Surrogate Assisted Multi-objective Optimisation Using Artificial Neural Networks

710046012

ABSTRACT

Traditional optimisation approaches quickly become overly complex and computationally infeasible as you increase the dimensionality of the optimisation problem; evolutionary algorithms provide a computationally feasible solution to these multi-objective optimisation problems. However, these problems can include computationally expensive function evaluations that significantly diminish the effectiveness of the algorithms. This report proposes an experimental method for overcoming the complex function evaluations by integrating machine learning into the optimisation process of the dominance-based multi-objective evolutionary algorithm, NSGA-II; resulting in a potential reduction in computation time compared to the standard algorithm. We will evaluate the surrogate-assisted NSGA-II algorithm, demonstrate its effectiveness, and discuss future work to improve it.

1 INTRODUCTION

Context and Motivation: As dimensionality increases, traditional optimisation approaches such as particle swarm quickly become overly complex, and the exponential increase in search space size can make them computationally infeasible [6]. Evolutionary algorithms (EA) provide a powerful method of finding solutions to problems that involve optimising multiple design variables in high dimensions. They use the principle of natural selection to iteratively select and mutate points to find a set of optimal solutions called the Pareto front; where an improvement in one objective will worsen the evaluation in another objective.

The evaluation of objectives in evolutionary algorithms can be a black box, meaning the problem is too complex, and it is infeasible to understand the computations occurring and how the computer is making the evaluations. Computationally expensive evaluations such as computational fluid dynamics, or finite element analysis can take a long time to perform the many function evaluations required by EA. Therefore we must explore methods of improving the efficiency of the function evaluations. A popular method of doing this involves a hybrid approach to EA that includes machine learning in the optimisation process; often predicting the output of the function evaluations, reducing the algorithm's computation time.

Objective of the Report: This report introduces the fundamentals of Multi-objective optimisation and explores surrogate-assisted evolutionary algorithms (SAEA), it showcases an experimental design using neural networks in combination with the dominance-based evolutionary algorithm, NSGA-II [3]. Furthermore, we analyse an implementation of the surrogate, assess its performance, and discuss potential improvements and future work in this field.

2 BACKGROUND

Multi-objective Optimisation: Multi-objective Optimisation problems (MOPs) involve optimising multiple objectives which are often conflicting. Solving MOPs aims to find optimal solutions that approximate the Pareto front, and therefore the Pareto set of solutions; where no objective can be improved without worsening another. Formally, a multi-objective optimisation problem (MOP) is denoted by:

$$\begin{aligned} &Minimise: \{f_1(x), f_2(x), ..., f_M(x)\} \\ &subject\ to: \\ &\quad x_i^L \leq x_i \leq x_i^U, \\ &\quad \forall i = 1, 2, ..., D \end{aligned} \tag{1}$$

Where $x = (x_1, x_2, ..., x_n)^T$, represents a decision vector in the decision space $\mathfrak{R}^n$ of n dimensions; the decision vector x is subject to the constraints $x_i^L \leq x_i \leq x_i^U$ that it must follow to be considered a feasible solution, and D denotes the number of decision vectors, and $f_i(x)$ represents the functions to be, producing a vector $y = (y_1, y_2, ...y_M)^T$ in the feasible objective space $\mathfrak{R}^M$.

Multi-objective evolutionary algorithms (MOEA) provide an effective approach to approximating the optimal solutions of a MOP; they employ a population-based approach, iteratively evolving solutions over generations using mechanisms inspired by natural selection. MOEAs are generally categorized into dominance-based, decomposition-based, and indicator-based approaches. Dominance-based algorithms focus on comparing solutions based on Pareto dominance, decomposition-based methods break down multi-objective problems into simpler sub-problems, and indicator-based algorithms use performance indicators to guide evolution [7]. Each approach has distinct advantages depending on the problem structure, though dominance-based MOEAs are particularly popular due to their ability to maintain diversity and accurately converge to the Pareto front. They use the property of dominance to evaluate points against each other, a point dominates another point if it is closer to the Pareto front in all objectives [7]. Formally, dominance can be described with:

- If $f_i(x) \leq f_i(x')\forall I$, and $f(x) \neq f(x')$, then x *dominates* x'; denoted $x \prec x'$.
- If $f(x) \not\prec f(x')$ and $f(x) \not\succ f(x')$, the x and x' are *incomparable* and mutually non-dominating.

