to play the simpler end game before approaching larger, more open game states. This is done by filling in random edges in new games, such that a specific ratio of all possible edges is already filled in when the game begins. When producing initially complete games, selection of edges is done avoiding producing completed or nearly completed boxes, by filling edges only if the boxes neighbouring an edge will have at most two edges completed.

The difficulty of the artificial player is set according to the performance of the agent system. Initially the agent system plays against a completely random player, and as it develops a policy that is better than random play the

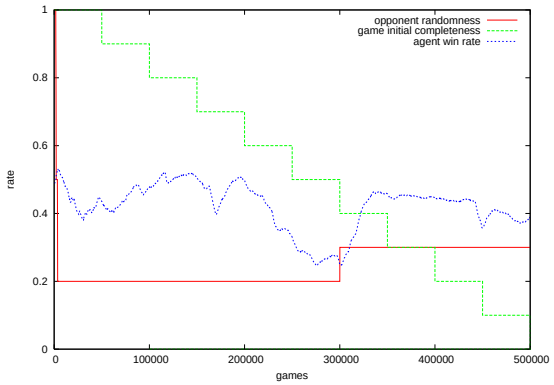

Figure 2. Performance of the agent system against the artificial player during the initial training period

percentage of random moves is decreased. The win rate threshold used is 0.7, when the agent starts winning more than 70% of games the difficulty is increased, and if the performance drops below 0.2 the difficulty is reduced. This is a similar threshold to that used in [12]

The initial completeness of games is controlled as an independent parameter according to a fixed schedule. An experiment is run over a fixed number of games, and over the course of the run the percentage of initial completeness of the game is reduced at an even rate so that early games are played only on end games and later games play the whole game.

At the end of the training schedule a number of games are played using an unrestricted artificial player and starting with an empty board, to assess the overall performance of the agent player against its opponent. The agent system is left to play a number of games against the AI to try and improve performance against the full strength opponent.

5 Results and Discussion

The agent player was able to develop a set of rules that were able to play effectively against the artificial player with a random play restriction as low as 20%. The agent player quickly developed strategies to beat the largely random player, causing the random factor to be reduced to 20% after 4,000 games. The agent player was able to maintain a competitive win rate until 200,000 games, however as the initial completeness of the game world was reduced the performance dropped, until the randomness of the agent player was increased to 30% and the agent player was able to maintain competitiveness again (figure 2).

The performance of the agent player was maintained close to 40% for the rest of the run, with a slight decrease in performance as the initial completeness of the game world was reduced further. The number of random moves performed by the agent player, as a result of not having any rules which match a given board state, quickly reached a

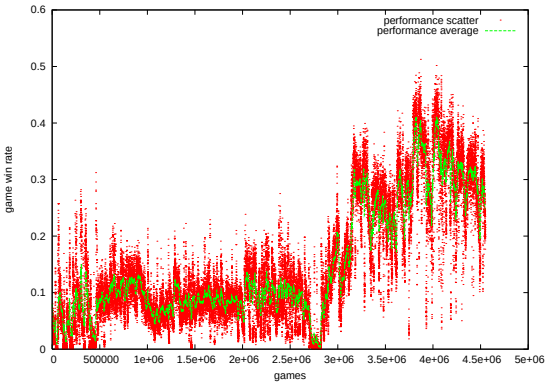

Figure 3. Performance of the agent system against the unrestricted artificial player

small level of 1% after 1500 games, and typically maintained a level below 2% throughout the run.

After the initial training period was completed, the agent system continued to play games starting from an empty board, against an opponent acting without any random move restrictions. The perfomance of the agent system is shown in figure 3, in terms of games won.

In previous runs the agent system was able to maintain a win rate of 12% of games directly after completing the training schedule, however in the experiment described here the agent system was able to win 5% of games against the unrestricted opponent directly after the training schedule.

Performance was fairly erratic as the agent system was left to develop against the full opponent, reaching peaks up to 20% before returning to low performance levels. This was likely due to collections of rules being developed where the removal or restricted activity of some rules led to the overall policy to break down, due to agents having a small amount of wealth and not having other similar agents to fall back on when an agent is removed.

After 500,000 games the performance was much more stable, maintaining a level around 10% with peaks up to 20%. After 3,000,000 games the performance began to improve in fairly distinct steps as new policies were developed. The best was able to maintain a performance of about 40% over a period of approximately 100,000 games, with peaks above 50% for short periods, showing the system had developed a strategy capable of competing with the given engineered opponent.

The current use of wildcard symbols allows rules to match as subgraphs of larger game state graphs. This allows the use of generally applicable rules that can operate in a wide variety of game states, however may lead to general rules dominating when rules specific to particular situations may be more applicable.

The training runs described were performed over a period of less than an hour, indicating there is room to increase the complexity of the system to handle more sophisticated tasks. This could also open the possibility of extending the language used in generating the rule states to allow more complex rules to be produced.

6 Conclusion

We developed a coarse-grained representation of the game of dots and boxes in a network dual. This representation enabled a surprising level of performance to be obtained by agents teams. The agents interoperate through an auction system, receiving rewards only when they win the auction and are allowed to make a move. Thus the behavioural rules emerge with no explicit description or training. The challenge now is to enable the representation to itself coevolve with the agent behaviour, perhaps through agents which “rewrite” the game space in progressively higher level conceptual structures.

Acknowledgements

Anthony Knittel was funded by Australian Research Council Discovery Grant DP0560207 Bossomaier et al.

References

[1] R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.*, 6(2):215–219, 1994.

[3] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, Vol. 10 Number 2, 1995.

[4] D. Wolpert and K. Tumer. An introduction to collective intelligence. *http://www.arxiv.org/pdf/cs.LG/9908014*, 1999.

[5] M. Harré and T. Bossomaier. Thinking outside the box: A multi-agent approach to complex games. In *Proceedings of IASTED International Conference on Artificial Intelligence and Soft Computing 2006*. IASTED, 2006.

[6] N. Israeli and N. Goldenfeld. Computational Irreducibility and the Predictability of Complex Physical Systems. *Physical Review Letters*, 92(7):074105–+, February 2004.

[7] E. Baum and I. Durdanovic. An evolutionary post production system. citeseer.ist.psu.edu/baum00evolutionary.html, 2000.

[8] E. R. Berlekamp. *The Dots-and-Boxes Game: Sophisticated Child’s Play*. AK Peters, Ltd., 2000.

[9] K. Scott and E. Berlekamp. Forcing your opponent to stay in control of a loony dots-and-boxes endgame. *More Games of No Chance*, 42:317–330, 2002.

[10] E. Baum and I. Durdanovic. Evolution of cooperative problem-solving in an artificial economy. *Neural Computation*, Vol. 12:2743 – 2775, 2000.

[11] Arnon Politi. Nonie's dots and boxes, http://dsl.ee.unsw.edu.au/dsl-cdrom/unsw/projects/dots. (Resource of artificial player used as training opponent).

[12] E. Baum and I. Durdanovic. An artificial economy of post production systems. *Lecture Notes in Computer Science*, 1996:3, 2001.