Toward Scheduling Over-Constrained Remote-Sensing Satellites

Joseph C. Pemberton
pemberto@psrw.com
Veridian Pacific-Sierra Research
Arlington, VA 22209

Abstract

The high demand for satellite imagery and the relative scarcity of commercial imaging satellites combine to produce scheduling problems in which one or more of the resources is over constrained. Over-constrained scheduling problems can be problematic for constraint programming methods because of the overhead associated with propagating constraints and the time spent backtracking over a large number of tasks that cannot be scheduled. We present a hybrid approach called priority segmentation that breaks the scheduling problem up into sub-problems based on the priority of the tasks, finds an optimal solution to each sub-problem, and then combines the sub-problem solutions. Our preliminary results indicate that priority segmentation produces solutions that are an improvement over a greedy solution when the problems are highly over constrained. When the problems are only slightly over constrained, the percent improvement over greedy is very small. Our results also indicate that greedy solutions are competitive with more computation intensive approaches under certain problem conditions.

Introduction

Satellite scheduling is the problem of mapping tasks (observations, communications, downlinks, control maneuvers, *etc.*) to resources (sensor satellites, relay satellites, ground stations, *etc.*). Although the term satellite scheduling has been applied to many different aspects of a satellite's operation (*e.g.*, design planning, launch control, lifecycle, *etc.*), our work has largely focused on scheduling mission operations, namely the day-to-day activities of an operational satellite. Mission operation activities include: payload operations (*e.g.*, using a sensor to collect data), bus operations (*e.g.*, maintaining the health and status of the vehicle), and communications operations (*e.g.*, transmitting data between satellites or from satellites to the ground and receiving information or commands from a ground station). The rest of this paper will focus on scheduling of payload operations for remote-sensing satellites – namely deciding which observations to perform.

Satellite mission operation schedules are typically over constrained because the commercial demand for satellite imagery to date has far exceeded the capabilities of commercial satellite systems. For this paper, the term "over constrained" refers to scheduling problems in which the capacity required by the tasks exceeds the total available capacity for at least one resource in the problem.

To further quantify this property, we have adopted the following definitions: The *average constrainedness* of a scheduling resource is the ratio of the sum of the resource capacity required by all tasks to the total capacity available. The *average constrainedness* of a scheduling problem is the maximum average constrainedness over all resources in the problem.

When the average constrainedness of a resource is greater than one, we say that the resource is over constrained. Note that a resource can be over constrained in one part of the scheduling problem and then under constrained in another part. In cases where the level of resource constrainedness differs greatly from one part of a problem to another, it may be necessary to evaluate the constrainedness of the different parts of the problem independently. For this paper, we have focused on problems where the average constrainedness of a resource does not vary significantly from one part of the scheduling problem to another.

The paper is organized as follows. Section 2 describes the remote sensing scheduling problem and presents a simplified satellite imagery scheduling problem. Section 3 describes a constraint programming approach to solving this class of scheduling problems and discusses why this approach can be problematic when the scheduling problem is over constrained. Section 4 describes a hybrid method that we have used to address over constrained scheduling problems. Sections 5 and 6 describe our preliminary experiments and initial results. Section 7 discusses the results and related work. Finally, Section 8 contains some conclusions and plans for future work.

Remote Sensing Scheduling

The term remote sensing refers to the class of satellite missions in which a satellite is used to make observations. The observations can be of either earth-bound or interstellar objects. Due to the high cost of building, launching and maintaining remote sensing satellites, it is typically the case that more than one user will share the use of a satellite's sensing capability. It is also typically the case that the demand for use of the satellite will exceed its capacity. This means that the scheduler must choose which subset of the tasks to execute. A good schedule is a tradeoff between the constraints of the problem and the needs of the users.

As an example, a group of farmers in Kansas has banded together to perform a land use analysis of their farmland and the surrounding communities to better assess the impact of the current drought. They have contacted a remote sensing satellite operator and agreed to purchase a set of satellite images that will be taken over the next three days. The farmers set a maximum three-day spacing between images to ensure that any unexpected storm would not have time to affect the crop growth assessment.

