Robust local search for spacecraft operations using adaptive noise

Alex S. Fukunaga, Gregg Rabideau, and Steve Chien
Jet Propulsion Laboratory
Californnia Institute of Technology
4800 Oak Grove Dr., M/S 126-347
Pasadena, CA 91109-8099
firstname.lastname@jpl.nasa.gov

Abstract.
Randomization is a standard technique for improving the performance of local search algorithms for constraint satisfaction. However, it is well-known that local search algorithms are sensitive to the noise values selected. We investigate the use of an adaptive noise mechanism in an iterative repair-based planner/scheduler for spacecraft operations. Preliminary results indicate that adaptive noise makes the use of randomized repair moves safe and robust; that is, using adaptive noise makes it possible to consistently achieve performance comparable with the best tuned noise setting without the need for manually tuning the noise parameter.

1 Introduction

Local search has been shown to be an effective algorithm for solving many difficult constraint satisfaction and combinatorial optimization problems. Deployed space applications that apply local search include the Hubble scheduler (Johnston and Miller 1994) and the on-board CASPER scheduler in the EO-1 Autonomous Science Experiment (Chien *et al.* 2003).

The performance of local search in any given domain usually depends greatly on the quality of domain-specific *heuristics* that guide the behavior of the algorithm. Such heuristics include procedures for selecting the next variable to modify, as well as procedures for selecting new values to apply to the selected variable.

Some local search heuristics are *domain-independent*, and can be concisely expressed in abstract terms, such as the ubiquitous *greedy* heuristic (make the change to the schedule that results in the neighboring state with the best objective function), or the well known min-conflicts heuristic (Minton *et al.* 1992) for constraint satisfaction problems (select some variable that is involved in a conflict, and assign a value to v such that the total number of conflicts remaining is minimized).

However, because scheduling problems are generally intractable (NP-hard), such elegant, domain-independent heuristics can not always be guaranteed to yield adequate performance. *Domain-specific* heuristics are usually the product of an iterative process in which domain experts and scheduling algorithm developers collaborate to improve the default behavior of a scheduling algorithm by deriving new domain-specific heuristics (this process is often initiated when it is discovered that domain-independent heuristics by themselves are failing to perform adequately in practice). Domain-specific heuristics for real-world scheduling systems require a more complex, domain-dependent representation. For example, in the ASPEN scheduler, it is possible to implement a heuristic that specifies that a certain type of activity should be scheduled as soon as possible (because it is known that this type of activity tends to generate many constraints on the placement of other activities).

Although scheduling algorithm developers can develop heuristics which are *usually* effective (i.e., the algorithms performs better with the heuristic than without the heuristic), it is inherent in the nature of heuristics (which literally mean "rules of thumb") that they are not *always* effective. For example, local search procedures that depend heavily on greedy heuristics can easily get stuck at local optima. Additional mechanisms are necessary to counterbalance the focusing effect of heuristics. One such mechanism is randomization, or *noise*, i.e., a mechanism that sometimes (with probability p) forces the local search algorithm to make a random move instead of one that would be prescribed by its heuristics. It is well-known that noise can significantly improve the performance of local search. For example, empirical studies such as (Selman and Kautz 1993; Selman *et al.* 1994) have shown that the addition of randomized moves significantly improves the performance of SAT local search.

Noise mechanisms have traditionally been *static*. That is, the probability of making a random noise p at any given point in time is determined a priori by setting the noise parameter p before the search algorithm is run. However, it has been shown that the performance of local search mechanisms is significantly affected by the value of this static noise parameter (c.f. (McAllester *et al.* 1997)). It was recently proposed that an adaptive noise mechanism which automatically adjusts the noise level depending on the perceived progress of the search algorithm may result in performance that is comparable to, or even better than a hand-tuned static

noise setting (Hoos 2002).

In this paper, we consider the problem of enhancing the robustness and performance of a scheduling algorithm for spacecraft operations using noise mechanisms. First, we give a brief overview of the ASPEN system for spacecraft operations planning/scheduling. Then, we characterize the impact of the standard, static noise on ASPEN local search for two prototype domains. Finally, we show that an adaptive noise mechanism can achieve performance that is competitive with static noise but is significantly more robust.

