

Stochastically Generated Trees and Non-Linear Time-Cost ¹

Louis Steinberg

Department of Computer Science, Rutgers University New Brunswick, NJ 08903
lou@cs.rutgers.edu

This Statement of Interest will discuss three aspects of the work at Rutgers on utility-guided search: the class of problems we have focused on, the way we model utility, and the the way we estimate and use utility to guide search.

Searching Stochastically-Generated Trees

First we will summarize the class of problems that we are focusing on.

In many kinds of engineering design tasks the design process involves working with candidate designs at several different levels of abstraction. E.g., in designing a microprocessor, one might start with an instruction set, implement the instructions as a “netlist” defining how specific circuit modules are to be wired together, etc.

A typical way for a program to produce a design at one level from a parent design at the previous level is to first translate the parent design into a correct but poor quality design at the lower level and then to optimize this design. Often the optimization is done by a stochastic process such as a genetic algorithm, simulated annealing, or random-start hill climbing. Since this process is stochastic, it is possible to run it multiple times on the same parent design and get multiple child designs. Each of these children can, in turn, serve as a parent to generate designs at the next level down. Thus, the design space is a (virtual) tree. Design in such a domain can be seen as a process of searching this tree for a high-quality leaf, e.g., for a leaf that represents a fast, cheap microprocessor.

This problem has a number of features that distinguish it from other tree search problems: (a) The branching factor is high (in the thousands for our example domains) (b) the cost to generate a single child is high (from half a second to tens of minutes in domains we have looked at); (c) we do not have an operator that will generate children in some systematic order, we can only get a random child; and (d) there is no concern for minimizing path length from the root to the goal node. All that matters is the quality of the design returned and the total amount of work involved in finding this design.

Furthermore, while we assume that we will be give a heuristic evaluation functions that can be used to compare design alternatives within a level, we cannot assume that these functions are comparable across levels. In engineering domains such as microprocessor design, the alternatives at different levels are of entirely different types and cannot easily be compared.

We use the term *Stochastically Generated (SG) trees* to refer to trees with these characteristics.

Utility with Non-Linear Time Cost

Next, we will discuss our model of utility. Our earlier work on utility guided search [Steinberg *et al.*, 1998] followed the approach taken by Russell and Wefald [Russell and Wefald, 1991], among others, in using a model of utility based on linear time costs. That is, the utility of a result was defined as

$$\text{intrinsic value of result} - \text{cost of time}$$

where the “intrinsic value of result” depends on the quality of the final design but not on when it is produced, while the cost of time is some fixed cost per unit of time the design process takes. While this formulation of utility has been a fruitful one for research, in the real world the cost of time is rarely linear. The cost of taking an extra week to finish a design, for instance, is usually quite different if the extra week is the week before the due date than if it is the week after the due date. In fact, the notion of a “due date” or “deadline” cannot even be expressed in a linear time-cost setting.

Therefore, in our more recent work [Steinberg and Rasheed, 1999] we have used a more general model of utility, allowing the utility of a design d to be a user-defined function of the form $U(S(d), t)$ where $S(d)$

¹The work presented here is part of the “Hypercomputing & Design” (HPCD) project, and benefited greatly from both the intellectual and software environments provided by our colleagues on that project. Special thanks also to Prof. Robert Berk for his contributions on the statistical theory.

The work presented here is part of the “Hypercomputing & Design” (HPCD) project; and it is supported (partly) by ARPA under contract DABT-63-93-C-0064 and by NSF under Grant Number DMI-9813194. The content of the information herein does not necessarily reflect the position of the Government and official endorsement should not be inferred.

is a real-valued “score function” that measures the intrinsic quality of design d (by convention, lower scores are better) and t is the amount of time spent finding this solution; we only require that U^0 is monotonic non-increasing in $S(d)$ and in t and that $\exists t_0 \forall t > t_0 \forall s U(s, t) = 0$, that is, there is a deadline t_0 beyond which a result, no matter how good, has no value.

Estimating Utility and Guiding the Search

We assume $U^0(S^0(d), t)$, the utility of having a lowest-level design alternative d at time t is given, as well as a heuristic score function $S_i(d)$ and the time, t_g^i , it takes to generate a child design at level i , for each level i .

For each design alternative d_i that we have generated, we estimate $U^i(d_i, t)$, the utility of a design process that starts working from design d_i on level i at time t , i.e., the utility of searching the subtree of design alternatives for which d is the root. These utility estimates are based, in turn, on estimates of $G(s|d)$, the probability distribution of the scores of the children of a design d . That is,

$$G(s|d) = P(S(d') = s \mid \text{parent}(d') = d)$$

We also estimate

$$G(s|S(d) = s_p) = P(S(d') = s \mid S(\text{parent}(d') >) = s_p)$$

which depends only on the score s_p of the parent, not on any other information about the parent.

For a discussion of how we estimate $G(s|d)$ and $G(s|S(d) = s_p)$ see [Steinberg *et al.*, 1998].

At time t , what is the utility of searching under a design alternative d_i at level i , where i not the lowest level? Suppose we generate a child d_{i-1} . The time will then be $t + t_g^{i-1}$. If we continue by searching under d_{i-1} , the utility of the search will be $U^{i-1}(d_{i-1}, t + t_g^{i-1})$. If we instead continue generating children from d_i , the utility will be $U^i(d_i, t + t_g^{i-1})$. Presumably we will make the choice with the higher utility, so

$$U^i(d_i, t) = \max(U^i(d_i, t + t_g^{i-1}), U^{i-1}(d_{i-1}, t + t_g^{i-1}))$$

But since at time t we have not generated d_{i-1} yet, we have to average this formula over all possible children. Actually, what we do is to take an average over all possible child scores, weighted by $G(s|d)$. Thus,

$$U^i(d_i, t) = \int G(s|d) \max(U^i(d_i, t + t_g^{i-1}), U_s^{i-1}(s, t + t_g^{i-1})) ds$$

where $U_s^{i-1}(s, t)$ is the expected utility of searching under an arbitrary design at level $i - 1$, given that the score of this design is s . By an argument similar to the one above,

$$U_s^i(s_p, t) = \int G(s|S(d) = s_p) \max(U_s^i(s_p, t + t_g^{i-1}), U_s^{i-1}(s, t + t_g^{i-1})) ds$$

These recurrence relations terminate at $t > t_0$ and at U^0 , and can be translated directly into recursive code to calculate $U^i(d_i, t)$.

The search process is then simple best-first search using $U^i(d_i, t)$ to define “best”. At each iteration we choose the alternative with the highest utility and generate one child from it. If the chosen alternative is at the lowest level we stop and return that alternative as the result.

We have tested this method in two very different domains, with very good results - see [Steinberg and Rasheed, 1999, Steinberg, 2000] for details.

References

- [Russell and Wefald, 1991] Stuart Russell and Eric Wefald. *Do the Right Thing*. MIT Press, 1991.
- [Steinberg and Rasheed, 1999] Louis Steinberg and Khaled Rasheed. Optimization by searching a tree of populations. In *Genetic and Evolutionary Computation Conference GECCO'99*, 1999.
- [Steinberg *et al.*, 1998] Louis Steinberg, J. Storrs Hall, and Brian Davison. Highest utility first search across multiple levels of stochastic design. In *Proceedings of the Fifteenth National Conference on AI*, pages 477–484, Madison, July 1998.
- [Steinberg, 2000] Louis Steinberg. Utility guided stochastic optimization. in preparation, 2000.