N93-18662

A Simulated Annealing Approach to Schedule Optimization for the SES Facility

S3-63
137229
P-4

Mary Beth McMahon and Jack Dean
Planning and Scheduling Technology Group
McDonnell Douglas Space Systems Co.
16055 Space Center Blvd
Houston, TX 77062

Introduction

The SES is a facility which houses the software and hardware for a variety of simulation systems. The simulators include the Autonomous Remote Manipulator, the Manned Maneuvering Unit, Orbiter/Space Station docking, and shuttle entry and landing. The SES simulators are used by various groups throughout NASA. For example, astronauts use the SES to practice maneuvers with the shuttle equipment; programmers use the SES to test flight software; and engineers use the SES for design and analysis studies.

Due to its high demand, the SES is busy twenty-four hours a day and seven days a week. Scheduling the facility is a problem that is constantly growing and changing with the addition of new equipment. Currently a number of small independent programs have been developed to help solve the problem, but the long-term answer lies in finding a flexible, integrated system that provides the user with the ability to create, optimize, and edit the schedule.

COMPASS is an interactive and highly flexible scheduling system. However, until recently COMPASS did not provide any optimization features. This paper describes the simulated annealing extension to COMPASS. It now allows the user to interleave schedule creation, revision and optimization. This practical approach was necessary in order to satisfy the operational requirements of the SES.

Statement of Problem

The SES facility is scheduled a week at a time. A work week consists of seven days, each of which is divided into six 4-hour "sessions." Each session has two sides, side-a and side-b. This allows two people to work in the facility at the same time. Each person requiring time at the facility makes a request telling what equipment is needed for the simulation. A request consists of the required simulators, the preferred days and sessions, and optionally, a preferred side. Each person may make one or more requests per week and may ask for multiple iterations of the same request. The SES scheduler satisfies their requests by creating a schedule based on priorities. The SES manager determines the priority of each request by the type of work being done and the number of repetitions requested. For example, mission related activities have a higher priroity than software development activities. And the fourth repetition of a request typically has a much lower priority than the first. Each week there are about 60 - 70 requests and 76 session slots to be filled. There are additional requests at the last minute for empty slots, as well as high priority requests coming through that may bump lower priority items.

There are a few guidelines by which the SES facility is scheduled. First, there is only one instance of each simulator; therefore the persons working on side-a and side-b must use mutually exclusive sets of equipment. Second, certain pieces of equipment reside only on certain sides; therefore side assignments must coincide with equipment requirements. Third, a person may state a preference for particular sessions, may state which sessions are acceptable, and may state which sessions are unacceptable. The schedule should try to accommodate the preferences, but can place the person in an acceptable session when the preferred sessions are not available. Under no circumstance should a person be placed in a session which has been marked as unacceptable. Fourth, a person can only work up to two sessions in one day, and if they do, the sessions should be consecutive so that a straight eight hour day is worked. Fifth, each person should have at least an eight hour break between non-consecutive scheduled sessions. Sixth, if a person works more than one third shift (session five or session six) then the third shift sessions worked should be on consecutive days or at least two days apart.

Each request has a primary and secondary requestor. The above rules must be satisfied in the event that either the primary or secondary requestor work the session.

There are two goals to consider when scheduling the SES facility. One is to produce a weekly schedule in which the largest number of requests are satisfied. The other is to fill the schedule with the highest priority items. These two goals must be satisfied simultaneously, but there are no rules defining the trade-offs be-

tween quantity and priority. It is left up to the scheduler to produce a schedule which, in his opinion, works the best. In fact, if so inclined, the scheduler may actually violate resource or timing constraints listed above when producing the schedule.

Currently, the requests are entered on a PC and then transferred to a Cyber computer where an optimization routine written in FORTRAN finds 10 candidate schedules. The SES manager then selects one of the 10 schedules and hand edits it. The editing usually consists of adding late assignments and moving assignments around for subjective reasons. This is done with paper and pencil to keep track of resource assignments. Finally the handwritten schedule is entered into a PC, using a drawing program, where it is printed out for distribution.

When providing an integrated solution for the SES problem, all phases of the scheduling process must be considered. First, the scheduling system must be able to accept and handle all of the constraints and preferences described by the requests. Second, the system must provide the SES manager with an initial feasible schedule which is at least as "good" as the initial schedules produced by the FORTRAN program. Third, the system must allow the schedule to be modified, even if it means overriding constraints. And fourth, the system must print the schedule in the prescribed SES format.

Background

An interactive scheduling system allows the user to impose subjective constraints such as the trade-off between the quantity of requests satsified and the priorities of the activities scheduled. A non-chronological system allows the user to place activities anywhere in the week, so that high priority items can be scattered throughout the week and low priority items can fill the leftover time slots. These two characateristics, along with the fact that COMPASS only produces feasible schedules, lay the ground work for solving the SES scheduling problem. The significance of these characteristics is described further.