2 Iterative Repair Scheduling in ASPEN

ASPEN is a planning and scheduling system for spacecraft operations (Chien *et al.* 2000), which has recently been deployed on board the EO-1 earth imaging satellite (Chien *et al.* 2003). In an ASPEN schedule, anything that indicates an "incomplete" or "unsatisfactory" schedule is considered a conflict (e.g., unscheduled activities, oversubscription of resources). The task of the scheduling algorithm is to produce a schedule with no conflicts (or alternatively, produce a schedule that maximizes some objective function (Rabideau *et al.* 2000)).

While ASPEN is an application framework and has been used to implement a range of planning and scheduling paradigms, most commonly users have utilized the iterative repair approach in ASPEN (Rabideau *et al.* 1999). The ASPEN Iterative repair algorithm works as follows: At each step, ASPEN chooses a conflict to resolve by applying a conflict selection heuristic. Then, a *repair method selection heuristic* is applied to decide how ASPEN will attempt to resolve the conflict (e.g., by moving an activity elsewhere, removing/unscheduling an activity, etc). The selected repair method may entail further choices, and at each such *decision point*, decisions are guided by a heuristic that is applicable at that decision point. Thus, each step in ASPEN iterative repair can be characterized by the application of a decision tree, where choices at the decision nodes are guided by a heuristic that applies at that node. The root node is conflict selection, the second level decision node is repair method selection, and so on.

For every decision node, ASPEN implements a domain-independent, *default* heuristic. However, as noted in Section 1, domain-independent heuristics sometimes fail. Therefore, ASPEN provides a mechanism for users to implement a domain-specific heuristic that is used instead of the default heuristic.

We recently performed an analysis of 30 prototype and fielded ASPEN applications in order to identify opportunities for improvements to the scheduling algorithm. This analysis revealed that users tended to rely on the default heuristic for most of the decision points. However, we found that in more than a third of the applications, the user had implemented a domain-specific repair method selection heuristic function, indicating that it might be worthwhile to focus our efforts on that particular choice point.

2.1 Experimental Domains

In the rest of the paper, we describe experiments performed with ASPEN in order to improve its repair method selection procedure by adding randomization to the heuristic. The experiments are performed on two prototype domains, ST-4 and EO-1.

The ST-4 domain models the landed operations of a spacecraft designed to land on a comet and return a sample to earth. The model has 6 shared resources, 6 state variables, and 22 activity types. Resources and states include battery level, bus power, communications, orbiter-in-view, drill location, drill state, oven states for a primary and backup oven state, camera state, and RAM. There are two activity groups that correspond to different types of experiments: mining and analyzing a sample, and taking a picture. The instance used in this paper has 4 mining activities and 5 picture experiments to be scheduled.

The EO-1 domain was a early prototype for the recently deployed ASE on-board scheduler (Chien *et al.* 2003). EO-1 is an earth imaging satellite featuring an advanced multispectral imaging device. EO-1 mission operations consists of managing spacecraft operability constraints (power, thermal, pointing, buffers, consumables, telecommunications, etc.) and science goals (imaging of specific targets within particular observation parameters). Of particular difficulty is managing the downlinks as the amount of data generated by the imaging device is quite large and uplink opportunities are a limited resource. The EO-1 domain models the operations of the satellite for a two-day horizon. It consists of 14 resources, 10 state variables, and 38 different activity types. The instance used in our experiments includes 7 downlinks.

3 Adding noise to repair method selection

It is well-known that adding randomness, or *noise* to a local search algorithm can potentially improve its performance. That is, instead of always making a decision based on a heuristic, the search algorithm can make a random move with probability p. Traditionally, this noise mechanism described above is *static* – the value of p is determined prior to running the algorithm, and does not change during the run.

