Is less more? Comparing Scalarization and Pareto Approaches in a Multi-Objective Game Problem

Evolutionary Computing | Standard Assignment 2: Generalist Agent | Team: 112 | October 20, 2024

Mikel Blom
Vrije Universiteit Amsterdam
2867792

Alexandra Genis
Vrije Universiteit Amsterdam
2834714

Daimy van Loo
Vrije Universiteit Amsterdam
2852535

Martino Zaratin
Vrije Universiteit Amsterdam
2860138

ABSTRACT

The optimization of autonomous game agents for a single objective is a well-researched topic. Yet, many problems have more than one objective. Such multi-objective problems (MOPs) are receiving increased interest from the optimisation community. We compared two popular approaches for solving such problems, Scalarization and Pareto in a game context. Based on prior research, we expected Pareto algorithms to perform better than Scalarization algotithms. However, we found no difference in performance for shorter training runs. Yet, results showcase a better robustness and a lower rate of premature convergence for Pareto. Additionally, we obtain some exploratory indications of a better performance for Pareto for longer training runs, suggesting this approach might be a better choice if longer training is feasible. Finally, we provide directions for future research to further explore these promising qualities of Pareto algorithms.

1 INTRODUCTION

The creation of autonomous game agents through Evolutionary Algorithms (EAs) is an important driver of Artificial Intelligence (AI). The goal of EAs is the optimization of generated solutions with respect to an objective function. However, many optimisation problems have not one but multiple and often conflicting objectives. These so-called multi-objective problems (MOPs) have been receiving increasing interest from the scientific community in recent years [6, p. 195]. To address MOPs, Scalarization and Pareto stand out as the most popular approaches, while fundamentally differing in how they handle these problems [7, p. 4].

In Scalarization, the algorithm optimises an individual fitness function for each objective and then combines them into a single fitness, usually through averaging or weighting [6, p. 196], systematically transforming the MOP into a series of single objective problems [8, p. 28]. This method is straightforward, yet, there is no opportunity to control the trade-offs between objectives among other problems [6, p. 196].

On the other hand, the Pareto approach compares the fitnesses of a solution for all objectives simultaneously, identifying solutions at the trade-off between objectives (Pareto Front) and selecting them for inclusion in the next generation [6, p. 197]. This process offers better control of performance on different objectives but requires more computations.

1.1 Related Work

Prior research was conducted on both methods. Miras compared multiple common EAs on their performance in training an autonomous game agent, using a Scalarized approach [3]. While this approach was successful, it is unclear whether a Pareto approach would have obtained better results. Prior research found that Pareto can perform better than Scalarization on the Knapsack Problem and Travelling Salesman Problem with two objectives [9, p. 21]. Further, Pareto algorithms were found to generalize better than Scalarization algorithms in the game Ms Pac-Man using two competing objectives [12, p. 6]. Interestingly, Pareto has been found to outperform Scalarization in some aspects of problems that are usually thought of as single-objective [4, p. 15].

While these authors provide some evidence for benefits of Pareto in games, they investigate problems with two objectives. Accordingly, it is still unclear how Scalarization and Pareto algorithms will perform on problems with a higher number of competing objectives.

To fill this gap, we compare Scalarization and Pareto for solving a problem with more than two objectives in a game context.

Based on prior literature, we expect the Pareto approach to improve performance compared to the Scalarization approach. To facilitate this comparison, we developed two algorithms using Scalarization and Pareto respectively, and compared their performance on training an autonomous agent in the benchmarking game EvoMan, using two different groups of objectives, implying the training of a generalist agent.

2 METHODS

The present study compares a Scalarization (SCAL) and a Pareto (PAR) algorithm in training an generalist autonomous agent in the game EvoMan, on a group of three and four objectives.

2.0.1 Implementation Details. Code written for the present study used Python Version 3.14 [10], the pymoo package [2], and the EvoMan game environment [1]. EvoMan is a Computational Intelligence game framework for the development of autonomous agents. The algorithms employed by the present study evolve the framework-internal controller neural network (NN) with 10 hidden nodes.

