To Appear In Proceedings of the IEEE Conference on Computational Intelligence in Games, August 31-July 2 2015.

Evaluating team behaviors constructed with human-guided machine learning

Igor V. Karpov, Leif M. Johnson and Risto Miikkulainen
Dept. of Computer Science, The University of Texas at Austin
2317 Speedway, 2.302, Austin, TX, 78712 USA
{ikarpov, leif, risto}@cs.utexas.edu

***Abstract*—Machine learning games such as NERO incorporate adaptive methods such as neuroevolution as an integral part of the gameplay by allowing the player to train teams of autonomous agents for effective behavior in challenging open-ended tasks. However, rigorously evaluating such human-guided machine learning methods and the resulting teams of agent policies can be challenging and is thus rarely done. This paper presents the results and analysis of a large scale online tournament between participants who evolved team agent behaviors and submitted them to be compared with others. An analysis of the teams submitted for the tournament indicates a complex, non-transitive fitness landscape, multiple successful strategies and training approaches, and performance above hand-constructed and random baselines. The tournament and analysis presented provide a practical way to study and improve human-guided machine learning methods and the resulting NPC team behaviors, potentially leading to better games and better game design tools in the future.**

I. Introduction

Evolutionary computation has shown promising results in many areas of machine learning research, particularly for tasks that are difficult to solve using formal optimization techniques, such as tasks involving teams of multiple, cooperating agents or environments with multiple, interacting sources of reward. Evolutionary approaches have been shown to work well for such tasks, even in the absence of a formal learning objective. There are even indications that explicit objectives can actually impede the ability of an evolutionary algorithm to produce interesting solutions [1].

Neuroevolution in games presents both exciting opportunities and interesting challenge [2]. The opportunities include enabling new kinds of games such as those with evolving content, those that adapt to their players, and those that allow the player to train NPCs as part of the game play. Among challenges is the objective evaluation of strategies and algorithms in these new complex types of environments.

The NERO video game [3] was originally developed to demonstrate that neuroevolution could be a powerful tool for constructing solutions to complex problems. A human player provides increasingly challenging goals, and a team of NPCs evolves to meet those goals, eventually excelling in the game. Complex behavior was demonstrated in a number of different challenge situations, such as running a maze, approaching enemy while avoiding fire, and coordinating behavior of small sub-teams. However, the final behavior of entire teams was never evaluated relative to each other, so it is not clear how complex the behaviors could become in this process and what successful behavior in the game might actually look like. Also, it is not clear whether there is one simple winning strategy that just needs to be refined to do well in the game, or whether there are multiple good approaches; similarly, it is unclear whether winning requires combining individuals with different skills into a single team, or perhaps requires on-line adaptation of team composition or behaviors.

In any case, such evaluations are difficult for two reasons: (1) designing teams takes significant human effort, and covering much of the design space requires that many different designers participate; (2) evaluation of the resulting behaviors takes significant computational effort, and it is not clear how it can be best spent. This paper solves the first problem by crowd-sourcing, i.e., running a NERO tournament online. Students in the 2011 Stanford online AI course[1] were invited to participate. About 85 of them did, many spending considerable effort to produce good teams, thereby resulting in a wide selection of approaches and solutions. The second problem was solved through running a comprehensive round robin tournament of 24,180 games in parallel in a Condor cluster, and by analyzing the strategies and performance of the submitted teams.

The results from the tournament were then used to identify complex and interesting behaviors that perform well on the task. Three main approaches were found, and interestingly, none of them dominated the others. In the final results, there were many circularities, where team A beats team B, which beats team C, which then beats team A. We believe that it is precisely such complex interactions that make the game interesting and fun to play. Games centered around machine learning may therefore be a viable game genre in the future as well as a productive platform for research in human-guided machine learning and multi-agent systems. Human-guided neuroevolution is a method to employ in designing such games, allowing humans to be creative at the high level while letting machine learning construct the actual behaviors.

II. Background

Before discussing the strategies that were evolved, the NERO game and the OpenNERO software implementing it is first described, followed by an overview of various methods

[1]www.ai-class.com

for evaluating complex behavior in tournaments, and the role of competitions in advancing the state of the art in AI research.

