Experiments with a Decision-Theoretic Scheduler*

Othar Hansson[1,2] and **Gerhard Holt**[1] and **Andrew Mayer**[1,2]

[1]Heuristicrats Research Inc.
1678 Shattuck Avenue, Suite 310
Berkeley, CA 94709-1631

[2]Computer Science Division
University of California
Berkeley, CA 94720

Abstract

This paper describes DTS, a *decision-theoretic scheduler* designed to employ state-of-the-art probabilistic inference technology to speed the search for efficient solutions to constraint-satisfaction problems. Our approach involves assessing the performance of heuristic control strategies that are normally hard-coded into scheduling systems, and using probabilistic inference to aggregate this information in light of features of a given problem.

BPS, the Bayesian Problem-Solver [2], introduced a similar approach to solving single-agent and adversarial graph search problems, yielding orders-of-magnitude improvement over traditional techniques. Initial efforts suggest that similar improvements will be realizable when applied to typical constraint-satisfaction scheduling problems.

1 Background

Scheduling problems arise in schools, in factories, in military operations and in scientific laboratories. Although many algorithms have been proposed, scheduling remains among the most difficult of optimization problems. Because of the problem's ubiquity and complexity, small improvements to the state-of-the-art in scheduling are greeted with enormous interest by practitioners and theoreticians alike.

A large class of scheduling problems can be represented as constraint-satisfaction problems (CSPs), by representing attributes of tasks and resources as variables. Task attributes include the scheduled time for the task (start and end time) and its resource requirements. A schedule is constructed by assigning times and resources to tasks, while obeying the constraints of the problem. Constraints capture logical requirements (a typical resource can be used by only one task at a time) and problem requirements (task T_x requires N units of time, must be completed before task T_y, and must be completed before a specified date).

One common approach to finding an assignment for the variables employs a preprocessing stage which tightens the constraints (e.g., by composing two constraints to form a third), followed by a backtrack search to find a satisfying assignment. Figure 1 illustrates the operation of such a search algorithm: searching depth-first until a dead-end is reached, and then backtracking to the nearest choice point to continue the search.

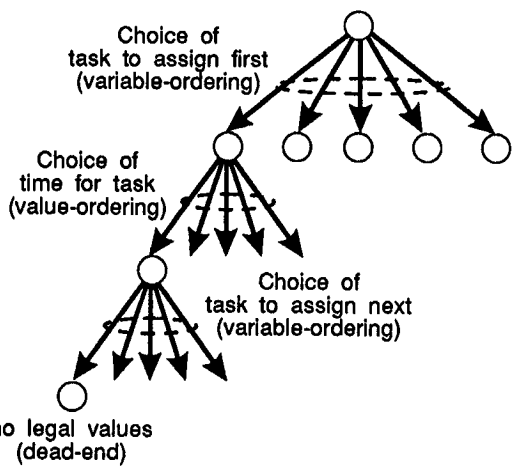

Figure 1: Basic CSP Algorithm

Heuristic functions guide the ordering of variables and values. For example, one heuristic for variable ordering counts the number of possible values for each variable, and chooses the variable with the smallest number of values as the next to instantiate. Typi-

*This research was supported by the National Aeronautics and Space Administration under contract NAS2-13340.

cally, the variable ordering in backtracking algorithms is static, determined prior to search by use of a heuristic function. As heuristics for variable and value ordering form the basis for the algorithm's performance, tremendous effort has been invested in developing good general-purpose heuristics. However, practitioners often bypass the general-purpose heuristics in favor of hand-crafted domain-specific heuristics (e.g., Sadeh's work [8]).

2 DTS Rationale

CSP heuristics are imperfect and exhibit highly domain-specific performance. Although they often provide useful search control advice, the possibility of error introduces uncertainty into the search algorithms which rely on them. Consequently, current techniques are forced to pay a large computational price in cases where the heuristic function makes incorrect classifications. Furthermore, the algorithms will repeat these costly mistakes, as there are no robust learning mechanisms designed to improve a CSP heuristic's performance over time.

Existing heuristic functions encode many different domain attributes. Some estimate the quality of partial schedules while others estimate the difficulty of finding a feasible solution. Unfortunately, there is no *sound* methodology for combining the information provided by an arbitrary number of heuristics for use in controlling a single search. This forces human schedulers to make an unpleasant choice:

- decide *a priori* on a particular heuristic, and thus concentrate on a single domain attribute. This can skew the system's performance at the expense of other domain attributes.
- hand-craft a composite heuristic which captures multiple domain attributes in a single function.