NSGA-II:. Non-dominated Sorting Genetic Algorithm II (NSGA-II) is one of the most widely used dominance-based MOEAs, valued for its efficiency, accuracy, and ability to maintain a well-distributed set of solutions [3]. NSGA-II introduces an approach to rank solutions into hierarchical fronts based on Pareto dominance, with the top-ranked solutions forming the best approximation of the Pareto front at each generation. To preserve solution diversity, NSGA-II calculates a crowding distance, which helps spread solutions across the Pareto front by favouring points in sparsely populated areas. Elitism is also a critical feature in NSGA-II, making use of natural selection principles by ensuring that high-quality solutions are retained across generations, which improves convergence toward

the optimal front. These combined features make NSGA-II especially effective in solving complex, high-dimensional problems and have led to widespread use where multi-objective optimization is essential.

Surrogate Models and Neural Networks: Although evolutionary algorithms prove incredibly effective in finding a Pareto set for multiple objectives, they have often been criticised for their slow convergence and reliance on many, lengthy and computationally expensive function evaluations. For example, optimising the crashworthiness of car construction materials and design using finite element analysis to validate their results [10]. To mitigate this problem, surrogate models can be used in place of the complex function evaluations to speed up the algorithms. Common surrogate-assisted evolutionary algorithms (SAEA) methods include kriging, polynomial regression, and neural networks, each offering a way to capture the underlying structure of the black-box problem found in function evaluations [2]. By using these surrogates, SAEAs can evaluate large populations or generations quickly, reducing the time and computational resources needed to converge toward the Pareto front.

3 EXPERIMENTAL DESIGN

This section demonstrates a surrogate-assisted version of NSGA-II (SA-NSGA-II) using a regression-based machine learning perception (MLP) to evaluate decision space vectors instead of the computationally expensive function evaluations. An MLP is a type of feed-forward neural network that is densely packed, it learns to predict an output based on given inputs by updating weights and biases using a learning process called backpropagation [5].

Configuration of NSGA-II with MLP Surrogate: Throughout the execution of the algorithm, the MLP will need to be trained many, possibly thousands of times. To speed up this process, it is beneficial to make it as simple as possible to train. We initialised the MLP with 2 hidden layers, with 100 nodes at each layer, this still fits the model well, whilst being very quick to train, as there is less back-propagation to perform at each iteration. The 'adam' solver is used for the MLP, it is a stochastic gradient descent solver which converges very quickly, although potentially hampers model accuracy [5]. The MLP was initialised with the warm start process. This means the machine learning regressor reuses the previous call to fit at the start of the next iteration. As we are potentially training the same model thousands of times, it is beneficial to use a warm start as it ensures the previous iterations are not wasted.

At the start of the algorithm, a population of decision vectors is initialised using Latin Hypercubes Sampling (LHS) [9], these points are then evaluated using the computationally expensive function evaluations. This results in correct data which can be used to train the surrogate model for future evaluations.

Updating the surrogate model: To ensure that the model's accuracy does not diminish over time as the objective vectors change, the surrogate model will need to be updated. In this implementation, function evaluations are performed every 10 iterations although this hyper-parameter can be specified to user requirements. At each re-assessment, if the mean error of a real objective vector exceeds a specified error threshold ϵ in comparison with the surrogate models' predicted objective vectors, it is appended to the training set and the MLP is called to fit. A higher error rating means the MLP made a worse prediction and therefore this data point must take high priority and be used in the training set with its real function evaluation. The error is evaluated as follows:

$$Error = \left| \frac{f_r(x) - f_s(x')}{f_r(x)} \right| \tag{2}$$

Implementaion and Simulation Setup: See below for a pseudocode implementation of the algorithm. In this implementation, the algorithm was written in Python, using Pymoo, a Python package developed for Multi-objective Optimisation [1]. The MLP was implemented using the scikit-learn *MLPRegressor* object, which provides an effective implementation of a regression-based MLP [8]. The algorithm was benchmarked on multiple test problems, including DTLZ1, DTLZ2, DTLZ5, and DTLZ7 [4] using a range of variable and objective dimensions for each problem; where $n_{var} \in \{2, 5, 10, 20\}$, $n_{obj} \in \{3, 5, 10\}$, ensuring $n_{var} \geq n_{obj}$. The SAEA was run for 10,000 iterations, with a retrain interval of 10 iterations, and an error threshold of 0.1 for each problem. Finally, the implementation was executed on an Apple Mac-book Air with their ARM-based M3 chip, this information is provided for replication purposes.

