Metaheuristic Design Pattern: Surrogate Fitness Functions

Alexander E.I. Brownlee
Computing Science and Mathematics
University of Stirling
FK9 4LA Scotland UK
sbr@cs.stir.ac.uk

John R. Woodward
Computing Science and Mathematics
University of Stirling
FK9 4LA Scotland UK
john.woodward@cs.stir.ac.uk

Jerry Swan
Computing Science
University of York
YO10 5GW England UK
jerry.swan@york.ac.uk

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

metaheuristics; surrogates; fitness approximation

1. PROBLEM STATEMENT

Metaheuristics are algorithms which search a space of candidate solutions for a given problem, seeking good quality solutions with respect to one or more objectives. This search proceeds by using operators that create new solutions (the decision space), guided by some concept of solution fitness (the objective space). The *fitness function* usually serves two distinct but related purposes. Firstly, the fitness function encapsulates the problem objectives, in the most restrictive case providing no more information than a partial ordering over the solutions. Secondly, it is desired that the fitness function *guides* the search, providing the metaheuristic with a gradient which directs it towards better quality solutions. However, it is not necessarily the case that these two purposes are aligned: the "true" objective function may not provide a suitable gradient for the search, or may simply be expensive to compute. In this context it may be more suitable to use a *surrogate* fitness function (also known as a meta-model, proxy or approximation function) to guide the search.

2. SOLUTION

Use the 'Surrogate Fitness' design pattern in the following situations:

1. The objective function is costly to execute, e.g., it is obtained via a long-running simulation or complex function, where evaluations make up the majority of the search run time.
2. The objective function is noisy, i.e., it can return different values for multiple evaluations of a single solution, and must be repeatedly sampled to compensate.
3. The objective function does not provide a useful search gradient (Figure 1), preventing the search algorithm from reaching the global optimum. Specifically:
   (a) The objective function is too rugged, causing the search to become trapped in a local optimum. This would be reflected by recently visited solutions having low diversity in the decision space (and hence in the objective space).
   (b) The objective function has plateaus, providing no direction in which the search should proceed. This would be reflected by recently visited solutions having low diversity in the objective space, but not the decision space.

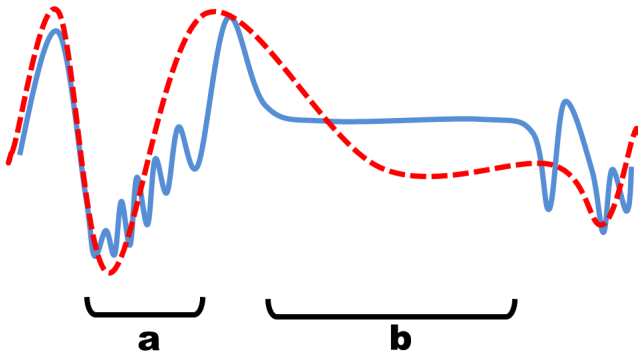

Figure 1: The objective function (solid blue) has several features such as local optima (e.g., region (a)) and a plateau (e.g., region (b)) which may hinder the search. The surrogate (dashed red) acts to mitigate these features and provide a more useful gradient.

Surrogates may be either static (explicit functions that are defined as part of the problem) or dynamic (models that are trained used samples of the "true" objective function). The former can be used in situations 3 to 3b (essentially as a means of introducing additional domain knowledge to the search process) and the latter can be used in any of the situations listed in Section 2. The UML class diagram in Figure 2 shows how a surrogate function relates to the other

major components of metaheuristic search. The problem description provides an objective function, and zero or more surrogate functions. Static surrogates are supplied as part of the problem definition: for example, a count of the number of optimal component parts of a solution can serves to add a gradient to the search. These are provided to the solver, and whether a function is an objective or surrogate is made visible (so they are not simply access through the EvaluationFunction interface). The solver can also have one or more dynamic surrogates, which are typically models built online during the search using machine learning techniques. The function within a dynamic surrogate can be updated or replaced over time. Such adaptive components can be seen as an instance of the 'Bridge' (a.k.a. Handle/Body) pattern [10]. A surrogate can also in turn be an ensemble of simpler surrogates [20], which can be seen as a further example of the 'Composite' pattern in metaheuristics [30]. Use of an ensemble of surrogates allows the strengths of several different approaches to be combined automatically.

