Towards Explainable Metaheuristic: Mining Surrogate Fitness Models for Importance of Variables

Manjinder Singh
manjinder.singh1@stir.ac.uk
University of Stirling
Stirling, UK

Alexander E.I. Brownlee
alexander.brownlee@stir.ac.uk
University of Stirling
Stirling, UK

David Cairns
david.cairns@stir.ac.uk
University of Stirling
Stirling, UK

ABSTRACT

Metaheuristic search algorithms look for solutions that either maximise or minimise a set of objectives, such as cost or performance. However most real-world optimisation problems consist of nonlinear problems with complex constraints and conflicting objectives.

The process by which a GA arrives at a solution remains largely unexplained to the end-user. A poorly understood solution will dent the confidence a user has in the arrived at solution. We propose that investigation of the variables that strongly influence solution quality and their relationship would be a step toward providing an explanation of the near-optimal solution presented by a metaheuristic.

Through the use of four benchmark problems we use the population data generated by a Genetic Algorithm (GA) to train a surrogate model, and investigate the learning of the search space by the surrogate model. We compare what the surrogate has learned after being trained on population data generated after the first generation and contrast this with a surrogate model trained on the population data from all generations.

We show that the surrogate model picks out key characteristics of the problem as it is trained on population data from each generation. Through mining the surrogate model we can build a picture of the learning process of a GA, and thus an explanation of the solution presented by the GA. The aim being to build trust and confidence in the end-user about the solution presented by the GA, and encourage adoption of the model.

CCS CONCEPTS

• **Theory of computation → Models of learning**; *Theory of randomized search heuristics.*

KEYWORDS

genetic algorithms, explainability, interpretable, surrogate model, fitness function, optimization

ACM Reference Format:
Manjinder Singh, Alexander E.I. Brownlee, and David Cairns. 2022. Towards Explainable Metaheuristic: Mining Surrogate Fitness Models for Importance of Variables. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3520304.3533966

1 INTRODUCTION

The growing ubiquity of machine learning (ML) and Artificial Intelligence (AI) models [34] to automate decision making in our day-to-day life has brought with it ethics concerns[42], and a lack of trust from those users directly impacted by the automated decision making [2, 38]. The interest in explainability is not a new one [37], however the resurgence in interest is being driven by users demanding an explanation of *why* a particular decision was reached. This demand for an explanation has also been enshrined in European Union (EU) law when the right to explanation was included in the General Data Protection Regulation (GDPR) [21] as a recognition of the rise in importance of AI systems in automated decision making.

The majority of AI models in use today can be thought of as *"black-boxes"*, with many complex layers consisting of nonlinear transformations [27]. It can be difficult for end users to understand the decision making process and hence, already, trust in the final output is eroded. This has led to decisions being made that were biased [25], and some that have led to real harm being done to people due to a lack of transparency and explainability of these models [36].

A large area within AI is metaheuristic search algorithms applied to optimisation problems. These algorithms look for solutions that either maximise or minimise a set of objectives, such as cost or performance. However most real-world optimisation problems consist of nonlinear problems with complex constraints and conflicting objectives. These algorithms find applications in many areas, including optimisation of transportation for positive environmental impact [9], and healthcare [33].

Metaheuristic algorithms are not problem dependent, which means we can use them for a variety of optimisation problems. Many of these algorithms are inspired by nature. A Genetic Algorithm (GA) takes its inspiration from evolution. To find an optimal or near-optimal solution for a problem, the GA is used to optimise a problem by initially creating many random solutions, populations, and then iteratively updating these solutions through selection, mutation and crossover, to reach a *good enough* solution or the most optimal solution possible. A GA performs well in combinatorial optimisation where we have a large search space [1]. However, despite recent advances in theoretical understanding of GAs [29, 31], the process by which a GA arrives at a solution remains largely unexplained to the end-user. Herein lies the problem, firstly a poorly understood solution will dent the confidence a user has in the arrived at solution. Secondly during our problem definition we may have excluded some critical criteria. This may be of particular relevance in a scheduling optimisation problem, where we may focus on only a distance or financial cost constrain and so fail to take into account personal preferences and/or convenience for the end-user. Thirdly, a GA follows a random process, which can lead to noise in the presented solution [17]. In this case it would be useful to know the characteristics of this solution, and be able to tease out this noise without impacting the solution quality. Metaheuristic algorithms in general are adept at finding *shortcuts* in a problem definition [28]. Therefore it is useful to know whether the solution presented genuinely solves the problem or if the algorithm merely found a *loophole* in the problem definition. As demonstrated, for the Vehicle Routing Problem (VRP)[3], being able to explain the characteristics that constitute a near-optimal or sub-optimal solution also goes a long way towards designing a more efficient algorithm.

Taking all of the above points into consideration we can see that there are many points at which we can lose a user's trust when designing a system. This loss of trust would ultimately lead to lack of adoption of the system, and hence, wasted resources and suspicion of any systems recommended in the future.