Algorithm 1 Pseudocode for Surrogate-assisted NSGA-II

```
procedure SA-NSGA-II
Inputs
    FE_max, The number of complex function evaluations for the training set
    Gen_max, The maximum number of generations
    Pop, The size of the population in the algorithm
    R, The number of SAEA iterations before re-evaluating the surrogate
    ϵ, The error threshold
Returns
    F ← Set of non-dominated points
Start
    Create FE_max inputs points using LHS
    Evaluate these points using complex FE evaluations
    Train the MLP Regressor using the FE evaluations
    while Gen_i < Gen_max do
        Crowded tournament selection to select parents
        Create offspring using crossover and mutation
        f_s ← Surrogate function evaluations on offspring
        if Gen_i mod R = 0 then
            f_r ← Real function evaluations on offspring
            Error ← Calculate Error between f_r and f_s
            if Error ≥ ϵ then
                Add the points to training set
                Re-train the surrogate model
        Select Parents with NDS and Crowding Distance
        Gen_i ← Gen_i + 1
```

Evaluation Metrics: To evaluate the algorithm, we will assess the accuracy of the surrogate model using the R^2 metric every time it is called to fit. Furthermore, we will be using the hypervolume metric to evaluate how well the model converges toward the Pareto front, comparing the solution set calculated using the surrogate algorithm to a solution set calculated using standard NSGA-II. Furthermore, we can visualise how well the model fits the Pareto front as the Pareto front for the DTLZ problems is known.

4 RESULTS AND ANALYSIS

Analysis of Convergence: The size of the hypervolumes after 10,000 iterations for SA-NSGA-II and standard NSGA-II algorithms was analysed. Although the ideal nadir points are known for these benchmark problems, the reference point used to calculate the hypervolume was the estimated nadir point obtained from the SAEA iteration of the algorithm. This is because *hvwfg* and *hv3d* would not execute if there were points outside the bounds of the nadir point, which was sometimes the case with the SAEA. However, using the same reference point for the SA-NSGA-II and the standard run of NSGA-II should give correlating results. The SAEA failed to converge towards the Pareto front of DTLZ1 and is therefore not shown in Figure 2 as the hypervolume was too large. Otherwise, the figure shows that the algorithms converge to a similar hypervolume size. The SAEA was consistently larger than NSGA-II, however, as shown by Figure 3, SA-NSGA-II is shown converged to negative points, which were outside the constraints of the problem and caused the hypervolume to appear bigger; this is a flaw in the surrogate and indicates that the error threshold may be too high. More work must be done in the future to analyse this further. Convergence onto the Pareto front can be observed from Figures 4 and 5, showing the initial points and objective vectors after 10,000 iterations respectively. These figures show that potentially more generations would be needed to converge completely onto the Pareto front.

n_{var}	n_{obj}	DTLZ1	DTLZ2	DTLZ5	DTLZ7
5	3	0.801	0.984	0.978	0.943
10	3	0.934	0.992	0.983	0.985
10	5	0.952	0.993	0.985	0.936
20	3	0.965	0.994	0.992	0.995
20	5	0.975	0.991	0.991	0.963
20	10	0.879	0.982	0.983	0.572

Table 1: Accuracy of the MLP at iteration 10,000

Surrogate Accuracy: Overall, the accuracy of the surrogate is very good, achieving high R^2 scores at generation 10,000 for each problem and combination of variables and objectives with the exception of DTLZ1 where $n_{var} = 5$ and $n_{obj} = 3$ and DTLZ7 where $n_{var} = 20$ and $n_{obj} = 10$; however, this does not generally indicate a good convergence toward the Pareto optimal front. Figure 1 shows the accuracy of the DTLZ1 surrogate model for $n_{var} = 5$ and $n_{obj} = 3$, and at approximately generation 1800 the model accuracy becomes negative, this means the model is objectively worse for the data provided; although it climbs back up to 0.801%. Analysing the figure further, the data points are very dense, indicating that the model is trained upon very frequently and therefore lots of points are constantly exceeding the error threshold. This could be due to poor model accuracy, or the error threshold is too low and this hyper-parameter must be tuned.

