Selection Schemes for Evolving Specialist Video Game Agents

Evolutionary Computing Task I

Group 42

Xiaojin Gu
2654391

Eui Yeon Jang
2569512

Tim van Loenhout
2525199

Lando de Weerdt
2652415

1 INTRODUCTION

Video and strategy games provide interesting challenges for developing artificial intelligence agents, requiring them to learn a variety of skills for a successful game-play. Whereas past approaches have focused on developing scripted agents [8], there are various machine learning approaches to develop game-playing agents. While there has been much attention on deep learning agents for video and strategy games, for example deep neural networks playing Atari games [10], it has been demonstrated that agents developed using evolutionary algorithms (EAs) can achieve competitive performance [12].

There are various components to an EA, one of which is the selection mechanism. The selection mechanism has a vital role as a driving force towards finding better individuals. It can narrow down the search area by discarding poorly performing individuals [11] and also aid in maintaining genetic diversity via selection pressure [7].

The goal of this paper is to investigate how the choice of a parent selection scheme in genetic algorithms (GAs) affects evolving a specialist game-playing agent in the EvoMan framework [4]. We examine two parent selection schemes – (1) proportionate stochastic universal sampling and (2) tournament selection – and compare the performance of the GAs on three fronts: population fitness, individual gain, and relative speed of convergence.

2 RELATED WORK

Developing a game-playing agent is a multimodal problem, where there exists multiple possible solutions. Selection schemes with lower selection pressure are preferable in such multimodal problems [1]. These "softer" selection schemes are slower at reducing the genetic diversity of the population an thus sufficiently explore the search space for a good optimum.

2.1 Proportionate Stochastic Universal Sampling

In traditional GAs, the probability of an individual being selected as a parent is proportional to its fitness value. Given the selection probability, we can sample the population to pick the parents. Stochastic Universal Sampling (SUS) is a sampling algorithm with zero bias and minimum spread [2], analogous to the roulette wheel algorithm with λ equally spaced arms. Consequently, the outcome of SUS is as close as possible to the expected value. This helps to preserve more diversity compared to, for example, roulette wheel selection, where there is a discrepancy between the expected value and the sampling probability. Nevertheless, the high selection pressure in proportionate SUS still enforces faster increase in fitness during the early stages of evolution. However, this may also lead to premature convergence [7].

2.2 Tournament Selection

Tournament selection is a widely used selection mechanism in GAs [9]. A tournament of size k is created by randomly selecting k individuals, the best among which is chosen as the parent. This gives all individuals a chance to be selected, preserving genetic diversity [11]. It does not require global knowledge of the population, and allows control over the selection pressure by adjusting k – larger the tournament, the higher the selection pressure [5]. When $k = 2$, it is equivalent to linear ranking with $s = 2$. However, because each tournament is carried out individually, it suffers the same sampling error as roulette wheel selection [7]. Nevertheless, it is a popular choice due to its simplicity and speed.

We hypothesise that tournament selection will perform better than proportionate SUS in evolving the population such that it contains better individuals, in terms of defined fitness function and individual gain. We believe that proportionate SUS may lead to premature convergence as there is not a well balanced trade-off between exploration and exploitation. We further hypothesise that this will also lead the algorithm with proportionate SUS selection to converge sooner than the one with tournament selection.

3 EXPERIMENTAL SETUP

In order to test our hypothesis, we implement two GAs, differing only by their parent selection mechanisms, to evolve specialist agents to compete in the games of EvoMan. We make use of the the Distributed Evolutionary Algorithms in Python (DEAP) library [6] to implement the GAs.

The EvoMan game environment is configured to be at level two, enemy mode is set to static, and the agent, or player controller, is a feed-forward neural network (FFNN) controller. The rest of the parameters are kept at the default [4].

