Mixed Initiative Scheduling for the Air Force Satellite Control Network

Adele E. Howe, L. Darrell Whitley,
Laura Barbulescu, Jean-Paul Watson
Computer Science Department, Colorado State University,
Fort Collins, CO 80523
{howe,whitley,laura,watsonj}@cs.colostate.edu

Abstract

Scheduling access to Air Force satellites is an over-constrained problem. Over 500 requests for access are received for each day's operation; after manual scheduling, over 100 conflicts typically remain in the schedule. We are developing a prototype scheduling system that will assist human schedulers in this problem. Automated scheduling promises to reduce the number of conflicts, but will never remove all of them. Thus, the system will allow the human schedulers to examine alternative schedules so that they can better negotiate changes with the users.

1 Introduction

The prevalent view of scheduling is that of automatically constructing the optimal or best solution to a single problem instance on a single objective. While this view works, more or less, for commonly studied scheduling problems such as static job-shop, it breaks down in applications that operate continuously, receive requests asynchronously, trade-off between multiple usually competing objectives or require judgment about priorities. Our application, scheduling access to remote satellite tracking stations (RSTS), possesses all four characteristics. The tracking stations operate 24 hours each day with emergency requests arriving at any time, and the demand outpaces the available resources. This necessitates tough decisions about which accesses are most important and whether some accesses can be partially accommodated.

In a recently initiated project, we are developing a prototype system to assist human schedulers in constructing schedules for the RSTS application. Several key issues must be resolved. Which scheduling algorithms and heuristics are best suited to this over-constrained problem? How can automated scheduling best function for dynamic scheduling over variable intervals? What objective function(s) match the requirements of the application and of the algorithms for automating the process? How a mixed-initiative scheduling system can improve the efficiency of the process and solutions? Although all of these issues need to be resolved ultimately, in this paper, we focus on the last issue and describe our methodology and design for a mixed-initiative system for RSTS scheduling.

2 RSTS Application

In the Air Force Satellite Control Network, Remote Tracking Stations provide the satellite-to-ground interface necessary for satellite command and control operations. Remote Tracking Stations are located at nine (9) sites with 16 antennas. The stations are placed so as to maximize area coverage and multiple support capability. Time on the various Remote Tracking Stations is a scheduled resource.

Scheduling is carried out by one of two Resource Control Complexes (RCCs), located at the Schriever Air Force Based (AFB) and the Onizuka AFB. Approximately 520 resource requests must be scheduled for each day of operations at Schriever.

Because the resources are scarce, requests may include backup time and resource specifications and may be subject to later negotiation. Thus, a client's request consists of a primary desired time window on a particular Remote Tracking Station, the amount of time required, the priority of the request and optionally, alternative time windows and resources. (Alternatives may also be determined automatically based on the availability of duplicate or similar resources and capabilities.)

Attempts to automate these tasks have either failed, or have yielded dubious results. Human schedulers are often still doing this task manually; only schedule entry, storage and display have been computerized. It can take a human one to two years to learn to schedule Satellite Remote Tracking Station requests at an expert level. Currently, a one day schedule is built manually by human experts over a one week period. On any given day, the schedulers manage seven sliding windows representing the next seven days of activity. During the course of a week, the schedule is gradually determined and frozen, except for high priority, emergency requests which may arrive at any time and must be included.

3 Mixed-Initiative Scheduling in SRTS

Shortening the scheduling window of SRTS from one week down to two or three days would mean managing fewer active scheduling windows at any point in time. In addition, shortening the scheduling time is also desirable because high priority requests may arrive late, causing disruption to the overall schedule.

Automation can help reduce the scheduling window, but by itself, automation is not the complete solution. If two requests require exactly the same resource at the same time, and there is no alternative way to satisfy both requests, then there is no feasible solution. Even when there are alternative ways to satisfy a request, contention for resources covering global "hot spots" may still preclude satisfying all requests. Human intervention is generally required to adjust the objectives of the schedule, to arbitrate what requests to abandon or to negotiate alternatives off-line. Emergency or extremely high priority requests for satellite time may also arrive in the last 24 hours before a schedule is to be deployed. This creates a need for fast schedule repair. What the automated part can do best is to anticipate problems and to provide the human scheduler with promising alternative schedules allowing him/her to dictate trade- offs in objectives and request satisfaction.

