2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

*Tego - a framework for adversarial planning

Daniel Ashlock
Department of Mathematics and Statistics
University of Guelph
Email: dashlock@uoguelph.ca

Philip Hingston
School of Computer and Security Science
Edith Cowan University
Email: p.hingston@ecu.edu.au

Abstract—**This study establishes a framework called *-Tego for a situation in which two agents are each given a set of players for a competitive game. Each agent places their players in an order. Players on each side at the same position in the order play one another, with the agent's score being the sum of their player's scores. The planning agents are permitted to simultaneous reorder their players in each of several stages. The reordering is termed *competitive replanning*. The resulting framework is scalable by changing the number of players and the complexity of the replanning process. The framework is demonstrated using iterated prisoner's dilemma on a set of twenty players. The system is first tested with one agent unable to change the order of its players, yielding an optimization problem. The system is then tested in a competitive co-evolution of planning agents. The optimization form of the system makes globally sensible assignments of players. The co-evolutionary version concentrates on matching particular high-payoff pairs of players with the agents repeatedly reversing one another's assignments, with the majority of players with smaller payoffs at risk are largely ignored.**

I. Introduction

In a number of gaming situations, a commander must deploy units in opposition to one another. The simplest version of this requires a 1:1 matching of the units. This can be represented by a specification of the available units and the order in which those are placed in conflict. When agents can adjust their player deployments competitively during the course of a conflict, the agents are said to be engaged in *competitive replanning*. This study presents a general framework for competitive replanning and provides and example of matching of units that play the iterated prisoner's dilemma.

The work presented generalizes the competitive replanning presented in [10]. competitive replanning is, itself, a generalization of *red teaming* [15]. Red teaming is a technique for highlighting vulnerabilities in systems or structures. Two 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. Competitive replanning removes the attacker/defender asymmetry to permit the treatment of a broader variety of situations.

The remainder of this study is structured as follows. Section II gives the formal definition of the general framework. Section IIIspecifies the specific problem use to test and demonstrate the framework. Section IV gives the experimental design while section V gives and discusses results. Section VI outlines possible future directions for the work.

The authors thank Edith Cowan University, the University of Guelph, and the Canadian National Science and Engineering Research Council of Canada (NSERC) for supporting this work.

II. Framework

*-Tego is a framework for a family of scalable adversarial planning problems, inspired by the strategy board game Stratego. Each instance defines the following parameters:

- A base game, G, which is a simultaneous game.
- A finite set of players, $P_i : i = 1..n$.
- An integer t, which is the number of stages in the instance.
- An integer s which is the number of player changes allowed between stages.

We might call a particular instance G(n, s, t)-Tego, or simply G-Tego when the other parameter choices are clear. An encounter between agents competing at G-Tego proceeds as follows:

1) Initialization:
   a) Each agent is randomly assigned a permutation π_1 of $1..n$. This determines a sequential ordering of the players $S_i : i = 1..n$.
   b) The base game G is played out between corresponding players S_{π_1}.
2) Stage:
   a) Each agent may swap $\leq s$ pairs of players. Agents make their choices of swaps simultaneously. Each agent is informed after the fact what swaps the other agent(s) made.
   b) The base game G is played between corresponding players with the new ordering.
3) t stages are played out.
4) The overall payoff for each agent is the sum of the payoffs over all the base games, over all the stages.

The initialization stage is not counted in the payoffs unless full information on the payoff matrix is given to both agents – its function is to provide each agent with some information about the sequence of players being played by the other agent(s). On the basis of this information, each agent can then attempt to obtain a better match-up of players, while recognizing that the other agent(s) will be trying to do the same, so as to maximize their overall payoff.

Each stage provides more information about the players being played by the other agents. As a special case, if the behaviours in the stage games should ever uniquely identify the players of the other agent(s), then an optimal sequence of swaps for the remaining stages can be calculated (in the sense of a Nash equilibrium).

The intention of this design is to capture the following features of an adversarial planning problem:

- A number of sides are competing.
- The competing sides find themselves in some situation, about which they have partial information (*initialization*).
- On the basis of this information, each side constructs a plan to deploy its resources, trying to anticipate the actions of the other side, so as to obtain the best payoff in the long term.
- As the scenario plays out, more information is gathered, and there is the opportunity to replan - i.e. to redeploy resources (*stages*).
- The current plan restricts the possible choices for deployment of resources for the next stage (*restriction on the number of swaps*), and each side can take advantage of this limitation to its opponents plans.

By varying the properties of the game, the players (both the number and nature of them), the number of stages and the number of swaps allowed, it is possible to control:

- the size of the strategy spaces for the agents (*the complexity of the planning scenario*,
- the amount of certainty (*fog of war*),
- how changeable plans can be (*the degree to which current choices restrict future choices*).