We represent our individuals as real-valued vectors containing the weights and biases of the FFNN. The FFNN has one hidden layer of 10 neurons with a tanh activation function, and the output layer using sigmoid activation function. All weights are uniformly initialised in the range $[-1, 1]$. The output is in the range $[0, 1]$, where the agent selects an action if the respective output is ≥ 0.5. The input provided to the network at each time step is normalised to unit length.

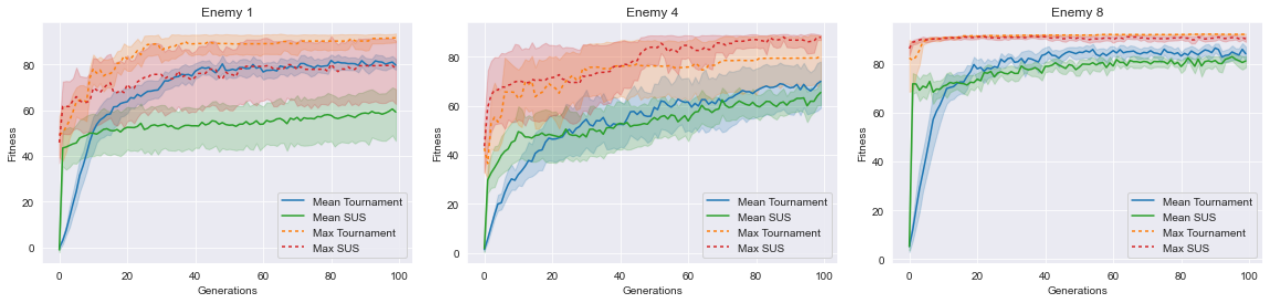

Figure 1: Means of the population-fitnesses and max-fitnesses, including their standard deviations, of 10 runs for enemies 1, 4, and 8 for GAs with proportionate SUS and tournament selection. Solid lines are the population-fitness, and dashed lines are the max-fitness. The shadow areas denote 1 standard deviation from the mean values.

The population size is set to 50 individuals. We perform two-point crossover with probability $p_c = 0.5$, where the offspring immediately replaces the parents after crossover. Gaussian mutation with $\mu = 0$ and $\sigma = 0.1$ with probability $p_m = 0.2$ is implemented. If an individual is to be mutated, then the independent mutation probability is 1.0, i.e. all values of the individual will be mutated. We have a termination condition of 100 generations.

An individual is evaluated by playing a full game against its enemy in EvoMan. The fitness of the individual is calculated by

$$fitness = 0.9 \cdot (100 - e\_energy) + 0.1 \cdot p\_energy - \log time\_step,$$

where p_energy and e_energy are the energy level of the player and the enemy agent, respectively, and $time\_step$ is the game time steps taken until end of the fight. The fitness values are to be maximised.

We implement one GA with proportionate SUS and another with tournament selection. The selection criteria for both mechanisms is the fitness values. This means that the segment sizes in proportionate SUS is proportional to the fitness values, and the individuals in a tournament are ordered in terms of their fitness values. The size of the mating pool is the same as the population.

We run an informal parameter tuning to select the tournament size. This is done by running the algorithm three times for $k \in \{2, 3, 5\}$ and observing how fast the mean fitness of the population increased and how fast it converged. We find that tournament size $k = 2$ gives the best result. Higher tournament sizes appear to result in more selection pressure causing a quick increase in fitness which then quickly flattens out. $k = 2$ yields more stable improvement of fitness across generations, resulting in a higher fitness in the long-run. We presume this was due to genes 'dying out' quickly with greater tournament sizes. Hence, we set $k = 2$ for our experiments. DEAP's implementation of SUS does not have further parameters to be tuned.

For each GA, we run the experiment independently ten times. For every generation during a run, we record the mean fitness value of the population, referred to as the population-fitness, as well as the maximum fitness value, referred to as the max-fitness.