An interactive scheduling system provides an environment where a mixed initiative is possible; that is, it lets the computer do what it does best (check constraints and calculate feasible intervals) and lets the human do what he/she does best (provide heuristic and subjective inputs into the schedule). Together the two can cooperatively produce a schedule which reflects both the hard constraints and subjective preferences. Subjective preferences may be controlled through input from the user. The input may reflect scheduling heuristics, such as the order in which to schedule the activities and whether to schedule as soon as possible or as late as possible. The input may reflect the desired look of the schedule, such as choosing where to place the activity from among the feasible intervals of time. Or the user may interactively direct the search, by specifying which items to freeze and which items to optimize.

In contrast, a fully automated system requires that all data be completely loaded before the system begins scheduling. All rules about scheduling preferences and optimization must be coded into the system before the scheduling process begins. The system then runs uninterrupted until it finds one solution (or many depending on the system) and then presents its findings as the final schedule. There is generally no effective way of editing the schedule once the solution is found. This method of scheduling is perfectly acceptable when the problem is bounded and the domain can be described completely. However, in a highly subjective scheduling domain, coding all of the rules (and exceptions-to-the rules) may become very laborious or even impossible.

A non-chronological scheduler allows the system to place activities anywhere on the timeline. The system has an omniscient view of time and can determine all the feasible intervals of time where the next activity may be placed. As each activity is placed on the schedule, constraints created by that activity must be propagated (either in the environment or directly to other activities). When new activities are placed on the schedule, they are constrained by the activities already on the schedule. A benefit of non-chronological scheduling is that high priority items may be placed on the schedule first and guaranteed that they be completed. Then the schedule may be filled with the lower priority items.

In contrast, a simulation-based scheduler starts at the beginning time of the schedule and as it progresses through time, it places activities on the schedule. When resources become available, the system has a choice about which item to place next on the schedule. Once the schedule reaches the ending time, the schedule can be evaluated and another pass may be made, perhaps making different choices about what to place at each decision point. Historically, simulation-based schedulers are very popular in the job shop arena as they naturally model the behavior of plant operations.

COMPASS, with its simulated annealing extension, searches only the feasible solution space. Some schedulers only search the feasible solution space, while others search both the feasible and infeasible solution space. It may be substantially easier to find good solutions if the scheduler is allowed to wander through the infeasible solution space. However, allowing infeasible solutions also greatly increases the size of the search space. There are far more infeasible solutions than feasible solutions. By prohibiting the search of the infeasible solution space, the scheduler has more time to spend evaluating feasible solutions. Deciding which solution space to search depends on the optimizing algorithm.

Approach

This section describes how the simulated annealing routine is used in conjunction with COMPASS.

Given a group of selected activities to optimize, the simulated annealing algorithm calls upon the COMPASS scheduling engine to unschedule then reschedule the selected activities in a different order. The activities are continuously rescheduled and the objective function is evaluated for each new schedule until a user specified time limit is up. When time is up COMPASS displays the schedule with the best score.

The user designates the focus of attention for the optimization by selecting a subset of activities. The user can select all of the activities, in which case the entire schedule is optimized with respect to the objective function. Or the user may select a subset of activities, in which case only part of the schedule is optimized. A benefit of this is that the user can selectively optimize parts of the schedule which need improvement, leaving the rest of the schedule intact.

The user may also specify the amount of time in which to run the simulated annealing algorithm. For simple schedules or small subsets of activities a small amount of time may be all that is necessary. COMPASS displays each new try as it is created. The user can actually sit and watch as the schedule is being modified. Once time is up, COMPASS redisplays the best schedule.

A scenario for using COMPASS and its simulated annealing extension is as follows. A first cut at the schedule can be created using the optimization function. The user can edit the schedule by unscheduling some activities or by forcing unscheduled high priority activities (overriding any constraints) onto the schedule. By evaluating the schedule, COMPASS will display all activities which now have conflicts. The user can unschedule the conflicting activities and reschedule them (using the optimization function or by placing each down interactively). The interaction between user placement and optimization continues until the final schedule is reached.

Implementation

Simulated annealing is an optimization technique which combines gradient descent with randomness to find global optima. The process used to control the optimization is analogous to the annealing of metal; hence the name simulated annealing. The annealing process is based on the laws of thermodynamics which state that atoms tend toward a minimum energy state. A metal is annealed by raising the metal to temperature over its melting point and then gradually cooling it. At high temperatures the atoms are in a high energy state, violently and randomly moving about. As the metal cools, lower and lower energy states become increasingly likely. By cooling the metal slowly, the lowest possible energy state, the global minimum, can be achieved.

