An Empirical Analysis of Local Search in Stochastic Optimization for Planner Strategy Selection

Barbara Engelhardt, Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
{firstname.lastname}@jpl.nasa.gov

Abstract

Optimization of expected values in a stochastic domain is common in real world applications. However, it is often difficult to solve such optimization problems without significant knowledge about the surface defined by the stochastic function. In this paper we examine local search techniques to solve stochastic optimization. In particular, we analyze assumptions of smoothness upon which these approaches often rely. We examine these assumptions in the context of optimizing search heuristics for a planner/scheduler on two problem domains. We compare three search algorithms to improve the heuristic sets and show that two local search algorithms perform well and also present empirical data that suggests this is due to smoothness properties of the search space.

1. Introduction

In many optimization applications, the optimization problem is made more difficult because the cost of determining the utility of a solution is expensive (e.g., high computational cost, limited data). This problem is exacerbated in stochastic domains where numerous samples are often required to accurately estimate the expected value (which is usually the optimization target) based on a probabilistic decision criteria. In many such applications, high dimensionality (e.g., large search space) and complex optimization spaces (e.g., non-convex) combine to make the problem difficult.

For many large-scale problems, local search and iterative improvement algorithms have been effective in finding good solutions. In particular, many gradient following approaches have been successfully applied to difficult real-world optimization problems [0]. However, these approaches rely on properties of the search space: that the surface has some notion of smoothness to enable the gradient to lead to a local maxima; and that a local maxima is likely to produce an adequate value. Furthermore, since the particular optimization approach often defines the search operators, it also defines the locale of the strategy search space. Consequently, some optimization strategies would result in a search space with smoothness properties while other generation strategies would not.

We examine this general approach applied to learning heuristics to guide search for a planner/scheduler that solves problems from a fixed but unknown problem distribution. We study the effectiveness of local search for optimizing planner strategies, where a strategy encodes the decision policy for the planner at each choice point in the search. In particular, we examine several issues of general interest:

1. We show that two different local search stochastic optimization methods find strategies that significantly outperform both the human expert derived strategy and a non-local search strategy.
2. We show that the smoothness property holds for both local search algorithms (despite their searching two quite different spaces)
3. Surprisingly, examining the learning trials showed that the learning runs had to modify the initial strategies considerable before showing significant improvement. This either meant that the learning algorithms were making poor initial steps and better later ones, or that the learned strategies lay within a valley. We show empirical results that show that the latter hypothesis is true.

Because our approach is modular to allow arbitrary candidate generation algorithms, we are able to examine the problem for vastly different generation strategies. In particular, we examine a local beam-search candidate generation strategy and an evolutionary computation strategy.

The remainder of this paper is organized as follows. First, we describe the general approach to stochastic optimization. Second, we describe how the planning application is an instance of stochastic optimization. As part of this, we describe the specifics of the control strategy encodings. Third, we describe the empirical results, focusing on the hypotheses outlined above. Finally, we describe related and future work in this area.

2. Stochastic Optimization

We now describe our general iterative framework for optimization of expected value in stochastic domains. First, hypotheses are generated by a local search, then these hypotheses are evaluated by testing them in the application domain and scoring the result (see Figure 1). This testing occurs under the direction of a statistical evaluation component (described below). When the best one or several hypotheses are known with the desired confidence, the process is repeated (e.g., generation of new hypotheses). This entire cycle is repeated until some termination condition is met (e.g., number of cycles, quiescence).

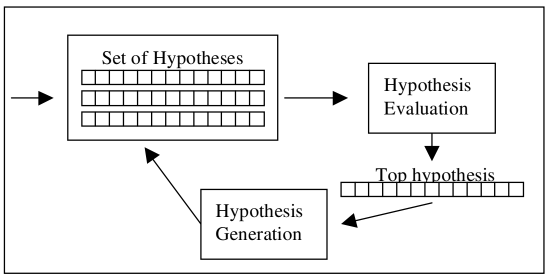

Figure 1 : Optimization cycle - given a set of hypotheses, ranks these hypothese, and generates a next generation based on the rank of the previous generation