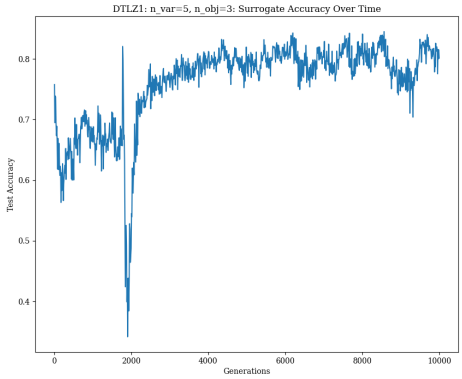

Figure 1: Surrogate accuracy improvement over time for DTLZ1 where $n_{var} = 5$ and $n_{obj} = 3$

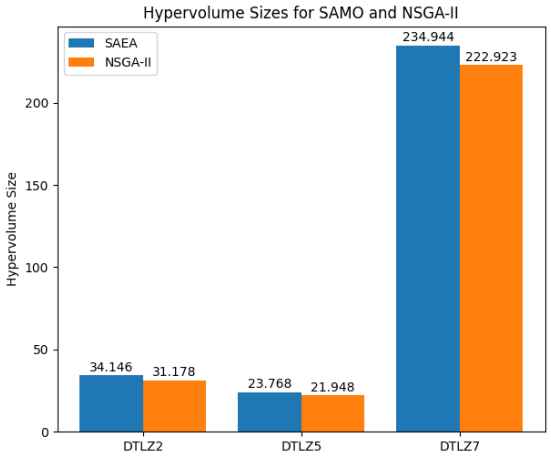

Figure 2: Hypervolume comparison for DTLZ2, 5, 7 where $n_{var} = 10$ and $n_{obj} = 5$

5 DISCUSSION AND FUTURE WORK

This report showcased the potential implementation of a surrogate model in place of computational function evaluations in NSGA-II.

Limitations. One limitation, as shown in Figure 6 is that the computation time for SA-NSGA-II exceeds that of standard NSGA-II, however, function evaluations for the DTLZ problems are very fast to solve, the benefit of computation time would be more felt with complex function evaluations in finite-element analysis or similar applications. Additionally, another limitation of SA-NSGA-II is its tendency to converge to objective vectors that were outside of the constraints of the problem itself, this is a critical flaw and more work should be done to mitigate this. This could be the result

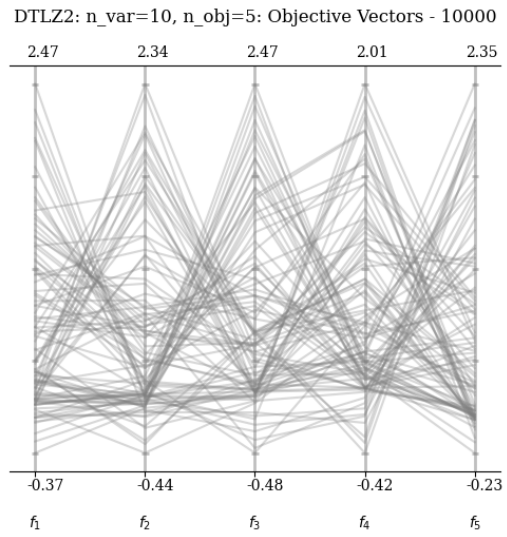

Figure 3: Objective vectors for SA-NSGA-II after 10,000 iterations for DTLZ2 where $n_{var} = 10$ and $n_{obj} = 5$

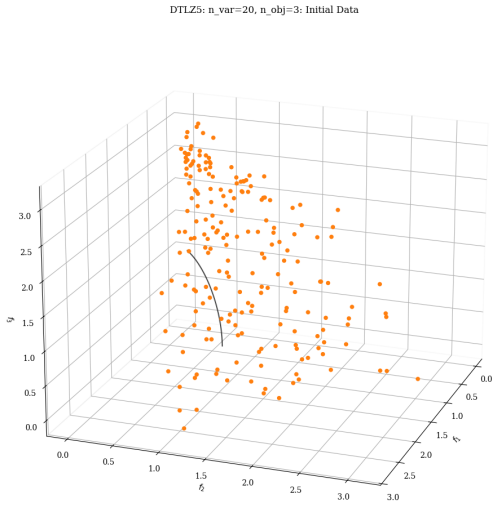

Figure 4: Inital Data provided to the SAMO algorithm for DTLZ5 where $n_{var} = 20$ and $n_{obj} = 3$

of the over-generalisation of the black-box model, as a single MLP attempts to predict the output of multiple function evaluations simultaneously. To mitigate this in the future, a separate model for each function and constraint could be trained, in a similar fashion to the kriging method [2]; although this will undoubtedly increase the computation time required, the models are more likely to converge and the objective vectors may be more likely to stay within the constraints of the problem.