In this paper we will outline an approach to identify the cardinal characteristics of metaheuristic derived solutions. We do this for a series of binary-encoded benchmark problems, by way of identifying which variables have the greatest influence on solution quality. Our approach will use *surrogate fitness functions* [11, 23]. Usually these are applied where the fitness function is costly; in parallel to the optimisation process we train a computationally cheap model, and the majority of calls to the costly fitness function are replaced by a call to the surrogate model. An additional benefit of this process is that the surrogate is an explicit representation of what the metaheuristic has learned about the search space and more crucially what comprises a *good* solution. In our study, we take a high fitness solution for each of the benchmark problems and probe it using the surrogate model to identify which variables are of greatest importance and to test their impact on fitness. The present paper extends the approach initially proposed in [40], by investigating the insights that might be gained at different stages of the algorithm's run, rather than just the final population.

In *Section 2* we will review related work around trust and explanation of metaheuristic optimisation, before going on to focus on the role of surrogate models in optimisation problems, and our use of them to explain metaheuristic algorithms. In *Section 3* we describe our approach to mining the surrogate model to explain solution quality. In *Section 4* we outline our approach to mining the surrogate model to extract information about variable importance of an optimal solution. In *Section 5* we demonstrate our methodology using four benchmark problems, and in *Section 6* we present our results and outline our conclusions and future work in *Section 8*.

2 RELATED WORK

Explainability in AI is not a new research topic, however, research conducted in the early 90's [37] tapered off for many decades. The recent growth and interest in this area is gaining momentum, mainly driven by a lack of trust and uptake by users of AI systems. The resurgence in this topic follows the thinking that making AI systems more transparent and explainable will engender end-user trust in the AI systems and hopefully lead to greater uptake and use of these systems [32].

There is, however, a gap in research centered around explainability of metaheuristic algorithms. The most relevant research within this area being innovization outlined in a paper by Deb et al. [14] [16], which proposed *"innovization"* to generate problem based knowledge alongside the normally generated near-optimal solutions, through identification of common principles among Pareto-optimal solutions for multi-objective optimisation problems. The impetus behind this being that most optimisation techniques adopted are used to surrender a single or small selection of optimal solutions, their research built on this to allow these optimisation techniques to find additional problem based knowledge in parallel to the generated optimal solution(s) via the innovization operation. By taking a generated set of *high-performing* trade-off solutions and identifying common principles hidden within them. The idea is that the main common thread amongst this set of *high-performing* solutions will represent properties that ensure Pareto-Optimality and by extension are valuable properties related to the problem globally.

More recently proposed by Urquhart et al. [39] is an application of Map-Elites to increase trust in metaheuristic algorithms. This paper aims to address the criticism that end-users have no role in the construction of the end solution. The authors applied Map-Elites, which allows for construction of a *high-performance* solution which is mapped onto a set of solutions defined by the user, such as cost or time. The solutions being generated by mutation and recombination, with each solution being assigned a *bin* within the solution space, with solutions that are assigned to an already occupied bin being either rejected or accepted based on a comparison of their relative fitness, with higher ranked fitness solutions being accepted, so only the highest fitness being assigned and retained within any particular *bin*. Ultimately the end-user is responsible for choosing the preferred solution, this ensures that input from the user is taken into account and they have a sense of ownership and a greater level of trust in the proposed solution. Map-Elites therefore provide a way to filter the solution space, and provides a set of solutions for the user, from which they can select the one most applicable to them and their needs. This has the benefit of increasing trust in the solution selected, because the user is provided an opening to the process and able to have some measure of influence as to what constitutes a *good* solution.

Gaier et al. [19] proposed a hybrid approach by using Map-Elites alongside a surrogate model to add efficiency to the Map-Elites process. They proposed reducing the need for the large number of checks normally required for Map-Elites. The proposed solution, Surrogate-assisted illumination (SAIL), aims to achieve this by way of integrating an approximation model (surrogate) alongside intelligent sampling of the fitness function. As with Map-Elites the search space is partitioned into *bins* each of which holds a map with a different layout of feature values. Firstly a surrogate is constructed based on an initial population of possible solutions including their fitness scores. Map-Elites is then used to produce solutions to maximise the fitness function, and generating an acquisition map. Thereafter, new solutions are sampled from this map and additional observations are used to iteratively improve the model. With the aim of looping through this process to generate increasingly better solutions with higher fitness functions. The performance predictions then being used by Map-Elites in place of the original fitness function to generate a prediction map of near optimal representations.

In contrast to the above approaches we do not aim to summarise or present a set of solutions, rather, we aim to explain a single solution chosen by the metaheuristic algorithm.

3 SURROGATE MODELS

Evolutionary Algorithms (EA) aim to minimise or maximise a given function *f(X)*. This is done by initially generating random solutions and evaluating them against a *fitness function*, and then generating new solutions, usually biased towards solutions with higher fitness values. The fitness function is often the most costly operation in the optimisation process, and to reduce this cost we can replace calls to the *fitness function* with calls to a much cheaper model. This cheaper model, *surrogate* [10, 22, 23, 40], also represents an explicit model of the population. We propose mining this model to capture the sensitivity of the *fitness function* to the problem variables. The premise being that the surrogate model is biased by the population of the EA, and therefore contains another view of the algorithms understanding of the problem.