To evaluate the set of candidate hypothesis steps, we use statistical methods that minimize resources used to satisfy a decision criteria [1][1]. While the algorithm can use an arbitrary decision criterion, in this paper we focus on the use of the Probably Approximately Correct (PAC) requirement, to determine when the utility of one hypothesis is superior to another based on pair-wise comparisons. With the PAC decision requirement, and algorithm must make decisions with a given confidence (expressed that the probability that its selection is correct is greater than δ) to select the and appropriate hypothesis (expressed that its utility must be within ε of the true best hypothesis) as expressed in Equation (1). Because any specific decision either satisfies or does not satisfy the requirement that the selected hypothesis is within ε of the true best hypothesis, the PAC requirement specifies that over a large number of decisions that the accuracy rate must meet δ. For a pair of distributions, it is relatively straightforward to calculate the probability that one has a higher than the other. However, for selection of a single hypothesis from a set of n hypotheses requires summation of a number of pairwise comparisons. To minimize resource usage, the algorithm allocates error to each pair-wise comparison based on the estimated cost of samples for

$$(1) \quad \sum_{i=1}^{k-1} \Pr[\hat{H}_i < \hat{H}_{sel} - \varepsilon \mid H_i > H_{sel} + \varepsilon] \leq \delta$$

$$(2) \quad \sum^{k-1} c_{sel,i} n_{sel,i}$$

those hypotheses, and allocates a greater error to costly comparisons. Thus, the overall error criterion is met using the fewest resources possible by minimizing Equation (2) after each sample where c is the cost of the best hypothesis and the cost of the i^{th} hypothesis, and n is the number of samples allocated to the comparison. The number of samples, n, can be generated, given a normal distribution of sample utility, by estimating the difference in expected utility and variance of each hypothesis. For more information regarding these techniques, see [1].

3. Learning Planner Heuristics as Stochastic Optimization

We investigate stochastic optimization in the context of learning control strategies for the ASPEN planner. ASPEN uses heuristics to facilitate the iterative search for a feasible and high utility plan. During each search step, a planner confronts a series of decisions such as which schedule conflict to repair or the action to take to repair it. The planner resolves these choices based on the heuristics and weights for the choice points to apply iterative repair [13] (thus defining the control strategy of the planner and hence the expected utility of the resulting plans).

Specifically, in our setup, a strategy hypothesis is a vectors of a weight for every heuristic function, with a weight of 0 for a heuristic not in use. The utility of a hypothesis can be determined by running the planner using the control strategy hypothesis on a certain problem instance and scoring the resulting plan. A problem generator for each domain provides a stochastic set of problem instances to enhance the robustness of the expected solution for the entire planning domain.

In our ASPEN setup, there are twelve choice points in the repair search space. Higher level choice points include choosing the conflict to resolve and choosing the resolution method, such as preferring open constraints before violated constraints, or preferring to add activities over moving them. Once a resolution method is selected, further choice points influence applications of the choice point (e.g.,

---

[1] In this paper we focus on the candidate hypothesis generation strategies and the outer loop. The statistical evaluation phase of the learning process is described in further detail in [1,2]

where to place a newly created activity and how to instantiate its parameters). For each choice point, there are many heuristics that might be used. The hypothesis vector is the list of relative weight that is given to each heuristic for that choice point, and by default the choice is random. Since the planner is stochastic, the order that the heuristics are used at each step is randomized, so multiple runs even for the same problem instance may yield a range of solutions (plans) and hence a range of utilities.

The repair heuristics were developed for individual domain search requirements from ASPEN applications [3]. There are also domain-specific heuristics, which reference particular features of a domain in order to affect the search. For each domain, the human expert strategy hypotheses was derived independently from (and prior to) our study by manual experimentation and domain analysis.