For this reason, the selection of heuristics and problem-solving techniques for any given CSP domain remains an art despite years of comparative study.

DTS, which is derived from previous work on BPS (the Bayesian Problem-Solver), is designed to address these problems. The first area of innovation is the *heuristic error model*: a probabilistic semantics for heuristic information, based on the concept of conditional probability in statistical decision-theory [3]. Heuristics are interpreted by correlating their estimates with the actual payoffs of problem-solving instances. When a problem is solved, the heuristic error model is updated, adapting it to the problem's specific characteristics. Multiple heuristics are combined by correlating payoffs with a *set* of heuristic estimates. This alleviates the human scheduler's dilemma by providing a dominating alternative, a sound framework for combining an arbitrary number of heuristic functions.

The second area of innovation is the use of *multi-attribute utility theory*, a formalized method for quantifying preference relationships among a set of uncertain outcomes. An important target application for DTS is experiment scheduling for the Hubble Space Telescope. Figure 2 depicts a partial set of utility attributes, whose non-linear tradeoffs can be encoded by a multiattribute utility function. In contrast to

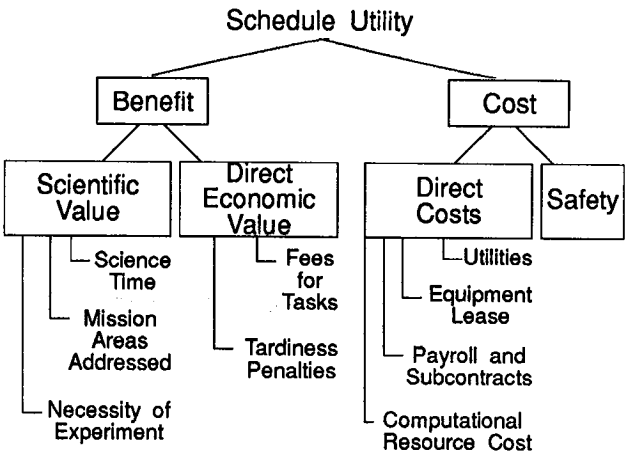

Figure 2: Utility Attributes for Experiment Scheduling

traditional CSP scheduling algorithms, which employ special-purpose control rules, DTS's control rule is the decision-theoretic *rationality* criterion of maximizing expected utility.

In DTS, domain information is encoded in heuristic functions and user preferences are encoded in utility functions. By combining domain-independent and domain-specific heuristics, and then using the user's utility function to make search control decisions, DTS provides a more efficient and flexible alternative to traditional scheduling techniques.

3 DTS: First Results

This section describes empirical results illustrating the performance advantages of these two DTS innovations.

3.1 Combining Heuristics

The primary strength of the DTS prototype is the method for combining information from separate heuristic evaluation functions to improve constraint-satisfaction search control. Experiments with the prototype on the Eight Queens and Bridge-Construction Scheduling [9] problems confirm that the combination of heuristic functions provides more information than any of the heuristics taken individually. This translates into significant reductions in overall search time.

Traditionally, CSP algorithms make use of a variable ordering heuristic and a value ordering heuristic. Figure 3 shows the performance of a standard CSP algorithm using all possible pairs (A1, A2, B1, B2)