The satellite system's ability to perform the farmers' tasks is constrained by several factors. First, the ground location must be visible to the satellite. Second, the quality of the image typically depends on the angle between the satellite and ground (the aspect angle). Typically, the best quality (highest resolution) picture is taken when the satellite is directly over the position to be imaged. For our example, the farmers might have specified a minimum quality for their images. The farmers could have also asked for stereo images, which are created by taking two separate images of the same ground location from slightly different viewing angles. To satisfy the constraints of a stereo image, the satellite must take two images within a few minutes of each other, and it must take both pictures in order to satisfy the stereo requirement.

The physical characteristics of the satellite components can also constrain the execution of tasks. The most common examples are power, momentum, slew, heat and memory. Slew refers to the movement of a satellite sensor that is necessary to set it up before a task is executed. In most cases satellite operators prefer to minimize the amount of slew because the time and power spent slewing the satellite is time and power that could be spent performing revenue-generating tasks. A satellite scheduling system must consider all of these factors at some level in order to produce a schedule that satisfies the problem constraints and also addresses the objective functions (*e.g.*, minimize slew, maximize profit, *etc.*).

Another important source of constraints is users' requirements. Since we have already assumed that the satellite will not be able to perform all tasks in the time available, the scheduling problem becomes an optimization problem. Operators of commercial remote sensing satellites want to maximize their profits; therefore, they are interested in generating schedules that maximize the amount of money that they can collect from their customers. For our example, the farmers might decide to pay only if all of the images are taken because anything less would make their analysis incomplete. Alternatively, they might create a priority ordering for the imaging tasks and agree to a cost scale based on the priority (*i.e.*, to pay more for higher priority tasks).

The remote sensing scheduling problem can be summarized as follows. The tasks are observations to be performed by the satellite(s). Each task has a start time and duration. Tasks require resources (*e.g.*, the satellites camera, power, *etc.*). Resources have finite capacity (*i.e.*, a camera can only take one picture at a time). When it is not possible to schedule every task, the quality of the schedule can be measured by an objective function (*e.g.*, maximize the number of tasks scheduled, maximize total revenue, minimize slew, *etc.*).

A Constraint Programming Approach

We have adopted a constraint programming approach to satellite scheduling problems. We model the satellite-scheduling problem using four basic objects: tasks, resources, events and constraints. Tasks are the activities and operations to be performed. Resources are the people, satellites, sensors, communication channels, *etc.*, that are required to accomplish tasks. Events are used to capture domain-specific occurrences that restrict when tasks can be scheduled (*e.g.*, satellite visibility windows). Constraints are further restrictions on when tasks can be scheduled due to interactions with other tasks and to resource capacity and availability. When the problem is over constrained, we assign a priority value to each task. The priority values can then be used by the scheduler as part of an objective functions (*e.g.*, optimize the priority sum of the scheduled tasks). For commercial imaging tasks, the task priority could be directly related to the revenue of the task.

A task is scheduled by assigning it a resource, start time and duration, which satisfy all of the relevant constraints. Tasks that require more than one resource (*e.g.*, a downlink task requires a satellite and a ground station) will need to have more than one resource assigned to it. More formally, a scheduled task is a tuple:

$$(task,\ \{resource\ set\},\ start\ time,\ duration)$$

In cases where the specific set of resources needed to perform a task are known *a priori*, then the scheduling problem is reduced to finding an execution time and duration for each task. When each task has a fixed duration, then the scheduling problem is further reduced to finding an execution time for each task. If the task start times are also fixed, then scheduling becomes a problem of deciding which tasks to perform. We will discuss the last problem class further in later sections.

We map satellite scheduling problems into constraint programming problems by associating a variable with each task. The values correspond to the start time, end time and resources assigned to the task. In general, our constraint-based scheduler explores the space of possible schedules by choosing a task to schedule and

Priority Segmentation:

1. Sort the set of tasks by their priority value.
2. Select the n highest priority tasks not yet scheduled.
3. Find an optimal schedule for the n tasks given the resource constraints and previously scheduled tasks.
4. Lock the tasks that were successfully scheduled and remove the tasks that could not be scheduled.
5. Loop to 3.

Figure 1: Priority segmentation algorithm.

then selecting values for the start time, end time and resource assignments. Our general purpose constraint-based scheduler is built on top of ILOG Solver and Scheduler, which performs the constraint propagation and directs the search based on the heuristics that we have designed (*e.g.*, the value and variable selection heuristics).