A. NERO and OpenNERO

NERO [3] was originally developed as an experimental platform for training teams of agents to accomplish complex tasks based on the rtNEAT [4] method for evolving artificial neural networks. The original NERO game was later extended into an open-source version called OpenNERO,[2] which is a general-purpose platform for AI research and education [5]. OpenNERO includes several different environments and AI methods in addition to the NERO game environment itself, but only the NERO environment in OpenNERO was used in this research.

Each NERO agent on a team has a fixed array of 15 sensors that detect agents on the same and opposite teams, placement of nearby walls, distance to a flag (if present), current motion, damage to opponents, and damage to the agent itself. Agents control their movement on the field using a two-dimensional control signal $u = < \ddot{r}, \ddot{\theta} >$, where $\ddot{r}$ is the linear acceleration of the agent in the direction of the agent's current orientation θ, and $\ddot{\theta}$ is the agent's angular acceleration.

Training teams in OpenNERO is similar to NERO. The user can dynamically change the virtual environment by adding, scaling, rotating or removing walls, moving a flag, and adding or removing immobile enemy agents. The user can also change the way the fitness function is computed by adjusting a (positive or negative) weight on each of the different available fitness dimensions. The available fitness dimensions are *stand ground* (i.e., minimize $\dot{r}$), *stick together* (minimize distance to the team's center of mass), *approach flag* (minimize distance to a flag on the field, if present), *approach enemy* (minimize distance to the closest enemy agent), *hit target* (successfully fire at an enemy), and *avoid fire* (minimize accrued damage).

It is entirely up to each player how to train their team. In this evaluation we did not record the training process, and so we only have indirect measures of how the teams were trained, such as how long (how many removal/replacement cycles) did the evolution last for, how large the neural networks were, how similar to each other they were, and what the networks and the resulting behavior was. These indirect measures do not seem to correlate significantly with team performance in our sample. In future such evaluations it would be very interesting to see the exact training process in each case and to see what correlates to good team performance.

In a previous evaluation with a smaller sample of undergraduate students (not presented here) we observed that a typical training session can last from 10 minutes to several hours, and can include varied sequences of training actions such as modifying the environment, modifying the fitness landscape, restarting the training process, waiting for a behavior to emerge, combining results of previous training sessions (genomes from different teams), evaluating an existing strategy against another one in battle mode. The creative process of coming up with a training strategy is an interesting subject of study in and of itself, however, it is outside of the scope of this paper.

A team of NERO agents can be serialized to a flat text file. The text file describes each of the 50 agents on a team. Agents that use rtNEAT serialize to a description of the genotype for each agent, and agents that use Q-learning serialize their (hashed) Q-tables directly to the file.

For the battle task, two teams—each consisting of 50 NERO agents—occupy a continuous, two-dimensional, virtual playing field of fixed size. The playing field contains one central obstacle (a wall), four peripheral obstacles (trees), and four walls around the perimeter to contain all agents in the same general area.

Each NERO agent starts a battle with 20 hit-points. At each time slice of the simulation, each agent has the opportunity to fire a virtual laser at the closest target on the opponent's team that is within two degrees of the agent's current orientation. If an agent fires and hits an opponent, the opponent loses one hit-point. The score for a team is equal to the number of hit-points that the opponent team loses in the course of the battle.

III. Tournament

Participants in an online tournament were asked to contribute teams of autonomous agents evolved through neuroevolution or learned through value function reinforcement learning. The submitted teams were then evaluated in a round-robin virtual combat tournament. The space of behaviors and their performance in the tournament are analyzed below. This machine learning game tournament is a competition between different training strategies devised by the human participants.

The OpenNERO platform was extended for this tournament to allow mixed teams of rtNEAT (neural network) and state-action value function approximator agents; in addition to representation (arbitrary topology neural networks vs. hashed state-action value function approximators), these rtNEAT and Q-learning agents used different algorithms during training. rtNEAT is a direct evolutionary policy search method that uses lifetime fitness of each agent to rank individuals. Q-learning is a temporal difference reinforcement learning method that uses step-by-step reward information to update an estimate of the value of performing different actions from observed states in order to maximize the expected lifetime reward. The methods also differed in how they combined the multiple and multi-scale components of the fitness/reward function during training. For rtNEAT–based training, individuals within the population were ranked based on the weighted sum of the Z-scores over the fitness components. For Q-learning–based training, each fitness dimension was scaled to $[0, 1]$, and then a linear weighted sum was used to assign a total reward to each individual.