2.1 U-NSGA-III Algorithms

We adapted the U-NSGA-III algorithm from [2], which is particularly suitable for our research question as it is designed for the implementation of both single and multi-objective methods [11]. Package-specific default values were used for hyperparameters and are explained in the sections below.

We built and trained two algorithms on groups of three and four enemies, resulting in: SCAL3 and SCAL4, using Scalarization, and PAR3 and PAR4, employing Pareto. The number of each setup indicates the enemy group used in training. Enemy groups are discussed in the Objectives section below.

A population size of n = 100 was adapted from [3], leaving room for diversity for the variation operators [11, p. 361]. Further, each model was trained for 200 generations, in anticipation of longer time to convergence for PAR, due its complexity, while balancing computational cost.

Notably, U-NSGA-III structures the population along predefined Reference Directions, a process called Reference-Directions-based Niching, which is aimed at increasing diversity and is one of the major advances of U-NSGA-III compared to its predecessors [11, p. 395]. More information on Niching can be found in [6, p. 91, p. 195]. The present study employed one Reference Direction for SCAL and 10 Reference Directions for PAR.

The algorithms have a largely shared setup but differ in their handling of evaluation and selection. Accordingly, the sections below describe the common components in each algorithm buildup, differentiating between algorithms where necessary, and finally elaborate on the two evaluation approaches separately.

2.1.1 Parent Selection. Parents were selected using Niching-based, Binary Tournament Selection. First, non-dominated sorting was applied. This includes the calculation of dominance for all solutions, meaning that each solution is ranked based on how many other solutions obtained a higher fitness on at least one objective. The less dominated a solution is, the lower rank it is assigned, with lower ranks being more preferable. Candidates that are not dominated by any other solution (non-dominated) are given the lowest rank, corresponding to 0, and represent the Pareto Front.

Second, two parent candidates are randomly selected and compete for the parent role based on the following criteria: a) if parent candidates share the niche but have different ranks, the lower-ranked candidate is chosen, b) if candidates share both niche and rank, the candidate with a lower distance to its niche is chosen, and c) if candidates are located in different niches, the parent is selected randomly. As SCAL includes only one Reference Direction, all individuals are automatically placed in the same niche and one solution obtains the lowest rank. As a consequence b) is always applied.

Two tournaments are performed to create two parent solutions, to which then recombination is applied, resulting in two children. Tournaments are repeated until the number of offspring reaches the population size n.

2.1.2 Recombination. We employed Simulated Binary Crossover (SBX) with $\eta = 30$ and $probability = 1.0$, a method translating the working principle of One-Point Crossover from bit-strings to real-valued vectors. This process is facilitated through a probability distribution function parameterized by η [5, p. 1187], which controls the amount of possible difference between children and parents, with higher values resulting in higher similarity between children and parents (exploitation) and lower values resulting in lower similarity (exploration). The probability parameter determines how likely crossover is to happen, $probability = 1.0$ implying that crossover takes place for all parent pairs.

2.1.3 Mutation. Polynomial mutation with $\eta = 20$ was applied. This variation of non-uniform mutation [6, p. 57] uses a polynomial probability distribution in place of the Gaussian distribution to define mutations [5, p. 1188]. The degree of mutation is parameterized by η_m, with larger values producing smaller mutations and smaller values producing larger mutations, similarly to SBX.

2.1.4 Population management. We employed $(\mu + \lambda)$ selection [11, p. 361], which creates a union P of all individuals in parent and offspring populations. Subsequently, non-dominated sorting is applied to this union in the same way as is used for parent selection. This results in ranked individuals. The new population is generated with an Elitist approach, namely by replacing the whole parent generation with a rank-based selection p_{best} of individuals from P.

2.1.5 Fitness Function. The standard fitness function of the EvoMan framework $fitness = \gamma(100 - e_e) + \alpha e_p - \log(t)$ with $\gamma = 0.9$ and $\alpha = 0.1$ was adapted for evaluation [1] . Fitness comprises a composite fitness score based on player energy points, enemy energy points, and the time it took to finish the game.

2.1.6 Scalarized Evaluation. For SCAL, the EvoMan environment was set to 'multi' mode, as applied in [3]. This means that for each evaluation, the algorithm plays all specified enemies sequentially and calculates fitnesses individually. Then the mean $\bar{f}$ and standard deviation σ are calculated for this sample, and ultimately a single group fitness is calculated as: $f' = \bar{f} - \sigma$.