As in [40], in the work presented here, the EA alternates between using the surrogate to evaluate solutions and the true fitness function. The surrogate model is re-trained each time new evaluations are carried out by the true fitness function, it is re-trained on the current and all previous populations at alternating intervals. So, we alternate between using the true fitness function and the surrogate at regular intervals. The surrogate is implemented by a Support Vector Regression (SVR) model, provided by scikit-learn (version 1.0.2). We use the default parameter settings for the SVR model. The model features are simply the problem variables X and the target for prediction is fitness *f(X)*. We hypothesise that the surrogate model will retain some of the key properties from the original fitness function, such as regions of high fitness. The surrogate model is able to evaluate the solution at a much cheaper cost compared to the real fitness function. The two benefits of this approach being that firstly the surrogate model can evaluate the potential solutions in a more timely manner and, secondly can give us an insight into the algorithms understanding of the search space.

The benefit of this approach is that the surrogate model, being an explicit model of the population, can be mined to gain insights into the learning of the EA of the search space. This approach has previously been demonstrated when looking at Estimation of Distribution Algorithms (EDAs), which sample probabilistic models after first constructing these models to reflect the distribution of high fitness solutions. It has been demonstrated that information gained from EDAs can in many cases offer additional insight into the problem as the solutions presented by an EA.[7, 8, 13]. The usefulness of the information gleaned from a surrogate model is of course closely coupled to the problem being analysed. The surrogate model could be inherently interpretable (such as a linear regression model similar to [7] or a decision tree), or we could exploit existing XAI approaches such as mining the model through probing. We make use of the latter approach in the present paper.

4 METHODOLOGY

To determine the rank and importance of individual variables for each of the benchmark problems, we will investigate the local sensitivity of the closest-to-optimal solution (*highest fitness*) found for each of the problems. We will take each of these *best solutions* and evaluate against our surrogate model, allowing us to determine the change in surrogate fitness due to mutating each variable in the solution. We will also set the sign of the *"importance"* measure to be negative if the corresponding bit in the starting seed solution was 1: providing an indication of the direction of the relationship between that variable and the fitness. This gives us a measure of the variables importance and its correlation with fitness, in the neighbourhood of the *high-fitness* solution.

We aim to highlight the variables that impact either negatively or positively on solution quality. It should be noted that this approach is applied to benchmark problems for demonstration purposes only, where the simple problem definition means that the explanations can be compared against known expectations, and gives a starting point for further research in this area for real-world problems. The surrogate is trained on data derived from the real fitness function, hence, it learns the local search space from, and is biased towards, the best solutions from the metaheuristic. In this way the surrogate reflects the understanding of the problem as learned by the algorithm.

We will compare the surrogate model trained on only the first generation against a surrogate trained on all populations over the run of 100 generations. Because the surrogate is biased toward the best solution, we aim to visualise the surrogates understanding of the problem between the first generation and over all generations.

A formal definition of our approach is shown in Algorithm 1.

Algorithm 1 Probing variables in a solution with respect to the surrogate fitness function

```
In: $x = (x_0 \dots x_n), x_i = \{0, 1\}$, near-optimal seed solution found by GA
In: $S(X) \to f$, surrogate fitness function to estimate fitness $f$ of a solution $X$
Out: $C = (c_0 \dots c_n), c_i \in \mathbb{R}$, absolute change to surrogate fitness for each variable in $x$
$C \leftarrow \emptyset$;
$f_{org} \leftarrow S(x)$                         ▷ surrogate fitness of solution
for for each variable $x_i$ do)$i = 0$ to $n - 1$
    $x_i \leftarrow (x_i + 1)$ mod 2                 ▷ flip variable $x_i$
    $\hat{f_i} \leftarrow S(x_i)$              ▷ surrogate fitness of mutated solution
    if $x_i = 1$ then $c_i = -1 * \hat{f_i}$     ▷ Change in surrogate fitness, optimum should have a 1
    else$c_i = \hat{f_i}$     ▷ Change in surrogate fitness, optimum should have a 0
    end if
    $C \leftarrow |c_i|$                              ▷ add to list
end for
```

5 EXPERIMENTS

We will focus on four well-known bit-string encoded benchmark functions (*1D Checkerboard, 2D Checkerboard, Trap5 and MAXSAT*).

The experiment setup is shown in Table 1. Where *P* is the population size, *n* the length of the bit-string, problem-size, *maxGen* is the maximum number of generations, *mutRate* the mutation rate. The Selection operator used is *Tournament* with a tournament size of 5. The *Crossover* operator is Uniform. We have made little effort to tune the algorithm beyond simple empirical exploration of these parameters, because the focus of this study is on explaining the results rather than maximal algorithm performance.

P	n	maxGen	mutRate	Selection	Crossover
100	100	100	0.01	Tournament(5)	Uniform

Table 1: Experiment Setup

5.1 Benchmark Problems

Our experiment focused on four benchmark functions using a bit-string representation. The four functions were chosen to contrast uni-variate and multi-variate problems. This will allow us to investigate variations in the importance attached to each variable and model's handling of the presence, or absence, of interactions.