We examine three different spacecraft domains, which satisfy the normality assumption of the evaluation method. The first domain, Earth Orbiter-1 (EO-1), is an earth imaging satellite. The domain consists of managing spacecraft operations constraints (power, thermal, pointing, buffers, telecommunications, etc.) and science goals (imaging targets and calibrating instruments with observation parameters). Each problem instance is to create a two-day operations plan with: a typical weather and instrument pattern, observation goals (between 3 and 16), and a number of satellite passes (between 50 and 175). EO-1 plans are preferring more calibrations and observations, earlier start times for the observations, fewer solar array and aperture manipulations, lower maximum value over the entire schedule horizon for the solar array usage, and higher levels of propellant. The Comet Lander domain models landed operations of a spacecraft designed to land on a comet and return a sample to earth. Resources include power, battery, communications, RAM, communications relay in-view, drill, and ovens. Science includes mining and analyzing a sample from the comet, and imaging. The problem generator includes between 1 and 11 mining activities and between 1 and 24 imaging activities at random start times. The scoring functions for the Comet Lander domain includes preferences for more imaging activities, more mining activities, more battery charge over the entire horizon, fewer drill movements, and fewer uplink activities.

The two local search types used were a local beam search method and an evolutionary computation method. The local beam search [9] defines a vector's neighborhood as changing the subset of the vector associated with a choice point by less than a certain step size. As opposed to propagating only highest-ranking vector, the search propagates a beam *b* of vectors, where *b* is greater or equal to 1. Samples for each individual candidate hypothesis are generated and scored using the planner, and ranking is done by pair-wise comparisons of these sample utilities for each candidate hypothesis in a generation. For each generation, the beam search takes the top ranking *b* hypotheses, creates *b/g* candidate neighbor hypotheses for each of them, and ranks the *g* candidate hypotheses to create the subsequent generation.

The evolutionary algorithm [5] uses three general operators (crossover, mutation, and reproduction) to generate the next set of hypotheses. Parents are chosen based on their relative ranking, where the higher-scoring hypotheses are more likely to be parents. The crossover operator was not aware of subsets of the hypothesis vector related to each choice point, so it could choose to split within one of those subsets. For all operators, the results are normalized to 100% before evaluation. Samples for each individual candidate hypothesis are generated and scored using the planner, and ranking is done by pair-wise comparisons of these sample utilities for each candidate hypothesis in a generation. For each hypothesis in a generation, the algorithm either reproduces one parent or crosses two parents based on their ranking in the previous generation, and mutates the resulting candidate hypothesis.

Random sampling is another (non-local) method of search. Vectors are generated at random and deep sampling is performed on these vectors for a planning domain. The results show a distribution of random hypothesis points and expected utility for these random points in the strategy space.

Although the local search algorithms are greedy given a correct ranking, due to sampling error the ranking algorithm can produce only an approximation of the correct ranking. Furthermore, as the overall utility of the candidate hypotheses continues to improve, ranking is more difficult because the hypotheses have higher variances relative to the differences in the mean (this is a phenomenon well understood related to the Least Favorable Configuration (LFC) in statistical ranking). Consequently, the highest overall expected utility hypothesis might not occur in the final iteration, and the optimization algorithm does not know the true utilities of the strategies sampled (it only has estimates). To address this problem, each of our algorithms (beam-search and evolutionary) select the highest estimated utility strategy from all seen during that run (e.g., potentially not the last strategy). When we report that strategy's utility, we report a true utility based on a deep sample of many more samples. Since each run takes several CPU days, we are continuing to perform more optimization runs to provide more detailed results.

4. Empirical Results

One simple question is "Can the local optimization techniques improve on the human expert strategies? In both the EO-1 domain and the Comet Lander domain, we compare expected utilities of: the handcrafted expert strategy and best and average strategies found by: random sample in Table 1. For local beam search and local genetic search we report on the top strategy in the final set of strategies (recall that the beam has several strategies retained and the genetic search has the population) as well as the average of the strategies in the final set. While the learned strategies outperformed the expert strategies, surprisingly the expert strategy in the EO-1 domain was worse than a randomly generated strategy. This is because the best strategies tend to be highly stochastic (e.g., have a strategy for a choice point where two heuristics are reasonably likely to be selected, such as 60% A 40% B). In contrast, we have observed that human experts have a tendency to develop choice point heuristics that are less stochastic (e.g., 95% A 5% B).

