Geometric Surrogate Model Based Optimisation for Genetic Programming: Initial Experiments

Alberto Moraglio[1] and Ahmed Kattan[2]

[1] School of Computing,University of Birmingham, UK
albmor@gmail.com
[2] College of Computer and Information Systems
Um Alqura University, Saudi Arabia
akattan@uqu.edu.sa

Abstract. Many real-world problems have expensive objective functions. In continuous optimisation, Surrogate Models (SMs) are often used as components of optimisation algorithms to tackle these types of problems. In previous work, we showed that such approaches can be naturally and *rigorously generalised* to combinatorial spaces based in principle on *any arbitrarily complex underlying solution representation*. This direct approach to representations, unlike previous approaches, does not require shoe-horning of the solution structure in a vector of features prior to its application. This enlarges greatly the scope of SMs to complex representations which cannot be naturally mapped to vectors of features. In this paper, we illustrate how this framework applies straightforwardly to tree-based Genetic Programming and report initial experimental results.

1 Introduction

Some typologies of tasks when cast as optimisation problems give rise to objective functions which are prohibitively expensive to evaluate. Oftentimes these problems are black-box problems, i.e., whose problem class is unknown, and they are possibly mathematically ill-behaved (e.g., discontinuous, non-linear, non-convex). For example, most engineering design problems are of this type (see e.g., [10]). They require experiments and/or simulations to evaluate to what extent the design objective has been met as a function of parameters controlling the design. The simulation can take many minutes, hours, or even days to complete.

There is an increasing number of optimisation problems naturally associated with complex solution representations which have also very expensive objective functions. In particular, Genetic Programming that normally uses a tree representation, has a number of application domains with expensive objective functions. For example, one of them is when genetic programs encode behavioral controllers of robots that may need to be tested in a virtual or real environment a number of times to assess how good the controller is at controlling the robot for certain target tasks (e.g., wall-following or obstacle avoidance).

Optimisation methods based on surrogate models, also known as response surface models, have been successfully employed to tackle expensive objective functions. For a survey on surrogate model based optimisation methods refer to [4]. A surrogate model is a mathematical model that approximates as precisely as possible the expensive objective function of the problem at hand, and that is computationally much cheaper to evaluate. The objective function is considered

Algorithm 1 Surrogate Model Based Optimisation

1: Sample uniformly at random a small set of candidate solutions and evaluate them using the expensive objective function (initial set of data-points)
2: **while** limit number of expensive function evaluations not reached **do**
3: Construct a new surrogate model using all data-points available
4: Determine the optimum of the surrogate model by search, e.g., using an evolutionary algorithm (this is feasible as the model is cheap to evaluate)
5: Evaluate the solution which optimises the surrogate model in the problem with the expensive objective function (additional data-point available)
6: **end while**
7: Return the best solution found (the best in the set of data-points)

unknown. The surrogate model is built solely from available known values of the expensive objective function evaluated on a set of solutions. We refer to the pair (solution, known objective function value) as data-point. The traditional procedure of surrogate model based optimisation (SMBO) [4] is outlined in Algorithm 1. The role of the evolutionary algorithm in the SMBO procedure is to infer the location of a promising solution of the problem using the surrogate model, and it is not directly applied to the original problem with the expensive objective function. This is feasible because the computational cost of a complete run of the evolutionary algorithm on the surrogate model is negligible (in the order of few seconds) with regard to the cost of evaluating a solution using the expensive objective function of the problem (in the order of minutes, hours or even days depending on the problem).

SMBOs are naturally suited to continuous optimisation. In this case, a host of techniques to build functions from data-points can be borrowed from statistics (i.e., multi-variate regression [1]) and machine learning (i.e., supervised learning by e.g., neural networks and support vector machines [6]), which can be used to build surrogate models. However, SMBOs do not seem to be applicable to combinatorial optimisation problems except in those cases in which solutions are naturally represented as vectors of integers, in which case adequately discretised versions of continuous surrogate models may be used.

To the authors's best knowledge there are no works in literature on surrogate models defined *directly* on more complex representations than vectors, and specifically on tree-based Genetic Programming. In literature, there is an approach [5] in which Genetic Programming is used to do symbolic regression to determine the best fitting function to the data-points. In this approach GP is not used to search the surrogate model (i.e., data-points are not programs) but to train the surrogate model on the known data-points with a vector representation.

