VU Amsterdam

Evolutionary Computing Task 2: Generalist Agent

Selection strategies and their performance for a multi-modal problem

Group 47:
Gijs Vermeulen (**2677769**)
Max van der Veer (**2666645**)
Gidon Quint (**2680204**)
Pelletreau-Duris Tom (**2774212**)
Mick Roumen (**2626348**)

October 20, 2023

Abstract

In the field of evolutionary computing the selection mechanisms play an important role. This paper explores the various ways of merging parents and their offspring before applying selection to this combined set. In this study, the EvoMan game framework is used as the test bed for the designed Evolutionary Algorithms (EAs), utilizing both a static training set and an adaptive training set. Optimising their parameter values using two SPO-based searches, one that includes crossover rates and combines parents and offspring before selection ($\mu + \lambda$), and one that uses a crossover rate of 1 and only selects from the offspring set(μ, λ). Testing the optimised EAs on a static and adaptive training set allows us to reach conclusions about the selection strategies in regard to combining parents and their offspring.

Introduction

In the field of Computational Intelligence in games, evolving efficient agents for multi-modal game spaces remains very challenging [1]. In such contexts, reaching a global optimum becomes increasingly more difficult as the essential strategies to learn to get more complex or when there are multiple objectives. A robust EA is then defined by its capability to generalise to such a multi-objective problem (MOP) and reach the Pareto front for the different objectives. Such an EA should strike the right balance between exploration and exploitation.

This study focuses on evolving a Neural Network that controls a generalist agent playing the 2D game EvoMan [1]. By considering two EAs with different selection strategies, our goal is to quantify their effect of exploration and exploitation induced by their difference in the selection of offspring. The first EA, favouring exploration as a form of elitism, utilises the (μ, λ) strategy, where selected individuals from the offspring (λ) replace the previous population (μ). The second EA, favouring exploitation, uses the $(\mu + \lambda)$ strategy, which selects the surviving individuals from a combination of children and parents. The expectation is that (μ, λ) will outperform the other method due to its increased capability to forget and hence get out of local optima easier.

Methods

The neural network controller, containing 10 neurons and 1 hidden layer, is represented as an array containing all the weights and biases of the network. The output node bias for shooting is initialised such that continuous shooting is guaranteed.

The two different EAs are compared for both an adaptive and static training set. The adaptive strategy works by checking for enemies in the training set for which the gain is bigger than a given threshold and switches them with a non-beaten enemy. To define the training and test set for the two EAs, the following section characterizes the specialist agent knowledge transfer, quantifying its performance when trained on one enemy and fighting against another. This helps us select a diverse training set and generalize better. Both EAs use a combination of N-point crossover and self-adaptive Gaussian mutation with one step size.

Transfer learning study By training and testing all 64 combinations of enemies, we produced Figure 1. It suggests that in general 3,4 and 6 are harder to beat while 4 and 8 have interesting generalisation possibilities. Since the goal is to optimize the breadth and diversity, the combinatorial approach, leads us to the training set: 1,3,7,8 as it could benefit from a transfer learning on 1,2,3,5,7,8 which is the widest combination. We decided to do the adaptive training initialised with 1, 4, 6, and 7, as they all have unique strategies to be learned in order to win [2].

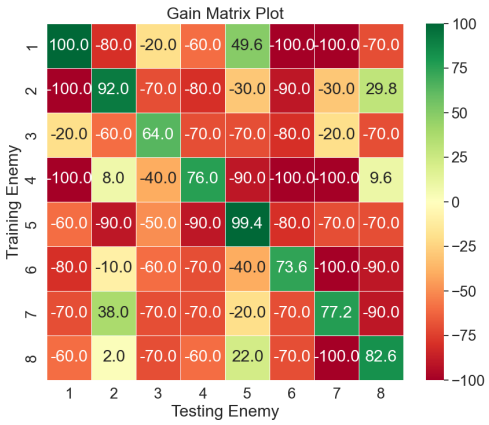

Figure 1: *Comparison of Gains for Evolutionary Algorithms train and test on different enemies*

Fitness function Since the optimisation problem at hand is multi-objective, a common approach to reducing it to a single-objective is to scalarize the fitness function [3]. Different methods include a weighted sum or the Tchebycheff Method [3]. It is desirable that the fitness score follows a smooth path since bigger steps could lead to the EA getting stuck. Regarding player and enemy life we found that optimal results were obtained when using an equal consideration of both (alpha=0.5) by tuning alpha in $\alpha * (100 - enemylife) + (1 - \alpha) * playerlife - log(time)$. However, only considering a weighted average of this gain value, showed to be too volatile. Therefore,

we added the duration of the fights, only including it when an enemy is not defeated. The idea is that surviving longer is only beneficial when you're not yet winning. Furthermore, the weights in the sum of the gain values are set to 1 when the corresponding enemy is not beaten and 0.1 when it is. This strategy allows for the prioritization to optimise on not-yet-beaten enemies. We define our aggregated fitness function as :

$$Fitness = \sum_{i \in E} w_i * gain_i + 2 * \sum_{i \in E} \frac{time_i}{max(time)}$$

Here E is defined as current training set and $time_i = max(time)$ if $gain_i > 0$. The final performance measure for both EAs is this fitness evaluation applied to the entire enemy set.