III. Computational Example

This section defines the game used to demonstrate *-Tego. Since the underlying game is prisoner's dilemma, we are playing PD-Tego and in fact will be using PD(20, 50,5)-Tego for demonstration purposes. This means that two agents are allowed up to fifty swaps of the order of their twenty players in each of five planning stages. At the end of each collection of swaps (planning stage) the score for playing 50 rounds of iterated prisoner's dilemma between each pair of matched players is totaled. The agent's fitness is the average score per play over all five planning stages, twenty opponents, and fifty rounds of play. The payoff matrix used is shown in Figure 1 so fitness values will be in the range 0-5.

The prisoner's dilemma [9], [8] is a classic model in game theory. Two agents each decide, without communication, whether to cooperate (C) or defect (D). The agents receive individual payoffs depending on the actions taken. The payoffs used in this study are shown in Figure 1. The payoff for mutual cooperation C is the *cooperation* payoff. The payoff for mutual defection D is the *defection* payoff. The two asymmetric action payoffs S and T, are the *sucker* and *temptation* payoffs, respectively. In order for a two-player simultaneous game to be considered prisoner's dilemma, it must obey the following pair of inequalities:

$$S \leq D \leq C \leq T \tag{1}$$

and

$$2C \geq (S + T). \tag{2}$$

In the *iterated prisoner's dilemma* (IPD) the agents play many rounds of the prisoner's dilemma. IPD is widely used to model emergent cooperative behaviors in populations of selfishly acting agents and is often used to model systems in biology [20], sociology [16], psychology [19], and economics [14].

$$\begin{array}{cc|cc} & & \mathcal{S} & \\ & & C & D \\ \hline \mathcal{P} & C & 3 & 5 \\ & D & 0 & 1 \\ \multicolumn{4}{c}{(1)} \end{array} \qquad \begin{array}{cc|cc} & & \mathcal{S} & \\ & & C & D \\ \hline \mathcal{P} & C & C & T \\ & D & S & D \\ \multicolumn{4}{c}{(2)} \end{array}$$

Fig. 1. (1)The payoff matrix for prisoner's dilemma used in this study – scores are earned by strategy $\mathcal{S}$ based on its actions and those of its opponent $\mathcal{P}$ (2) A payoff matrix of the general two player game – C, T, S, and D are the scores awarded.

A number of terms need to be defined to specify the player strategies used to generate the optimization and co-evolution targets in this study. The game will be played with a collection of $n = 20$ players, each of which is a distinct prisoner's dilemma strategy. We begin by defining the novel types of strategies used as players in this study. The strategies are represented by finite state machines using the Mealy architecture [7], [3], [4], [2].

Definition 1: A **sugar** strategy is defined in terms of a final strategy $\mathcal{S}$ and a **password** from $\{c|d\}^*$. If an opponent executes the sequence of prisoner's dilemma moves in the password then the sugar strategy plays according to the exploitable final thereafter. Any deviation from the password returns the sugar strategy to its initial state. The final strategies of a sugar strategy are chosen to be one that will cooperate or permit themselves to be exploited.

A treasure hunt strategy has a few initial transitions that choose a communicating class of the FSM defining the player's "treasures". An interesting treasure hunt will have treasures that are equivalent to agents with different margins of exploitation.

Definition 2: A **treasure hunt** strategy has an initial set of states structured as a tree. At each leaf of the tree is a strategy that, once the leaf is reached, the agent will play thereafter. Under the convention that the left branch is labeled by "c" while the right branch is labeled with "d", a treasure hunt strategy is specified by a delimited binary tree. All responses of a treasure hunt strategy before it enters a leaf are "c". This makes it impossible to tell the difference between treasure hunt strategies until transitions reach a leaf. The notation for this type of strategy is $(TFT, (ALLD, ALLC))$.

Notice that, in a competitive environment, sugar strategies with exploitable final strategies are quite unlikely to arise via evolution. Treasure hunt strategies might be competitive, and so could arise via evolution, but are unlikely to do so because their intrinsic connectivity is very unlikely. To see this notice the number of pairs of states within the same leaf strategy as opposed to the number of pairs in which both are not in the same leaf strategy. The first type of pair have a size proportional to the sum of the squares of the numbers of states in the leaf strategies while the second are proportional to the square of the number of states in the machine as a whole. This means that a majority of possible mutations must not occur to preserve the treasure hunt strategy. In a tree with multiple treasures a majority of mutations that affect transitions disrupt the structure of the player to something outside of the definition of a treasure hunt.