Currently, if one wants to use surrogate models on search problems naturally based on structured representations, the original representation has to be shoehorned to a vector form in a pre-processing phase known as *features extraction* in the Machine Learning literature. However, extracting features from structured representations, such as Genetic Programming trees, can be inherently problematic as oftentimes it does not appear to be a natural way to map these types of structures to vectors of features. For example, what would it be a natural set of features to consider to map a tree representing a symbolic regression formula or a boolean formula to a fixed-size vector? This question does not seem to have any obvious answer.

Is there a systematic and rigorous way to adapt a surrogate model for the continuous domain to a new representation which does not require us to rethink the surrogate model, or make ad-hoc adaptation to the model, for any target representation considered however complex it is? In very recent work [8], we have proposed a generalisation based on geometric ideas [7] of a well-known class of surrogate models – Radial Basis Function Networks [3] – which answers in the affirmative the question above. The general surrogate model was applied to the binary string representation. In this paper, we illustrate how the SMBO with the general surrogate model can be straightforwardly applied to tree-based Genetic Programming, and report initial experimental results.

2 Generalised Radial Basis Function Networks

A radial basis function (RBF) is a real-valued function $\phi : \mathbf{R}^n \rightarrow \mathbf{R}$ whose value depends only on the distance from some point $\mathbf{c}$, called a *center*, so that $\phi(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$. The point $\mathbf{c}$ is a parameter of the function. The norm is usually Euclidean, so $\|\mathbf{x} - \mathbf{c}\|$ is the Euclidean distance between $\mathbf{c}$ and $\mathbf{x}$. The most frequently used types of radial basis functions are Gaussian functions of the form:

$$\phi(\mathbf{x}) = \exp(-\beta\|\mathbf{x} - \mathbf{c}\|^2)$$

where $\beta > 0$ is the width parameter. Radial basis functions are used to build function approximations of the form:

$$y(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{c}_i\|).$$

In an RBFN there are three types of parameters that need to be determined to optimise the fit between $y(\mathbf{x})$ and the data: the weights w_i, the centers $\mathbf{c}_i$, and the RBF width parameters β_i. A widely applied procedure to fit RBFNs to the data consists of choosing the centers $\mathbf{c}_i$ to coincide with the known points $\mathbf{x}_i$, and choosing the widths β_i according to some heuristic based on the distance to nearest neighbors of the center $\mathbf{c}_i$ or to the maximum distance between the chosen centers. The bias w_0 is set to the average of the function values b_i at the known data-points (i.e., function values of the points in the training set). The weights w_i can be determined by solving the system of N simultaneous linear equations in w_i obtained by requiring that the unknown function interpolates exactly the known data-points, which can be solved using simple linear algebra, involving a matrix inversion (see [3] for details).

In very recent work [8], we used a geometric methodology to generalise RBFNs to any solution representation. The gist of the idea is that all aspects of RBFNs that allow us to use them as surrogate models, i.e., model definition and representation, training, querying and searching of RBFNs, can be naturally generalized from Euclidean spaces to general metric spaces, simply by replacing the Euclidean distance with a generic metric. The generalized model applies to any underlying solution representation once a distance function rooted on that representation is provided (e.g., Edit distance on trees). In particular, this method can be used *as it is* to learn in principle any function mapping directly complex structured representations to reals without introducing any arbitrary ad-hoc adaptation to the RBFNs. There is no special requirement of pre-processing the target representation and shoehorn it in a vector of features.

Table 1. Results on the Unimodal Problems (w.r.t. SD and SHD), Parity and Symbolic Regression Problems for the comparison of SMBO-SD, SMBO-SHD, Random Search and GP. Mean and best fitness over 20 independent runs of the best solution found by each algorithm, for $md = 3, 4, 5, 6, 7$ (lower fitness is better).

Unimodal-SD

	SMBO-SD			**SMBO-SHD**			**Random Search**			**GP**		
MD	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
3	0.0095	0.0000	0.0219	-	-	-	0.0285	0.0000	0.0407	0.0560	0.0000	0.0495
4	0.0078	0.0000	0.0134	-	-	-	0.0228	0.0000	0.0364	0.0493	0.0010	0.0470
5	0.0008	0.0000	0.0022	-	-	-	0.0027	0.0001	0.0040	0.0143	0.0010	0.0231
6	0.0011	0.0000	0.0030	-	-	-	0.0041	0.0001	0.0046	0.0259	0.0000	0.0399
7	-	-	-	-	-	-	-	-	-	-	-	-