Both types of controllers could be submitted to the online tournament: artificial neural network controllers of arbitrary weight and topology, and hash tables approximating the value function of game states. The competitors could extend and/or modify the available OpenNERO training methods as well as create their own training environments and regimens. It was this training that determined the fitness of each team when pitted against other teams submitted to the tournament.

[2]opennero.googlecode.com

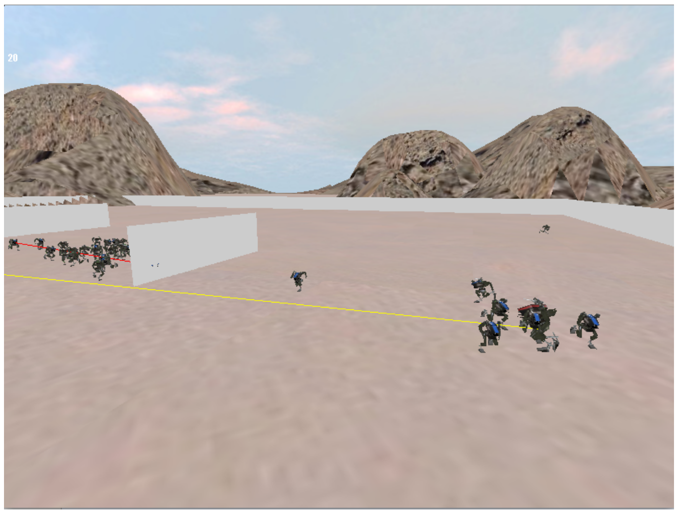


Fig. 1: A screenshot of a single NERO match. Two teams of agents are shown as bipedal robots in a playing arena with obstacles and boundaries. More information, including videos of sample games, can be viewed at code.google.com/p/opennero/wiki/TournamentResults2011

About 85 participants submitted 156 teams to the tournament. Of these, 150 teams contained neural network-controlled agents and 11 contained value table-controlled agents. Mixed teams were also allowed; four of the submitted teams contained both agent types.

Each team could include anywhere between 1 and 50 individual controllers, with any number of repeated controllers. If the number of controllers in the team file was smaller than 50, controllers were replicated in turn until 50 individual agents could be created. If the number of submitted controllers exceeded 50, only the first 50 were used.

To speed up the tournament, each match was played off-screen and without AI frame delays (necessary if trying to observe a match) and limited to 5 minutes of game time. In practice, good teams were able to eliminate most opponents in less than 5 minutes. The team with the highest remaining number of hit points at the end of each match was declared the winner. Ties were rare and were broken by a pseudo-random coin toss. The matches were run in parallel on 100 nodes of a Condor compute cluster, each a virtual machine with a 2GHz Intel Core 2 processor and 2GB of RAM [6].

Each of the 156 submitted teams was matched against the 155 other teams. Each pair of teams was matched up twice, allowing each team to play once as the blue team and once as the red team. This resulted in 24,180 separate games, with the whole tournament taking under 24 hours to complete.

IV. Results and Analysis

The results of the tournament and an analysis of the strategies exhibited by the submitted teams are presented below. Team behaviors were categorized based on state statistics and also qualitatively by running selected matches on-screen. In this work we did not collect or consider the individual training processes that went into the training of each team in the tournament - this is an interesting direction for future work.

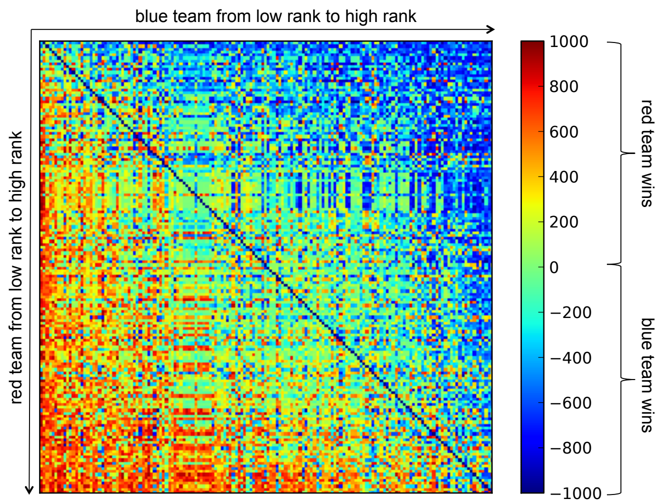


Fig. 2: Results from the round-robin NERO tournament. Teams are sorted by average score differential over all matches. Rows and columns in the matrix represent teams in the tournament, and colors represent score differentials for the respective individual matches between two teams. Red indicates victory by the row team, and blue indicates victory by the column team.