We now list the 20 player strategies that form the collection of players available to the PD-Tego agents being evolved. Some of the strategies are well known, others are sugar or treasure hunt strategies defined above.

1) Always cooperate (ALLC). This strategy always cooperates.
2) Always defect (ALLD). This strategy always defects.
3) Tit-for-tat (TFT). This strategy cooperates initially and then repeats its opponent's last action thereafter.
4) Tit-for-two-tats (TF2T). This strategy cooperates unless its opponent's last two actions are defects.
5) Two-tits-for-tat (2TFT). This strategy only cooperates if its opponents last two actions are both cooperates.
6) Pavlov (PAV). This strategy cooperates initially and, thereafter, cooperates if its action and its opponent's action on the previous round of play are the same.
7) Fortress 3 (Fort3). This strategy defects until its opponent defects twice in a row. The strategy then cooperates and continues to do so if the opponent does so as well. Any deviation from cooperation resets the machine to its initial state.
8) Fortress 4 (Fort 4). This strategy is like Fortress 3 except it defects three times in a row instead of two.
9) Fortress 5 (Fort 5). This strategy is like Fortress 3 except it defects four times in a row instead of two.
10) Sugar, password DDD, strategy ALLC.
11) Sugar, password CDC, strategy ALLC.
12) Sugar, password DDD, strategy TF2T.
13) Sugar, password CDC, strategy TF2T.
14) Sugar, password DDD, strategy TFT.
15) Sugar, password CDC, strategy TFT.
16) Treasure hunt (TFT,(ALLD,ALLC)).
17) Treasure hunt ((TF2T,2TFT)ALLC).
18) Treasure hunt ((TF2T,2TFT),ALLD).
19) Treasure hunt ((ALLD,ALLC),(TF2T,2TFT)).
20) Treasure hunt (Fort3,PAV).

The situation treated is thus co-evolutionary optimization of the order in which to place the 20 player strategies. Each agent is working on their own order while aware of the opponent's order.

IV. Experimental Design

A number of choices must be made in setting up an instance of *-Tego. In the initial demonstration on PD-Tego we chose, somewhat arbitrarily, use 50 swaps and 5 planning stages yielding PD(20,50,5)-Tego. In addition to the co-evolutionary evolution we performed a baseline study in which the opponent's order was held fixed. This is intended to verify that the agent representation can solve the problem in the relatively clean environment of an optimization problem. It also serves as a baseline study that establishes an upper bound on an agent's performance in planning against an opponent that is helpless (that cannot plan).

```
States:8.
Start:C->4
   If C |If D
--------------
 0)D-> 0|D-> 6
 1)C-> 5|C-> 1
 2)D-> 3|C-> 0
 3)D-> 6|C-> 1
 4)C-> 4|D-> 3
 5)D-> 3|C-> 2
 6)C-> 6|D-> 2
 7)C-> 0|C-> 2
--------------
```

Fig. 2. An example of an evolved finite state prisoner's dilemma player.

```
  0) if(Swap(9,7)>8.06815)go 27 else go 63
  1) if(Swap(17,19)>8.07613)go 9 else go 23
  2) if(Swap(15,18)>20.1899)go 33 else go 75
  3) if(Swap(12,5)>0.903448)go 11 else go 71
  4) if(Swap(10,8)>3.21103)go 122 else go 113
  5) if(Swap(8,2)>18.1873)go 125 else go 105
  6) if(Swap(7,6)>2.01642)go 109 else go 105
  7) if(Swap(4,13)>13.9161)go 27 else go 25
  8) if(Swap(19,18)>6.74807)go 12 else go 126
  9) if(Swap(19,15)>17.939)go 103 else go 49
 10) if(Swap(11,1)>-10.0888)go 29 else go 32
 11) if(Swap(19,11)>5.39423)go 30 else go 108
...
126) if(Swap(8,16)>3.82702)go 108 else go 117
127) if(Swap(6,3)>-12.367)go 120 else go 91
```

Fig. 3. Shown is a partial listing of an evolved BDA.

A. Agent Representation

The agents used in this study are a type of augmented finite state machine called a binary decision automata (BDA). The transitions of the automata are driven by Boolean variables. A partial example is shown in Figure 3. In the past BDAs have been used as agents in a model of stress in the workplace [18] and as player agents for a simple card game [5]. BDAs use the Mealy architecture that associates actions with transitions. BDAs are capable of dealing with a large number of different possible inputs via the abstraction of those inputs into Boolean form and, because of their finite state structure, remembering simulation context in the form of state information. They are a relatively simple agent representation given their high degree of generality. BDAs are a simplified form of *GP-automata* [1].