5.1.1 1D Checkerboard. The objective of the 1D-Checkerboard problem [4] is to realise a checkerboard pattern with alternating 1s and 0s. The function scores the chromosome based on the sum of adjacent variables that do not share the same value. A formal description of the function is shown in Equation (1).

$$f(x) = \sum_{i=0}^{l-2} \begin{Bmatrix} 1, & x_i \neq x_{i+1} \\ 0, & x_i = x_{i+1} \end{Bmatrix} \tag{1}$$

5.1.2 2D Checkerboard. The 2D-Checkerboard problem [4, 26] introduces bivariate interactions into the problem. While these interactions are weighted equally in the fitness function, implicitly those in the centre of the grid have greater impact on fitness. A solution represents the rows of a *s* x *s* grid concatenated into one string, the objective being to realsie a grid with a checkerboard pattern of alternating 1s and 0s. A formal description of the function is shown in Equation (2)

$$f(x) = 4(s-2)^2 - \sum_{i=2}^{s-1}\sum_{j=2}^{s-1} \begin{Bmatrix} \delta\left(x_{ij}, x_{i-1j}\right) + \delta\left(x_{ij}, x_{i+1j}\right) \\ +\delta\left(x_{ij}, x_{ij-1}\right) + \delta\left(x_{ij}, x_{ij+1}\right) \end{Bmatrix} \tag{2}$$

5.1.3 Trap5. The Trap-5 problem is designed to be intentionally deceptive [15], by rewarding steps towards local optima in small groups of variables in order to force, or deceive, an EA away from the global optimum. This is of particular concern in algorithms that do not consider interactions between variables[6, 20]. The problem construction consists of bit-strings partitioned into blocks with their fitness scored separately. A formal description of the function is shown in Equations (3a) and (3b)

$$f(x) = \sum_{i=1}^{n/k} trap_k(x_{b_{i+1}} + \ldots + x_{b_{i+k}}) \tag{3a}$$

$$trap_k(u) = \left\{ f_{\text{high}} \text{ if } u = k, f_{\text{low}} - u\frac{f_{\text{low}}}{k-1} \text{ otherwise} \right\} \tag{3b}$$

5.1.4 MAXSAT. The Maximum Satisfiability or MAXSAT problem [12, 24, 35] attempts to find a set of values which maximises the number of satisfied clauses of a fixed predicate logic formula expressed in conjunctive normal form (CNF). It is known to be NP-complete in its general form. The MAXSAT is useful for modeling high order interactions, because each instance of the problem uses a known predefined structure. The problem construction involves a bit-string, where each bit encodes a predicate variable in the CNF formula. An individuals fitness is therefore just equal to the number of satisfied clauses.

5.2 Experimental Procedure

For each of the benchmark problems, we started with an initial population of 100 individuals and an Genetic Algorithm (GA) was used to determine a near-optimal solution. A problem size of 100, population size of 100, tournament selection with size 5, mutation rate of 0.01, crossover rate of 0.95 and was run over 100 generations. We used elitism, to guarantee that the single fittest solution was carried forward for each generation without mutation.

A normal run of a GA would consist of generating an initial population for which a fitness value is calculated and through selection, crossover and mutation a second population is generated. This process iterates for a predetermined number of generations, or until a solution with the possible fitness, as defined by our fitness function, is found, whichever occurs first. At the end of this run, the most optimal solution found is presented to the user. However, the solution presented is devoid of an explanation, as to how it was found nor why it is considered the fittest solution, other than it has the *best fitness function value*, which was determined by the algorithm designer.

We will attempt to provide an intuitive understanding of the solution generation process, with the aim of providing an explanation for the end-user as to how the solution was chosen. We will do this via construction of a surrogate model, using the population and fitness values for each generation of the GA run as training data for the surrogate. The aim in this experiment is not to generate a surrogate to speed up the fitness evaluation process, but rather to mine the surrogate for importance of variables. We will examine the surrogate model trained on only the population form the first generation and one trained on the the population data for all of the runs over 100 generations.

For each problem we used a Support Vector Regression (SVR) model, provided by scikit-learn (version 1.0.2), as the surrogate. We used the default hyper-parameters for this model, and made no attempt to tune any of the parameters, so as to enable reproducible experiments and focus on the learning of the search space by the surrogate model between the first and all generations of the GA run.

6 RESULTS AND DISCUSSION

6.1 Surrogate Trained on First Generation Data

In this section we will consider the results of mining a surrogate fitted to the first population of the algorithm run. Each plot in Figures 1, 3, 5 and 7 shows the mean contribution to surrogate fitness for each variable. The positive values indicate that the optimal value was a 0 and negative values indicate that it was a 1. In Figure 1 we can observe that the model has not managed to grasp enough of the problem structure, there is a clear partial chain structure visible, however, it is difficult to see the relative contribution of each variable due to random noise. We would expect to see an equal contribution from all the variables. There is a similar picture with Trap5 (Figure 7) and MAXSAT (Figure 5). Although with the latter there should be some variation, as variables appear in different clauses and so have different contributions to overall fitness, there is only a small variation in the number of clauses a variable appears in, and not as much as suggested by the bar plot. For 2D checkerboard there is a clearer picture of the problem structure emerging, suggesting that some variable have twice the importance of others. This appears to be driven by the suboptimal solutions in the first generation; as we will see, the model fitted to all generations weights all variables similarly and shows clearer signs of the checkerboard pattern.

In there results, the model is trained on only the first population, at this point the GA is just starting an exploration of the search space, and so has not explored enough of the problem structure to generate meaningful data on which to train the surrogate model.

Our aim here was to set a starting point to which we can then compare the results as outlined in Section 6.2

6.2 Surrogate Trained on All Generation Data

In this section we will consider the results of mining a surrogate fitted to all populations of the algorithm run. Each plot in Figures 2, 4, 6 and 8 shows the mean contribution to surrogate fitness for each variable. The positive values indicate that the optimal value was a 0 and negative values indicate that it was a 1.