	Expert	Random Sample		Local Beam Search		Genetic Search	
Domain		High	Mean	High	Mean	High	Mean
Comet Lander	0.538	0.531	0.461	0.584	0.549	0.593	0.569
EO-1	0.161	0.341	0.196	0.446	0.426	0.444	0.382

Table 1: Summary Utility Results

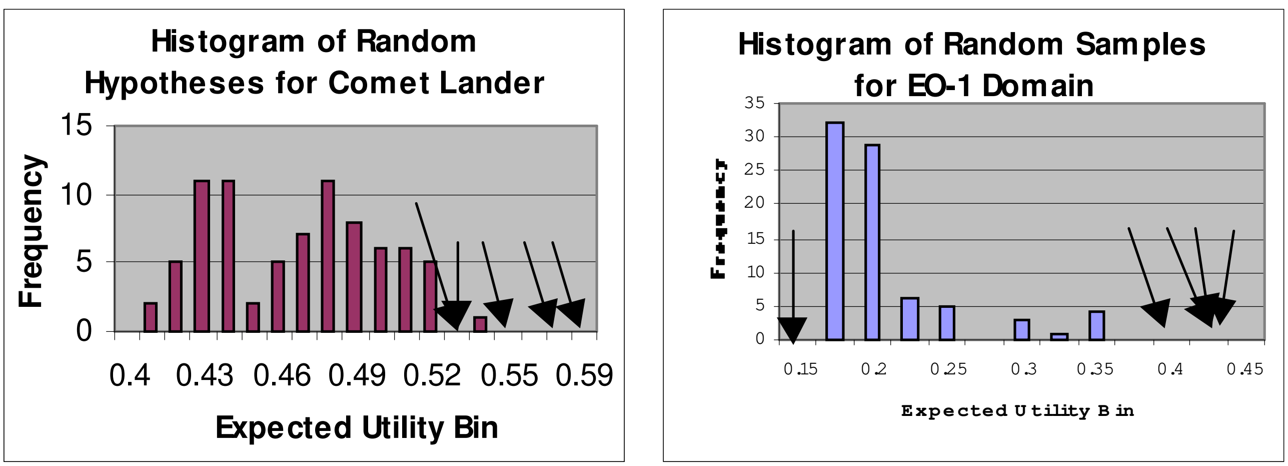

Table 2: Histogram Summaries

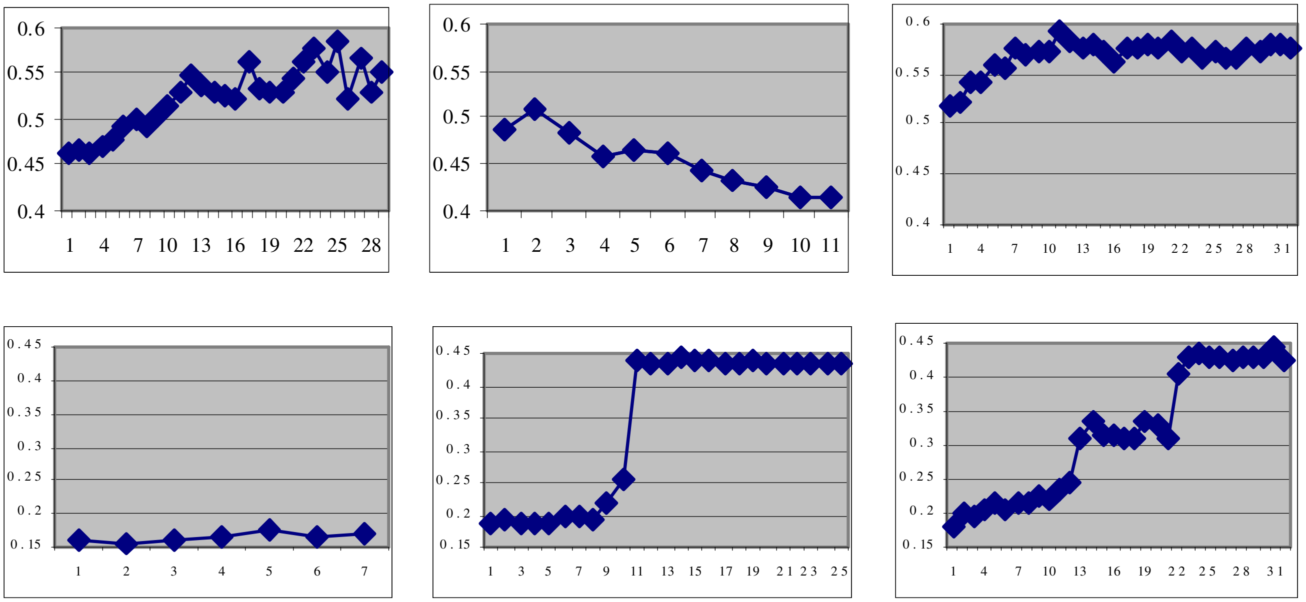

Table 3: Utility v. Iteration- Lander Random Walk Beam , Lander random walk GA, Lander GA
EO1 Random Walk Beam, EO1 Beam, EO1 GA

The results show that the local search optimization was able to find strategies that significantly improved on the expert strategies. We plot histograms (Table 2) for randomly selected strategies in the Comet Lander and EO-1 domains (key values from Table 1 - expert, and learned strategies) indicated with arrows). These show that the local search optimization techniques found very good strategies overall in the space, among the best possible strategies. The traces of the two local search techniques operating on each of the domains is shown below (e.g., deep sample utility versus iteration). The shapes of these graphs (little early improvement) led us to believe that the expert strategies are located in a valley of the search space. In order to test this conjecture, we generated random walks in the strategy spaces. This data (Table 3 graphs towards left) confirms that the expert strategies lay in a valley but that sufficient gradient information existed to allow the learning to escape the valley. One potential explanation could be that the variance of the problems from a single domain requires a large amount of flexibility in the planner heuristics (e.g. stochasticity), whereas the expert designed the set of heuristics such that it would choose a single non-random strategy for each choice point every time (because it is easier to understand such a strategy).

How did the local search techniques find their way out of the valley? Local search algorithms are effective on these domains if the search spaces are smooth with respect to the candidate hypothesis generation functions. Smoothness in a discrete domain can be determined by measuring the difference in expected utility between adjacent points, (wrt a search step definition) is small compared to the difference in expected utility between two randomly selected