ASPEN provides a randomization parameter for each decision point, and a majority of ASPEN applications have applied noise at the conflict selection and/or repair method selection heuristics. In the case of the repair method selection heuristic, a "random decision" means that instead of choosing the repair method suggested by the heuristic, one of the applicable repair methods for a conflict type is selected randomly (Figure 1 shows each of the the conflict types linked to each of the applicable repair methods applicable to them).

Figures 2 and 3 shows the *noise response* of ASPEN local search on one instance each of the DS-4 domain and EO-1

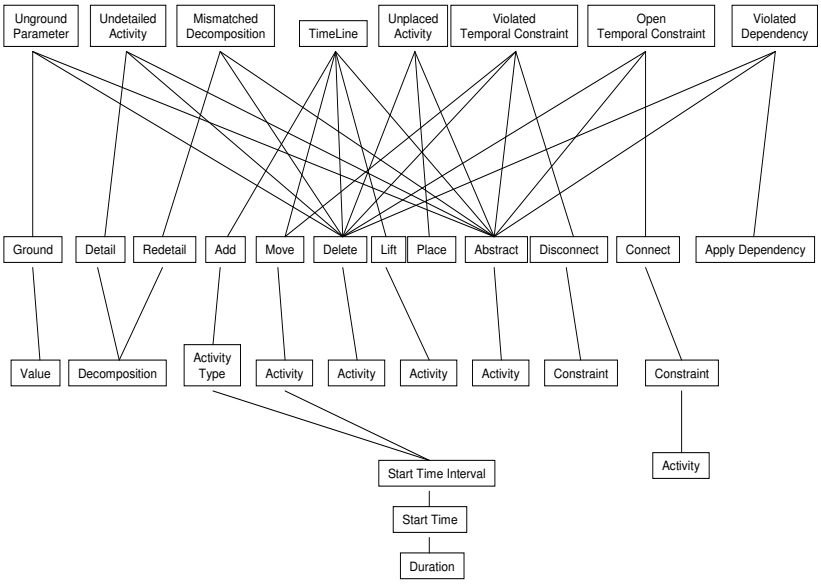

Figure 1: Hierarchy of ASPEN iterative repair decision points. The top row shows the conflict types (conflict selection decision chooses one of these). The second row shows the repair methods available (linked to the conflicts to which they are applicable). The third row and below show further decision points available, depending on the type of repair method.

domain, respectively.[1] The `default-heuristic` lines show the mean runtime required to solve the problem instance as p was varied.[2].

Figure 2 also includes a line showing the performance of ASPEN iterative repair using a hand-coded, domain-specific repair method selection heuristic. In this case, adding static noise to this domain-specific heuristic only degrades the performance, and never helps.

Note that the performance of the scheduling algorithm depends significantly on the noise setting Noise values that are too low or too high clearly degrades the performance of ASPEN iterative repair.

4 Adaptive Noise

Although static noise yields significant benefits there seems to be room for improvement. In the previous section, we observed that ASPEN with static noise is very sensitive to the p setting. In our experiments above, we had the luxury of being able to determine the optimal value for p experimentally. In practice, such experimentation is usually not an option. Therefore, a more robust mechanism that does not require extensive tuning is desirable.

