Active Statistical Learning of Heuristic Scheduling Strategies

Jonathan M. Gratch[2], Steve A. Chien[1], and Darren Mutz[1]

[1] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA

[2] Information Sciences Institute, University of Southern California, Marina del Rey, CA

Email: {steve.chien,darren.mutz}@jpl.nasa.gov, gratch@isi.edu

Abstract

Although the general class of most scheduling problems is NP-hard, in practice, domain-specific techniques frequently solve problems in much better than exponential time. Unfortunately, constructing special-purpose systems is a knowledge-intensive and time-consuming process that requires a deep understanding of the domain and problem-solving architecture. In *adaptive problem-solving* the system automatically learns an effective domain-specific search strategy given a general problem solver with a flexible control architecture. In this approach, a learning system explores a space of possible heuristic methods for one well-suited to the eccentricities of the given domain and problem distribution. In this article, we discuss an application of the approach to scheduling satellite communications. Using problem distributions based on actual mission requirements, our approach identifies strategies that not only decrease the amount of CPU time required to produce schedules, but also increase the percentage of problems that are solvable within computational resource limitations.

1 Introduction

With the maturation of automated problem-solving research has come grudging abandonment of the search for "the" domain-independent problem solver. General problem-solving tasks like planning and scheduling are provably intractable. Although heuristic methods are effective in many practical situations, an ever growing body of work demonstrates the narrowness of specific heuristic strategies (e.g., (Baker, 1994, Frost & Dechter, 1994, Kambhampati, Knoblock & Yang, 1995, Stone, Veloso & Blythe, 1994, Yang & Murray, 1994)). Studies repeatedly show that a strategy that excels on one task can perform abysmally on others. These negative results do not entirely discredit domain-independent approaches, but suggest that considerable effort and expertise is required to find an acceptable combination of heuristic methods, a conjecture supported by the few published accounts of real-world implementations (e.g. (Wilkins, 1988)). The specificity of heuristic methods is especially troubling when we consider that problem-solving tasks frequently change over time. Thus, a heuristic problem solver may require expensive "tune-ups" as the character of the application changes. *Adaptive problem solving* is a general method for reducing the cost of developing and maintaining effective heuristic problem solvers. Rather than forcing a developer to choose a specific heuristic strategy, an adaptive problem solver adjusts itself to the idiosyncrasies of an application. This can be seen as a natural extension of the principle of least commitment (Sacerdoti, 1977). When solving a problem, one should not commit to a particular solution path until one has information to distinguish that path from the alternatives. Likewise, when faced with an entire distribution of problems, it makes sense to avoid committing to a particular heuristic strategy until one can make an informed decision on which strategy performs better on the distribution. An adaptive problem solver embodies a space of heuristic methods, and only settles on a particular combination of these methods after a period of adaptation, during which the system automatically acquires information about the particular distribution of problems associated with the intended application.

More rigorously, the adaptive problem–solving problem can be describes as follows. Given a flexible perfor-

mance element PE with control points $CP_1 ... CP_n$, where each control point CP_i corresponds to a particular control decision and for which there is a set of alternative decision methods $M_{i,1} ... M_{i,k}$,[1] a control strategy is a selection of a specific method for every control point (e.g., STRAT = $<M_{1,3}, M_{2,6}, M_{3,1}, ...>$). A control strategy determines the overall behavior of the scheduler. It may effect properties like computational efficiency or the quality of its solutions. Let PE(STRAT) be the problem solver operating under a particular control strategy. The function U(PE(STRAT), d) is a real valued utility function that is a measure of the goodness of the behavior of the scheduler over problem d. The goal of learning can be expressed as: given a problem distribution D, find STRAT so as to maximize the expected utility of PE. Expected utility is defined formally as:

$$\sum_{d \in D} U(PE(STRAT), d) \times probability(d)$$

For example, in a planning system such as PRODIGY [Minton88], when planning to achieve a goal, control points would be: how to select an operator to use to achieve the goal; how to select variable bindings to instantiate the operator; etc. A method for the operator choice control point might be a set of control rules to determine which operators to use to achieve various goals plus a default operator choice method. A strategy would be a set of control rules and default methods for every control point (e.g., one for operator choice, one for binding choice, etc.). Utility might be defined as a function of the time to construct a plan, cost to execute the plan, or some overall measure of the quality of the plan produced.