This general approach to scheduling works well for a range of problem parameters. However, we have found that when the number of tasks requested by users exceeds the number of tasks that can be reasonably performed, the performance of the constraint engine suffers. This is due to the fact that it must propagate constraints over a large set of tasks, most of which cannot be scheduled. In addition, the scheduler can spend a lot of time backtracking over a relatively large number of tasks that have very little chance of being scheduled. Thus the runtime of a constraint-based approach can depend more on the number of tasks requested than on the number of tasks that can ultimately be executed. We have found this overhead cost to be intolerable for the time demands of scheduling operational satellites. For this reason, we have developed a hybrid algorithm as a first step toward addressing this problem.

A Hybrid Approach

In this section, we present a hybrid approach for generating solutions to over-constrained satellite scheduling problems.

Priority segmentation reduces the constraint propagation overhead by breaking up the problem into a subset of problems. The idea is to sort the tasks by their priority value, divide them into groups of n tasks (where n is a small integer), and then schedule one group at a time. The algorithm details are shown in Figure 1.

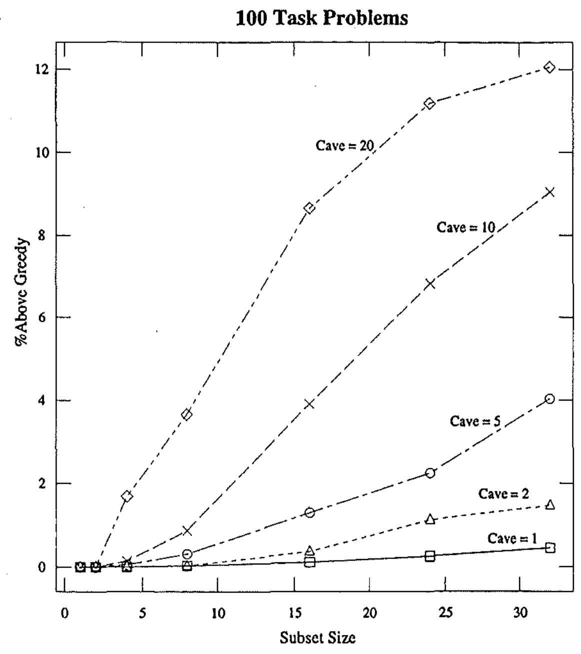
Priority segmentation can be viewed as an application of staged search (Ibaraki 1978) to a constraint-based formulation of scheduling problems. Because the algorithm commits to scheduling decisions before all tasks are considered, it is necessarily a suboptimal algorithm. For large scheduling problems (*i.e.*, greater than 1000 tasks), the computation cost of finding optimal solutions is prohibitive so we must consider suboptimal scheduling algorithms. Suboptimal algorithms must also be considered for problems with real-time constraints on the scheduling. Obviously, we could apply additional optimization methods (*e.g.*, local search) to the output of priority segmentation. These issues are not addressed in this paper and are the subject of future work.

Experimental Design

We implemented priority segmentation using branch-and-bound search to generate the optimal schedules for task subsets. We also implemented a random problem generator to provide us with a set of test problems. For our initial investigation, we made some simplifying assumptions about the remote sensing scheduling problem. For example, we assumed fixed duration tasks with fixed start times. This reduces the over-constrained scheduling problem to deciding which of the tasks to schedule (since the start time and duration cannot change). The objective is to choose a set of tasks to schedule such that the sum of the task priorities is maximized.

There are two justifications for simplifying the problem in this manner. One is to better focus our attention on the over-constrained resource problem. The other is that one of our customers is interested in solving a remote sensing problem where a large number of tasks (*e.g.*, greater than 1000) need to be scheduled in less than 30 minutes. The problem description includes a model to calculate the slew required by the satellite to reposition its camera between tasks. The slew model provides the slew time between a given pair of tasks, which depends on the imaging locations and the start times of the two tasks. The time constraint on the scheduler means that the slew must calculated ahead of time and stored in a table. The size of the slew table is $(cn)^2$ where c is the number of possible start times for individual tasks and n is the number of tasks. We chose to require fixed start times for tasks so that the slew table could fit in the computer's memory.

We generated a set of random remote-sensing scheduling problems with the following characteristics. The satellite system resources were simplified to a single resource (*e.g.*, the camera) which has a capacity of one (*i.e.*, only one picture can be taken at a time). Tasks were generated to simulate different levels of demand for the satellite resource that directly correspond to different levels of average constrainedness. The task duration was fixed at 10 seconds for all tasks. Each fixed start time was randomly selected from a uniform distribution over the schedule period. The test sets did not include any stereo tasks, precedence or logical constraints between tasks. The tasks priorities were randomly selected integers from the range [1, 50] (where 1 is the lowest priority value). We generated problem sets by varying the number of tasks and the scheduling period to control the average constrainedness of the problem (C_{ave}).