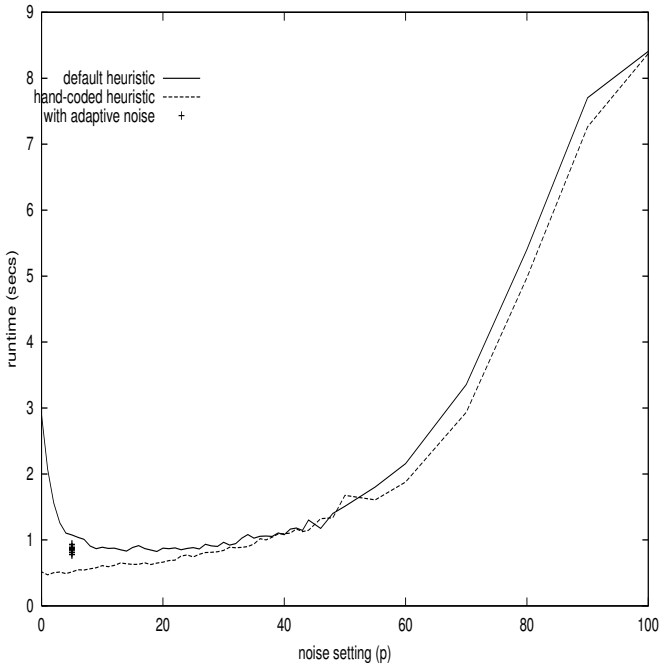

Figure 2: DS-4 Noise response: runtime vs noise level p. N=100

[1] We have repeated all of the experiments here with several instances of each domain, and the results are similar.

[2] In the experiments in this paper, we ran ASPEN in "repair" mode, where a valid "solution" is any schedule which satisfies all constraints, i.e., all valid solutions have the same objective function value. ASPEN also has another, preference-based optimization mode which can make finer distinctions among the set of solutions that satisfy the hard constraints (see (Rabideau *et al.* 2000)

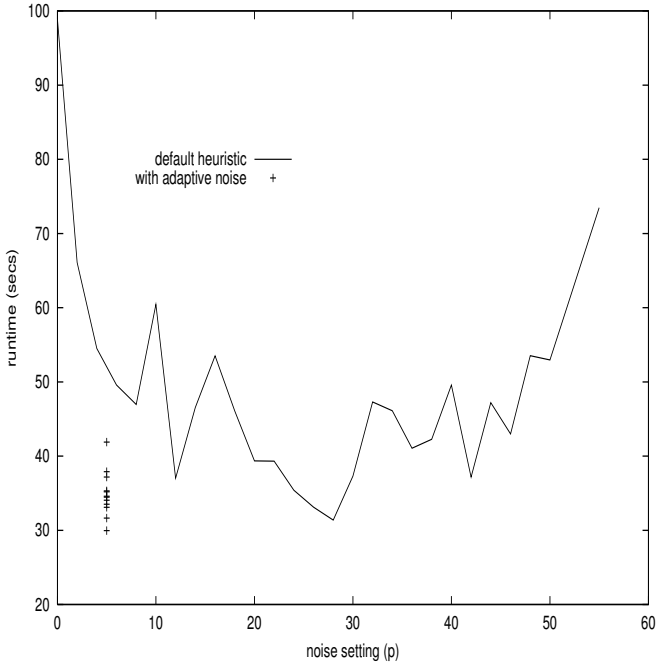

Figure 3: EO-1 Noise response: runtime vs noise level p. N=40

Intuitively, heuristics that have been developed over several years, such as ASPEN's domain-independent heuristics should, in principle, work most of the time. On the other hand, noise provides a last-resort escape mechanism when the heuristic occasionally leads to incorrect behavior. Thus, it seems that we should rely on the heuristics as long as they are working, and use noise only when necessary.

This intuition can be directly implemented as an *adaptive* noise mechanism for iterative repair. Hoos (Hoos 2002) recently proposed an adaptive noise mechanism for SAT local search, which can be viewed as an instance of iterative repair. We implemented Hoos' mechanism in ASPEN, as follows: [3]

At the beginning of the search, the noise p is set to 0 (i.e., at the beginning of the search, we start by relying on the heuristic).

If stagnation is detected (no improvements in the objective function within θ steps), p is increased. When p is increased, it is adjusted using the update formula: $p = p + (1 - p)\phi$. Whenever progress is made, p is updated as: $p = p - 2p\phi$. Increases and decreases in p are asymmetrical. Each time after a noise increase, there is a period of time while the sufficiency of the noise is tested (the delay of θ), whereas noise decreases are applied every time there is an improvement. Also, the magnitude of the noise increases is smaller than the magnitude of the noise decreases.