Selection strategies For selection, we used tournament selection since it has less stochastic noise and a controllable selection pressure [4]. The size of the tournament and its exponential increase factor are both controllable parameters. The tournament increase parameter determines the multiplication of the initial value in the latest generations. Tournament selection was done without replacement. In each generation, we generated $6 * population$ children and we only evaluate them once selected in a tournament after which we save the fitness values for possible future tournaments to decrease computation time.

For (μ, λ) only children are considered in the selection, and for $(\mu + \lambda)$ the 100 parents of the previous generation are added to the children before selection. Additionally, the crossover-rate parameter determines how many children are generated with cross-over and how many children are generated by copying the parents and then mutating. As discussed in the introduction, (μ, λ) is expected to outperform $(\mu + \lambda)$ since it is able to forget and therefore better at leaving local optima [5]. Furthermore, excluding the parents from selection also prevents bad $\sigma - values$ from surviving if their parent is very fit. After the last generation, we compute the fitness by playing all enemies, and not the previously selected ones, in order to select the network that generalizes the best.

Parameter optimization The parameter values used are noted in Table 1, where the ranges correspond to the SPO search. These are tuned with Sequential parameter optimization (SPO). In SPO, 10 samples are taken for each parameter range. Using 8 cores, the EvoMan EA runs 8 times, returning average best fitness values. Using all the information up until the current generation a Gaussian model is fitted determining which values to pick next, based on the most promising combinations and areas where there has been very little exploration. In total 10 samples are taken. Using SPO we tuned the mutation rate, the number of cross-overs, the initial tournament size and the rate at which the tournament size increased. For the static set, we also included the crossover rate. The tournament size increases in an exponential manner. The parameters and their ranges are denoted in table 1.

Baseline We initialized 8,000 individuals, derived from 40 generations with 200 individuals each, for the baseline performance.

Parameter	Value (EA)
Population	200
Generations	40
Hidden neurons	10
Elitism	2
Enemy gain	10
Children	600
Init. tour. size	[1, 4]
Tour. increase	[1, 10]
Mutation	[0.01, 0.5]
Crossovers	[2, 10]
Cross. rate	1 & [0.2, 0.8]

Table 1: *Default parameter settings and ranges used by SPO for sampling.*

Results and Discussion

We first conducted SPO on our two EAs in order to find the best parameters.

Param.	SPO (μ, λ)	SPO $(\mu + \lambda)$
Tour. size	4	1
Tour. increase	2	10
Mutation rate	0.05	0.5
# Crossovers	2	10
Cross. rate	NA	0.8

Table 2: *Resulting parameter values from SPO tuning for adaptive methods.*

We can observe that tournament selection parameters are already quite different, the (μ, λ) starts slightly higher, which shows it skews towards selection a little more as low k values are used in most generations as k increases exponentially. The mutation rate in $(\mu + \lambda)$ is 10 times the value of the other. This could be explained by the fact that the $(\mu + \lambda)$ EA needs a high mutation rate to maintain variation while (μ, λ) has this attribute inherently. The same principle is true for the cross-over rate which is a factor 5 times smaller in (μ, λ). Interestingly, the cross-over rate in $(\mu + \lambda)$ is tuned to 0.8 which is the end of the given range, this would indicate that generating more children through cross-over is better which indicates the (μ, λ) cross-over rate of 1 is probably most favourable. It would be interesting to look

into tuning the full range in future research to test this hypothesis. This would also be an argument in favour of (μ, λ) EA performing better, which is not what we have found.

We then conducted the comparative analysis between our two EAs and our static and dynamic set of enemies to characterise the role of keeping parents in the reproduction process and thus have more insights into the exploitation/exploration trade-off. It turned out the best performing EA was $(\mu + \lambda)$ on the static enemy set as can be seen in the two figures below.

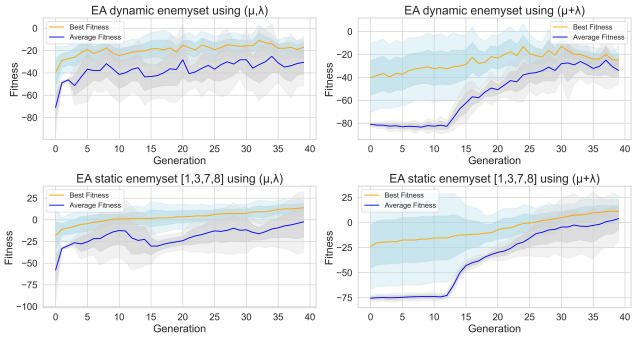

Figure 2: *Line-plot across the generations, with the average best fitness (orange) and the average mean fitness (blue), σ is indicated by the dark cloud and 2σ by the light cloud.*