Unimodal-SHD

	SMBO-SD			**SMBO-SHD**			**Random Search**			**GP**		
MD	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
3	-	-	-	0.47	0.11	0.19	0.43	0.11	0.16	0.50	0.22	0.18
4	-	-	-	0.14	0.07	0.05	0.37	0.22	0.13	0.49	0.11	0.25
5	-	-	-	0.07	0.04	0.03	0.24	0.11	0.08	0.48	0.05	0.21
6	-	-	-	0.04	0.01	0.04	0.14	0.02	0.08	0.46	0.14	0.21
7	-	-	-	0.02	0.008	0.04	0.18	0.10	0.04	0.32	0.06	0.20

4-Odd Parity

	SMBO-SD			**SMBO-SHD**			**Random Search**			**GP**		
MD	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
3	47.50	37.50	3.85	45.00	37.50	6.45	45.00	37.50	6.45	48.75	37.50	3.95
4	41.25	37.50	5.55	40.00	37.50	5.27	41.25	37.50	6.04	42.50	37.50	6.45
5	38.75	37.50	2.80	37.50	37.50	0.00	37.50	37.50	0.00	47.50	37.50	5.27
6	37.50	37.50	0.00	37.50	37.50	0.00	37.50	37.50	0.00	41.25	37.50	6.04
7	-	-	-	33.75	25.00	6.04	37.50	37.50	0.00	37.50	37.50	0.00

Symbolic Regression

	SMBO-SD			**SMBO-SHD**			**Random Search**			**GP**		
MD	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std	Mean	Best	Std
3	5.17	3.01	1.84	4.88	3.44	0.82	4.88	3.44	0.78	5.17	2.64	1.35
4	2.94	1.93	2.56	6.35	4.46	1.17	5.78	4.27	1.41	6.39	4.46	1.58
5	4.33	2.98	1.04	5.58	3.84	1.21	5.18	3.51	1.21	5.39	4.05	1.27
6	4.10	2.34	1.14	3.74	2.95	0.73	3.52	2.99	3.48	4.39	3.48	0.57
7	-	-	-	4.50	3.45	0.77	4.96	3.81	0.54	4.62	3.71	0.67

3 Experiments

Experiments have been carried out on two standard GP test problems – symbolic regression and parity problems – and on two types of unimodal problem, in which the fitness of a tree (to minimise) is given by its distance to an arbitrary but fixed tree. We will consider the problems in the test-bed as having costly objective functions. We treated all problems as minimisation problems (i.e., solutions with lower fitness values are better). As distance functions between GP trees, we have used the well-known Structural Distance (SD) for GP trees [2] with parameter $K = 15$ and the Structural Hamming Distance (SHD) [9]. Since we consider two distance functions, we have two types of SMBO (SMBO-SD and SMBO-SHD) and two types of unimodal functions (Unimodal-SD and Unimodal-SHD). The choice of a distance well suited to the problem at hand is crucial to obtain a surrogate model able to make meaningful predictions and guide appropriately the search of the SMBO.

We use a standard surrogate model based optimisation algorithm (see Algorithm 1) and the learning procedure presented in the previous section. As search spaces of different size require different parameter settings, we set the parameters as a function of the maximum allowed depth of trees in the initial population md. Consequently, the maximum number of nodes of a tree with maximum depth md and function sets of arity at most two is $2^{md} - 1$. The number of total available expensive function evaluations is set to $n = 2^{md}$. So, essentially our aim is to

find the best solution to the problem the algorithm can produce in linear time on the maximum size of the trees in the initial population. We set the size of the initial sample of data-points to two, and the number of sample points suggested by the surrogate model to $n - 2$. To search the surrogate model we use standard genetic programming with tournament selection with tournament size two, subtree crossover with crossover rate 0.8, subtree mutation with mutation rate 0.17 and reproduction operator at 0.03. The population size and the number of generations are both set to n, which provide GP with enough trials to locate a good solution of the surrogate model.

We compared the SMBO algorithms with standard GP and with Random Search (RS) applied directly on the problem with the expensive objective function. Random search is considered because with small samples it can perform relatively well. We gave all algorithms in the comparison exactly the same number of expensive objective functions, which is n trials, and report the best solution found. The GP used has a population of $\sqrt{n}$ individuals, it runs for $\sqrt{n}$ generations. It uses tournament selection with size two, subtree mutation with probability of 0.17, subtree crossover with crossover rate 0.8 and reproduction operator at 0.03. For any of the problem in the test-bed, we conducted experiments for maximum depth md ranging from 3 to 7, and did 20 independent runs.