Thus, the system must support fast schedule proposal, schedule visualization, rapid consideration of alternatives and on-line editing for new requests. We are developing a system that has a suite of algorithms and heuristics for building schedules integrated with a visualization tool for examining and manipulating proposed schedules.

3.1 Algorithms for SRTS Scheduling

Scheduling of remote tracking stations is a constraint satisfaction problem in which no conflict-free solution exists and yet some solution must be generated. Often, an attempt is made to turn this kind of constraint satisfaction problem into a combinatorial optimization problem: a weighting is used to indicate which constraints are more important–and then an optimization routine is run to minimize the weighted set of violated constraints.

Search and combinatorial optimization methods are not enough to support a fielded system for resource scheduling. The combinatorial optimization approach has several problems in real world applications. First, it may be difficult or impossible to impose a meaningful weighting. Tuning a linear set of weights so that the weighted evaluation function reflects the same kinds of (possibly nonlinear) relative evaluations made by a human expert can be extremely difficult. Second, all the information needed to resolve a scheduling conflict may not be available to the automated scheduling system. In this case, it is impossible to build an evaluation function that captures the scheduling abilities of the human expert. For example, clients requesting a resource tend to make their request the highest allowable priority, so that priority information is less accurate and useful than one might expect. Given two requests of equal priority, an automated system may give both requests 70 percent of the requested time in other to minimize an objective functions. This may not at all be the kind of compromise that is appropriate.

At present, we are extending heuristics and algorithms from job-shop scheduling. In particular, we have implemented slack-based [9] and texture-based heuristics [1] coupled with several types of search algorithm, e.g., LDS [4] and HBSS [2]. The min-slack heuristic with greedy search can construct a preliminary schedule in approximately 2 minutes. Given 500 requests with 120 initial conflicts, we have been able to resolve about 80 percent of the conflicts using this combination. We have found that texture-based heuristics can further reduce the number of conflicts but at the cost of significantly more computation.

3.2 Interface Aiding Human SRTS Schedulers

When scheduling the Remote Tracking Stations, the human schedulers may negotiate changes with the individuals making the resource request. Such negotiation is beyond the abilities of any automated scheduler. Humans may also accumulate information that is very hard to quantify (or which should not be quantified). For example, if clients X and Y are in conflict over a resource, X may outrank Y and so usually win the conflict despite the fact that the requests are of the same priority. Human schedulers can anticipate and informally exploit this type of information.

Thus, instead of trying to fully automate the scheduling process, an alternative approach is to build an interactive system that *aids* the human scheduler by doing what a computer does best: tediously generating and trying out alternatives and managing information about the schedule. Currently, it takes a human expert about five hours to build a preliminary schedule for satellite scheduling, which our current algorithms can build in a few minutes.

Once an initial schedule is constructed, a user should then be able to interactively select, reject or fix parts of the schedule. When a scheduling decision is made that differs from the proposed schedule, the automatic scheduler can be re-run to find a new solution that optimizes around the fixed parts of the schedule. Since automated scheduling methods are to be used to generate initial schedules and to do rescheduling, very fast rescheduling methods may sometimes (but not always) be required.

Such an environment can do more than just provide an interface to the actual schedule and scheduling algorithms. It can also provide a data analysis tool to better understand the scheduling problems and statistic data about client requests and past schedules which may be useful in constructing new schedules.

It is common to find references to "interactive scheduling" in the literature, but there appears to be a limited number of ideas about how best to do this. There is general agreement that the user should interact with a schedule at an abstract, ideally graphical level that hides schedule implementation and optimization details to an appropriate degree [7] [8] [10]. Interactive Gantt Charts are a popular way to achieve this goal [6]. Generally, the user can make a change to the schedule; in response, the scheduler may be called upon to propagate the effects of the change and perhaps to optimize in response to the change [11] [3]. These ideas are incorporated in our prototype system.

3.3 Example of Scheduling Interface

To demonstrate the basics of our system, we will step through an example showing how it will look to a user. In the example terminology, the "Clients" will be those requesting time on a Remote Tracking Station; the "Users" are the schedulers who assign satellite resources to clients. The figures that follow are mock-ups of proposed visualizations for the user.