The adaptive problem–solving technique is applicable in cases where the following conditions apply:

1. The control strategy space can be structured to facilitate hillclimbing search. In general, the space of such strategies is so large as to make exhaustive search intractable. Adaptive problem–solving requires a transformation generator that structures this space into a sequence of search steps, with relatively few transformations at each step.

2. There is a large supply of representative training problems so that an adequate sampling of problems can be used to estimate expected utility for various control strategies.

3. Problems can be solved with a sufficiently low cost in resources so that estimating expected utility is feasible.

4. There is sufficient regularity in the domain such that the cost of learning a good strategy can be amortized over the gains in solving many problems.

Due to space limitations we now turn to a description of the DSN application domain and results. For further details on the statistical learning techniques we use for adaptive problem–solving the reader is referred to (Chien et al. 1995, Gratch & Chien 1996).

2 Results of Applying Adaptive Problem–solving to Deep Space Network Scheduling

In order to assess the viability of adaptive problem–solving to real–world scheduling problems, we have applied adaptive problem–solving to a testbed scheduler which solved Deep Space Network Antenna Allocation problems. In this section we describe a formulation of the Deep Space Network antenna allocation problem, and outline results of applying adaptive problem–solving to learning control heuristics for a scheduler which solves this problem.

The Deep Space Network (DSN) is a multi-national collection of ground-based radio antennas responsible for maintaining communications with research satellites and deep space probes. DSN Operations is responsible for scheduling communications for a large and growing number of spacecraft. This already complex scheduling problem is becoming more challenging each year as budgetary pressures limit the construction of new antennas. As a result, DSN Operations has turned increasingly towards intelligent scheduling techniques as a way of increasing the efficiency of network utilization. As part eof this ongoing effort, the Jet Propulsion Laboratory (JPL) has been given the responsibility of automating the scheduling of the 26-meter sub-net; a collection of

1. Note that a method may consist of smaller elements so that a method may be a set of control rules or a combination of heuristics. Note also that a method may also involve real–valued parameters. Hence, the number of methods for a control point may be infinite, and there may be an infinite number of strategies.

26-meter antennas at Goldstone, CA, Canberra, Australia and Madrid, Spain.

2.1 DSN Scheduling Problem

Scheduling the DSN 26-meter subnet can be viewed as a large constraint satisfaction problem. Each satellite has a set of constraints, called project requirements, that define its communication needs. A typical project specifies three generic requirements: the minimum and maximum number of communication events required in a fixed period of time; the minimum and maximum duration for these communication events; and the minimum and maximum allowable gap between communication events. For example, Nimbus-7, a meteorological satellite, must have at least four 15-minute communication slots per day, and these slots cannot be greater than five hours apart. Project requirements are determined by the project managers and tend to be invariant across the lifetime of the spacecraft.

In addition to project requirements, there are constraints associated with the various antennas. First, antennas are a limited resource – two satellites cannot communicate with a given antenna at the same time. Second, a satellite can only communicate with a given antenna at certain times, depending on when its orbit brings it within view of the antenna. Finally, antennas undergo routine maintenance and cannot communicate with any satellite during these times.

Scheduling is done on a weekly basis. A weekly scheduling problem is defined by three elements: (1) the set of satellites to be scheduled, (2) the constraints associated with each satellite, and (3) a set of *time periods* specifying all temporal intervals when a satellite can legally communicate with an antenna for that week. Each time period is a tuple specifying a satellite, a communication time interval, and an antenna, where (1) the time interval must satisfy the communication duration constraints for the satellite, (2) the satellite must be in view of the antenna during this interval. Antenna maintenance is treated as a project with time periods and constraints. Two time periods conflict if they use the same antenna and overlap in temporal extent. A valid schedule specifies a non-conflicting subset of all possible time periods where each project's requirements are satisfied.

The automated scheduler must generate schedules quickly as scheduling problems are frequently over-constrained (i.e. the project constraints combined with the allowable time periods produces a set of constraints which is unsatisfiable). When this occurs, DSN Operations must go through a complex cycle of negotiating with project managers to reduce their requirements. A goal of automated scheduling is to provide a system with relatively quick response time so that a human user may interact with the scheduler and perform "what if" reasoning to assist in this negotiation process. Ultimately, the goal is to automate this negotiation process as well, which will place even greater demands on scheduler response time (see (Chien & Gratch, 1994) for some preliminary work on this later problem). For these reasons, the focus of development is upon heuristic techniques that do not necessarily uncover the optimal schedule, but rather produce adequate schedules quickly.

