Utility Functions and Heuristics for Resource-Bounded Planning Applications

Ted Linden

Myriad Software
2245 Tasso St.
Palo Alto, CA 94301
linden@netcom.com

Abstract

This paper proposes a structure for utility functions that are useful when modeling and solving important classes of practical planning applications—problems that are not adequately characterized by an additive utility function, but still have some additive structure. The structure is $\Sigma u(t)*mod(t,...)$ where u(t) is a function of how and when a single task is completed and mod(t,...) captures the effects of dependencies between tasks. This structure makes it relatively easy to represent many important problems and allows decision theoretic techniques to be used to merge the available evidence when making variable and value order decisions during heuristic search.

1. Introduction

Experience from practical planning applications indicates that the formalisms needed to represent and solve many problems are quite different from those of classical planning. While some problems fit the classical planning mold; other important problems have the following features:

1. There are many goals or tasks (often hundreds or thousands) and not all of them can be accomplished with the available resources. Deciding what to do is part of the problem. With some difficulty, user preferences among the goals can be approximated by a utility function.
2. The utility function is non-linear and non-additive. There are hard and soft constraints between task assignments, and some goals support other goals. Furthermore, some sets of goals have a conjunctive all-or-none property, and some sets of goals are alternatives with decreasing additional value once M out of N are accomplished.
3. The planning situation is dynamic with ongoing changes in goals, preferences, and resources. Knowledge of current and projected states is partial and uncertain.
4. The formal problem model only approximates the user's real planning problem. As a particular problem is being solved, the user often discovers constraints that had not been foreseen. Thus, the planning must be interactive. The user may find it hard to formulate general constraints and often prefers to prescribe part of the solution.

Examples of problems that have these four features are:

- Complex project planning and scheduling applications.
- Military mission planning and crisis action planning.
- Disaster relief planning
- Sensor planning and allocation.
- Job shop scheduling in the presence of complex, multivariate utility functions.

Much recent work on planning addresses the third feature above. Operations research handles large problems with the first feature, but does not formalize methods for dealing with the second through fourth features. The fourth feature is a consideration for theoretical work because some problem solving methods are more able to accommodate and benefit from user interaction than others. Decision theoretic approaches appear capable of dealing with all these issues. The challenge is in scaling up decision theory to handle the interrelated decisions involved in assigning resources and choosing parameters for hundreds of tasks.

There has been little or no domain-independent planning research that allows all four of the above problem features to be handled within a single formalism. Some practical planning applications have addressed all these features, but have not used a separable, domain-independent planning engine.

For example, Rome Laboratory's Advanced Planning System (APS) is now an operational system that helps teams of Air Force personnel plan, coordinate, and schedule up to 2000 daily missions. The planning component of this system has an AI architecture, implements constraint propagation, and uses a non-additive utility function to guide its search. In this specific domain, it addresses all the problem features listed above. (The third issue is handled in a replanning extension of APS that is not yet operational but does exist in prototype form.)

This paper generalizes techniques originally developed for APS and tries to conceptualize them in domain-independent form. It uses ideas from decision theory, and extends them in ways that will work for large scale planning applications. This is an attempt to define a general approach, specific algorithms generally take advantage of additional features of the problem structure.

2. Exploiting Simplifying Features of Typical Problems

The four problem features listed above are not handled by classical planning, but most real applications are easier than classical planning in other ways. Practical applications often require relatively little emphasis on dynamic creation of task plans (how to accomplish a goal) and more emphasis on resource allocation (what to accomplish) and scheduling (when to accomplish it). For example, in manufacturing applications and job shop scheduling, the possible process plans are usually known at design time. The hard problem is in allocating resources to instantiate a generic process plan. This characteristic is common in a wide variety of applications that I have dealt with including the air mission planning in APS, planning Army Corps-level maneuvers, managing containers for a shipping company, scheduling training missions for pilots, an associate system to support submarine commanders, and other applications.

One simplification comes from exploiting the dominant role of resources in many practical applications. Dependencies between tasks and some need for dynamic task creation mean that the applications are not simple resource allocation problems. However, problem solving methods that focus on resource allocation are effective and some form of statistical look-ahead that projects resource contention is useful when making value ordering and variable ordering decisions during search [Muscettola & Smith 87, Fox & Sadeh 90, Sycara et al. 90, Sadeh 91, Linden 91, Linden & Vrotney 92].

Another common simplification is that the parameters assigned to each task are constrained only by the resources and by direct dependencies on a relatively small number of other tasks. Once the resource constraints are factored out, each task is independent of most other tasks.