2.1.7 Pareto Evaluation. In PAR, the algorithm plays all enemies sequentially, storing all resulting fitnesses of the solution per enemy in a vector f'. Note that in the EvoMan environment, this can only be facilitated by setting the enemy mode to 'single' and then playing enemies separately as 'multi' performs Scalarization automatically. Based on the information in the individual fitness scores, the algorithm performs non-dominated sorting using the Pareto Front as specified in sections Parent Selection and Population Management.

2.1.8 Objectives. EvoMan features eight enemies in total, which represent the objectives in the present study. A full list of enemies can be found in [1]. As our research question involves more than two objectives, we selected groups of three and four enemies for training. This decision was based on balancing computational feasibility with the interest of comparing possible improvements in generalizability resulting from inclusion of more objectives.

To obtain the best enemies for training, we created sets of all possible groups of three and four enemies, trained SCAL once on each group, and tested the best solutions on all eight enemies. For each set, we selected the enemy group that had led to the solution with the highest individual gain sum during testing. Note that a SCAL was employed for this due to being less computationally costly than Pareto.

This comparison yielded the enemy groups (1, 2, 7) and (1, 2, 3, 8) to be the most promising for training a generalized agent.

2.2 Experiments

The performance of SCAL3, SCAL4, PAR3, and PAR4 in training a generalist controller NN was compared in an experimental setup.

2.2.1 Algorithm Training. Each algorithm was trained 10 times on each enemy group. Across these 10 runs, the mean and maximum fitness of the population of generated solutions were compared in a line plot. We also stored the best final solution generated by each algorithm setup, resulting in 40 final solutions.

2.2.2 Algorithm Testing. For testing, the 40 final solutions were used to play once against all eight enemies. Subsequently, we compared the results of all setups on the individual gain metric, which is the difference between player energy points (e_p) and enemy energy points (e_e): $g = \sum_{i=1}^{8}(e_{pi} - e_{ei})$.

2.2.3 Statistical Tests. Due to a sample size of 10 runs per algorithm, we compared the final individual gain of both algorithms and enemy groups with a Kruskal-Wallis test, which is the non-parametric equivalent of ANOVA. Post-hoc analyses were conducted with a Mann-Whitney test.

3 RESULTS

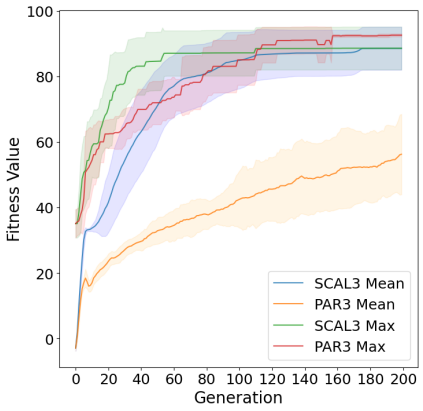
3.0.1 Training. Results for SCAL3 and PAR3 as well as SCAL4 and PAR4 are presented in panels a) and b) of Figure 1 respectively.

PAR3 obtained a lower mean fitness than SCAL3 and a similar max fitness as SCAL3. Further the mean and max fitnesses of SCAL3 are very similar, while for PAR3, max fitness is much higher than its mean fitness.

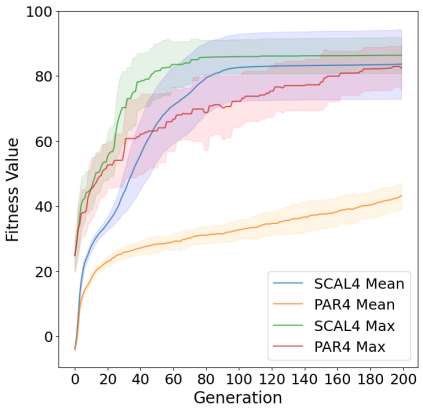
PAR4 obtained a much lower mean fitness than SCAL4 and a similar max fitness as SCAL4. Again, the mean and max fitnesses of SCAL4 are very similar, while for PAR4, max fitness is much higher than its mean fitness.