points in the search space. Table 2 shows the relative smoothness of the two domains for the three search algorithms. For random search, adjacent points are any two vectors in the strategy space (Genetic Search EO-1 data forthcoming). The average difference in expected utility is measured between two adjacent points, where the initial point is a randomly generated hypothesis, and the adjacent point is one step (as defined by the candidate hypothesis generation function) from that point.

5. Related Work, Future Work, and Conclusions

There is significant related work on efficient search techniques. The Q2 algorithm optimizes the expected output of a noisy continuous function, but does not have guarantees on the results [14]. Response Surface Methods [0] have been applied to optimization problems in continuous domains, our but require modification for discrete domains (as in our planning heuristics domain). Evaluating control strategies is a growing area of interest. Horvitz [6] described a method for evaluating algorithms based on a cost versus quality tradeoff. Russell, Subramanian, and Parr[10] used dynamic programming to rationally select among a set of control strategies by estimating utility, including cost. MULTI-TAC [8] considers all k-wise combinations of heuristics for solving a CSP in its evaluation, which also avoids problems with local maxima, but at a large expense to the search. Previous articles describing work in adaptive solving described general methods, which have been developed for transforming a standard problem solver into an adaptive one. Gratch & Chien [5a] illustrated the application of adaptive problem solving to real world scheduling problems and showed how adaptive problem solving can be cast as a resource allocation problem. Zhang and Dietterich used reinforcement learning to learn applicability condition for scheduling operators, using a sliding time window of applicability for those operators [12]. Our optimization approach is equivalent to learning a naïve bayesian model using an Expectation Maximization approach [15,16]. One difference is that our model attempts to minimize resource usage by updating the model after each sample, as opposed to sampling in bulk, simply because of the high sample cost and the low cost to update the model.