Comparing these plots to those generated for only the first population data, we see a marked contrast. We observe that the surrogate model has picked up key characteristics of the problem; for Trap5 we can now start to see clearer groupings of variables as we would expect for this problem. However the surrogate is reflecting that the GA has converged on a local optima, and hence the clear grouping that has settled on all 0 for variables 85-90.

For the 1D Checkerboard we see a clearer pattern of alternating 1s and 0s and this pattern is more pronounced, and each variable seems to be contributing equally to fitness.

For 2D Checkerboard, we see that the variables share a similar relationship to fitness and fall on or around the middle, and we see some regular groupings of slightly more dominant variables, which seem mostly clustered around the central region of our checkerboard.

For MAXSAT, we see similar grouping of clauses and clustering of the variables that contribute more to overall fitness.

We can note that the explanatory method of using a surrogate model does reveal that the model has begun to detect the key components of the problem.

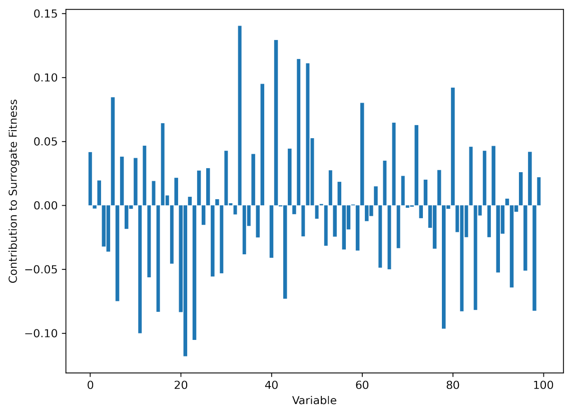

Figure 1: 1D Checkerboard Contribution to Surrogate Fitness per Variable (First Generation)

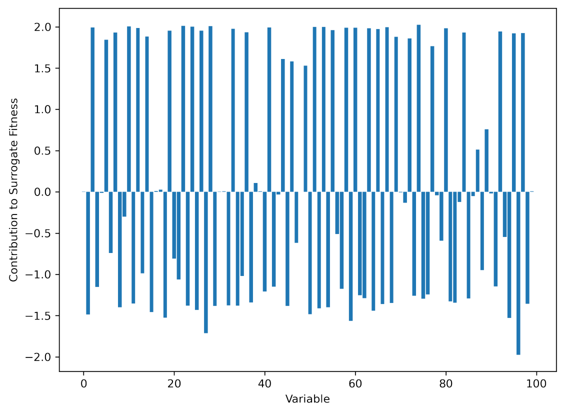

Figure 2: 1D Checkerboard Contribution to Surrogate Fitness per Variable (All Generations)

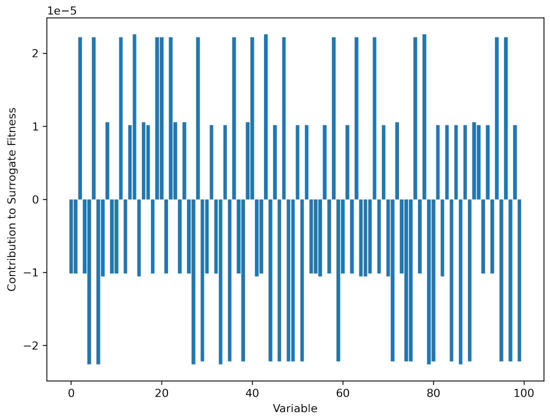

Figure 3: 2D Checkerboard Contribution to Surrogate Fitness per Variable (First Generation)

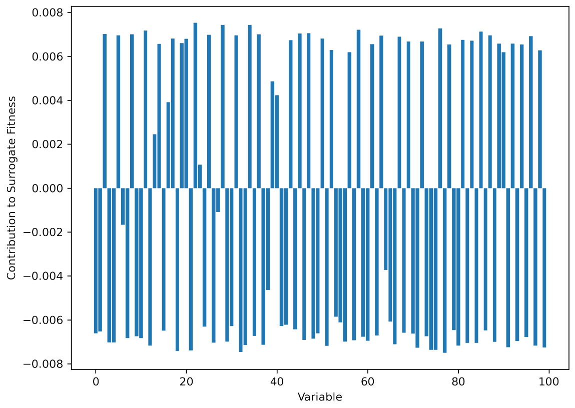

Figure 4: 2D Checkerboard Contribution to Surrogate Fitness per Variable (All Generations)

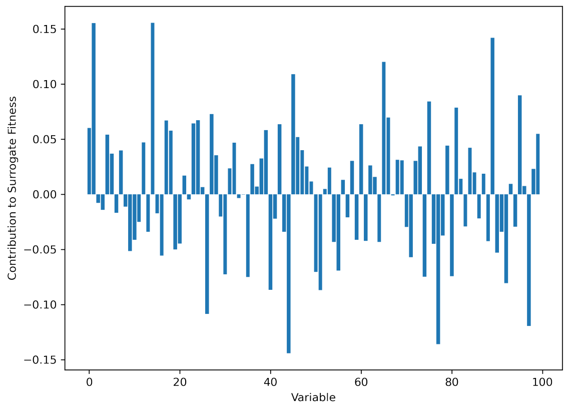

Figure 5: MAXSAT Contribution to Surrogate Fitness per Variable (First Generation)