For each GA and each run, we select the best individual, referred to as the champion, according to their individual gain,

$$individual\_gain = p\_energy - e\_energy,$$

yielding ten champions for each GA. Each champion plays additional five games against its enemy, and the mean individual gains of those games are recorded. We then perform an independent t-test between the GAs using the ten mean individual gains of each GA. Using these test results, we attempt to conclude statistical significance between the performance of the two GAs.

This entire process is repeated for each of the three selected EvoMan enemies: enemy 1, 4, and 8.

4 RESULTS AND DISCUSSION

4.1 Fitness Values and Convergence

The mean of the population-fitnesses and the mean of the max-fitnesses across all ten runs, including their standard deviations, are plotted in Figure 1. We observe that for all enemies in the first a few generations, both of population- and max-fitness values of SUS increase faster than tournament selection, and then slows down, almost becoming stagnant, in the later generations. Tournament selection takes over SUS in terms of mean population-fitness for all enemies, albeit the differences are somewhat minor for enemies 4 and 8. The mean population-fitness of tournament selection is even able to match the max-fitness of SUS for enemy 1. SUS wins over tournament selection for max-fitness only for enemy 4, although appears that for enemy 4, neither GAs have converged as the values continue to increase near generation 100. It is possible that given more generations, tournament selection would take over SUS.

Tournament selection's outperformance could be explained by SUS converging on a local optimum due to its tendency to exploit rather than explore during the early stages of evolution due to the higher selection pressure. Thus SUS evolves faster and converges earlier but does not necessarily achieve higher fitness against tournament selection.

We also observe varying standard deviations per enemy. Against enemy 1 and 4, we observe a high standard deviation compared to enemy 8. Such high standard deviation was obtained due to the population-fitness of a few runs greatly falling behind, relative to the other runs. They appeared to get stuck performing some tactic that kept failing and the population as a whole stopped evolving. This was most prominent against enemy 4, which required quite a difficult tactic to win. We presume some populations got 'stuck' in a local optima.

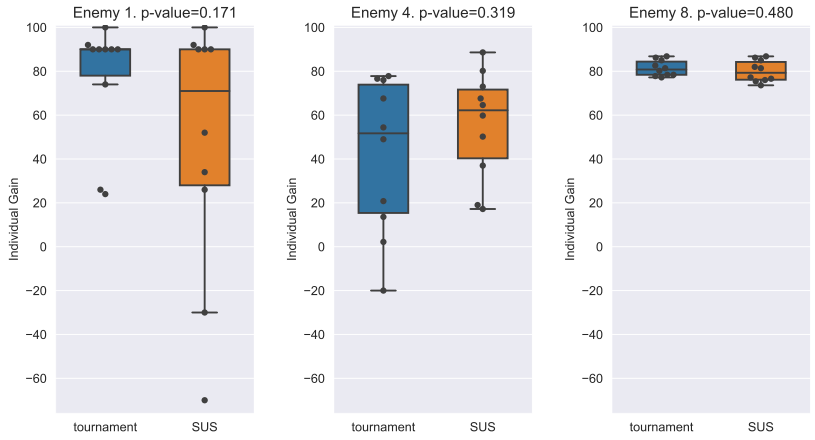

Figure 2: Box plots of the mean individual gains of the 10 champions of each GA after playing five games each.

Enemy	Baseline	SUS	TNMT
1	**90**	60.40	79.60
4	30	**55.72**	46.80
8	**83**	80.02	81.40

Table 1: Mean of the remaining player energy of the 10 champions of each GA compared to the baseline of GA10 in [3]. The value is 0 when the player loses. SUS denotes GA with proportionate SUS, and TNMNT with tournament selection. Highest value marked bold.

4.2 Individual Gains

Figure 2 displays the box plots of the individual gains of the champions of each GA for each enemy. Table 1 compares the mean of the individual gains of the 10 champions of each GA for each enemy against a baseline provided in [3].