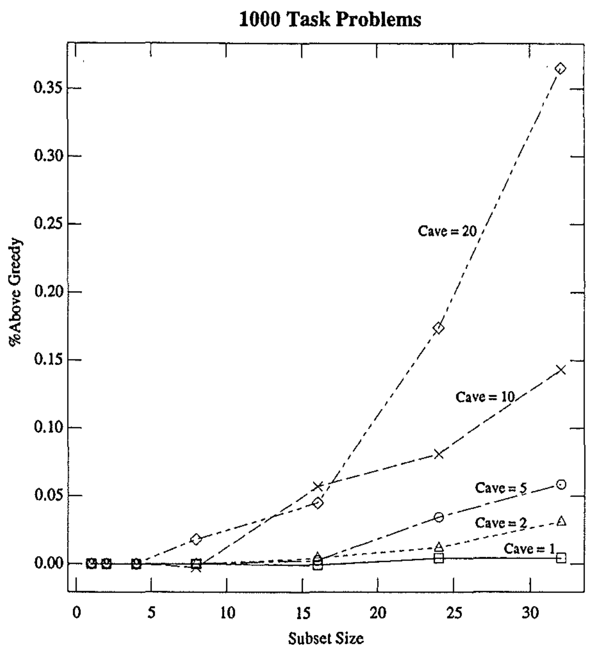


Figure 2: Percent average priority sum improvement relative to greedy (100-task and 1000-task problems).

Initial Experimental Results

We ran priority segmentation on a set of randomly generated scheduling problems that were built using the simplifying assumptions described above. The priority segmentation subset size was varied as well as the size of the problem (*i.e.*, number of tasks) and the average constrainedness (C_{ave}). For a given problem size, we varied the average constrainedness by choosing the length of the scheduling period so that the average number of tasks per unit time was equal to the desired constrainedness. We performed experiments on 100-task and 1000-task problems.

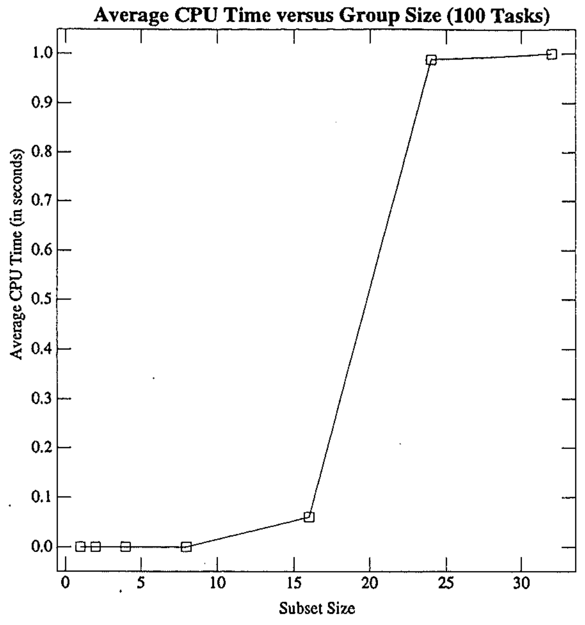
To provide a baseline for comparison, we implemented a simple greedy algorithm (*i.e.*, consider tasks for scheduling one at a time from highest to lowest priority). Priority segmentation with a subset size of 1 is equivalent to this greedy algorithm. It is also interesting to note that priority segmentation with a subset size of 2 also produces the same solutions as this greedy algorithm. This is because the order in which two tasks are considered for scheduling doesn't affect which one is scheduled.

Figure 2 shows the average solution quality as a percentage above the greedy solution for both the 100-task and 1000-task problems. Each point is the averaged over 100 problem instances. The results show that the average solution quality (*i.e.*, average schedule priority) increases with the size of the priority segmentation subset. For the 1000-task problems, the average solution quality only increases very slightly for smaller C_{ave} values and sometimes the average solution quality is lower than greedy (*e.g.*, when $C_{ave} = 10$ and the subset size is 8). In general, the results shown in the graphs illustrate the trend that we expected, namely a gradual increase in average solution quality as the subset size is increased. The 100-task graph also shows a gradual approach to optimal (12-percent above greedy) for the $C_{ave} = 20$ case as the subset size is increased.