2.2 Adaptive LR–26 Control Points

Pseudocode for the LR–26 scheduler is given below with the four control points indicated parenthetically. Control point one controls which partial schedule is chosen from the agenda, two determines in what order children in S are explored, three dictates which constraint is chosen to satisfy next, and control point four determines how schedules are refined. The details of how 1, 2, 3, and 4 are implemented define the runtime efficiency characteristics of the algorithm almost entirely (for a more detailed discussion see Gratch & Chien, 1996).

LR-26 Scheduler

Agenda := $\{S^0\}$;
While *Agenda* $\neq \emptyset$
(1) Select some partial schedule $S \in$ *Agenda*; *Agenda*:=*Agenda*–$\{S\}$
(2) Weight search for some $S^*(u) \in S$;
IF $S^*(u)$ satisfies the project requirements (*PR*) Then
Return $S^*(u)$;
Else
(3) Select constraint $c \in PR$ not satisfied by $S^*(u)$;
(4) Refine S into $\{S^i\}$, such that each $S^G \in S^i$ satisfies c
and $\cup\{S^i\} = S$;
Perform constraint propagation on each S^i
Agenda := *Agenda*$\cup\{S^i\}$;

Figure 1: The basic LR-26 refinement search method.

1)	Value ordering:	penalize-conflictedness
	Weight search:	first-solution
	Primary constraint ordering:	penalize-unforced-periods
	Secondary constraint ordering:	prefer-total-conflictedness
	Refinement method:	systematic-refinement
2)	Value ordering:	prefer-gain
	Weight search:	first-solution
	Primary constraint ordering:	penalize-unforced-periods
	Secondary constraint ordering:	prefer-total-conflictedness
	Refinement method:	systematic-refinement
3)	Value ordering:	penalize-conflictedness
	Weight search:	first-solution
	Primary constraint ordering:	penalize-unforced-periods
	Secondary constraint ordering:	penalize-satisfaction-distance
	Refinement method:	systematic-refinement

Figure 2: The three highest utility strategies learned by Adaptive LR-26.

2.3 Overall Results — DSN DISTRIBUTION

In this section we summarize the results of adaptive problem solving over the constructed DSN problem distribution. The results support that the system learned search control strategies that yielded a significant improvement in performance. Adaptive problem solving reduced the average time to solve a problem (or prove it unsatisfiable) from 80 to 40 seconds (a 50% improvement).

Due to the stochastic nature of the adaptive scheduler, different strategies were learned on different trials. All learned strategies produced at least some improvement in performance. The best of these strategies required only

24 seconds on average to solve a problem (an improvement of 70%). The fastest adaptations occurred early in the adaptation phase and performance improvements decreased steadily throughout. It took an average of 62 examples to adopt each transformation. Adaptive LR-26 showed some improvement over the non-adaptive scheduler in terms of the number of problems that could be solved (or proven unsatisfiable) within the resource bound. LR-26 was unable to solve 21% of the scheduling problems within the resource bound. One adaptive strategy substantially reduced this number to 3%.

In Figure 2 we give the three highest utility strategies learned by the adaptive algorithm. Most of the performance improvement (about one half) can be traced to modifications in LR-26's weight search method (i.e., how the algorithm chooses which child to explore next). The rest of the improvements are divided equally among changes to the heuristics for value ordering (the order in which partial schedules are prepended to the agenda), constraint selection, and refinement.

Another claim is that, in practice, adaptive problem–solving can identify strategies that rank highly when judged with respect to the whole strategy space. A secondary question is how well does the expert strategy perform. The improvements of Adaptive LR-26 are of little significance if the expert strategy performs worse than most strategies in the space. Alternatively, if the expert strategy is extremely good, its improvement is compelling.

As a way of assessing these claims we estimate the probability of selecting a high utility strategy given that we choose it randomly from one of three strategy spaces: the space of all possible strategies (expressible in the transformation grammar), the space of strategies produced by Adaptive LR-26, and the trivial space containing only the expert strategy. This corresponds to the problem of estimating a *probability density function* (p.d.f.) for each space: a p.d.f., $f(x)$, associated with a random variable gives the probability that an instance of the variable has value x. More specifically we want to estimate the density functions, $f_s(u)$, which is the probability of randomly selecting a strategy from space s that has expected utility u.