Rank	Team	Total wins
1	synth.pop	137
2	synth_flag.pop	130
3	me - Rambo	126
4	lolwutamidoing	126
5	PollusPirata	125
6	Cyber-trout	124
7	CirclingBullies	123
8	SneakySnipers	121
9	Tut	121
10	coward1	120

TABLE I: Top 10 teams submitted to the tournament. Overall winner of the tournament was decided according to number of wins, with any ties broken by score difference. Although there are teams that dominate in terms of number of wins, there are other teams that are able to defeat these strategies.

A. *Match statistics*

Figure 2 shows the complete results of the final round-robin tournament. Black squares along the diagonal represent matches that were not played, blue squares indicate a win by the column team, and red squares indicate a win by the row team. A group of near-duplicate teams shows up as the band of similar-colored games about one-third of the way through the matrix. The teams in the Figure are enumerated on both axes in order of increasing average match score differential.

Table I lists the top ten teams. After each team plays every possible opponent, the number of wins accrued by each team provides a reasonable estimate of the probability of that team winning in a future match [7]. Despite the large number of teams, no single competitor emerged that significantly outperformed all others (though some outperformed most). This is sometimes referred to as a paradoxical tournament (see Section IV-E).

The overall distribution of teams by number of wins is approximately normal with mean 76.7 (approximately equal to half the number of games that each team plays) and standard deviation 28.0; the fact that the distribution is normal implies that number of wins that each team accrues can be thought of as repeated samples from a common underlying process.

Teams also follow an approximately normal distribution by score differential. They cluster around zero, with some teams performing significantly better than the mean, others performing significantly worse than the mean, and most teams performing near zero. Teams in the top ten by average match score were at least 1.5 standard deviations above the mean, and the team with the largest average score differential exceeded the mean by 2.2 standard deviations.

Collecting statistics of the tournament matches in this way allows us to discover structure in the submission pool. For example, it is possible to cluster teams by their relative score vectors, finding teams that are similar because they win and lose against the same opponents. It is also useful to see the overall gradient of the score difference to notice if any teams are particularly "dark" or "bright", i.e. if they win or lose with low or high score difference.

B. Correlation Analysis

An insight into training successful teams can be obtained by measuring how well each team's average score difference correlates with the score difference when playing against a weak, medium, and strong opponent (Figure 3).

Three trends can be observed from this data. First, the correlation is higher when playing against a team of average strength. Second, higher-scoring teams are only good at discriminating the other high scorers in the group. Third, lower-scoring teams are only good at discriminating other low-scoring teams.

This insight can be used when deciding which teams to play against during the training process. As the team starts out training, it is most useful for the team to train against an average opponent (or perhaps even a low-scoring one) to quickly identify and eliminate bad strategies. Then, as the team improves, it may become beneficial for it to train against higher scoring teams to continue to improve.

C. Successful strategies

In NERO, agents are "shaped" towards more complex behaviors by progressively changing the environment and the fitness function progressively during training. This process can be used to create teams of agents that perform specific tasks during a battle. Given the complexity of the environment and the task, many different strategies can arise in this process, and they can interact with each other in complex ways. Considering this potential complexity, evolved strategies in the tournament turned out to be surprisingly easy to analyze.

Because fitness is evaluated similarly for each team member during training, teams generally consist of agents that perform similar actions in a given world state[3]. It was therefore possible to characterize the most common strategies used in the tournament by considering the histograms of agent positions during sample games as described below.

D. Strategy clusters

In order to determine the different kinds of approaches taken by team trainers, teams were clustered using the following procedure. First, positions of team members during the match were recorded and a coarse 32x32 heat-map formed for each team (Figure 4a). These histograms were clustered using K-medoid clustering around 10 representative behaviors. The distribution of wins and scores within each cluster (Figure 4b) was then considered to determine if any of these coarse behavior clusters is significantly better within the tournament.