Adaptive noise uses two control parameters, ϕ and θ, Intuitively, ϕ should be large enough so that increasing the noise level by applying the update formula in Section 4 will result in a significant change. On the other hand, if ϕ is too large, then the resulting behavior will be a degenerate policy where the repair algorithm switches back and forth between heuristic-driven and totally randomized modes. Likewise, θ must be small enough so that noise values can increase rapidly enough to minimize time wasted on fruitless applications the heuristic; yet, θ should be large enough that some time is being spent at the current noise level before further increasing p.

We therefore believe that $0.1 \leq \phi \leq 0.2$ and $5 \leq \theta \leq 40$ covers a *domain-independent* range of intuitively "reasonable" settings for ϕ and θ. More extreme parameter settings would lead to behavior that does not conform with our intuitions of how adaptive noise is intended to behave.

We evaluated adaptive noise on the DS-4 and EO-1 domains, for the cross-product of parameter settings ϕ and θ, where $\phi \in 0.10.15, 0.2, 0.25$, $\theta \in 5, 10, 20, 40$. (i.e., 12 data points, each data point representing $n = 100$ runs for the DS-4 domain, $n = 40$ runs for EO-1 domain).

In Figures 2 and 3, these are shown as the cluster of values shown for $p = 5$ (although p is actually adaptive).

For DS-4, the average performance across all 12 settings was 0.85 seconds, with a standard deviation of 0.05. The worst control parameter setting had a mean runtime of 0.926 seconds, while the best control parameter setting had a mean runtime of 0.777 seconds.

For EO-1, the average performance across the 12 settings was 34.90 seconds with a standard deviation of 3.08 seconds. The worst control parameter setting had a mean runtime of 41.89 seconds, while the best control parameter setting had a mean runtime of 31.64 seconds.

From this we see that the performance of iterative repair with adaptive noise is remarkably robust, clustering close to the performance obtained by the best static noise setting. This performance seems to be quite robust with respect to the ϕ and θ control parameter settings, since the 12 control parameter sets cover the range of reasonable parameter settings.

For DS-4, the domain-independent heuristic combined with adaptive noise achieves performance comparable to that of the hand-coded, domain-specific heuristic.

In addition, we also ran the DS-4 model using adaptive noise using the same 12 sets of control parameter settings. For this combination of hand-coded heuristic + adaptive noise, the average performance across all 12 settings was 0.63 seconds, with a standard deviation of 0.13. The worst control parameter setting had a mean runtime of 0.86 seconds, while the best control parameter setting had a mean runtime of 0.48 seconds. In this case, adaptive noise tends to minimizes the performance degradation observed when

[3] We first implemented a similar mechanism that we developed independently, but then discovered Hoos' work and found that it was simpler and performed just as well.

using static noise.

All experiments reported here were run on a 2.7GHz Pentium-4 processor. This is orders of magnitude faster than on-board processors for spacecraft that are flying presently and in the near future. Furthermore, on-board scheduling systems are usually not allocated 100% of the CPU resources at any given time. In fact, our benchmarks have shown that the Linux workstation used here is 500-1000 times faster than the MIPS R3000 Mongoose 5 running at 12MHz that we are using on the EO-1 to run ASPEN for the ASE experiment (Chien *et al.* 2003). Therefore, depending on processing constraints, the relative disparities between the runtimes for the iterative repair variants considered here can have significant impact on spacecraft operations.

5 Related Work

In addition to the adaptive noise mechanism proposed by Hoos which was used in this work(Hoos 2002), various adaptive mechanisms for setting control parameters for search/optimization algorithms can be found in the Artificial Intelligence and Operations Research literature. We describe some of the most closely related techniques below.

Local search with noise is similar to simulated annealing (Kirkpatrick *et al.* 1983). Simulated annealing (SA) is essentially a local search procedure that will move to a state that has a worse objective function value than the current state with some probability, where the p is dependent on an annealing schedule. The random moves introduced by noise mechanisms in iterative repair are intended to serve a role similar to the probabilistic acceptance of worse states in SA (escaping local optima or cycles). However, the basic repair/local search in ASPEN is inherently non-greedy and very different from SA. The hierarchical scheme of first making a non-greedy commitment to address a particular conflict, means that the resulting state will frequently be worse than the previous state, because all applicable methods of resolving that particular conflict will lead to a worse state.