3. CONSEQUENCES

- A surrogate provides an alternative search landscape. This means that it can provide a gradient for the search process where no useful gradient was present. Furthermore this allows the introduction of additional domain information to the search process: for example a surrogate can be added to indicate when a solution has known building blocks of solutions for the problem.
- Approximation errors in the surrogate can cause the search to diverge from a trajectory towards optimum (termed *evolution control* [13]). To avoid this, the solver must make periodic reference to the objective function. This can be achieved by either using the surrogate as a filter to choose promising offspring prior to their evaluation by the objective function, or by evaluating some solutions with the surrogate and others with the objective function.
- A surrogate can reduce calls to the costly objective function, but model training introduces overhead. Balancing these also takes care and experimentation.
- An additional benefit of a surrogate that has been constructed by machine learning processes in parallel with the optimization run is that it represents an explicit model of the problem. This can be mined to support decision making [4, 11, 24].

4. IMPLEMENTATION

Static surrogates are entirely problem-dependent and are implemented alongside the true objective function. For dynamic surrogates, a wide variety of regression or machine learning methods can be employed to model the objective function. Common implementations used in the literature are:

- Polynomials (response surface methodology).
- Kriging models.
- Artificial neural networks.
- Interpolation models.

The most suitable approach depends on the specific problem. For example, Kriging can produce a high-quality model for continuous functions, but can be prohibitively costly to compute for problems with many dimensions. Although it does not involve the construction of an explicit model, a simple alternative which can be considered to be a surrogate is fitness inheritance. This is the passing of fitness values from parents to offspring to reduce the number of fitness evaluations [8, 23, 27].

A further alternative which could be considered as a surrogate is to relax the constraints of a problem [17, 21], modifying the search space. Certain constraints can either be removed or altered, effectively making a different problem which should be easier to solve. The constraints can be adjusted in order to solve the original problem. A similar situation exists with multi-objective problems. Often, many objectives are collapsed into a single objective which is a weighted sum of each of the individual objective. Manipulating the weights changes the search space, creating a surrogate function. A model with high fidelity can be trained once and used in place of the objective function, but more commonly it is updated as the search process proceeds.

5. EXAMPLES

Examples in which the objective function is a complex simulation and hence computationally expensive to execute are given in [16, 5]. Related to this, for some problems there is no explicit mathematical function which models the problem domain (e.g., human-in-the-loop evaluation [25, 2]). In this context, samples of the "objective function" are very time-consuming: awaiting human input or real-world measurement. Therefore a computational or mathematical model of the domain can be built in order to provide an abstraction of the problem.

Static surrogates were used in [29] for solving the Eternity II puzzle, measuring the number of partial completed puzzles were present in a given solution. These guided the search, with the algorithm infrequently making reference to the full solution.

In the bin-packing problem (both online and offline [6, 7]), the aim is to pack a number of items into the smallest number of bins possible. The objective function therefore scores a packing solution as the number of bins used. However a function which only counts the number of bins is a rather coarse measure and will consider two solutions as indistinguishable regardless of how the items are packed within the bins. For example, in the first situation in which a pair of bins are both equally occupied (i.e., contain the same total contents), and the second situation in which one bin is almost full, while the second bin is almost empty. This leads to large plateau in the search space, where many solutions are assigned the same value and therefore such blunt functions fail to guide the search process. In the former case a further item may not be able to be accommodated in the current bins, while in the latter case there may well be space in the second bin which is almost empty. This function has proved to be a better driver in the evolutionary search process.

A surrogate function can add noise to the objective function so the search process is less likely to become stuck in local optima [12]. The effect of adding a small amount of noise to an objective function (the degree of noise could of course be determined adaptively) is to smooth small undulations and local optima making the landscape less rugged. It has been noted that adding too much noise, or the wrong sort of noise could have a negative effect [28].