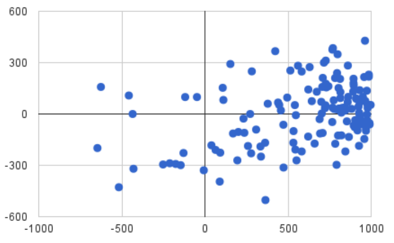
The clustering does produce several qualitatively distinct categories of strategies. Many teams opted for moving towards the wall, navigating around it to the left or the right, and attacking the enemy team. However, other teams stayed near their original location or even back-pedaled to the back wall, leading to a more defensive strategy. Within each of these categories, there were solutions that performed well as well as strategies that did not (Figure 4b). In other words, what mattered most was not just what high level strategy was used, but also how well it was implemented. These fine differences may be part of what makes the tournament interesting for participants.

E. Strategy cycles

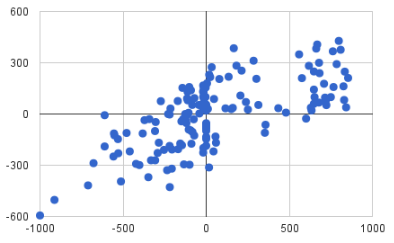
Perhaps most interestingly, the strategies do not form a strict dominance hierarchy, but instead are highly cyclic. For instance, the third-place `me-Rambo` reliably defeats the first-place `synth.pop`, apparently due to subtle differences in timing. On the other hand, `synth.pop` wins over the 24th-place `EvilCowards`, because the `synth.pop` pack splits into two and breaches the wall from both edges simultaneously. However, `EvilCowards` handily defeats `me-Rambo`, because agents in the `me-Rambo` train are eliminated one-at-a-time as they come around the wall!

The amount of cyclicality within a tournament can be measured formally by representing the results of the tournament as a graph, where the teams are the vertices and the edges are directed from winner to loser in each match. A tournament graph is K-paradoxical, if for every K-sized subset of the teams, S, there is some team $t_0 \notin S$ such that for every $t \in S$ t_0 beats t. In other words, a K paradoxical tournament is K-removed from having a single dominant strategy [8]. The OpenNERO tournament is paradoxical by this definition with a $K >> 1$. Such cyclic landscape makes this an interesting domain to train for, both from the point of view of a human player, and as a technical question of how to train an AI agent to do well in such a tournament.

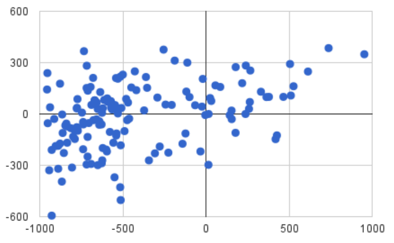
[3]In principle, multiple teams can be trained using different shaping strategies, and single agents from those teams then combined into one team by copying the appropriate parts of the serialized team files (as was suggested in the online tournament instructions). However, most teams submitted to the tournament did not (yet) take advantage of this possibility; instead, agents on a single team usually performed similar actions in response to each game state.



(a) Low scoring team ($r = 0.38$)

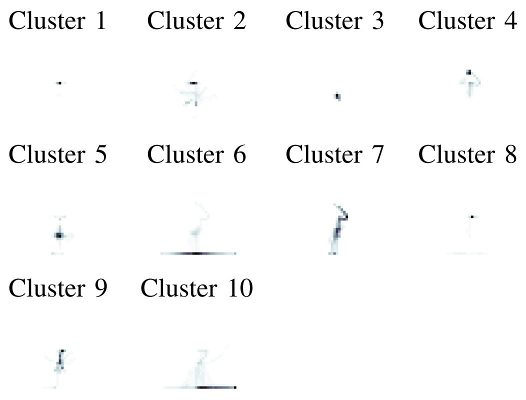


(b) Medium scoring team ($r = 0.69$)

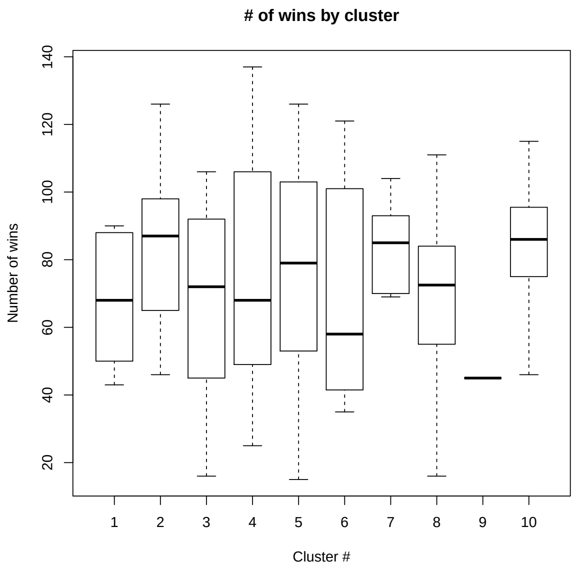


(c) High scoring team ($r = 0.39$)

Fig. 3: Correlations between teams' average round-robin score difference (x-axis) and their score difference when playing against a weak team (a), a medium team (b), and a strong team (c). Correlations are strongest against a team of similar strength, which suggests that training could most effectively proceed with opponents from low to high-scoring teams.