In this study the possible actions of the BDAs are to swap the positions two of the twenty players in the agent's lineup. Both agents have the full payoff matrix for 50 plays of IPD between any two of the twenty players available. The $Swap(x, y)$ primitive returns the amount that exchanging players x and y would change the agent's score against its opponents current lineup of players. If it exceeds the constant in the if clause, making the Boolean value true, then those player's orders are exchanged. In addition, there are are two transitions. The first is followed if the Boolean value is true, the second is followed in the Boolean value is false. When initial populations of agents are generated, the values of the two players to consider swapping and the threshold value are generated at random. The constant for comparison is chosen uniformly at random in the rage [-25,50], a reasonable range given the values in the payoff matrix.

The agents used have 128 states, a number chosen by preliminary experimentation. This number must be large because each state can only swap one pair of agents. There are $\binom{20}{2} = 190$ possible pairs of players that can be swapped. Group theory [17] tells us that a minimum of nineteen available pairwise swaps are available to achieve any permutation of the agents, but these individual swaps may need to be reused several times. Using 128 states gives the machine many pairs of exchanges it can use.

B. Analysis Techniques

The best outcome of a *-Tego simulation is always an assignment of one agent's players to the other agent's players. A *permutation scattergram* is a way of displaying the ensemble assignment across many experiments. The scattergram is a matrix with rows indexed by the first agent's players and column's indexed by the second agent's players. The entries of the matrix are disks whose size is proportional to the number of times the player indexing the row was assigned to the player indexing the column. The radii of the disks are proportion to the natural log of the number of times that one player was matched against another.

The largest disk in a row indicates the most popular choice. In the baseline experiments, a large disk indicates that having the player indexing the row play the player indexing the column is a good choice. Having an player play itself, a fixed point of the permutation and a diagonal entry of the matrix, gives neither agent an advantage in the co-evolutionary version of *-Tego. In the baseline optimization problem, it shows that the player in question plays better against itself than anything any other available player. Evolved players often have passwords like those used in the sugar strategies. This recognition-of-self property makes assignments of players of this type to themselves a natural choice.

C. Evolutionary Algorithm Specification

Two very similar evolutionary algorithms were used whose design follows those in [7], [3], [4], [2]. A population of agents plays a round-robin tournament of PD-Tego. An elite consisting of two-thirds of the population is saved and the remainder of the population is replaced. Pairs of parents are chosen from the elite by fitness-proportional selection. The parents are duplicated, the duplicates undergo two point crossover of their lists of states, and then 1-15 point mutations are performed with the number of mutations selected uniformly at random. A point mutation modifies a single state, changing a single parameter. The parameters that can change are the players selected for comparison, the threshold value used for comparison, and the two transitions for the true and false Boolean outcomes. The algorithm is run for 250 generations.

The algorithms differ in that the population size for the baseline experiment was set to 30 and the baseline used PD(20,50,1)-Tego. The lack of additional planning states follows from the fact that the opponent in the baseline study cannot change the order of its players, so there is no need for additional planning stages - the agent rapidly attains a desired order and stops moving players. The full, co-evolutionary version of the algorithm with competing agents used a population size of 120. Two collections of thirty independent runs were performed for the baseline and full PD(20,50,5)-Tego game.

V. Results and Discussion

Figure 4 shows examples of the evolution of agent fitness over time for both the baseline and PD-Tego experiments. The baseline experiment is, in effect, an optimization experiment in which the agent refines its deployment against a helpless opponent. The fitness plots show that this is in fact what is happening. The maximum fitness remains constant for a substantial portion of evolution suggesting that the choice not to run multiple rounds of updating in the baseline experiment was a correct one. In any case, the baseline shows that the agent representation is functioning in a nominal fashion in a simple environment.

Comparison of the fitness plots of the baseline with those of the co-evolving agents shown that the baseline scores are a weak upper bound on the co-evolved scores. In addition, while the baseline maximum fitness shows the fitness ratchet of an elitist algorithm (which the baseline algorithm is) the co-evolved agents usually show a decline in average fitness, and in the later generations, maximum fitness. This contrast suggests that robust competition is taking place during evolution.

The decline of the average fitness in the co-evolved agents is barely visible in the plots shown in Figure 4 so Figure 6 shows a close-up of one of the average fitness plots. This behavior is typical but not universal. Two of the thirty PD-Tego evolutionary runs showed trends that had no clear direction; twenty-eight trended downward.

The large difference between maximum and average fitness suggests that the population is maintaining a diversity of behaviors. This is probably made possible by the use of agents with a large number of states - required because of the large number of possible swaps - but this in turn suggests that 250 generations is simply not enough time for agents with such large genomes to converge.