The similarity of mean and max fitnesses for SCAL3 and SCAL4 is indicative of low population diversity, which leads to premature convergence of these models. Additionally, the lower mean fitness of PAR3 and PAR4 is unsurprising, as these models have a higher complexity and thus optimisation takes longer than in SCAL3 and SCAL4. The shape of their curves further indicates that PAR3 and PAR4 are not fully converged yet.

We were interested in exploring this aspect and trained both SCAL and PAR on all eight enemies for 2000 generations. PAR8 obtained a solution that was able to consistently beat all eight enemies, while SCAL8 was terminated early after 640 generations



(a) Enemies [1,2,7]



(b) Enemies [1,2,3,8]

Figure 1: Mean and Maximum Fitness against each group of enemies

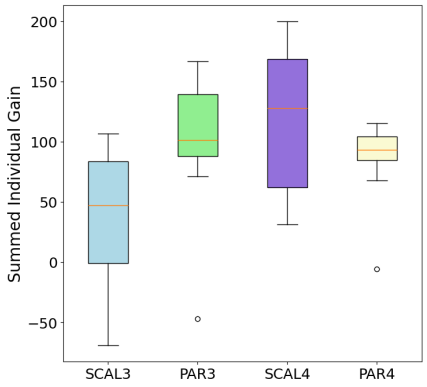


Figure 2: Summed Individual Gain for both EAs on the two groups of enemies

due to premature convergence. The final results of this setup can be found in Appendix A.

3.0.2 Testing. Results for testing are presented in Figure 2. SCAL4 obtained the highest individual gain sum on all eight enemies (Mdn=128, IQR=107), followed by PAR3 (Mdn=102, IQR=51.2), PAR4 (Mdn=93.3, IQR=19.5), and SCAL3 (Mdn=47.1, IQR=84.5). The Interquartile Range (IQR) for SCAL3 and SCAL4 is very large, indicating that results are less robust than for PAR3 and PAR4. Additionally, both algorithms and their setups have overlapping IQRs, suggesting that their differences could be insignificant.

In terms of generalizability, PAR3, PAR4, and SCAL4 were able to beat five of the eight enemies at least once during testing. SCAL3 was only able to beat 4 enemies. The very best individual was obtained by SCAL4. Table 1 displays the player and enemy life of this individual after testing against each enemy.

3.0.3 Statistical Tests. The Kruskal-Wallis test indicated a difference in at least two of the means ($p < 0.05$). Yet, post hoc tests revealed no significant differences in the means of any pair of algorithms (all $p > 0.05$). This indicates that the difference between the groups is statistically not large enough for significance after adjustment for multiple comparisons. Full statistical results can be found in Appendix B.

Enemy	1	2	3	4	5	6	7	8
Player Life	100	80	78	0	78	0	0	63
Enemy Life	0	0	0	70	0	100	30	0

Table 1: Player and Enemy life of the very best individual from all 40 training runs, obtained from SCAL4

4 ANALYSIS AND DISCUSSION

The present study compared the performance of Scalarization and Pareto approaches on problems with more than two objectives. Two algorithms were developed, using a Scalarization and a Pareto approach respectively, and both trained on groups of three and four objectives. We expected PAR to perform better than SCAL with the same number of objectives. Results revealed that SCAL and PAR perform equally well on problems with more than two objectives. However, PAR appears to showcase better robustness than SCAL.

This result can be explained by the number of generations used in this experiment, which appears insufficient for convergence of PAR due to its complexity. It is likely that PAR would overtake SCAL on performance after a higher number of generations, especially considering the low diversity in SCAL. The exploratory results, which compared the two approaches for more generations, additionally point in this direction. Thus, further research could replicate the present study with a larger number of generations.

Similarly, experiments were performed with only 10 runs due to resource considerations. Accordingly, statistical test results need to be handled carefully. Further research is encouraged to reproduce this experiment using more experimental runs, to ensure any differences present can be detected.

Another limitation is the process of enemy group selection for this study, during which algorithm performance was tested only once on each enemy group. Results of these test are therefore likely not robust. Further studies could employ a more rigorous process for selecting enemy groups by testing for multiple times, for example 20.