(a) 10 cluster medians



(b) Distribution of victories vs losses within each cluster.

Fig. 4: K-medoid clustering was used to determine 10 clusters of 32x32 heat-maps of team behavior during matches against a random opponent (a). As with individual teams, none these clusters dominate the others.

F. Tournament Stability

In order to test wether the tournament provides a stable ranking over teams in the face of environment randomness and other sources of noise, a second tournament was run among the 10 top-ranked teams but now with 100 matches per team pair instead of two.

Most of the match-ups in this top-10 tournament were stable (Figure 5). However, given a particular ranking of teams (either from the original tournament or from the top-10 tournament) how do we know that the ranking is not due to chance?

For any given tournament instance, it is possible to define a *tournament stability index* $\mathcal{T}$ between -1 and 1 [9]:

$$\mathcal{T} = \frac{\sum_{i=1}^{K} v_i}{N}$$

where K is the number of matches played and N is the number

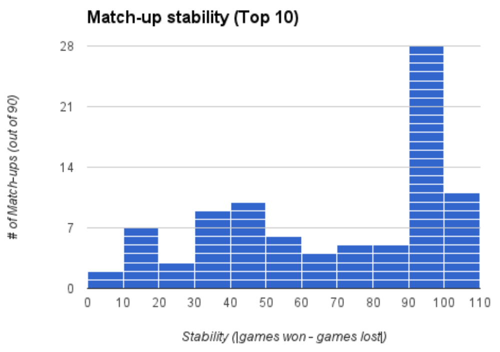


Fig. 5: Stability of match-ups between the top 10 teams from the tournament. Stability is measured as $|w - l|$, where w is the number of wins and l is the number of losses (out of 100 repeated games).

Baseline	Wins	Ties	Losses	score-Δ	Rank
Random	79	21	56	-51.58	98/156
NN	23	1	125	-225.15	123/156
RL	77	22	59	-62.66	100/156

TABLE II: Performance of baseline teams.

of matches, and v_i is defined:

$$v_i = \begin{cases} 1, & \text{if higher-ranked team won the match} \\ -1, & \text{if lower-ranked team won the match} \\ 0 & \text{otherwise} \end{cases}$$

The index is further normalized to ensure that a tournament decided by random coin flips has expected stability index of 0.

Defining this index makes it possible to compare the stability of different tournaments. For example, the full tournament of all the submitted teams has a stability index of 0.455, while the top-10 tournament with 100 repeats has a stability index of .533. As a comparison, the most stable tournament Lundh, et. al report an NBA tournament from 1995-96 as having a stability index of 0.31.

G. Baseline Comparisons

In order to provide a baseline level of performance in the battle tournament task, several artificial teams were introduced into the tournament. The uniform random team (*random*) simply returns a pseudo-random number within the range of possible action values at each step. The untrained neural network team (*NN*) consists of fully connected single-layer neural networks initialized with normally distributed random weights (as in the starting population for rtNEAT). The untrained reinforcement learning team (*RL*) baseline is the performance of the policies represented by the sparsely initialized function approximator and epsilon-greedy action selection. The results of these baseline comparisons are summarized in Table II.

Overall, the trend seen within the submitted teams still holds, i.e. no team dominates all others nor is any team absolutely dominated by others. Surprisingly, the uniform random baseline performs relatively well.

The explanation for the relatively high performance of the uniform random baseline may be that it is a fundamentally different type of behavior from all the others in the tournament, one that is not commonly the result of the training processes available to the participants of the tournament.

While the random behavior performs surprisingly well (perhaps by being equally prepared for any strategy it plays against), it is not particularly interesting to watch. The baseline comparisons add support the idea that human trainers develop their teams based on a rational model of what they will encounter in the tournament, and that the random baseline is not what they expect. However, the low ranking of the RL and NN baselines shows that the submissions were, by and large, an improvement over the starting point of the training process.