As expected, the time required to perform priority segmentation increases with the size of the problem and the size of the subset. An example of the CPU time for the case where $C_{ave} = 20$ is shown in Figure 3. Notice that the CPU time levels off for the 100-task case when the subset size is 32, in part because priority segmentation is finding optimal solutions.

At first we didn't expect to see the slight decrease in average solution quality that occurred in some of the results. This result can be explained by the fact that priority segmentation with different subset sizes will search different parts of the scheduling space, thus it is possible for a larger subset to result in a lower average solution quality for any given problem instance. We expect that larger priority subsets will generate better solutions on average, although better solutions are not guaranteed.

The results also show that the percent improvement in solution quality over greedy increases with C_{ave} for a given priority subset size. This is because as C_{ave} becomes larger, the set of tasks to choose from for a given resource availability time slot increases. This also increases the expected value of the maximum priority of these tasks. In the limit as C_{ave} approaches infinity, the probability that there is a task with maximum priority for every time slot approaches one. Thus as C_{ave}



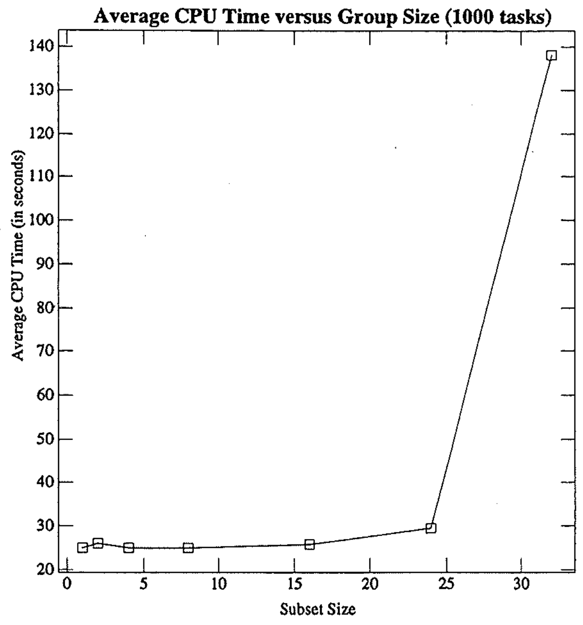


Figure 3: Average CPU time versus priority segmentation subset size ($C_{ave} = 20$, 100-task and 1000-task problems).

increases, we expect the optimal schedule score and the average priority segmentation score to increase.

Discussion and Related Work

Admittedly, the results presented here are just a first step toward understanding both this class of problems and the priority segmentation algorithm. Although we have used priority segmentation to address over-constrained scheduling problems for at least two of our customers, we are just beginning to evaluate its performance on more general problems. Our initial results indicate that priority segmentation can produce better quality solutions that greedy under certain conditions (*e.g.,* when the average constrainedness is large). Our result also suggest that, for this class of problems, greedy solutions are often very competitive with more sophisticated algorithms. Additional experiments and analysis are necessary to clearly determine the conditions under which priority segmentation is worth the computational effort.

We are aware of other work on over constrained scheduling problems. For example, the work by Bressina *et al.* (*e.g.*, (Bresina, Morris, & Edgington 1997; Drummond, Bressina, & Swanson 1994)) on ground-based telescope scheduling used a dynamic priority weighting scheme that depends on both the value and age of the tasks (*i.e.,* older tasks are given a higher priority). Sobiesk *et al.* (*e.g.*, (Olawsky, Kregsbach, & Gini 1995; Sobiesk, Kregsbach, & Gini 1996)) have investigated a stochastic dynamic programming approach to solving over-constrained product-quality planning problems. We have not yet evaluated either of these approaches on our over-constrained remote-sensing problems. As mentioned above, priority segmentation can be viewed as a variation on Ibaraki's work on hybrid search (Ibaraki 1978) and thus is related to other hybrid search algorithms (Pearl 1984).