As a highly-illustrative example of the art of construct-

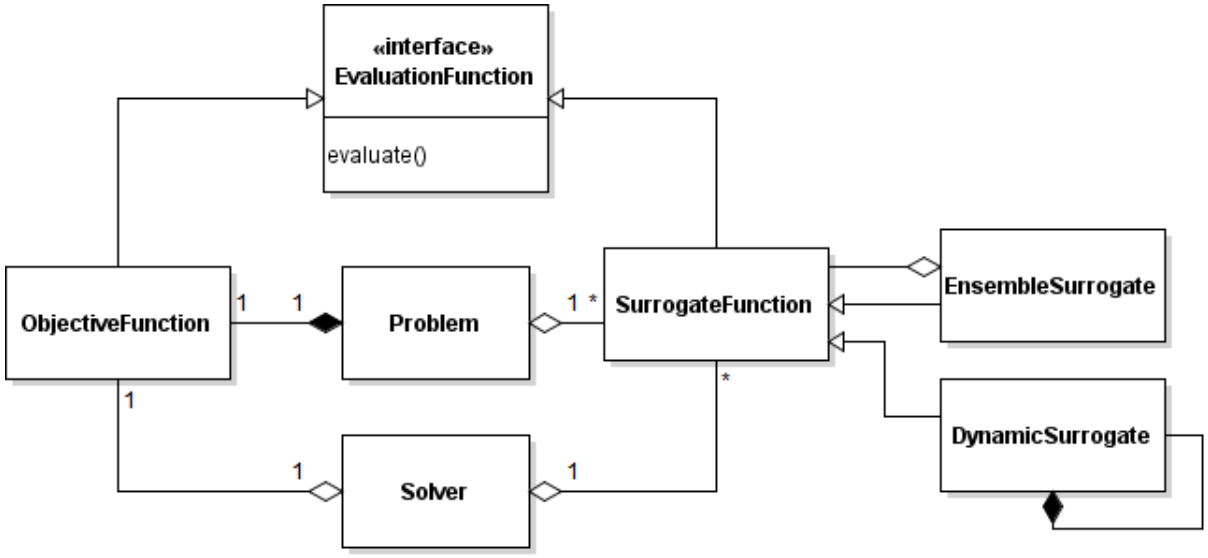

Figure 2: UML class diagram for the Surrogate Fitness Design Pattern

ing effective static surrogates, Cramer [9] evolves a program which multiplies two natural numbers together. The natural choice of objective function is the error. The author states "after much experimentation, the following scheme for giving an evaluation score was used", which indicated the difficulty of arriving at an effective surrogate function even for a very simple synthetic problem. A multi-tiered function was used where the following types of behaviour were noted and each successive type given more credit:

- Output variables changed from their initial values. (Is there any activity in the function?)
- Simple functional dependence of an output variable on an input variable. (Is the function accounting for the input?)
- The value of an input variable is a factor of the value of an output variable. (Are useful loop-like structures developing?)
- Multiplication. (Is an output variable exactly the product of two input variables.)

The fact that the fitness function played such a vital role in this first paper on Genetic Programming highlights the tension between being given an objective function and designing or generating a suitable surrogate.

Many other examples exist in the literature including noisy objective functions [1], plateaus and multi-modality [22, 18, 19, 26], deceptive gradient [3] and constraints [15]. Further examples of surrogate functions can be found in the extensive reviews by Jin [13, 14].

Acknowledgment

Work funded by UK EPSRC grant EP/J017515 (DAASE).

6. REFERENCES

[1] Maumita Bhattacharya. Reduced computation for evolutionary optimization in noisy environment. In *Proc. GECCO*, pages 2117–2122. ACM, 2008.

[2] John Biles, Peter Anderson, and Laura Loggi. Neural network fitness functions for a musical IGA. In *Proc. of the Int'l ICSC Symp. on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96)*, pages 131–137. Int'l. Computing Sciences Conferences (ICSC), 1996.

[3] A. E. I. Brownlee, O. Regnier-Coudert, J. A. W. McCall, and S. Massie. Using a Markov network as a surrogate fitness function in a genetic algorithm. In *Proc. IEEE CEC*, pages 4525–4532, Barcelona, Spain, 2010. IEEE Press.

[4] A.E.I. Brownlee, J.A.W. McCall, and Q. Zhang. Fitness Modeling With Markov Networks. *IEEE T. Evolut. Comput.*, 17(6):862–879, 2013.

[5] A.E.I. Brownlee and J.A. Wright. Constrained, mixed-integer and multi-objective optimisation ofbuilding designs by NSGA-II with fitness approximation. *Applied Soft Computing*, (0):In press., 2015. http://dx.doi.org/10.1016/j.asoc.2015.04.010.

[6] Edmund K Burke, Matthew R Hyde, and Graham Kendall. Evolving bin packing heuristics with genetic programming. In *Parallel Problem Solving from Nature-PPSN IX*, pages 860–869. Springer, 2006.