One current line of investigation is to translate all the binary and low order constraints between tasks into resource constraints. If Task A has to be completed before Task B, this can be modeled by a phantom resource that is produced by A and consumed by B. This allows the entire problem to be solved as a resource allocation problem; however, it is not yet clear whether this simplification, which has the side effect of increasing the number of resource types, actually simplifies the practical aspects of problem solving.

3. A Structure for Utility Functions

A utility function that is practical for planning and scheduling problems should be a compromise between several conflicting goals:

- It should represent user preferences in a reasonably natural and direct way.
- It should be useful during heuristic search when evaluating partial solutions.
- It should enable effective variable and value ordering decisions during heuristic search.

In approaching the utility function, it is useful to separate:

1. The utility of achieving a goal (performing a task).
2. Binary and low order constraints on the solution–both hard and soft constraints.
3. Resource constraints and high order constraints on the solution.

Resource constraints are common in practical applications and are not effectively handled by general purpose constraint satisfaction techniques. Resource constraints are complex, N-way constraints between many tasks. N is typically large.

Viewed from the perspective of search strategies, resource constraints are nasty in that they tend to prune a branch of the search tree only after it is

almost fully expanded. For example, if there are k instances of a resource, simple propagation of this resource constraint does not have an impact until after the k^{th} assignment of that resource. Then it immediately has the dramatic effect of constraining many of the remaining tasks.

While resource constraints are not handled effectively by generic constraint satisfaction techniques, specialized heuristics are effective with resource constraints. A heuristic approach, which is now widely practiced in AI scheduling applications, projects resource contention using statistical look-ahead techniques and uses these contention estimates in variable and value ordering heuristics. Statistical look-ahead techniques have been used in work on Opis [Muscettola & Smith 87], Cortes [Fox et al. 90, Sycara et al. 90] and Micro-Boss [Sadeh 91] and in work on Rome Laboratory's Advanced Planning System (APS) [APS 89].

There is a very fine line between hard and soft constraints. Users will often describe hard constraints on the problem solution, but when asked whether they would ever violate the constraint, they will find situations in which the constraint can be violated. It is really a soft constraint with a strong penalty for violating it. A utility function should enable a smooth transition between hard constraints that can't be violated and soft constraints that carry a large penalty when violated.

These and other considerations lead to utility functions that are structured in the form Σu(t)*mod(t,...) where u(t) is a function of how and when a single task is completed and mod(t,...) captures the effects of dependencies between tasks. For more details, see [Linden 91]. The mod(t,...) factor is a function of all the assignments made to t and to all tasks that are involved in binary or other low order constraints with t. The value of mod(t,...) should be 0 when the assignments made to t and related tasks violate a hard constraint. A soft constraint is represented by a value between 0 and 1. Values outside the range [0,1] can also be meaningful. This approach supports continuity between hard and soft constraints while still recognizing a difference between hard and soft constraints.

Typically, mod(t,...) is structured as a product where each factor in the product captures the effect of one constraint between t and other tasks. A product is one specific way of combining the effect of multiple soft constraint violations. Other combining functions are possible.

A utility function in the proposed form is a fairly natural way of representing real problems. The u(t) factor combines the effects of multiple evaluation criteria that depend only on the assignments made to the parameters of this single task (e.g., resource costs, timeliness of task completion, appropriateness of the resources for the task, etc.). The effect of dependencies between tasks (for example, one task must be performed before another) are captured in the mod(t,...) factor. Essentially, each hard or soft constraint between tasks becomes a factor in the mod(t,...) component of each of the constrained tasks. The effects of conjunctive and disjunctive goals can also be captured in the mod(t,...) component.

4. Evaluating Partial Solutions

The proposed structure for utility functions allows many equivalent formulations, and often one formulation is more useful than others when evaluating partial solutions. For example, consider the case of two tasks t1 and t2 where t1 establishes a precondition for t2. Assume that t2 achieves 10 units on the utility scale, and t1 has no independent utility except for its role in enabling t2. The utility function for these two tasks would then be 0*mod(t1,t2) + 10*mod(t2,t1) where mod(t2,t1) is 1 if t1 is accomplished before t2 (and maintained) and 0 otherwise. When evaluating a partial solution, this utility function gives no importance to t1; however, an equivalent utility function is x*mod(t1,t2) + (10-x)*mod(t2,t1) where mod(t1,t2) is 0 if t2 does not come after t1. By choosing appropriate values for x, this form of the utility function can decompose the problem of choosing parameter values for t1 and t2 and support more effective least commitment strategies.