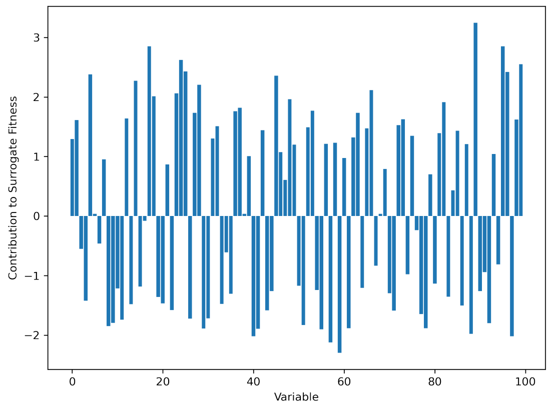

Figure 6: MAXSAT Contribution to Surrogate Fitness per Variable (All Generations)

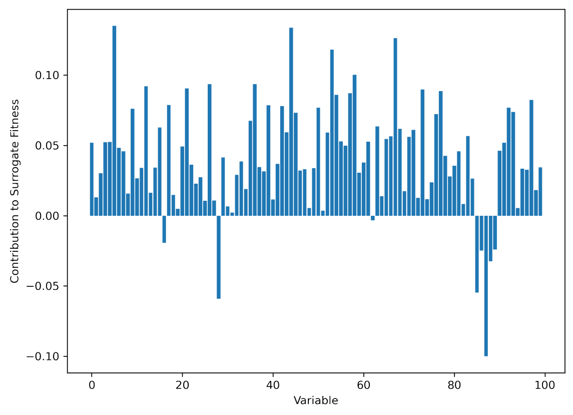

Figure 7: Trap5 Contribution to Surrogate Fitness per Variable (First Generation)

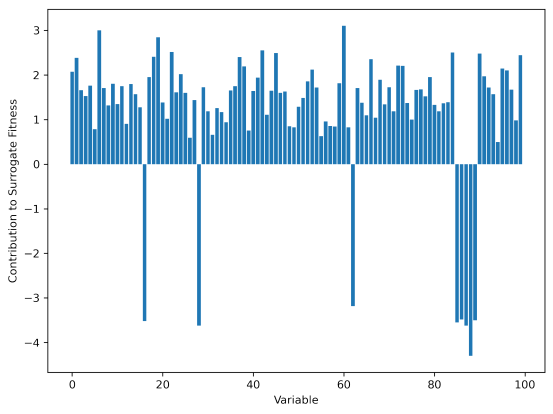

Figure 8: Trap5 Contribution to Surrogate Fitness per Variable (All Generations)

7 ACKNOWLEDGMENTS

The Data Lab funded project (Grant: 191047)

8 CONCLUSION

This paper set out to investigate the possibility of mining surrogate models for explanations about solution quality at different stages of an evolutionary run. We looked at the surrogate models learning of the search space for four benchmark problems; *1D Checkerboard, 2D Checkerboard, MAXSAT and Trap5*. We did this through using the population and fitness values for individuals generated by a GA. We used this population data as input to train an SVR model, which we mined for a deeper understanding of the solutions generated by the GA for each generation.

This work has demonstrated a starting point for further exploration and more importantly the generation of explanations around the near-optimal solution presented by a GA. We focused on mining the surrogate to understand how the mean contribution of each variable changed as the GA generated more insights into the population and problem at hand. In particular, we compared a surrogate model trained on the initial population against one trained on population data after 100 generations. These experiments suggest that, for the purposes of explanation, it is preferable to train the surrogate on all solutions visited over the course of the GA run. Although, there is also some evidence to suggest that useful information about the problem can be mined during the first generation. Further investigation into what is possible at each stage of the run is thus needed.

Future work will involve analysis of interaction of variables and investigation of multi-modal problems. As understanding of this approach to explanations grows we will apply the above method to real-world problems. We will also further investigate the performance of a surrogate model at different stages of a GA run; to generate comparisons at different generations of the GA run and investigate at what point it would be beneficial to switch from using a costly fitness function to the surrogate model and comparison of solutions generated between the two scenarios.

We also propose to further extend this work to exploit recent but already well known techniques for explaining machine learning models, such as permutation feature importance measurement[5, 18, 41] and SHapley Additive exPlanations (SHAP) [30] as a way to explain variable selection for individual solutions. The work presented here is a first step towards building a more generalised framework for explainability within the evolutionary computation (EC) domain.

The ultimate aim of this work is to generate explanations of the solution generated, by mining the surrogate model, and presenting these explanations to the end-user to instill more confidence and trust in our model. In turn, this increased trust in the solutions should lead to greater uptake and increased benefit from the use of GAs in practice.

REFERENCES

[1] Edward J. Anderson and Michael C. Ferris. 1994. Genetic Algorithms for Combinatorial Optimization: The Assemble Line Balancing Problem. *ORSA Journal on Computing* 6, 2 (May 1994), 161–173. https://doi.org/10.1287/ijoc.6.2.161

[2] Naomi Aoki. 2020. An experimental study of public trust in AI chatbots in the public sector. *Government Information Quarterly* 37, 4 (Oct. 2020), 101490. https://doi.org/10.1016/j.giq.2020.101490

[3] Florian Arnold and Kenneth Sörensen. 2019. What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. *Computers & Operations Research* 106 (June 2019), 280–288. https://doi.org/10.1016/j.cor.2018.02.007

[4] Shumeet Baluja and Scott Davies. 1997. Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space. In *Proceedings of (ICML) International Conference on Machine Learning*. 30 – 38.