Another approach to enhancing the performance of heuristic local search scheduling is to learn/adapt a static heuristic strategy to optimize its expected performance on the class of problems for which the solver is intended (Gratch and Chien 1996). This is *off-line* learning approach is complementary to the on-line adaptive approach taken in this paper. For example, the default heuristic could be learned by applying the off-line learning algorithm.

An alternate approach to on-line improvement of local search is STAGE, (Boyan and Moore 2000)— which learns good start states for multi-start local search (local search which periodically "restarts" after local optima are identified). The selection of restart states for local search is orthogonal to move selection, which is addressed by adaptive noise. Note that STAGE depends on the identification of state features that can be used in the functional mapping between search states and objective function values. This may be difficult in general for a system like ASPEN due to the complexity of the state representation, compared to the problems to which STAGE has been applied so far.

6 Conclusions

In this paper, we showed that adding random moves (noise) to iterative repair can significantly improve the performance of an iterative-repair scheduling algorithm. However, the traditional, static noise mechanism requires tuning of the noise parameter, and we showed that in some cases (e.g., on the DS-4 domain using the hand-coded heuristic), non-zero noise can always hurt performance.

We then showed that an adaptive noise mechanism first proposed for satisfiability testing (Hoos 2002) can yield performance comparable to static noise using the optimal parameter noise value, while also demonstrating significant robustness to control parameter variations. This is a promising result, since for on-board applications, a mechanism that can adapt to unforeseen situations is preferable to mechanisms involving a control parameter that must be optimized *a priori*.

In addition, we have shown empirically that while the performance of adaptive noise may not be quite as good as using the optimal static noise setting, the worst-case performance (performance with the worst control parameter settings) is quite close to the best-case performance. This robustness is desirable for on-board operations, where mitigating the worst case-behavior is arguably more important than optimizing best-case behavior.

In this paper, we have considered the application of noise to just one level of decision making in ASPEN heuristic repair. In future work, we will investigate how noise can be simultaneously, automatically tuned for all levels of decision making, including conflict selection, repair method selection, and repair parameter value selection.

Acknowledgments

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

J.A. Boyan and A.W. Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, 1(2), 2000.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, and G. Stebbinsand D. Tran. Aspen - automating space mission operations using automated planning and scheduling. In *International Conference on Space Operations (SpaceOps 2000).*, Toulouse, France, June 2000.

S. Chien, R. Sherwood, R D. Tran, Castano, B. Cichy, A. Davies, G. Rabideau, N. Tang, M. Burl, D. Mandl, S. Frye, J. Hengemihle, J. Agostino, R. Bote, B. Trout, S. Shulman, S. Ungar, J. Van Gaasbeck, D. Boyer, M. Griffin, H. Burke, R. Greeley, T. Doggett, K. Williams, V. Baker, and J. Dohm. Autonomous science on the eo-1 mission. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*, Nara, Japan, 2003.

J. Gratch and S. Chien. Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research*, 4:365–396, 1996.

H. Hoos. An adaptive noise mechanism for walksat. In *Proc. AAAI*, pages 655–660, 2002.

M. Johnston and G. Miller. Spike: Intelligent scheduling of hubble space telescope observations. In M. Zweben, editor, *Intelligent Scheduling*. Morgan Kaufman, 1994.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *Proc. AAAI*, pages 459–465, 1997.

S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Minimizing conflicts: A heuristic method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations in the aspen system. In *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space (I-SAIRAS)*, Noordwijk, The Netherlands, June 1999.

G. Rabideau, B. Engelhardt, and S. Chien. Using generic preferences to incrementally improve plan quality. In *Proc. Fifth International Conference on Artificial Intellignece Planning and Scheduling (ICAPS)*, Breckenridge, CO, 2000.

B. Selman and H. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proc. AAAI*, 1993.

B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proc. AAAI*, 1994.