5. Utility Functions and Heuristic Search

Most variable and value ordering heuristics are sensitive to only one feature of the search state; for example, the minimum domain variable ordering heuristic finds the variable with the smallest domain of feasible values. But complex resource-bounded planning problems involve optimization in the presence of constraints. A fixed heuristic that is sensitive only to constraints and ignores utility considerations involves a discontinuity between hard and soft constraints and will be effective only for problems where hard constraints are the critical fea-

ture. Heuristics should depend on the utility function as well as the constraints.

What is needed is a general way to merge the evidence available when making variable and value ordering decisions during search. This evidence comes from the utility function, binary constraints, and resource constraints. A decision theoretic viewpoint seems to provide justifiable semantics for computations that combine this evidence. A utility function in the form suggested above with separate components for the task utility and for the effects of interactions between tasks supports this decision theoretic approach.

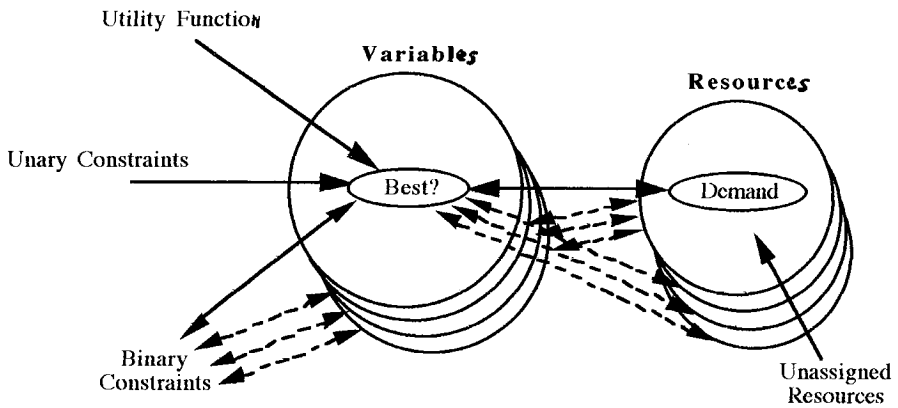

Figure 1: Sources of Evidence for Variable and Value Ordering Decisions.

6. The Decision-Theoretic View.

The variable and value ordering decisions that need to be made repeatedly during search can be viewed as problems in decision theory where evidence for each decision comes from the problem's features and the current state of the problem solving. While heuristics are frequently thought of as inexpensive computations, any computation that is not exponential in problem size can be a useful heuristic, and simpler heuristics can always be selected once the generic approach is understood.

6.1 Past Research on Probabilistic Computations of Heuristics

Many domain-independent heuristics have been proposed to solve constraint satisfaction problems (CSP). These include the variable with smallest feasible domain, the most constraining variable, the least constraining value, and the value that participates in the most solutions to a relaxation of the problem. Work by Hansson et al. [92] computes a combination of heuristics that is effective for a specific CSP. Heuristics for constrained optimization problems should extend these CSP heuristics.

When there is a utility function as well as constraints, a key question is how to combine the evidence from the utility function and the constraints; for example, how much extra utility is needed to compensate for consuming a highly constrained resource. Sycara et al. [90] addressed this question by experimenting first with the two extreme cases (picking the value that gives the most utility for this task vs. picking the value that most reduces resource contention). A compromise is probably better than either extreme. Rather than trying to find the most appropriate compromise through experiments, theoretical results may be able to predict the compromise that is most effective overall.

Figure 1 is a simplified representation of the recent state of research on using probabilities to combine evidence when making variable and value ordering decisions during search. At any intermediate stage of the search process, one considers the remaining tasks (variables) and the remaining resources (values)—each shown as stacks of circles in the figure. Evidence for the variable and value ordering decisions comes from the utility function, the unary constraints, binary constraints, and from statistics about resource contention. The heavy arrows show the flows of evidence involving the first variable and the first resource; the dashed arrows indicate other flows of evidence to and from the other variables and resources.