For enemy 1, the maximum values of individual gains are 100 for both tournament and SUS, indicating that the individual beat the enemy without receiving any damage, however, both GAs have poorly performing outliers. SUS also has a greater variance. For enemy 4, SUS has higher maximum, median, Q3 and minimum values than tournament selection. For enemy 8, both the GAs have same maximum individual gains, and tournament has slightly higher max, median and min.

We observe that both our implementations outperform the baseline against enemy 4, and perform worse against enemy 1 and 8. However, we do observe in the box plots that the maximum performance of the champions is greater than the results presented as the baseline. This shows that the population is capable to reach such performance, but they do so with much less consistency.

In either case, all results obtained greatly depend on the selected parameters. Larger population sizes may result in one selection method clearly gaining the upper hand against the other. Similar variation may be observed with more or less mutation rates.

However, we do note that none of these results are with statistical significance and so we cannot derive formal conclusions from it nor confirm our hypothesis. Further research, possible with larger sample sizes, is required should one wish to have a greater chance of achieving statistical significance.

5 CONCLUSION

In this paper, we compared proportionate Stochastic Uniform Sampling (SUS) against tournament selection with a tournament size of 2. We further compared the obtained results given by the best individual, in terms of remaining agent's energy, against the baseline results from [3].

We observe trends in our results which align with our initial hypotheses. The mean population-fitness obtained by SUS increases faster in early generations, but is subsequently taken over by tournament selection. Looking at the performance of the champions, we found that tournament selection outperforms SUS against enemies 1 and 8, but loses against enemy 4. Unfortunately, none of these results are with statistical significance, and therefore we can neither accept nor reject our hypotheses. We conclude that more data is needed to derive any formal conclusion regarding the relative performance of both selection mechanisms.

We note that the rate at which the fitness of a population improves, and how fast it flattens out, greatly depends on the given opponent. Therefore, the performance of one mechanism compared to another may depend on the problem at hand. To better compare the influence of selection schemes, additional repetitions against more enemies is warranted. This may also shed light into how different algorithms may be better suited to develop agents against different enemies.

Moreover, further research is required to understand the interaction of other parameters of the algorithms with the selection mechanism. For example, we employ a steady-state model in this paper, whereas it is possible that a generational model may provide better performance. Parameter tuning is highly recommended to observe and compare the full extent of the influence of the selection mechanisms.

REFERENCES

[1] Thomas Bäck and Frank Hoffmeister. 1991. Extended selection mechanisms in genetic algorithms. (1991).

[2] James E Baker. 1987. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the second international conference on genetic algorithms*, Vol. 206. 14–21.

[3] K. da Silva Miras de Araujo and F. O. de Franca. 2016. Evolving a generalized strategy for an action-platformer video game framework. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 1303–1310.

[4] Karine da Silva Miras de Araújo and Fabrício Olivetti de França. 2016. An electronic-game framework for evaluating coevolutionary algorithms. (2016). arXiv:cs.NE/1604.00644

[5] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer. https://doi.org/10.1007/978-3-662-44874-8 Gebeurtenis: 2nd edition.

[6] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research* 13 (jul 2012), 2171–2175.

[7] Peter J. B. Hancock. 1994. An empirical comparison of selection methods in evolutionary algorithms. In *Evolutionary Computing*, Terence C. Fogarty (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 80–94.

[8] S. M. Lucas and G. Kendall. 2006. Evolutionary computation and games. *IEEE Computational Intelligence Magazine* 1, 1 (2006), 10–18.

[9] Brad L. Miller, Brad L. Miller, David E. Goldberg, and David E. Goldberg. 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems* 9 (1995), 193–212.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013). arXiv:cs.LG/1312.5602

[11] Noraini Mohd Razali, John Geraghty, et al. 2011. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering*, Vol. 2. International Association of Engineers Hong Kong, 1–6.

[12] Dennis G Wilson, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. Evolving simple programs for playing Atari games. (2018). arXiv:cs.NE/1806.05695