Table 1 reports the results of the comparison. As a general remark, we notice that as md grows all algorithms in the comparison tend to get better solutions. This trend is not surprising as the number of expensive fitness evaluations provided is an increasing function of md. Let us now consider each problem. On both Unimodal problems, looking at the mean values, the corresponding SMBO is consistently the best, followed by Random Search, and finally by GP. This suggests that a SMBO performs well on a problem which is unimodal w.r.t. the distance used as a base for the SMBO. This may suggest that a criterion for choosing a distance for SMBO could be picking one with a good fitness-distance correlation for the problem at hand (since the unimodal problem considered is a cone and by construction has maximum fitness distance correlation). Perhaps surprisingly, Random Search does better than Genetic Programming. This is because GP does not have enough fitness evaluations available to "get the evolution started", whereas the solutions found by random search exhibits a large variance in quality so the best solution found can be competitive "by a stroke of luck", especially with small sample size and in small problems. On the Parity problem, looking at the means, SMBO-SHD wins over Random Search and GP, but with a smaller margin, whereas SMBO-SD performs worse than random search. Again, GP is worse than RS, but for larger budget of expensive evaluations (i.e., $md = 7$), its performance matches RS. Notice that also the performance of both SMBOs are getting better relative to RS with a larger budget. It would be interesting to see how the relative performance of the algorithms in the comparison changes for larger md. We will do this analysis in future work. On the parity problem, it would seem that SMBO-SHD will still lead over the others, and that RS will become last. Let us now consider the symbolic regression problem. SMBO-SD performs best, RS second best, followed by SMBO-SHD and GP worst. Also, in this case it would be interesting to do experiments with larger md. Symbolic regression seems to be harder for SMBO-SHD and easier for SMBO-SD, and the other way around for the parity problem. This may suggest that SHD and SD are well-suited for the parity problem and for symbolic regression,

respectively. In future work, we will consider alternative distances and try to understand how to chose distances as a basis for SMBO for a given problem.

In summary, analogously to the case of continuous space, the surrogate model on the Genetic Programs has helped at finding better solutions than using standard search algorithms. These results are by all means preliminary. However, this is initial evidence that makes promising the application of this framework to real-world problems using complex solution representations associated with non-trivial discrete spaces, such as GP, which cannot be approached with more traditional methods.

4 Conclusions and Future Work

A direct approach to representations greatly enlarges the scope of SMBO to complex representations (e.g., Genetic Programming trees) which cannot be naturally mapped to vectors of features. In previous work, we have outlined a conceptually simple, formal, general and systematic approach to adapt a SMBO algorithm to *any* target representation.

As a preliminary experimental validation of the framework on a non-trivial discrete space and structured representation, we have considered the Genetic Programming trees endowed with the structural distance and structural hamming distance and tested the SMBO on a test-bed on standard GP problems, obtaining that with the same budget of expensive function evaluations, the SMBO performs well in a comparison with other search algorithms. This shows that this framework has potential to work well on real-world problems using complex solution representations associated with non-trivial discrete spaces.

Much work remains to be done. Firstly, we will test the framework on other GP problems and with different settings for the number of expensive evaluations available. We will also experiment with other non-vectorial representations, such as permutations and variable-length sequences. Then, we will test how the system performs on a number of challenging real-world problems.

References

1. N. A. C. Cressie. *Statistics for Spatial Data (revised edition)*. Wiley, 1993.
2. A. Ekart and S. Z. Nemeth. A metric for genetic programs and fitness sharing. In *Genetic Programming, Proceedings of EuroGP'2000*, pages 259–270, 2000.
3. L. C. Jain. *Radial Basis Function Networks*. Springer, 2001.
4. D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:4:345–383, 2001.
5. T. L. Lew, A. B. Spencer, F. Scarpa, K. Worden, A. Rutherford, and F. Hemez. Identification of response surface models using genetic programming. *Mechanical Systems and Signal Processing*, 20:1819–1831, 2006.
6. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
7. A. Moraglio. *Towards a Geometric Unification of Evolutionary Algorithms*. PhD thesis, University of Essex, 2007.
8. A. Moraglio and A. Kattan. Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *EvoCop*, 2011.
9. A. Moraglio and R. Poli. Geometric landscape of homologous crossover for syntactic trees. In *Proceedings of IEEE congress on evolutionary computation*, pages 427–434, 2005.
10. S. Tong and B. Gregory. Turbine preliminary design using artificial intelligence and numerical optimization techniques. *Journal of Turbomachinery*, 114:1–10, 1992.