Examining the permutation scatterplots in Figure 5 gives us more information. Different runs of the baseline algorithm yielded different results, but with a number of common features. This suggests that the fitness landscape has multiple

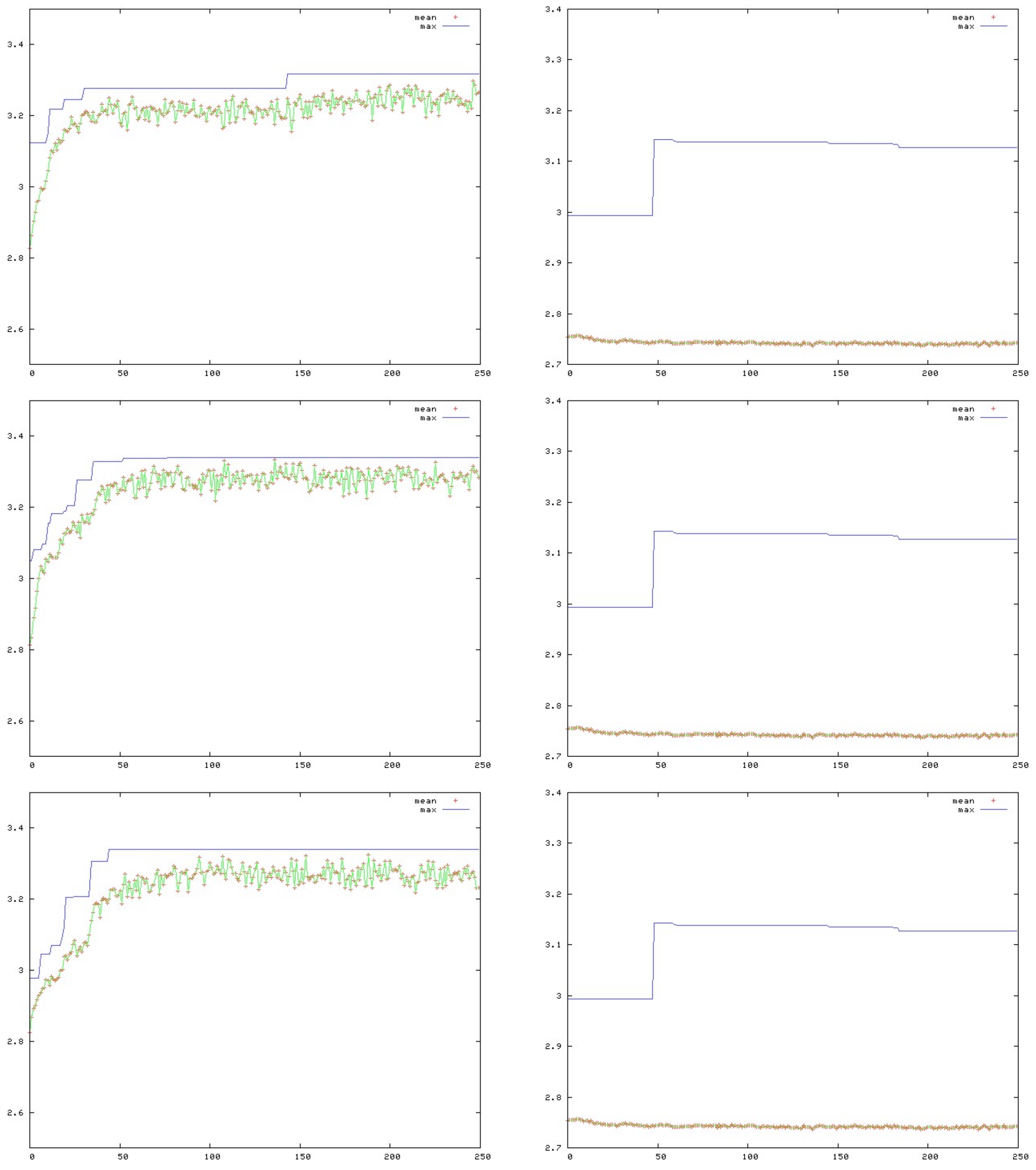

Fig. 4. Shown are plots of fitness over evolution of the agent populations for three exemplary runs from the baseline (left) and PD(20,50,5)-Tego (right) experiments.