[7] EK Burke, MR Hyde, G Kendall, and J Woodward. The scalability of evolved on line bin packing heuristics. In *2007 IEEE Congress on Evolutionary Computation*, pages 2530–2537. 2007.

[8] J.-H. Chen, D.E. Goldberg, S.-Y.Ho, and K.Sastry. Fitness Inheritance in Multiobjective Optimization. In *Proc. GECCO*, pages 319–326, New York, 2002. ACM Press.

[9] Nichael Lynn Cramer. A representation for the adaptive generation of simple sequential programs. In John J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and the Applications*, pages 183–187, Carnegie-Mellon University, Pittsburgh, PA, USA, 24-26 July 1985.

[10] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable*

Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[11] Mark Hauschild, Martin Pelikan, Kumara Sastry, and Claudio Lima. Analyzing probabilistic models in hierarchical BOA. *IEEE T. Evolut. Comput.*, 13(6):1199–1217, December 2009.

[12] L. Holmstrom and P. Koistinen. Using additive noise in back-propagation training. *Neural Networks, IEEE Transactions on*, 3(1):24–38, Jan 1992.

[13] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[14] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.*, 1(2):61–70, 2011.

[15] Yaochu Jin, Sanghoun Oh, and Moongu Jeon. Incremental approximation of nonlinear constraint functions for evolutionary constrained optimization. In *Proc. IEEE CEC*, pages 2966–2973, July 2010.

[16] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE T. Evolut. Comput.*, 6(5):481–494, Oct 2002.

[17] Il Yong Kim and OL De Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and multidisciplinary optimization*, 29(2):149–158, 2005.

[18] Ko-Hsin Liang, Xin Yao, and C. Newton. Combining landscape approximation and local search in global optimization. In *Proc. IEEE WCCI*, volume 2, pages 1514–1520, 1999.

[19] Ko-Hsin Liang, Xin Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-based Intelligent Engineering System*, 4(3):172–183, 2000.

[20] Dudy Lim, Yaochu Jin, Yew-Soon Ong, and Bernhard Sendhoff. Generalizing Surrogate-Assisted Evolutionary Computation. *IEEE T. Evolut. Comput.*, 14(3):329–355, June 2010.

[21] R Timothy Marler and Jasbir S Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6):853–862, 2010.

[22] Yew-Soon Ong, Zongzhao Zhou, and Dudy Lim. Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *Proc. IEEE WCCI*, pages 2928 –2935, 2006.

[23] Martin Pelikan and Kumara Sastry. Fitness Inheritance in the Bayesian Optimization Algorithm. In *Proc. GECCO*, pages 48–59, 2004.

[24] Roberto Santana, Concha Bielza, Jose A. Lozano, and Pedro Larrañaga. Mining probabilistic models learned by EDAs in the optimization of multi-objective problems. In *Proc. of the 11th Annual Conf. on Genetic and Evolutionary Comp. (GECCO 2009)*, pages 445–452, New York, NY, USA, 2009. ACM.

[25] Mark R.N. Shackelford and Christopher L. Simons. Metaheuristic design pattern: interactive solution presentation. *Proc. GECCO Comp. 2014*, pages 1431–1433, 2014.

[26] Siddhartha K. Shakya, John A. W. McCall, and Deryck F. Brown. Solving the Ising Spin Glass Problem using a bivariate EDA based on Markov random fields. In *Proc. IEEE WCCI*, pages 908–915. IEEE Press, 16-21 July 2006.

[27] Robert E. Smith, B. A. Dike, and S. A. Stegmann. Fitness inheritance in genetic algorithms. In *SAC '95: Proc. of the 1995 ACM symp. on applied computing*, pages 345–350, New York, NY, USA, 1995. ACM Press.

[28] Jürgen Van Gorp, Johan Schoukens, and Rik Pintelon. Adding Input Noise to Increase the Generalization of Neural Networks is a Bad Idea. In *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling, Leuven, Belgium*, pages 127–132, 1998.

[29] Tony Wauters, Wim Vancroonenburg, and Greet Vanden Berghe. A Guide-and-Observe Hyper-Heuristic Approach to the Eternity II Puzzle. *Journal of Mathematical Modelling and Algorithms*, 11(3):217–233, Feb 2012.

[30] John Woodward, Jerry Swan, and Simon Martin. The 'Composite' Design Pattern in Metaheuristics. In *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp '14, pages 1439–1444, New York, NY, USA, 2014. ACM.