V. Discussion and Future Work

Even though a number of distinct approaches were identified among the successful NERO teams, it was also clear that there was no single best strategy. At this point it is unclear whether such a strategy even exists, and this is indeed what makes the game interesting. There is room for innovation and creativity, and the outcomes often turn out to be surprising.

With more multi-objective evolutionary methods [10], it might also be possible to develop multi-modal behaviors that identify what strategy the opponent is using, and select a counter-strategy accordingly. It might also be possible in principle to adapt to opponents online, while the battle is taking place.

Larger tournaments could be supported by using a hybrid structure; round-robin pools could be run in parallel to identify the proper seeds for top-ranking teams, and then a double-elimination tournament could be used to identify the overall winner. Thanks to the independence of individual matches in round-robin tournaments and within each level of a knockout tournament, it should be possible to scale up to even larger tournaments by running games on more compute nodes or carefully designing a tournament structure to optimize the use of available resources.

In the future, the demonstrated crowd sourcing of teams of artificial agents can be instrumented to record **how** the teams are trained, in order to improve the human-guided aspects of the training algorithms. What methods of training work well? This evaluation strategy can be combined with human subject studies similar to [11] in order to contrast and compare alternative methods for combining human creativity and the speed and breadth of machine learning approaches.

VI. Conclusion

The results of the online NERO tournament demonstrate that successful behaviors in a complex multi-agent game can be highly diverse, which is precisely what makes training such teams an interesting game component. The results also show that complex behaviors can be constructed effectively by shaping neuroevolution, demonstrating its power as an approach for constructing NPCs for complex games. Finally, the machine learning game tournament format and the associated analysis techniques are a promising experimental tool in studying human-guided machine learning methods. On the whole, machine learning games are both a viable genre for game developers and a useful tool for artificial intelligence researchers.

VII. Acknowledgements

Special thanks to all participants of the OpenNERO online tournament, Sebastian Thrun and Peter Norvig for allowing us to run the tournament in their online AI class, and Ken Stanley, Wes Tansey, and Adam Dziuk for help with OpenNERO. This research was supported in by Google, Inc. and NSF under grants DBI-0939454, IIS-0915038, and IIS-0757479.

References

[1] B. Woolley and K. Stanley, “On the Deleterious Effects of A Priori Objectives on Evolution and Representation,” in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, ser. GECCO’11, Dublin, Ireland, July 2011, pp. 957–964.

[2] S. Risi and J. Togelius, “Neuroevolution in Games: State of the Art and Open Challenges,” *arXiv preprint arXiv:1410.7326*, 2014. [Online]. Available: http://arxiv.org/abs/1410.7326

[3] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, “Real-time Neuroevolution in the NERO Video Game,” *IEEE Transactions on Evolutionary Computation*, pp. 653–668, 2005. [Online]. Available: http://nn.cs.utexas.edu/?stanley:ieeetec05

[4] K. O. Stanley and R. Miikkulainen, “Evolving Neural Networks through Augmenting Topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[5] I. V. Karpov, J. Sheblak, and R. Miikkulainen, “OpenNERO: A game platform for AI research and education,” in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2008.

[6] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: the Condor experience.” *Concurrency - Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[7] H. Daniels, “Round-robin tournament scores,” *Biometrika*, vol. 56, no. 2, pp. 295–299, 1969.

[8] P. J. Cameron, “The random graph,” in *The Mathematics of Paul Erdös II*. Springer, 1997, pp. 333–351.

[9] T. Lundh, “Which Ball is the Roundest? - A Suggested Tournament Stability Index,” *Journal of Quantitative Analysis in Sports*, vol. 2, no. 3, July 2006.

[10] J. Schrum and R. Miikkulainen, “Evolving agent behavior in multiobjective domains using fitness-based shaping,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, Portland, Oregon, July 2010, pp. 439–446. [Online]. Available: http://nn.cs.utexas.edu/?schrum:gecco10

[11] I. V. Karpov, V. K. Valsalam, and R. Miikkulainen, “Human-Assisted Neuroevolution Through Shaping, Advice and Examples,” in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, ser. GECCO ’11, N. Krasnogor and P. L. Lanzi, Eds. Dublin, Ireland: ACM, July 2011.