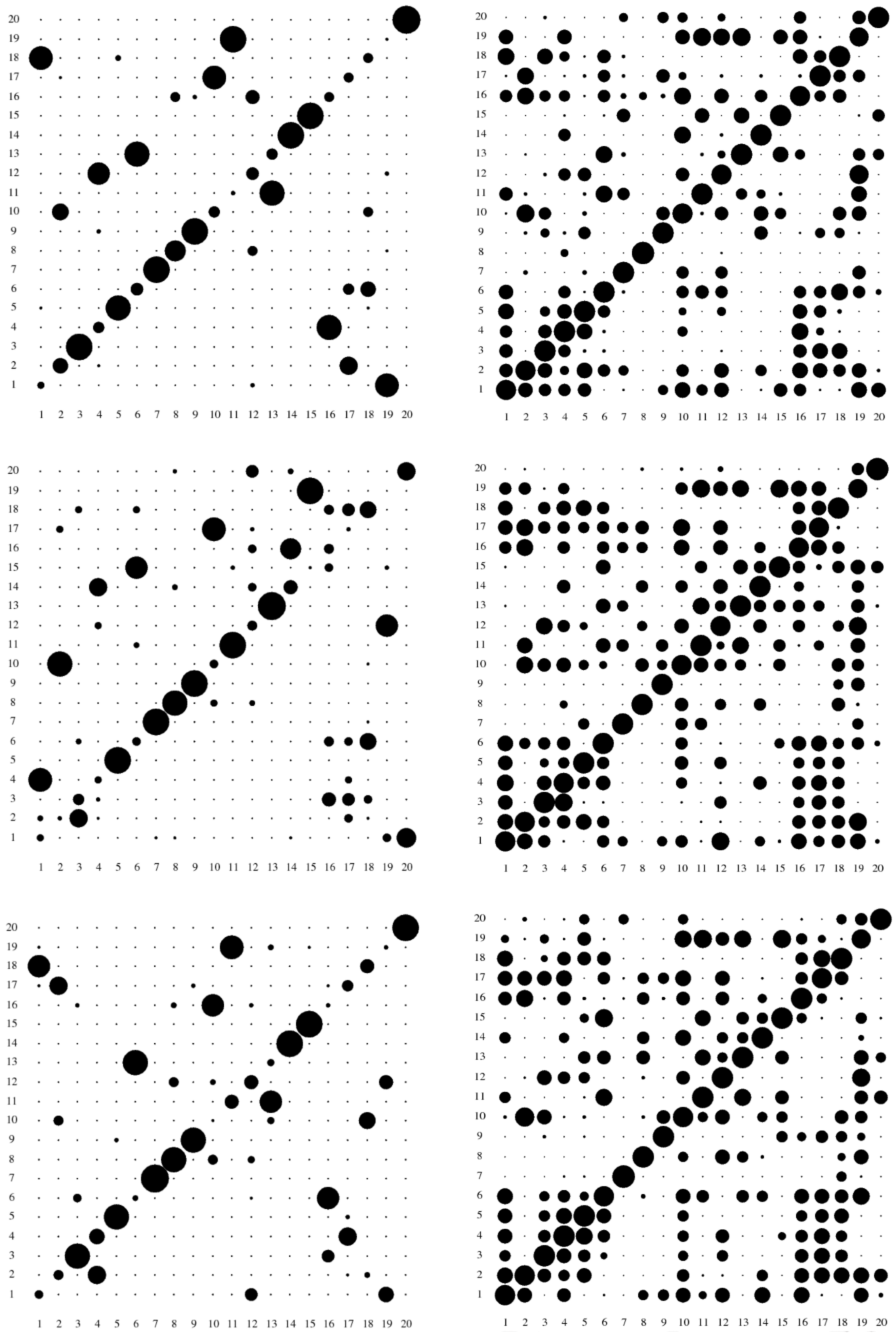

Fig. 5. Shown are permutation scattergrams of the matchings made in three exemplary runs from the baseline (left) and PD(20,50,5)-Tego (right) experiments. The scattergrams are in row-chooses-column as opponent order.

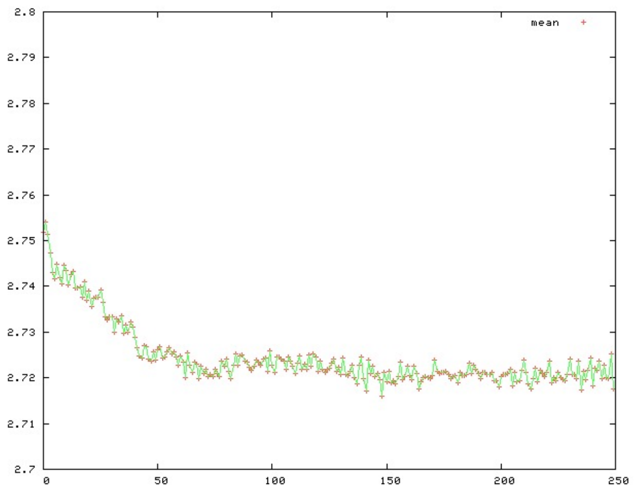

Fig. 6. Average fitness for one run of the PD-Tego runs with the fitness scale adjusted to make the trend easier to observe.

local optima, but that these contain common choices of player matching in common. All three permutation scatterplots shown match the Fortress players against themselves with high probability. Treasure hunt strategy 20 plays better against itself than any other available strategy, something all three depicted runs found, albeit to different degrees. Matching 18, a treasure hunt with always cooperate is a 2:3 majority choice; a sensible outcome as always cooperate can induce cooperative behavior from strategy 18.