Figure 1 illustrates a hypothetical situation where the initial set of requests has been displayed, with darker levels of gray-scale indicating greater conflict (this would be in color online). The user can examine the set and then select a specific request to be "checkpointed". The dark rectangle outline highlights a specific client request. If this particular request appears to represent a significant bottleneck, the request can be checkpointed to allow the user to look at the partial schedule as it exists at the point when the system is trying to schedule this particular request.

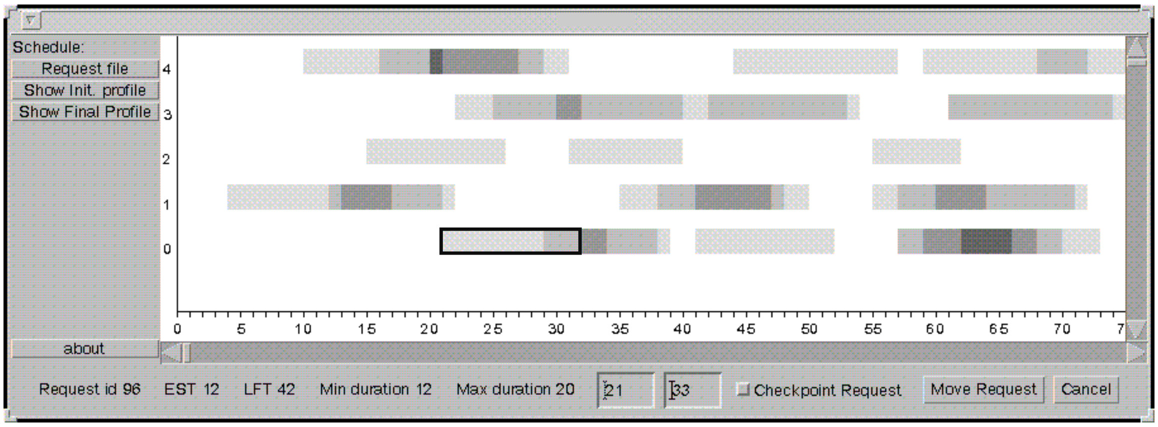

Figure 1: User can select a checkpoint in the schedule. A checkpoint indicates a key client request for which the user wishes to examine options when the system tries to schedule it.

Once the system has scheduled up until the checkpointed request, the user has some latitude in how to handle it. For example, the user could decide to look at historical information about the client making the request. Figure 2 illustrates a hypothetical situation where historical data is available about a client requesting a resource. The graph shows the actual duration allocated as a function of the original resource request. We know from human schedulers that some clients are more flexible than others; flexibility may also be a function of the size of the resource requested (as shown in this graph) or a function of the type of resource requested (which is not shown here).

Figure 3 illustrates suggested scheduling alternatives for a specific client request which has been highlighted and checkpointed as illustrated in the previous figure 1. The system shows various scheduling alternatives for this client request and data on how this impacts the schedule (i.e., the change in evaluation expected for each action).

Finally, figure 4 illustrates the set of recent emergency requests. One might define emergency requests to be those client requests which arrive in the last 24 hours before a schedule becomes active, or any requests that arrive after the schedule is active. In this case, the idea is that such requests are not random. A possible scheduling strategy is to avoid scheduling tasks on resources and specific time slots that have a recent history