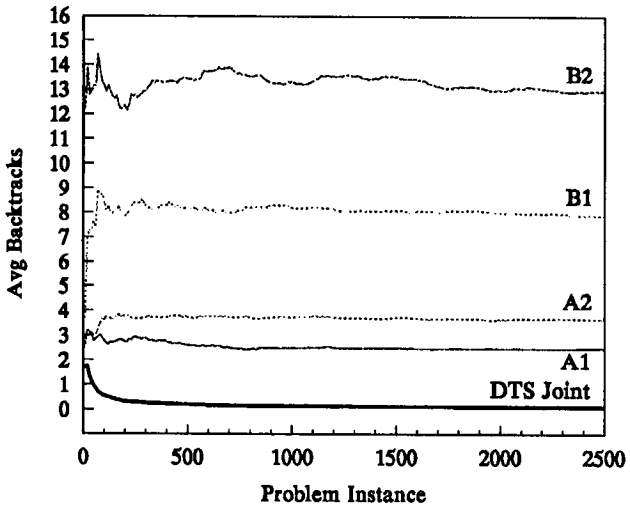

Figure 3: Eight Queens: Combining Heuristics vs. Heuristics in Isolation

drawn from two well-known variable ordering heuristics (Most Constraining Variable (A), Minimum Domain Variable (B)) and two well-known value ordering heuristics (Least Constraining Value (1), Dechter's Value Heuristic (2)[1]). Also shown is the DTS prototype (DTS-Joint), which dominated the competition by using all four heuristics in combination. The horizontal axis plots the number of problem instances solved and the vertical axis plots the running average of search time over the entire experiment. The plot, but not the average, beging with the tenth problem instance.

Figure 4 shows a corresponding graph for the Bridge-Construction Scheduling problem. The variable ordering heuristic used was Minimum Domain Variable and the value ordering heuristics were Least Constraining Value (curve A1) and ASAP, "as soon as possible" (curve A2). Also shown are the corresponding individual DTS performance curves (DTS A1, DTS A2) as well as the combined heuristic performance curve (DTS-Joint).

To summarize both graphs, the improvement is seen to be nearly 50% on average for Bridge Construction Scheduling, and over 95% for the Eight-Queens problem. Note that the sharp downward slope of the DTS-Joint *running average* in Figure 4 demonstrates the performance improvement accrued by learning, unattainable using traditional techniques.

3.2 Learning Heuristic Error Models

Figure 5 displays an example heuristic error model learned over the course of 2500 Eight-Queens problem

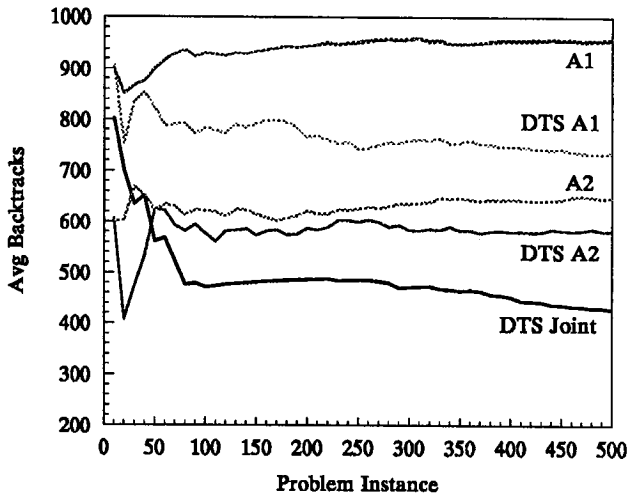

Figure 4: Bridge-Construction Scheduling: Combining Heuristics vs. Heuristics in Isolation

instances (for the Minimum Domain heuristic). The horizontal axis plots the heuristic function estimate and the vertical axis plots the preference for that estimate. In DTS, preference is based upon the expected utility associated with a heuristic estimate (dashed line). In traditional algorithms, the heuristic is assumed to rank-order alternatives perfectly, and therefore, preference is a monotonic function of the heuristic estimate.

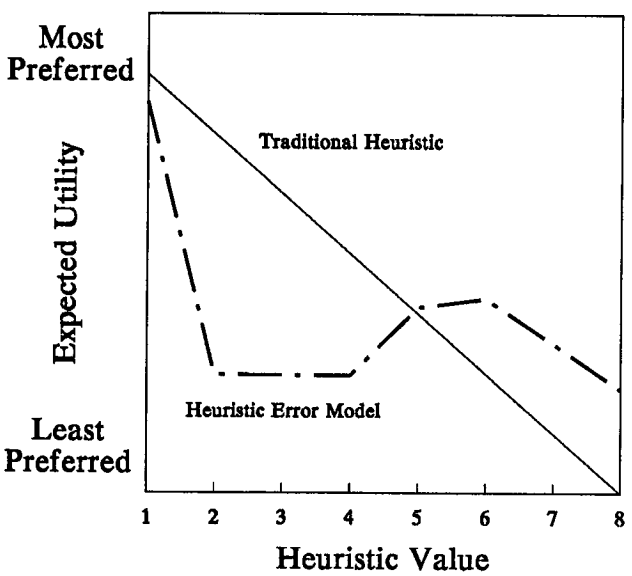

Figure 5: Sample Heuristic Error Model

The discrepancy between the heuristic estimates and the actual utilities explains the poor performance of

traditional approaches, which assume perfect heuristic estimates. Further, it explains why DTS outperforms these techniques, as it does not make this assumption, and instead learns to correct for the discrepancy.