If the population in a baseline run had converged to a single agent type then the corresponding permutation scatterplots would have one dot in each row and each column. While there is often a large dot, most rows have multiple non-trivial dots. This is additional evidence that the population is converging on a final genotype quite slowly.

The permutation scatterplots for the PD-Tego runs show a far greater diversity of matchings than the baseline runs. They also show a fairly high weight on the diagonal of the scattergram. While the fitness in the baseline runs was absolute, the fitness in the PD-Tego runs is relative. This means that matching a player against itself yield no advantage to either agent. Juxtapose this against the fact that some player mismatches are worth more than others and it is clear that leaving players matched against themselves is a safe place to put the less critical players while the agents work on getting the high-value mismatches in place.

Since agents are co-evolving in a single population it is likely that there will be genetic transformation of information about which mismatches are highly desirable or, from the opponent perspective, undesirable. This means that the adversarial planning steps include a good deal of exchanges that are performed simply to counter one another. The large jump in fitness in PD-Tego runs represents the discovery of a critical mismatch that an opponent does not have access to. The large genome size, together with the fact that most states are accessed by a series of "false" Boolean outcomes, means that genetic transmission of information is inefficient. In biological terms, when an agent arises with a good exploit, that exploit is not all that heritable, leading to a creature that can happily eat most of its young.

VI. Conclusion and Next Steps

This study presents the *-Tego framework for adversarial planning and demonstrates it on prisoner's dilemma as PD-Tego. Baseline experiments show that the optimization problem itself is complex with multiple optima. The PD-Tego co-evolution demonstrate that the agents are engaged in competition, not cooperation, with matching players against copies of themselves a common strategy. The ration of swaps per planning stage, 50, seemed large during the planning stages of this experiment but, in retrospect, may actually be small. This results from the fact that different player type mismatches can have substantially different associated payoff differentials. This means that agents can get into a dogfight over critical players and, largely, ignore the others. This places a parameter study on the number of swaps per planning stage in the early part of the queue for additional work.

Prisoner's dilemma was chosen for the initial demonstration of *-Tego because both authors have a great deal of experience with the game. The framework being general for competitive games, there is a need to test it for other games. Natural choices include other mathematical games such as rock-paper-scissors or the snow-drift game. Small board games like dodgem or hex with a small board would also make interesting experiments.

A. The Number of States

The choice to use a large number of states was made because it was assumed that it should be easy for agents to achieve almost any permutation of their players. Each state encodes only one pairwise swap and making an agent use many states to achieve a particular order seemed burdensome. At least for the set of players used in this initial demonstration, the fact that only some pairs of players were of interest suggests that the agents are indifferent to much of the ordering. This indifference manifests as equivalence classes of permutations which, in turn, would permits agents to function efficiently with far fewer states. Reducing the number of states could be used as a probe to discover critical pairs. With a limited number of states, the critical pairs would be those that the states have the ability to move at all.

B. Genetic Convergence

Even the baseline runs in this study have residual genetic diversity at 250 generations. This is clear evidence, somewhat contradicting the flat maximum fitness plot, that evolution might beneficially be run for a longer time in the baseline runs. The PD-Tego runs are strongly competitive and so unlikely to converge at all. These runs are subject to negative density dependent selection [13], in which traits are valuable in direct proportion to their rarity. If an agent finds an exploit, in the form of a player pairing other agents cannot interfere with, that grants it substantial fitness. As that trait in inherited by offspring the trait becomes less valuable.

In evolutionary optimization there is selection for high fitness, but also a secondary selection for heritability of that fitness. In the situation in the PD-Tego simulation an almost opposite effect occurs. The fact that a behavior may require many or few state-transitions to engage means that the heritability of particular variables is both highly variable and evolvable. If a high value trait is difficult to inherit then the agent possessing it may have a very long lifespan. This peculiar situation is currently hypothetical, but consistent with the persistent difference between mean and maximum PD-Tego fitness. Testing this hypothesis is a topic for future research.