In Figure 1, for each variable there is a probability distribution (called **Best?**) that captures the available evidence about which resource will turn out to be the best choice for assigning to this variable. For each resource there is a probability distribution (called **Demand**) that summarizes the demand for that resource from all the variables. The double arrow between **Best?** and **Demand** characterizes one of the problems that needs to be solved: the **Best?** distribution is used to project statistics about resource demand, and the **Demand** for a resource influences the probability that a resource is the best global choice for assignment to a variable. Similarly, binary constraints involve evidence flowing in both directions between two variables—probabilities

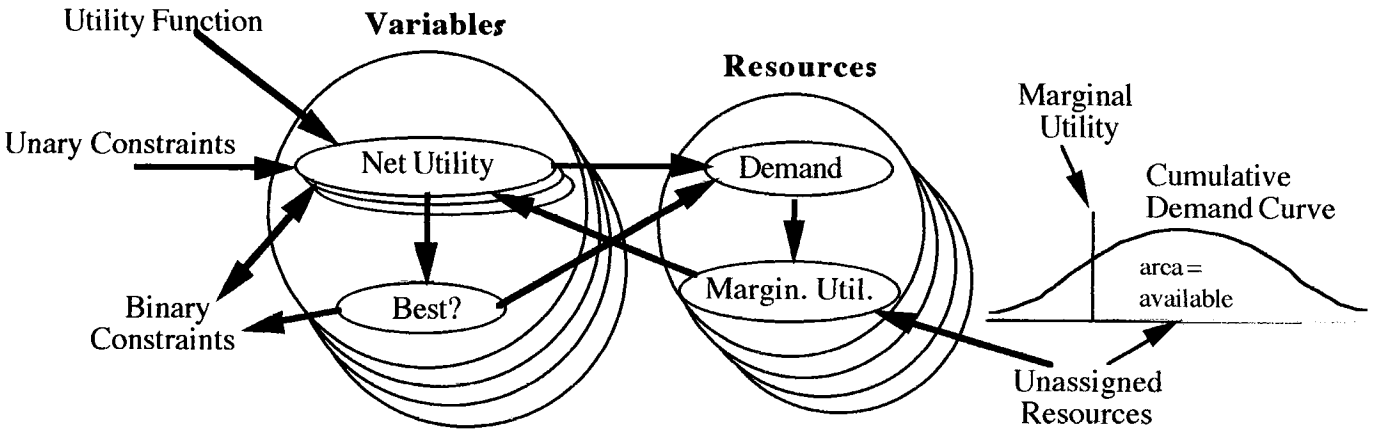

Figure 2: Evidence for Each Option is Accumulated in Probability Distributions about Net Utility.

about assignments to one variable influence the probabilities about the best assignments to the other variable.

6.2 Overview of Approach

Figure 2 elaborates Figure 1 with two additional concepts: the **Net Utility** of each potential assignment of a value to a variable and the **Marginal Utility** of a resource. The double arrow between **Best?** and **Demand** is eliminated, but there are still cycles in the influences between the probability distributions shown in Figure 2. The elimination of these cycles is discussed in Section 6.7 and depicted in Figure 3.

6.3 Net Incremental Utility

The first key concept is **net incremental utility**—or more simply the **net utility**. Intuitively, the net utility of a potential assignment of a value to a variable is the utility of that assignment less the opportunity loss for other variables caused by consumption of the resource and by constraints involving the assigned variable.

The net utility cannot be known exactly without a full search, but it can be estimated. Uncertainty about the net utility is subjective. Net utility can be calculated in exponential time, but with a computation that uses less than exponential time, its value is uncertain. With each variable and each possible value assignment to the variable, one associates a probability distribution that captures the available evidence about the net utility that will result if this value is chosen for this variable. The net utility that is being estimated is defined as follows:

- The **net incremental utility** of assigning a value to a variable, when given an existing partial problem solution, is the difference between the best complete extension that includes that assignment and the best complete extension that assigns no resource to that variable.
- More precisely, **net incremental utility** is relative to the search strategy that is being used. When the search strategy being used is not guaranteed to find the best solution, then net incremental utility is defined relative to the expected values of the best solution that will be found by the given search strategy. Thus, given a search strategy, the net incremental utility of assigning a resource to an operation is the difference between the expected value of the best complete solution that will be found by following that search strategy after including that assignment less the expected value of the best complete solution that will be found by following that search strategy after assigning no resource to that operation.

6.4 Net Utility and the "Best?" Choice

The unary constraints and the u(t) component of the utility function provide a local estimate of utility. This local estimate is the initial evidence for the probable net incremental utility. When a utility function is additive, the local utility comes directly from the utility function. Since a probability distribution about the local utility is all that is needed, the additivity assumption is not required. Most utility functions—as long as they have some partially additive structure—yield a probability distribution about the utility that will be achieved by assigning a value to a variable.