[5] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[6] Alexander Brownlee. 2009. Multivariate Markov networks for fitness modelling in an estimation of distribution algorithm. (05 2009).

[7] Alexander E.I. Brownlee. 2016. Mining Markov Network Surrogates for Value-Added Optimisation. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*. ACM, Denver Colorado USA, 1267–1274. https://doi.org/10.1145/2908961.2931711

[8] Alexander E.I. Brownlee, Olivier Regnier-Coudert, John A.W. McCall, Stewart Massie, and Stefan Stulajter. 2013. An application of a GA with Markov network surrogate to feature selection. *International Journal of Systems Science* 44, 11 (Nov. 2013), 2039–2056. https://doi.org/10.1080/00207721.2012.684449

[9] Alexander E.I. Brownlee, Michal Weiszer, Jun Chen, Stefan Ravizza, John R. Woodward, and Edmund K. Burke. 2018. A fuzzy approach to addressing uncertainty in Airport Ground Movement optimisation. *Transportation Research Part C: Emerging Technologies* 92 (July 2018), 150–175. https://doi.org/10.1016/j.trc.2018.04.020

[10] Alexander E.I. Brownlee, John R. Woodward, and Jerry Swan. 2015. Metaheuristic Design Pattern: Surrogate Fitness Functions. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, Madrid Spain, 1261–1264. https://doi.org/10.1145/2739482.2768499

[11] Alexander E.I. Brownlee and Jonathan A. Wright. 2015. Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation. *Applied Soft Computing* 33 (Aug. 2015), 114–126. https://doi.org/10.1016/j.asoc.2015.04.010

[12] Alexander E. I. Brownlee, John A. W. McCall, and Deryck F. Brown. 2007. Solving the MAXSAT problem using a multivariate EDA based on Markov networks. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation - GECCO '07*. ACM Press, London, United Kingdom, 2423. https://doi.org/10.1145/1274000.1274005

[13] Alexander E. I. Brownlee, Olivier Regnier-Coudert, John A. W. McCall, and Stewart Massie. 2010. Using a Markov network as a surrogate fitness function in a genetic algorithm. In *IEEE Congress on Evolutionary Computation*. IEEE, Barcelona, Spain, 1–8. https://doi.org/10.1109/CEC.2010.5586548

[14] Kalyanmoy Deb, Sunith Bandaru, David Greiner, António Gaspar-Cunha, and Cem Celal Tutum. 2014. An integrated approach to automated innovization for discovering useful design principles: Case studies from engineering. *Applied Soft Computing* 15 (Feb. 2014), 42–56. https://doi.org/10.1016/j.asoc.2013.10.011

[15] Kalyanmoy Deb and David E. Goldberg. 1994. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence* 10, 4 (Dec. 1994), 385–408. https://doi.org/10.1007/BF01531277

[16] Kalyanmoy Deb and Aravind Srinivasan. 2006. Innovization: innovating design principles through optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*. ACM Press, Seattle, Washington, USA, 1629. https://doi.org/10.1145/1143997.1144266

[17] A. Di Pietro, L. While, and L. Barone. 2004. Applying evolutionary algorithms to problems with noisy, time-consuming fitness functions. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*. IEEE, Portland, OR, USA, 1254–1261. https://doi.org/10.1109/CEC.2004.1331041

[18] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2018. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. (2018). https://doi.org/10.48550/ARXIV.1801.01489 Publisher: arXiv Version Number: 5.

[19] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2017. Data-Efficient Exploration, Optimization, and Modeling of Diverse Designs through Surrogate-Assisted Illumination. (2017). https://doi.org/10.48550/ARXIV.1702.03713 Publisher: arXiv Version Number: 2.

[20] David E Goldberg, Kalyanmoy Deb, and Jeffrey Horn. 1992. Massive Multimodality, Deception, and Genetic Algorithms.. In *PPSN*, Vol. 2.

[21] Bryce Goodman and Seth Flaxman. 2017. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Magazine* 38, 3 (Oct. 2017), 50–57. https://doi.org/10.1609/aimag.v38i3.2741

[22] Y. Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing* 9, 1 (Jan. 2005), 3–12. https://doi.org/10.1007/s00500-003-0328-5

[23] Yaochu Jin. 2011. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation* 1, 2 (June 2011), 61–70. https://doi.org/10.1016/j.swevo.2011.05.001

[24] David S. Johnson. 1974. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.* 9, 3 (Dec. 1974), 256–278. https://doi.org/10.1016/S0022-0000(74)80044-9

[25] Anja Lambrecht and Catherine Tucker. 2019. Algorithmic Bias? An Empirical Study of Apparent Gender-Based Discrimination in the Display of STEM Career Ads. *Management Science* 65, 7 (July 2019), 2966–2981. https://doi.org/10.1287/mnsc.2018.3093