Lemaitre *et al.* (Lemaitre & Verfaillie 1997; Bensana, Lemaitre, & Verfaillie 1999) have looked at a similar problem of scheduling imaging tasks for the Spot5 satellite. For their problem, tasks have fixed duration and fixed start times. They present results that compare a variant of depth-first branch-and-bound called Russian Doll Search to branch-and-bound and an implementation using ILOG Solver. Their results show that Russian Doll Search takes significantly less time on average than the other algorithms to generate good quality solutions. Although the ILOG-based approach requires more computation time, they argue that it may still be the better choice because it is a more general approach. Their work helps to justify our simplification of the more general satellite scheduling problem. In addition, their results support our belief that algorithms tailored to exploit the structure of a class of problems can often outperform general purpose approaches.

Conclusions and Future Work

We have presented a hybrid scheduling algorithm called priority segmentation. Our initial results indicate that it can produce better results than a simple greedy algorithm when the average number of tasks competing for a resource at any given time is high (*i.e.*, greater than 10). Our results also suggest that greedy solutions can be competitive for this general class of problems.

We intend to extend the work presented in this paper in several ways. First, we are planning to create a more sophisticated random problem generator that will allow us to evaluate scheduling algorithms on a wider range of problems (and more realistic problems). Second, we are planning to consider other suboptimal scheduling techniques such as real-time search (Korf 1990). Finally, we intend to implement and evaluate other approaches (*e.g.,* local search, Russian Doll Search (Lemaitre & Verfaillie 1997; Verfaillie, Lemaitre, & Schiex 1996), dynamic programming, etc.) to scheduling satellite resources.

Acknowledgments

This work has benefited from discussions with members of the Veridian-PSR Resource Optimization Group. Greg Heberle wrote the software that generates random remote sensing problems. Jesse Chappell helped with the algorithm implementation. John Calvit provided helpful comments on a version of this paper. This work has also benefited from discussions with some of our customers. We like to thank the reviewer for his comments and suggestions. We would also like to thank David Harrison, William Cheng, Leslie Lamport and the Free Software Foundation for their graphics and text formatting software.

References

Baptiste, P.; Pape, C. L.; and Nuijten, W. 1995. Incorporating efficient operations research algorithms in constraint-based scheduling. In *Proceedings of the First International Joint Workshop onArtificial Intelligence and Operations Research.*

Bensana, E.; Lemaitre, M.; and Verfaillie, G. 1999. Earth observation satellite management. *Constraints* 4:293–299.

Bresina, J. L.; Morris, R. A.; and Edgington, W. R. 1997. Optimizing observation scheduling objectives. In *Proceedings of the NASA Workshop on Planning and Scheduling for Space.*

Drummond, M.; Bressina, J.; and Swanson, K. 1994. Just-in-case scheduling. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94).*

Fox, M. S., and Sycara, K. P. 1990. Overview of CORTES: A constraint based approach to production planning, schedulingand control. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.*

Ibaraki, T. 1978. Depth-m search in branch-and-bound algorithms. *International Journal of Computer and Information Sciences* 7:315–343.

ILOG Corporation, Gentilly Cedex, France. 1998. *ILOG Optimization Suite: White Paper.*

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2–3):189–211.

Larson, W. J., and Wertz, J. R., eds. 1992. *Space Mission Analysis and Design.* Kluwer Academic Publishers, second edition.

Lemaitre, M., and Verfaillie, G. 1997. Daily management of an earth observation satellite: Comparison of ILOG Solver with dedicated algorithms for valued constraint satisfaction problems. In *Proceedings of the 1997 ILOG Optimization Suite International Users' Group.*

Olawsky, D.; Kregsbach, K.; and Gini, M. 1995. An analysis of sensor-based task planning. Technical Report 95-51, Department of Computer Science, University of Minnesota.

Pacific-Sierra Research Corporation, Arlington, VA. 1997. *GREAS Application Framework Reference Manual, Version 4.0.*

Pearl, J. 1984. *Heuristics.* Reading, MA: Addison-Wesley.

Sadeh, N., and Fox, M. S. 1990. Variable and value ordering heuristics for activity-based job-shop scheduling. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.*

Smith, S. F. 1994. Reactive scheduling systems. In Brown, D., and Scherer, W., eds., *Intelligent Scheduling Systems.* Kluwer Publishers.

Sobiesk, E.; Kregsbach, K.; and Gini, M. 1996. Overconstrained scheduling using dynamic programming. In *Proceedings of the AI and Manufacturing Research Workshop, National Conference on Artificial Intelligence (AAAI-96).*

Verfaillie, G.; Lemaitre, M.; and Schiex, T. 1996. Russian doll search for solving constraint optimization problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96).*