We use a non-parametric density estimation technique called the kernel method to estimate $f_s(u)$ (as in (Smyth, 1993)). To estimate the density function of the whole space, we randomly selected and tested thirty strategies. All of the learned strategies are used to estimate the density of the learned space. (In both cases, five percent of the data was withheld to estimate the bandwidth parameter used by the kernel method.) The p.d.f. associated with the single expert strategy is estimated using a normal model fit to the 1000 test examples from the previous evaluation.

2.3.1 *DSN Distribution*

Figure 3 illustrates the results for the DSN distribution. In this evaluation the learned strategies significantly outperformed the randomly selected strategies. Thus, one would have to select and test many strategies at random before finding one of comparable expected utility to one found by Adaptive LR-26. The results also indicate that the expert strategy is already a good strategy (as indicated by the relative positions of the peaks for the expert and random strategy distributions), indicating that the improvement due to Adaptive LR-26 is significant and non-trivial.

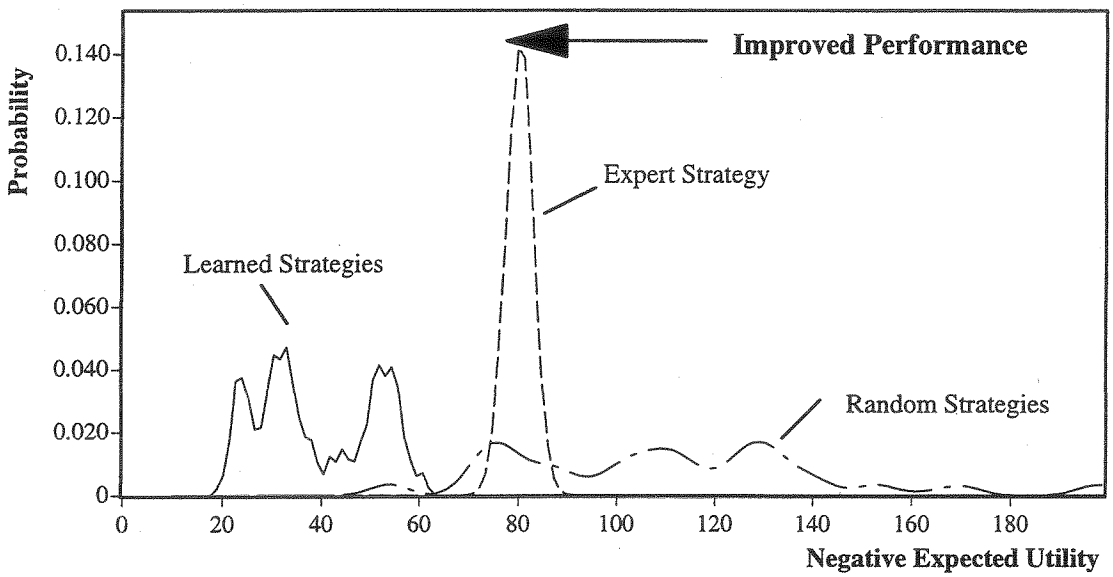

Figure 3: The DSN Distribution. The graph shows the probability of obtaining a strategy of a particular utility, given that it is chosen from (1) the set of all strategies, (2) the set of learned strategies, or (3) the expert strategy.

The results provide additional insight into Adaptive LR-26's learning behavior. That the p.d.f for the learned strategies contains several peaks, graphically illustrates that different local maxima exist for this problem. Thus, there may be benefit in running the system multiple times and choosing the best strategy. It also suggests that techniques designed to avoid local maxima would be beneficial.

2.3.2 *Full Augmented Distribution*

Figure 4 illustrates the results for the full augmented distribution. The full augmented distribution differs from the DSN distribution in that it also includes unsolvable problems posed to the scheduler (e.g., those problems requiring relaxation of project constraints or for which no known strategy was able to solve the problem). The results are similar to the DSN distribution: the learned strategies again outperformed the expert strategy which in turn again outperformed the randomly selected strategies. The data shows that the expert strategy is significantly better than randomly selected strategies. Together, these two evaluations support the claim that Adaptive LR-26 is selecting high performance strategies. Even though the expert strategy is quite good when compared with the complete strategy space, the adaptive algorithm is able to improve the expected problem solving performance.