Broader Impact. This report introduced a potential implementation of surrogate-assisted evolutionary algorithms using NSGA-II

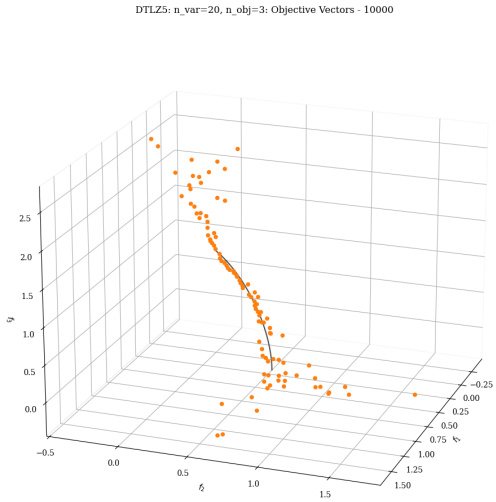

Figure 5: Objective vectors after 10,000 iterations for DTLZ5 where $n_{var} = 20$ and $n_{obj} = 3$

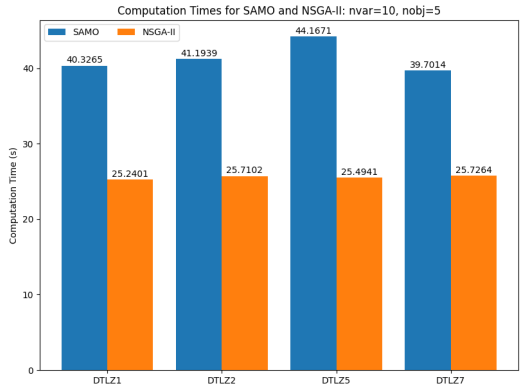

Figure 6: Comparison in computation times between the SAMO algorithm and standard NSGA-II for 10,000 generations where $n_{var} = 10$ and $n_{obj} = 5$

as a baseline model, it was shown to be mostly effective, although more research needs to be done to ensure its outputs remain within the bounds of the problem definition. Overall, SAEAs hold significant potential to revolutionise the industry by enabling faster, more efficient optimisation in fields that rely on high-complexity simulations and function evaluations such as aerospace, automotive, and energy [10]. SAEAs reduce the need for extensive computational resources, accelerating the optimisation process, and lowering costs. For industry, this means quicker design cycles, enhanced product performance, and the ability to optimize complex, multi-objective problems that were once too resource-intensive. Overall, SAEAs enable industries to tackle high-dimensional problems with greater precision and speed, transforming how sectors approach research, design, and decision-making.

REFERENCES

[1] Julian Blank. 2024. pymoo: Multi-objective Optimization in Python. https://pymoo.org

[2] Tinkle Chugh, Yaochu Jin, Kaisa Miettinen, Jussi Hakanen, and Karthik Sindhya. 2016. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation* 22, 1 (2016), 129–142.

[3] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 2002. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, Vol. 1. 825–830 vol.1. https://doi.org/10.1109/CEC.2002.1007032

[5] M.W Gardner and S.R Dorling. 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment* 32, 14 (1998), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

[6] Elre T. Oldewage. 2017. *The Perils of Particle Swarm Optimisation in High Dimensional Problem Spaces.* Ph. D. Dissertation. https://uoelibrary.idm.oclc.org/login?url=https://www.proquest.com/dissertations-theses/perils-particle-swarm-optimisation-high/docview/2911299317/se-2

[7] Luis V Santana-Quintero, Alfredo Arias Montano, and Carlos A Coello Coello. 2010. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. *Computational intelligence in expensive optimization problems* (2010), 29–59.

[8] scikit learn. 2024. MLPRegressor. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[9] Michael Stein. 1987. Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics* 29, 2 (1987), 143–151. https://doi.org/10.1080/00401706.1987.10488205

[10] Guangyong Sun, Tong Pang, Jianguang Fang, Guangyao Li, and Qing Li. 2017. Parameterization of criss-cross configurations for multiobjective crashworthiness optimization. *International Journal of Mechanical Sciences* 124-125 (2017), 145–157. https://doi.org/10.1016/j.ijmecsci.2017.02.027