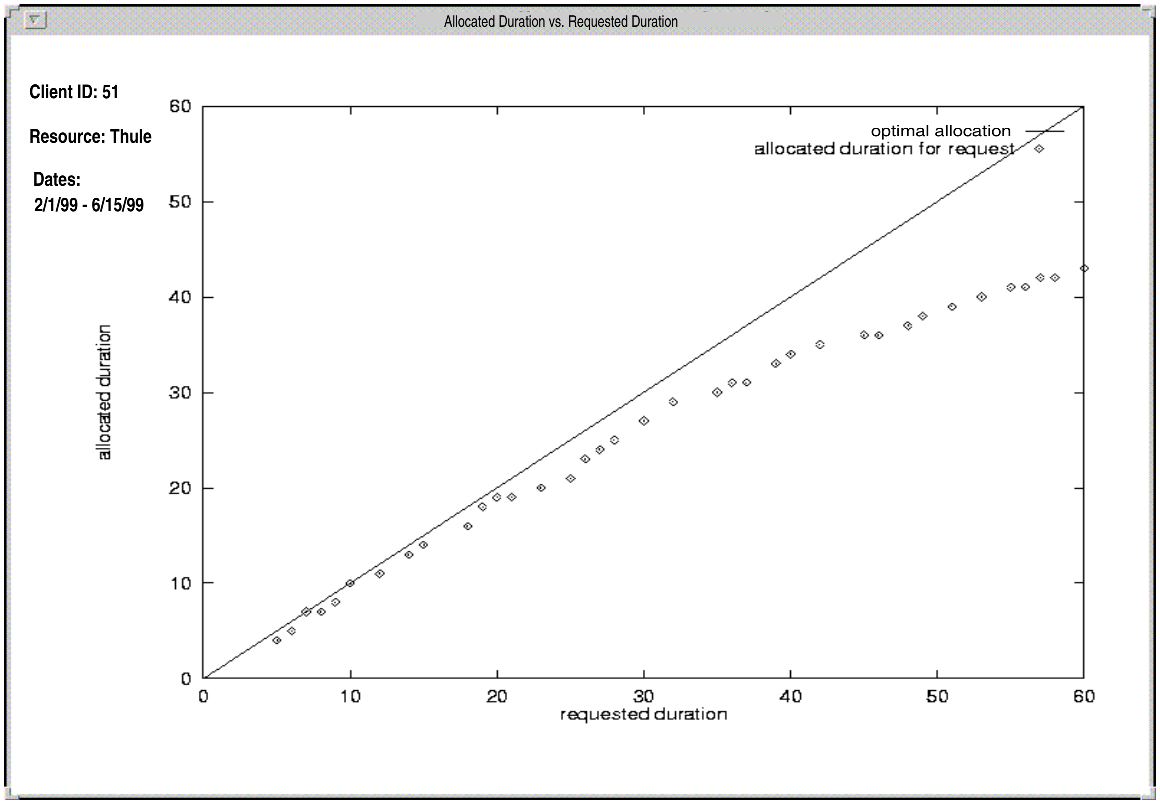

Figure 2: Graph for a single client mapping hypothetical previous resource requests by size on the x-axis against the actual time allocated on the y-axis. A scheduler could infer from this data that this client is more flexible about receiving less than the request time on a resource when the original request is larger.

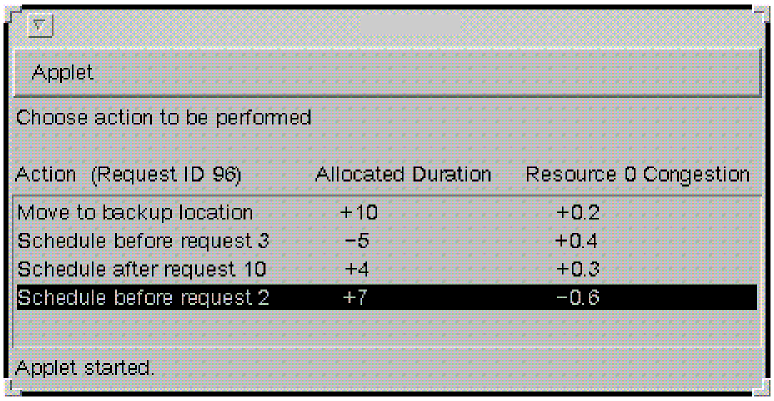

Figure 3: Display of suggested actions for scheduling a particular request. After a checkpoint has caused a stop, the user can determine what should be done with the request.