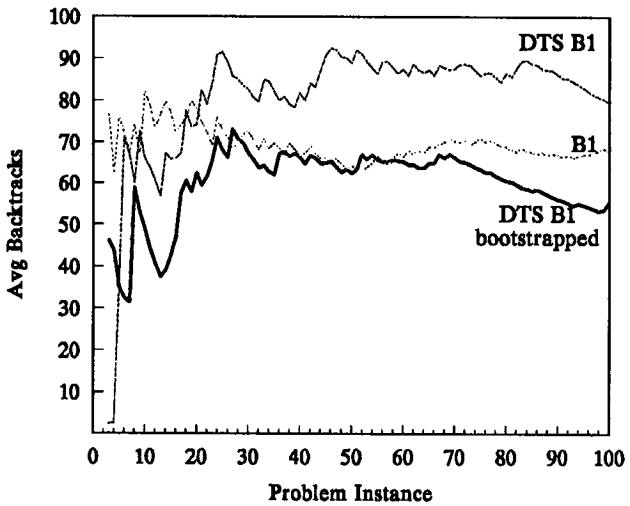

Figure 6: Generalizing Data to Larger Domains

An additional benefit of the heuristic error model is the ability to generalize learned data across domains. For example, Figure 6 depicts the performance of DTS on the Thirty-two-Queens problem with 1) no prior heuristic error model, and 2) a heuristic error model generalized (or "bootstrapped") from the 2500 Eight-Queens examples solved in Figure 3. Generalizing data from the simpler domain has reduced search complexity. This is particularly important as the time required to calibrate heuristic error models increases with problem complexity.

3.3 Decision-Theoretic Backtracking

The DTS prototype employed a simplified decision-theoretic control mechanism which was adapted to a conventional backtracking search algorithm: this allowed for controlled experiments on DTS vs. traditional algorithms. The application of decision theory to backtracking elucidates many important ideas.

The only search control decisions made in traditional backtracking systems are the selections of which subtrees of the search graph to explore next. Once a subtree is selected (by selecting the next variable or value), it is explored exhaustively unless a solution is found. Such an ordering problem can be viewed as a decision-tree. Figure 7 depicts the choice of ordering two subtrees A and B. We have proven a theorem [4] which shows that the system's expected utility (search time to first solution) is maximized if variables (or values) are ordered by the quantity $P(v)/C(v)$, where $P(v)$ indicates probability of finding a solution in the subtree, and $C(v)$ indicates the cost of searching the subtree (whether or not a solution is found). $P(v)$ and $C(v)$ are attributes of the payoff mentioned above. Experiments confirmed that once $P(v)$ and $C(v)$ are learned, this rule outperforms traditional backtracking search algorithms which interpret heuristic estimates at face value. This result indicates that decision-theoretic search-control improves overall system performance. A similar analysis can also be performed for iterative improvement [4].

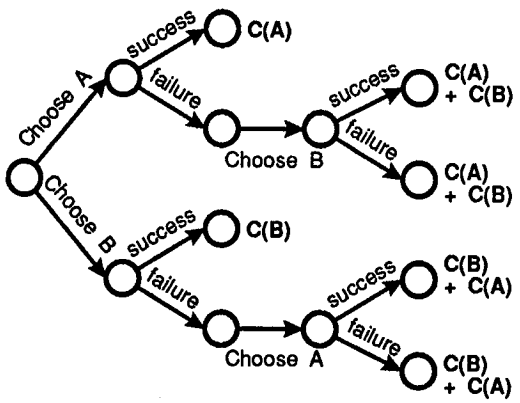

Figure 7: Decision Tree for Value-Ordering Problem (Values A and B)

As is evident from this discussion, DTS must convert raw heuristic estimates at a node into estimates of (1) probability of finding a solution in the subtree under that node, and (2) the cost of search in that subtree. We note here that while heuristics are usually very good at rank-ordering nodes based on (1) and (2) individually, the rank-ordering for the combination is typically incorrect. DTS' heuristic error model corrects for this.