Future work includes determining how to adjust search rates, learning composite strategies which involve logical decisions about the relative usage of heuristics as opposed to statistical methods, and a portfolio approach, which combines heuristics and chooses which set to use based on domain features judged statically or at run time. Additional work has been proposed for hypothesis evaluation based on a different set of stopping criteria, which can be resource bounded (specifically considering time as the resource), as in previous works on a similar topic [4].

In this paper we have focused on selecting the planner strategy with the highest expected utility. However other aspects of the strategy might be relevant. Fore example, consistent (e.g., predictable) performance might be desired. In this case probabilistic decision criteria incorporating undesirability of a high utility variance strategy would need to be used. Specifically, the PAC requirement does not incorporate any preference or disliking for high variance strategies.

This paper has presented an approach to optimization of expected values in a stochastic domain is common in real world applications. Specifically, we presented an approach based on local search of the optimization space. We presented empirical results from an application to learning planner heuristics in which learned strategies significantly outperformed human expert derived strategies. And we also presented empirical evidence that these local search techniques performed well because smoothness properties held in these applications.

Domain	**Random Search**		**Local Beam Search**		**Genetic Search**	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
Comet Lander	0.0435	0.0293	0.0086	0.0066	0.0134	0.0093
EO-1	0.0442	0.0466	0.0145	0.0244	-	-

Table 2: Mean Δexpected-score between adjacent compared to random points

7. Bibliography

[0] Box, G.E.P., Draper, N. R. 1987. *Empirical Model-Building and Response Surfaces.* Wiley.

[1] Chien, S., Gratch, J., Burl, M. 1995. "On the Efficient Allocation of Resources for Hypothesis Evaluation: A Statistical Approach." In *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(7), p. 652-665.

[2] Chien, S., Stechert, A., Mutz, D. 1999. "Efficient Heuristic Hypothesis Ranking." *Journal of Artificial Intelligence Research* Vol 10: 375-397.

[3] Chien, S., Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., Tran, D. 2000. "ASPEN – Automating Space Mission Operations using Automated Planning and Scheduling." *SpaceOps 2000*, Toulouse, France.

[4] Fink, E. 1998. "How to Solve it Automatically: Selection among Problem-Solving Methods." In *Proceedings of the Fifth International Conf.AI Planning Systems*, 128-136.

[5] Goldberg, D. 1989. *Genetic Algorithms: In Search, Optimization and Machine Learning*. Reading,

Massachusetts: Addison-Wesley.

[5a] J. M. Gratch and S. A. Chien, "Adaptive Problem-solving for Large Scale Scheduling Problems: A Case Study," Journal of Artificial Intelligence Research Vol. 4 (1996), pp. 365-396.

[6] Horvitz, E. 1988. “Reasoning under Varying and Uncertain Resource Constraints.” In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 111-116.

[7] Lin, S., and Kernighan, B. 1973. “An Effective Heuristic for the Traveling Salesman Problem,” *Operations Research Vol. 21.*

[8] Minton, S. 1996. “Automatically Configuring Constraint Satisfaction Programs: A Case Study.” In *Constraints* 1:1(7-43).

[9] Russell, S., Norvig, P. 1995. *Artificial Intelligence: A Modern Approach.* Upper Saddle River, NJ: Prentice Hall.

[10] Russell, S., Subramanian, D., Parr, R. 1993. “Provably Bounded Optimal Agents.” *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.*

[12] Zhang, W., Deitterich, T.G., (1996). “High-Performance Job-Shop Scheduling With a Time-Delay TD(λ) Network *Proc. NIPS* 8, 1024-1030.

[13] Zweben, M., Daun, B., Davis, E., and Deale, M. 1994. “Scheduling & Rescheduling with Iterative Repair.” In *Intelligent Scheduling*. Morgan Kaufmann. 241-256.

[14] Moore, A., Schneider, T., Boyan, J., Lee, M. S. 1998. “Q2: Memory-based Active Learning for Optimizing Noisy Continuous Functions*.” Proc. ICML 1998.*

[15] Heckerman, David. 1996. A Tutorial on Learning with Bayesian Networks. MSR-TR-95-06, Microsoft Corporation.

[16] Bishop, C. 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.