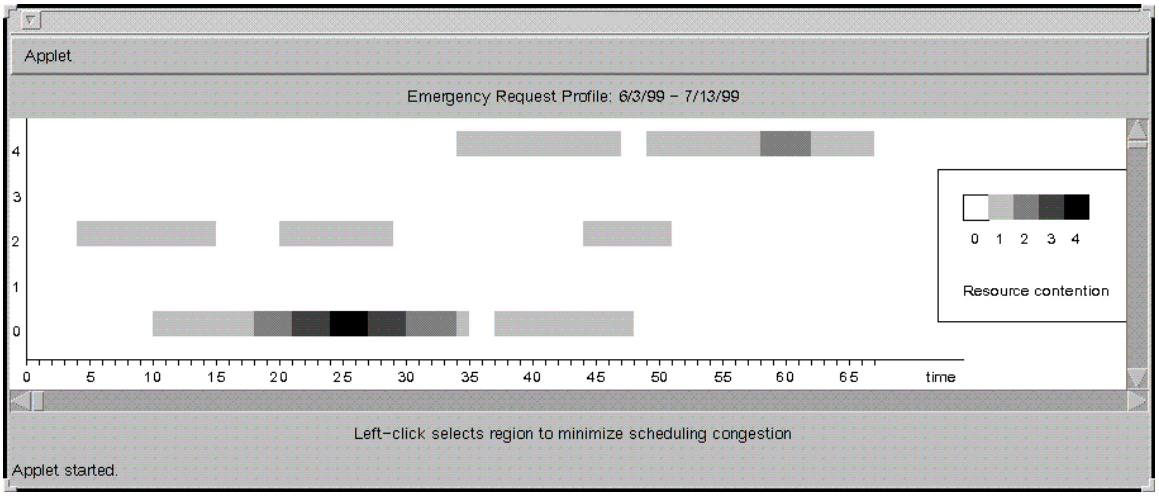

Figure 4: Profile of emergency requests submitted during the previous 40 days

of high emergency incidence. Human schedulers currently use a similar strategy for this application.

4 Project Status

At present, we have implemented a problem generator and a web based interface for viewing and manipulating schedules. The problem generator was necessary due to the sensitive nature of the application. We can automatically construct problems that are typical of the types received by the Air Force without compromising security. In addition, the generator allows us to exert experimental control during testing and evaluation. The generator [5] can be parameterized with the number of requests, disparity in their time window and duration sizes, and amount of clustering (contention at same times on the same resource). Based on the results of our evaluations, the best set of algorithms and heuristics will be included into the final system, allowing for particular algorithm/heuristic combinations to be selected. We recognize that users may wish to favor speed over optimality and thus iterative repair over optimization algorithms.

Because of the importance of the interface for these types of systems and the need to solicit feedback from the users and experts for the applications, the interface has been implemented in Java and will be accessible via the World Wide Web. This will allow Air Force users all over the world to access our system and demo the interactive scheduling environment. At present, the system generates initial schedules using min-slack/greedy search and allows the user to view, add, delete or move requests within the proposed schedule.

Acknowledgments We would like to thank Alex Kilpatrick of AFOSR and Brian Bayless of Falcon Air Force Base for providing us with the information on the application. This work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-97-1-0271. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] J.C. Beck, A.J. Davenport, E.M. Sitarski, and M.S. Fox. Texture-based heuristics for scheduling revisited. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-97)*, 1997.

[2] John L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[3] M. Deale, M. Yvanovich, D. Schnitzuius, D. Kautz, M. Carpenter, M. Zweben, G. Davis, and B. Daun. The space shuttle ground processing scheduling system. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, pages 423–449. Morgan Kaufmann, 1994.

[4] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[5] Adele E. Howe, L. Darrell Whitley, Jean-Paul Watson, and Laura Barbulescu. A study of air force satellite scheduling. In *World Automation Congress 2000*, June 2000. to appear.

[6] N. Sadeh. Micro-opportunistic scheduling: The micro-boss factory scheduler. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, pages 99–135. Morgan Kaufmann, 1994.

[7] S. Smith and O. Lassila. Toward the development of flexible mixed-initiative scheduling tools. In *ARPA-Rome Laboratory Planning Initiative Workshop*. 1994.

[8] S. Smith, O. Lassila, and M. Becker. Configurable, mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, pages 235–241. 1994.

[9] Steve Smith and C.C. Cheng. Slack-based heuristics for constraint satisfaction problems. In *Proceedings AAAI-93*, pages 139–144, 1993.

[10] A. Tate, B. Drabble, and R. Kirby. O-plan2: An open architecture for command, planning and control. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, pages 213–239. Morgan Kaufmann, 1994.

[11] B. Yen and M. Pinedo. On the design and development of scheduling systems. In *4th International Conf. of Computer Integrated Manufacturing and Automation Technology*. 1994.