[26] P. Larrañaga. 2002. A Review on Estimation of Distribution Algorithms. In *Estimation of Distribution Algorithms*, Pedro Larrañaga and Jose A. Lozano (Eds.). Vol. 2. Springer US, Boston, MA, 57–100. https://doi.org/10.1007/978-1-4615-1539-5_3 Series Title: Genetic Algorithms and Evolutionary Computation.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (May 2015), 436–444. https://doi.org/10.1038/nature14539

[28] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Lee Altenberg, Julie Beaulieu, Peter J. Bentley, Samuel Bernard, Guillaume Beslon, David M. Bryson, Nick Cheney, Patryk Chrabaszcz, Antoine Cully, Stephane Doncieux, Fred C. Dyer, Kai Olav Ellefsen, Robert Feldt, Stephan Fischer, Stephanie Forrest, Antoine Fŕenoy, Christian Gagńe, Leni Le Goff, Laura M. Grabowski, Babak Hodjat, Frank Hutter, Laurent Keller, Carole Knibbe, Peter Krcah, Richard E. Lenski, Hod Lipson, Robert MacCurdy, Carlos Maestre, Risto Miikkulainen, Sara Mitri, David E. Moriarty, Jean-Baptiste Mouret, Anh Nguyen, Charles Ofria, Marc Parizeau, David Parsons, Robert T. Pennock, William F. Punch, Thomas S. Ray, Marc Schoenauer, Eric Schulte, Karl Sims, Kenneth O. Stanley, François Taddei, Danesh Tarapore, Simon Thibault, Richard Watson, Westley Weimer, and Jason Yosinski. 2020. The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. *Artificial Life* 26, 2 (May 2020), 274–306. https://doi.org/10.1162/artl_a_00319

[29] Per Kristian Lehre and Pietro S. Oliveto. 2017. Theoretical Analysis of Stochastic Search Algorithms. *arXiv:1709.00890 [cs]* (Sept. 2017). http://arxiv.org/abs/1709.00890 arXiv: 1709.00890.

[30] Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. (2017). https://doi.org/10.48550/ARXIV.1705.07874 Publisher: arXiv Version Number: 2.

[31] Katherine Mary Malan. 2021. A Survey of Advances in Landscape Analysis for Optimisation. *Algorithms* 14, 2 (Jan. 2021), 40. https://doi.org/10.3390/a14020040

[32] Tim Miller. 2017. Explanation in Artificial Intelligence: Insights from the Social Sciences. (2017). https://doi.org/10.48550/ARXIV.1706.07269 Publisher: arXiv Version Number: 3.

[33] Gabriela Ochoa, Lee A. Christie, Alexander E. Brownlee, and Andrew Hoyle. 2020. Multi-objective evolutionary design of antibiotic treatments. *Artificial Intelligence in Medicine* 102 (Jan. 2020), 101759. https://doi.org/10.1016/j.artmed.2019.101759

[34] S. C. Olhede and P. J. Wolfe. 2018. The growing ubiquity of algorithms in society: implications, impacts and innovations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376, 2128 (Sept. 2018), 20170364. https://doi.org/10.1098/rsta.2017.0364

[35] Soraya Rana and Darrell Whitley. 1998. Genetic algorithm behavior in the MAXSAT domain. In *Parallel Problem Solving from Nature — PPSN V*, G. Goos, J. Hartmanis, J. van Leeuwen, Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel (Eds.). Vol. 1498. Springer Berlin Heidelberg, Berlin, Heidelberg, 785–794. https://doi.org/10.1007/BFb0056920 Series Title: Lecture Notes in Computer Science.

[36] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (May 2019), 206–215. https://doi.org/10.1038/s42256-019-0048-x

[37] William R. Swartout and Johanna D. Moore. 1993. Explanation in Second Generation Expert Systems. In *Second Generation Expert Systems*, Jean-Marc David, Jean-Paul Krivine, and Reid Simmons (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 543–585. https://doi.org/10.1007/978-3-642-77927-5_24

[38] Ehsan Toreini, Mhairi Aitken, Kovila Coopamootoo, Karen Elliott, Carlos Gonzalez Zelaya, and Aad van Moorsel. 2020. The relationship between trust in AI and trustworthy machine learning technologies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM, Barcelona Spain, 272–283. https://doi.org/10.1145/3351095.3372834

[39] Neil Urquhart, Michael Guckert, and Simon Powers. 2019. Increasing trust in metaheuristics by using MAP-elites. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, Prague Czech Republic, 1345–1348. https://doi.org/10.1145/3319619.3326816

[40] Aidan Wallace, Alexander E. I. Brownlee, and David Cairns. 2021. Towards Explaining Metaheuristic Solution Quality by Data Mining Surrogate Fitness Models for Importance of Variables. In *Artificial Intelligence XXXVIII*, Max Bramer and Richard Ellis (Eds.). Vol. 13101. Springer International Publishing, Cham, 58–72. https://doi.org/10.1007/978-3-030-91100-3_5 Series Title: Lecture Notes in Computer Science.

[41] Pengfei Wei, Zhenzhou Lu, and Jingwen Song. 2015. Variable importance analysis: A comprehensive review. *Reliability Engineering & System Safety* 142 (2015), 399–432. https://doi.org/10.1016/j.ress.2015.05.018

[42] Adrienne Yapo and Joseph Weiss. 2018. Ethical Implications of Bias in Machine Learning. https://doi.org/10.24251/HICSS.2018.668