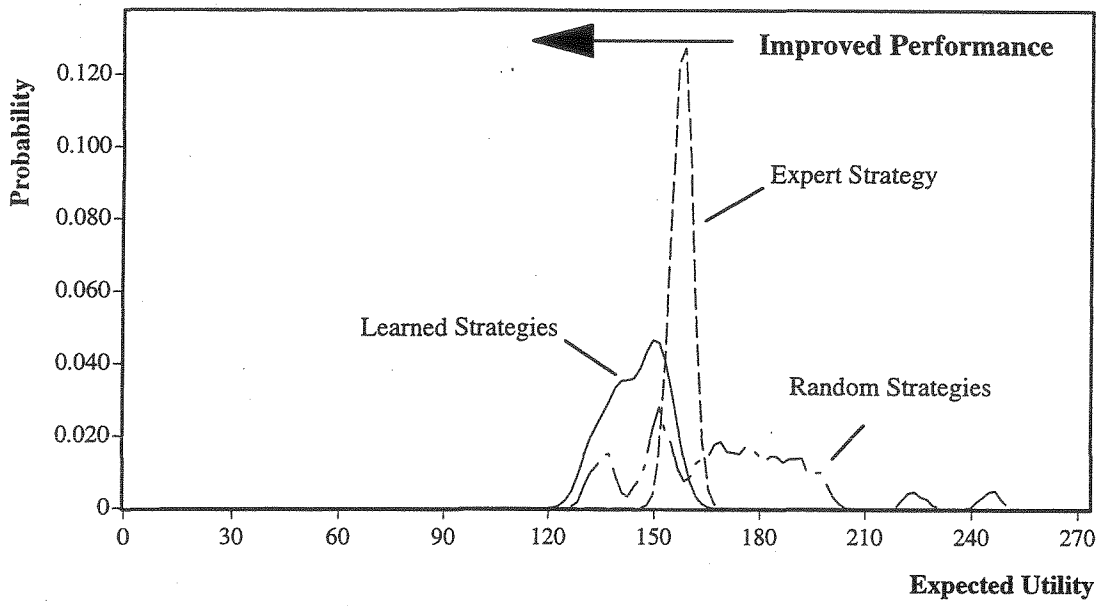

Figure 4: The Full augmented distribution. The graph shows the probability of obtaining a strategy of a particular utility, given that it is chosen from (1) the set of all strategies, (2) the set of learned strategies, or (3) the expert strategy.

3 Related Work and Conclusions

Related efforts include Heuristic–Biased Stochastic Sampling (HBSS) (Bresina, 1996) and GENH (Morris, Bresina, Rodgers, 1997). The approach described in this paper attempts to learn the optimal hypothesis to apply to a class of problems, based on a number of sample problem instances. HBSS and GENH differ in that they are designed to speed up search on individual problem instances.

Other related work includes more statistical techniques, which explicitly reason about performance of different heuristic strategies across a distribution of problems. These are generally statistical generate-and-test approaches that estimate the average performance of different heuristics from a random set of training examples, and explore an explicit space of heuristics with greedy search techniques. Examples of such systems are COMPOSER (Gratch & DeJong, 1992), PALO (Greiner & Jurisca, 1992), and the statistical component of MULTI-TAC (Minton, 1993). Similar approaches have also been investigated in the operations research community (Yakowitz & Lugosi, 1990). These techniques are easy to use, apply to a variety of domains and utility functions, and can provide strong statistical guarantees about their performance. They are limited, however, as they are computationally expensive, require many training examples to identify a strategy, and face problems with local maxima. Furthermore, they typically leave it to the user to conjecture the space of heuristic methods (see (Minton, 1993) for a notable exception).

This paper has described the application of statistical learning techniques to the refinement of heuristics for scheduling in an approach we call adaptive problem–solving. We first described the overall approach, in which the adaptive problem–solver searches through a space of potential control strategies for the scheduler. We then described results from applying adaptive problem–solving to a Deep Space Network Antenna Scheduing system. In this application, adaptive problem–solving was able to significantly improve upon the best human expert derived control strategy.

Acknowledgements

Portions of this work were performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration and portions at the Beckman Institute, University of Illinois under National Science Foundation Grant NSF–IRI–92–09394.