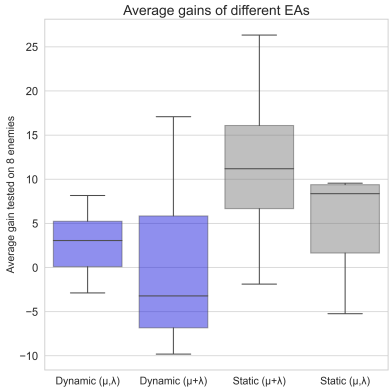

Figure 3: *Comparison of Average Gains for the different EAs with static enemy set and the dynamic enemyset*

This shows that conservative exploitation is a better setting for solving this MOP. In contrast to our hypothesis, the $(\mu + \lambda)$ EA shows better results. The effect can potentially be ascribed to a better balance of exploitation and exploration in $(\mu + \lambda)$. As shown in the plateau of the (μ, λ) results in figure 2, replacing the whole population might prevent optimization once a very good local optimum is found. On the contrary, being able to select well-performing parents from the previous generation does allow for exploited exploration even if the beginning has more trouble increasing (low initial tournament size, high mutation rate, low mutation step sizes (self-adaptive)). But we see that around generation 12 the average goes up. This is explained by the k-tournament parameter increase.

Table 3: *T-test results for the EA using (μ, λ) strategy vs. the EA using $(\mu + \lambda)$ strategy. Both EAs are trained on the enemyset 1,3,7,8 and on a dynamic enemyset.*

Enemyset	T-statistic	P-value
Dynamic	0.923	0.375
1,3,7,8	0.919	0.370

Table 3 presents the results of a two-sided t-test. The reported p-values do not indicate a significant difference in performances, considering a significance level of 10%.

To ensure our EAs are effective we tested their performance against a random initialised population of 8000 individuals and we found the following results

$AverageEnemiesBeaten$: 0.61

$StandardDeviationofEnemiesBeaten$: 0.82

$MaximumEnemiesBeaten$: 4

which helps validate our EAs' performance.

Finally, we study the average energy points of our best solutions on the table 4. We can see it beats 5 out of 8 enemies.

Enemy	1	2	3	4	5	6	7	8
Player energy	100	70	0	0	77.2	0	88.6	44.8
Enemy energy	0.	0.	20.	70.	0.	80.	0.	0

Table 4: *Average energy points retained by EvoMan of 5 runs against all 8 enemies.*

Using scalarization for MOP is only a valid concession when the tasks do not compete [6]. Therefore, future research methods could implement NSGA-II which is able to optimise multiple objectives at once without the need for scalarization [7].

Conclusions

In contrast to our initial hypothesis, the $(\mu + \lambda)$ EA shows better results. The effect can potentially be ascribed to a better balance of exploitation and exploration in $(\mu + \lambda)$. Indeed, following [8] and [9], evolution converges to an optimized solution when all the ingredients are here, reproduction, mutation, and selection. Wide exploration allows more novelty but conservative selection mechanisms are necessary in order to converge to more generalization abilities in a context of limited resources.

Finally, future research could focus on non-scalarization fitness aggregation methods on the first hand and study the effect of adding more generations on the second hand while training on all 8 enemies to generalize a maximum.

References

[1] K. da S. M. de Araújo and F. O. de França. "An electronic-game framework for evaluating coevolutionary algorithms". In: *arXiv* (2016), arXiv:1604.00644. URL: http://arxiv.org/abs/1604.00644.

[2] K Da Silva Miras De Araujo and FO De Franca. "Evolving a generalized strategy for an action-platformer video game framework". In: *2016 IEEE Congress on Evolutionary Computation (CEC)* (2016), pp. 1303–1310. DOI: 10.1109/CEC.2016.7743938.

[3] Panos M. Pardalos, Antanas Žilinskas, and Julius Žilinskas. In: *Non-Convex Multi-Objective Optimization*. Springer International Publishing, 2017.

[4] David E. Goldberg and Kalyanmoy Deb. "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms". In: ed. by GREGORY J.E. RAWLINS. Vol. 1. Foundations of Genetic Algorithms. Elsevier, 1991, pp. 69–93.

[5] J.E. Smith A.E. Eiben. *Introduction to Evolutionary Computing*. Springer Berlin, Heidelberg, 2015.

[6] Koltun V. Sener O. "Multi-Task Learning as Multi-Objective Optimization". In: (2018).

[7] "Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review". In: *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Ed. by Arun Kumar Sangaiah, Michael Sheng, and Zhiyong Zhang. Academic Press, 2018, pp. 185–231.

[8] D. C. Dennett. "Darwin's Dangerous Idea". In: *The Sciences* 35.3 (1995), pp. 34–40. DOI: 10.1002/j.2326-1951.1995.tb03633.x.

[9] A. E. Eiben and J. Smith. "From evolutionary computation to the evolution of things". In: *Nature* 521.7553 (2015), pp. 476–482. DOI: 10.1038/nature14544.