Binary constraints and resource constraints also influence estimates of the net incremental utility. Before dealing with these constraints, we need to derive the probability that an assignment is the best choice from distributions about net incremental utility.

By reformulating the problem to focus on net incremental utility, the probability that an assignment to a variable is the best choice becomes a derived concept rather than an intuitive concept as in [Sadeh & Fox 89, 90, Sadeh 91].

Best? is a probability distribution over the possible resource assignments that is computed as the probability that a random utility value selected from the net utility distribution about the resource assignment is bigger than any other such utility value.

The variances in the probability distributions about net utility control whether one assignment is slightly or dramatically better than alternatives.

6.5 Computing Marginal Utility

The next issue is the interaction between assignments to variables and the projection of resource contention statistics. Previous work has used the probability that an assignment is the best choice to project demand for resources, but demand also influences the probabilities about the best choice. A key insight derives from Wellman's work on applying economic theory to transportation scheduling [Wellman 92]. The marginal utility of a resource establishes a price for the resource, and an agent evaluating the options for assignment to a variable should favor using a resource to the extent that the local utility of using a resource exceeds the globally determined price of the resource. Choosing the best resource to assign to a variable is no longer based on the relative size of the local utilities; rather it is based on the amount by which the local utility exceeds its resource's price.

6.6 Propagating the Influence of Binary Constraints

Binary constraints between variables also influence the net utility computations. Processing evidence from binary constraints can be thought of as an extension of arc consistency concepts from CSP problems. The utility function makes the constraint propagation more complex. The effect of constraints on probabilities about the best choice has been studied by Muscettola & Smith [87] and Sadeh & Fox [89, 90]. Net utility should include the utility lost when an assignment constrains the utility achievable by a dependent variable. For each assignment that a related variable may make, a variable estimates the utility it will lose and communicates that estimated loss to the dependent variable.

In addition to estimating the utility lost by other variables, it may also be useful to estimate the degree to which an assignment restricts the choices available for other variables. Consideration is being given to using entropy concepts to measure the distance between a partial solution and a complete solution or between two consistent partial solutions. Computing the entropy of a partial solution is often straightforward. The entropy involved in each Best? distribution may capture the flexibility that is left to make assignments to that variable. When a value assignment to one variable decreases the entropy of another variable that it constrains, this may be taken as a measure of the constraining effect of that proposed value assignment. These may be useful measures that generalize the most constraining variable and least constraining value heuristics.

6.7 Rumor Control and Convergence

Figure 2 showed a cycle in the evidence being passed from net utility to resource demand to marginal utility and back to net utility. This cycle needs to be broken by using the standard approach to rumor control from Bayesian nets. Essentially, the net utility information that the variable's agent passes to the resource agent must not include previous evidence received from the resource agent. The same restriction applies to evidence passed to other variable agents through the binary constraints. Figure 3 is a more detailed version of Figure 2 showing that the local net utility and local best choice (which do not reflect evidence received from the resource agents) is computed and passed to the resource agents. While not shown in Figure 3, the information passed between variable agents through the binary constraints must also be restricted so evidence previously received from another agent is excluded from all evidence passed to that agent.

There are still some longer cycles in the information flows. Intuition says they are not significant; however, further research is needed to derive conditions under which they can be proven to be insignificant.