C. Automatic Analysis

In [2], [6] techniques such as fingerprinting and agent-case embeddings are developed for the automatic analysis of evolved structures. Give the large number of states potentially required in agents for PD-Tego, and probably *-Tego for other games, automatic behavioral analysis techniques are badly needed. These techniques could be used to figure out how many different agent types there are. Given the similarities in different results in the baseline study, it is likely that evolution is discovering similar strategies. The BDA encoding, however, is almost ludicrously many-one with a given strategy have an entire combinatorial space of different encodings. Agent-case embeddings are constructively representation independentwith the assessment of an agent being based purely on its behavior for different cases of the problem its solving. For *-Tego, the cases are other agents, yielding a natural avenue for automatic analysis.

D. Genetic Isolation

It seems likely that the sharing, by opponents in a competitive game, of genetic information has the potential to cause problems. This can easily be addressed by using genetically distinct populations. An example of this type of technique is the *multiple worlds* algorithm [11], [12]. Multiple worlds has multiple populations. Fitness evaluations use one agent from each population but all reproduction takes place within the worlds. This means that each population is genetically isolated from the others. An item on the agenda for future research is comparing results from experiments on well mixed and genetically isolated populations.

E. Potential Cooperation

While the agents in PD-Tego are clearly in a state of adversarial competition, there is no reason that *-Tego must be adversarially competitive. Suppose we have players each of which yields the highest score when playing against a copy of itself.

In that case, the agents are engaged in a matching game and obtain the best score by cooperating. This means that the *-Tego framework can be used to assess the degree to which a group of players of strategies is naturally cooperative or competitive.

References

[1] D. Ashlock. *Evolutionary Computation for Opimization and Modeling*. Springer, New York, 2006.

[2] D. Ashlock, E.Y. Kim, and W. Ashlock. Fingerprint analysis of the noisy prisoner's dilemma using a finite state representation. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):157–167, 2009.

[3] D. Ashlock, E.Y. Kim, and L. Guo. Multi-clustering: avoiding the natural shape of underlying metrics. In C. H. Dagli et al., editor, *Smart Engineering System Design: Neural Networks, Evolutionary Programming, and Artificial Life*, volume 15, pages 453–461. ASME Press, 2005.

[4] D. Ashlock, E.Y. Kim, and N. Leahy. Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints. *IEEE Transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews*, 36(4):464–475, 2006.

[5] D. Ashlock and E. Knowles. Deck-based prisoner's dilemma. In *Proceedings of the 2012 Conference on Computational Intelligence in Games*, pages 461–478, Piscataway NJ, 2012. IEEE Press.

[6] D. Ashlock and C. Lee. Agent-case embeddings for the analysis of evolved systems. *IEEE Transactions on Evolutionary Computation*, 17(2):227–240, 2013.

[7] D. Ashlock and B. Powers. The effect of tag recognition on non-local adaptation. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 2, pages 2045–2051, Piscataway, NJ, 2004. IEEE Press.

[8] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[9] R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[10] Daniel Beard, Philip Hingston, and Martin Masek. Using monte carlo tree search for replanning in a multistage simultaneous game. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[11] J. A. Brown. Multiple worlds evolution for motif discovery. In *Proceedings of the 2012 IEEE Symposium on Bioinformatics and Computational Biology*, page 9299, 2012.

[12] J. A. Brown. More multiple worlds evolution for motif discovery. In *Proceedings of the 2013 IEEE Symposium on Bioinformatics and Computational Biology*, pages 168–175, 2013.

[13] S. A. Dudley and J. Schmitt. Testing the adaptive plasticity hypothesis: Density dependent selection on manupulated stemp length in *impatiens capensis*. *The American Naturalist*, 147(3):445–465, 1996.

[14] M. Hemesath. Cooperate or defect? Russian and American students in a prisoner's dilemma. *Comparative Economics Studies*, 176:83–93, 1994.

[15] P. Hingston and M. Preuss. Red teaming with coevolution. In *Proceedings of the 2011 Congress on Evolutionary Compuation*, pages 1155–1163, 2011.

[16] J. M. Houston, J. Kinnie, B. Lupo, C. Terry, and S. S. Ho. Competitiveness and conflict behavior in simulation of a social dilemma. *Psychological Reports*, 86:1219–1225, 2000.

[17] T. W. Hungerford. *Albegra*. Springer-Verlag, New York, 1974.

[18] M. Page and D. Ashlock. Binary decision automata modelling stress in the workplace. In *Proceedings of the 2013 Congress on Evolutionary Computation*, pages 3331–3338, Piscataway NJ, 2013. IEEE Press.

[19] D. Roy. Learning and the theory of games. *Journal of Theoretical Biology*, 204:409–414, 2000.

[20] K. Sigmund and M. A. Nowak. Evolutionary game theory. *Current Biology*, 9(14):R503–505, 1999.