3.4 Implementation Synopsis

The prototype performs a backtracking search, using the standard optimizations of forward-checking and dynamic search rearrangement. The search is ordered by the expected utility selection criteria ($P(v)/C(v)$) discussed above. The estimates of $P(v)$ and $C(v)$ are derived from the heuristic error model, using traditional CSP heuristics. The heuristic error model is updated during and between trials using a bucketed histogram, and interpreted by a Laplacian estimation.

4 Future Directions

Our initial study of CSP and scheduling domains demonstrates that applying even the simplest modeling techniques of statistical decision theory can yield significant payoffs. There are many other aspects of CSP algorithms which would benefit from a similar decision-theoretic approach. We conclude with two such examples.

4.1 Preprocessing and Caching of Learned Constraints

Our decision-theoretic approach could be applied equally well to the control of scheduling subproblems. For example, Minton [5] has considered a simple utility-based model of the selective caching of learned problem-solving rules.

Minton demonstrated that the caching of too many rules acquired from problem-solving instances leads to a substitution of knowledge-search (searching the rule cache for an applicable rule) for problem-solving search. Similarly, in a CSP problem, any number of implicit constraints can be generated by preprocessing or constraint-recording and cached in the constraint graph. But additional constraints, while reducing problem-solving search, increase the number of consistency checks per search tree node (knowledge search). Choosing to generate and record a constraint is, again, a decision made under uncertainty, and it would be interesting to consider a decision-theoretic approach to the problem. We feel that decision-theoretic modeling and the simple structure of CSPs can provide a firmer theoretical foundation for this area of research.

4.2 Selective Value Generation

A common problem among search algorithms is selective expansion of successors. The textbook description of most search algorithms calls for a full expansion of all successors of a given node. For constraint-satisfaction problems, this is clearly inadequate, as many variables such as task start and end times have an infinite number of infinitesimally-spaced values.

One possible approach employs heuristics for value generation. While we have applied decision theory to search by designing an algorithm which evaluates all successors and then selects among them, it is equally possible to apply these tools to selective expansion of successors. If several heuristics (dispatch rules) can be used to suggest plausible values, our approach can be applied to the heuristics trivially. If no such heuristics exist, one possibility is to employ a tree of values, and perform an auxiliary search of this tree to select a particular value. This brings on a new learning task: clustering values of similar merit into a hierarchy of values.

5 Conclusion

The use of Bayesian probability theory in DTS underscores that scheduling involves decision-making under uncertainty, and illustrates how imperfect information can be modeled and exploited. The use of multiattribute utility theory in DTS underscores that scheduling involves complex tradeoffs among user preferences. By addressing these issues, DTS has demonstrated promising performance in preliminary empirical testing.

References

[1] R. Dechter and J. Pearl. Network-Based Heuristics for Constraint-Satisfaction Problems. In *Search in Artificial Intelligence*, L. Kanal and V. Kumar, eds., Springer-Verlag, New York, 1988.

[2] O. Hansson and A. Mayer. Heuristic Search as Evidential Reasoning. In *Proceedings of the the Fifth Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, August 1989.

[3] O. Hansson and A. Mayer. Probabilistic Heuristic Estimates. *Annals of Mathematics and Artificial Intelligence*, 2:209–220, 1990.

[4] O. Hansson and A. Mayer. Decision-Theoretic Control of Artificial Intelligence Scheduling Systems. HRI Technical Report No. 90-1/06.04/5810, September 1991.

[5] S. Minton. *Learning Effective Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic, Dordrecht, 1989.

[6] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Processing Letters*, vol. 7, 1974.

[7] N. Sadeh. Lookahead Techniques for Activity-Based Job-Shop Scheduling. Technical Report TR CMU-RI-TR-89-2, CMU, the Robotics Institute, 1989.

[8] N. Sadeh. *Lookahead Techniques for Micro-Opportunistic Job-Shop Scheduling.* PhD Thesis, CMU, the Robotics Institute, 1991.

[9] P. van Hentenryck. *Constraint-Satisfaction in Logic Programming.* MIT Press, Cambridge, MA, 1989.

[10] M. Zweben, M. Deale and R. Gargan. Anytime Rescheduling. in *Proceedings of the DARPA Planning Workshop*, Morgan Kaufmann, San Mateo, CA, 1990.