References

Baker, A. (1994). The Hazards of Fancy Backtracking. In *Proceedings of AAAI–94.*

Bell, C., & Gratch, J. (1993). Use of Lagrangian Relaxation and Machine Learning Techniques to Schedule Deep Space Network Data Transmissions. In *Proceedings of The 36th Joint National Meeting of the Operations Research Society of America, the Institute of Management Sciences.*

Biefeld, E., & Cooper, L. (1991). Bottleneck Identification Using Process Chronologies. In *Proceedings IJCAI–91.*

Bresina, J. (1996) Heuristic–Biased Stochastic Sampling. In *Proceedings of AAAI–96.*

Chien, S. & Gratch, J. (1994). Producing Satisficing Solutions to Scheduling Problems: An Iterative Constraint Relaxation Approach. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems.*

Chien, S., Gratch, J. & Burl, M. (1995). On the Efficient Allocation of Resources for Hypothesis Evaluation: A Statistical Approach. *Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence 17(7),* 652–665.

Dechter, R. & Pearl, J. (1987). Network–Based Heuristics for Constraint–Satisfaction Problems. *Artificial Intelligence 34(1)* 1–38.

Dechter, R. (1992). Constraint Networks. In *Encyclopedia of Artificial Intelligence*, Stuart C. Shapiro (ed.).

Fisher, M. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science 27 (1)* 1–18.

Frost, D. & Dechter, R. (1994). In Search of the Best Constraint Satisfaction Search. In *Proceedings of AAAI–94.*

Gratch, J. & DeJong, G. (1992). COMPOSER: A Probabilistic Solution to the Utility Problem in Speed–up Learning. In *Proceedings of AAAI–92.*

Gratch, J., Chien, S., & DeJong, G. (1993). Learning Search Control Knowledge for Deep Space Network Scheduling. In *Proceedings of the Ninth International Conference on Machine Learning.*

Gratch, J., Chien, S. (1996) Adaptive Problem–solving for Large –Scale Scheduling Problems: A Case Study, Journal of Artificial Intelligence Research 4 pp. 365–396.

Gratch, J. & DeJong, G. (1996). A Decision–theoretic Approach to Adaptive Problem Solving. *Artificial Intelligence (Winter 1996).*

Greiner, R. & Jurisica, I. (1992). A Statistical Approach to Solving the EBL Utility Problem. In *Proceedings of AAAI–92.*

Kambhampati, S., Knoblock, C. & Yang, Q. (1995). Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial Order Planning. *Artificial Intelligence: Special Issue on Planning and Scheduling 66,* 167–238.

Kwak, N. & Schniederjans, M. (1987). *Introduction to Mathematical Programming*, New York: Robert E. Krieger Publishing.

Minton, S. (1993). Integrating Heuristics for Constraint Satisfaction Problems: A Case Study. In *Proceedings of AAAI–93.*

Mackworth, A. (1992). Constraint Satisfaction. In *Encyclopedia of Artificial Intelligence*, Stuart C. Shapiro (ed.).

Minton, S. (1988). *Learning Search Control Knowledge: An Explanation–Based Approach*, Norwell, MA: Kluwer Academic Publishers.

Morris, R., Bresina, J., & Rodgers, S. (1997) Automatic Generation of Heuristics for Scheduling. In *Proceedings of IJCAI–97.*

Sacerdoti, E. (1977). *A Structure for Plans and Behavior.* New York: American Elsevier.

Smyth, P. (1993). Probability Density Estimation and Local Basis Function Neural Networks. *Computational Learning Theory and Natural Learning Systems 2.*

Stone, P., Veloso, M., & Blythe, J. (1994). The Need for Different Domain–Independent Heuristics. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems.*

Taha, H. (1982). *Operations Research, an Introduction,* Macmillan Publishing Co.

Wilkins, D. (1988). *Practical Planning: Extending the Classical Artificial Intelligence Planning Paradigm.* San Mateo, CA: Morgan Kaufman.

Yakowitz, S. & Lugosi, E. (1990). Random Search in the Presence of Noise, with Application to Machine Learning. *Society for Industrial and Applied Mathematics Journal of Scientific and Statistical Computing 11, 4,* 702–712.

Yang, Q. & Murray, C. (1994). An Evaluation of the Temporal Coherence Heuristic for Partial–Order Planning. *Computational Intelligence 10, 2.*