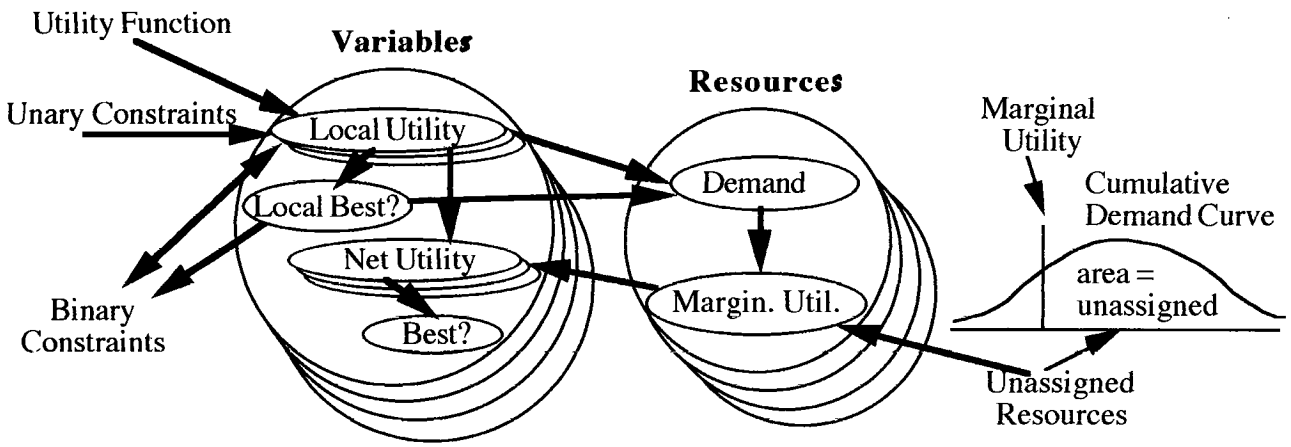

Figure 3: Additional distributions are introduced to break cycles and avoid rumor propagation.

7. Experimental Results and Conclusions

Initial experiments to test the decision-theoretic computations for variable and value ordering decisions are reported in [Linden & Vrotney 92]. The experiments use a statistical look-ahead algorithm that starts from a utility function, projects probable demand for each resource, computes a probable marginal utility for each resource, and uses these probabilities to make variable and value ordering choices during search. Initial experiments tested these concepts on assignment problems and compared the solution found on the first branch explored using these heuristics with the solution found by a simple greedy algorithm. The statistical look-ahead algorithm found solutions that average 4-7% higher utility. When an optimal solution was known, the statistical look-ahead, almost always either found it on the first branch or was within 0.5% of optimal The frequency of better results improved on larger problems. The statistical look-ahead improved the result (relative to greedy on 85% of the small problems (16 tasks and 12 resources), on 95% of the medium size problems (30 tasks and 24 resources) and on all runs of the larger problems (56 tasks and 48 resources).

This paper has argued that concepts from decision theory provide a theoretical foundation for the statistical look-ahead techniques used to make variable and value ordering decisions during heuristic search. The theory will extend current practice to handle resource-bounded planning problems with complex, non-additive utility functions with both hard and soft constraints.

8. References

[APS 89] Software Design Document for the Advanced Planning System. Unisys Corporation, Prepared for Rome Air Development Center, Griffiss Air Force Base, NY. CDRL B012, 1989.

[Fox et al. 89] Mark S. Fox, Norman Sadeh, & Can Baykan, "Constrained Heuristic Search," AAAI-89, pp. 309-315.

[Hansson et al 92] Othar Hansson, Gerhard Holt, and Andrew Mayer, "Experiments with a Decision-Theoretic Scheduler," 1992 AAAI Spring Symposium on Practical Approaches to Scheduling and Planning, NASA Ames TR FIA-92-17, May 1992.

[Linden 91] Theodore A. Linden, "Preference-directed, Co-operative Resource Allocation and Scheduling." Final Technical Report, DARPA Order No. 6685, Advanced Decision Systems Report TR-1270-3, Sept. 1991.

[Linden & Vrotney 92] Theodore A. Linden and William Vrotney, "Transformational Planning for Resource Constrained Problems," Annual Technical Report to DARPA, Advanced Decision Systems Report TR-18246-1, November. 1992.

[Muscettola & Smith 87] Nicola Muscettola and Stephen F. Smith, "A Probabilistic Framework for Resource-Constrained Multi-agent Planning." Proc. Tenth Inter. Joint Conf. on Artificial Intelligence, Morgan Kaufman, Publ. 1987, pp. 1063-1066.

[Sadeh & Fox 89] Norman Sadeh and Mark S. Fox, Preference Propagation in Temporal/Capacity Constraint Graphs. The Robotics Institute, Carnegie Mellon Univ., CMU-RI-TR-89-2, 1989.

[Sadeh & Fox 90] Norman Sadeh and Mark S. Fox, "Variable and Value Ordering Heuristics for Activity-based Job-shop Scheduling." Proc. of the Fourth Inter. Conf. on Expert Systems in Production and Operations Management, 1990, pp 134-144,.

[Sadeh 91] Norman Sadeh, "Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling." PhD Thesis, School of Computer Science, Carnegie Mellon University, 1991.

[Sycara et al. 90] Katia P. Sycara, S. Roth, N. Sadeh, and M. Fox, "Managing Resource Allocation in Multi-Agent Time-constrained Domains." DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control., Morgan Kaufman Publ., San Mateo, CA, Nov. 1990, pp. 240-250.

[Wellman 92] Michael P. Wellman, "A General-Equilibrium Approach to Distributed Transporation Planning." AAAI-92, AAAI Press, 1992.