5 CONCLUSIONS

Multi-objective problems (MOPs) receive much attention in the EA community, as problems often require optimisation on multiple objectives. This study compared two common techniques for handling MOPs, Scalarization and Pareto, on their performance on a problem with more than two objectives. We expected PAR to improve performance. Contrary to the expectations, there appears to be no difference between SCAL and PAR for smaller numbers of generations. Yet, exploratory results suggest the possibility of improved performance for PAR for longer training runs, and future researchers are encouraged to investigate this promising avenue. Additionally, the interesting insights about the higher robustness of the Pareto approach, could be useful to research and industry when selecting either of these methods for their applications.

A EXPLORATORY RESULTS

Enemy	1	2	3	4	5	6	7	8
Player Life	100	70	46	44	91	32	91	67
Enemy Life	0	0	0	0	0	0	0	0

Table 2: Player and Enemy life of best individual, obtained from PAR8

Enemy	1	2	3	4	5	6	7	8
Player Life	0	70	8	0	65	0	0	6
Enemy Life	60	0	0	60	0	10	10	0

Table 3: Player and Enemy life of best individual, obtained from SCAL8

B COMPLETE STATISTICAL TESTS

χ^2	df	p
9.8	3	0.020

Table 4: Krushkal-Wallis rank sum test

Comparisons	PAR3	PAR4	SCAL3
PAR4	0.423	-	-
SCAL3	0.056	0.068	-
SCAL4	0.481	0.215	0.056

Table 5: P-values of pairwise comparisons using Wilcoxon rank sum test

REFERENCES

[1] Karine da Silva Miras de Araújo and Fabrício Olivetti de França. 2016. An electronic-game framework for evaluating coevolutionary algorithms. (April 2016). http://arxiv.org/abs/1604.00644 arXiv:1604.00644 [cs].

[2] J. Blank and K. Deb. 2020. pymoo: Multi-Objective Optimization in Python. *IEEE Access* 8 (2020), 89497–89509.

[3] Karine Da Silva Miras De Araujo and Fabricio Olivetti De Franca. 2016. Evolving a generalized strategy for an action-platformer video game framework. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Vancouver, BC, Canada, 1303–1310. https://doi.org/10.1109/CEC.2016.7743938

[4] Kalyanmoy Deb. [n. d.]. Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction. ([n. d.]).

[5] Kalyan Deb, Karthik Sindhya, and Tatsuya Okabe. 2007. Self-adaptive simulated binary crossover for real-parameter optimization. 1187–1194. https://doi.org/10.1145/1276958.1277190

[6] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44874-8

[7] Nyoman Gunantara. 2018. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering* 5, 1 (Jan. 2018), 1502242. https://doi.org/10.1080/23311916.2018.1502242

[8] Stephan Helfrich, Arne Herzel, Stefan Ruzika, and Clemens Thielen. 2024. Using scalarizations for the approximation of multiobjective optimization problems: towards a general theory. *Mathematical Methods of Operations Research* 100, 1 (Aug. 2024), 27–63. https://doi.org/10.1007/s00186-023-00823-2

[9] Mohammed Mahrach, Gara Miranda, Coromoto León, and Eduardo Segredo. 2020. Comparison between Single and Multi-Objective Evolutionary Algorithms to Solve the Knapsack Problem and the Travelling Salesman Problem. *Mathematics* 8, 11 (Nov. 2020), 2018. https://doi.org/10.3390/math8112018

[10] Python Software Foundation. 2023. Python Language Reference, version 3.11. https://www.python.org/. (2023). Accessed: 2024-09-28.

[11] Haitham Seada and Kalyanmoy Deb. 2016. A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives. *IEEE Transactions on Evolutionary Computation* 20, 3 (June 2016), 358–369. https://doi.org/10.1109/TEVC.2015.2459718

[12] Tse Guan Tan, Jason Teo, and Kim On Chin. 2013. Single- versus Multiobjective Optimization for Evolution of Neural Controllers in Ms. Pac-Man. *International Journal of Computer Games Technology* 2013 (2013), 1–7. https://doi.org/10.1155/2013/170914