Simulated annealing is used to find global minima in optimization problems in the following fashion. An initial solution to the optimization problem is found by some means. The search space of solutions becomes the state space of the simulated annealing algorithm. An objective function, for which a global minimum is to be found, is defined over the search solution space. The objective function corresponds to the energy function. For each iteration, a random change is made to the state to obtain a new state. If this new state has a lower energy than the previous state, the new state is kept. If the new state has a higher energy than the old state, it is kept with a probability that varies with the simulated temperature. Continuing the analogy to the annealing of metal, this probability is proportional to the exponential of -c/kT, where c is the change in the energy level, k is constant analogous to the Boltzmann's constant for physical systems, and T is the simulated temperature. At very high temperatures, most changes in state are accepted, and the result approaches a random walk through the solution space. At very low temperatures, the probability of accepting a change that increases the total energy vanishes, and the random walk is limited to changes which decrease the total energy. This results in a gradient descent to the local minima. To achieve the global minimum, the temperature is started off very high and gradually reduced. For each local minimum there is a temperature which will allow the random walk to escape the local minimum, but not the global minimum.

In order to apply this algorithm to the SES optimization problem, the following have to be defined: (1) the state space of searched solutions, (2) the energy or objective function to be minimized, (3) the method for calculating the initial solution, (4) the method for randomly changing from one state to the next, and (5) the temperature decay algorithm.

The solution space consists of all feasible schedules. A feasible schedule is one that satisfies all the constraints. The constraints that are applicable to the SES scheduling problem are the resource availability constraints, the temporal constraints, and the rules discussed in the Statement of Problem section of this paper.

The objective function is the negative of the sum of the values of the scheduled activities. (The negative is used so that minimization of the objective function indicates improving schedules.) The value for each activity is derived from the priority input field of the schedule request. The priority is an integer between 1 and 22 inclusive, with 1 being the highest priority (most important). The value for the activity is set to 23 minus the request priority. Thus increasing value means increasing importance of the task.

An initial solution is found using a first fit decreasing algorithm. The activities to be scheduled are sorted into decreasing value order. The sorted activities are then scheduled using a front loading, or first

fit, scheduling algorithm.

Once a feasible schedule is found, a new random schedule is calculated in the following fashion. First, the probability that a scheduled task should be removed is calculated, based upon the current simulated temperature. The probability of removal is calculated using an equation of the form of the Boltzmann equation described above. Thus the probability of removal is higher at high temperatures than it is at low temperatures. This has the effect of allowing larger state changes at high temperatures and minor changes at low temperatures.

Next, each activity is examined in decreasing value order. If the activity is already scheduled, and a randomly generated number is less than the probability of removal, the activity is removed from the schedule. If the examined activity is previously unscheduled, and a randomly generated number is less than a constant probability of placement, the activity is placed on a list of activities to be scheduled.

Once the entire activity list is examined, with some of the activities randomly selected and unscheduled, the list of activities to be scheduled is examined. For each activity, the program first tries to schedule the activity in one of the preferred sessions. If that fails (the activity is not scheduled), the program attempts to schedule the activity in any one of the acceptable sessions.

Once the new schedule has been created, the energy value for this new schedule is calculated. If the new energy value is lower than the current energy, the new schedule is kept since it reflects an improved schedule. If the new energy is greater than the current energy (reflecting a poorer schedule), the probability of accepting this schedule is calculated using the Boltzmann equation described above.

Finally, the temperature decay used is a simple inverse linear function. The simulated temperature is set according to the equation $T = T0 / (1 + t)$, where T is the current temperature, T0 is the initial temperature, and t is the simulated time.

Conclusions

The simulated annealing algorithm has been successfully implemented and integrated into the COMPASS architecture. This new addition allows the user to automatically, as well as interactively, create schedules. This combination of automatic and interactive capabilities provides the user with greater functionality and control over the development of the schedule. The user can define the level of interaction/automation necessary in order to produce the best schedule.

The user selects the activities that are to be optimized. The user may optimize the whole schedule by selecting all of the activities or part of the schedule by selecting a subset of the activities. (In the SES problem the objective function is based on the priorities of the activities, so it is feasible to apply it to subsets of activities as well as the entire set.) The previously scheduled activities that have not been selected remain frozen on the schedule. This is especially beneficial in rescheduling once the initial schedule is underway and an event occurs which requires parts of the schedule to be reworked.

The SES scheduling problem requires an integrated system which will create an initial feasible schedule, allow the user to alter or optimize parts of the schedule, and will print out the schedule in the desired format. COMPASS now provides all of these capabilities in one cohesive package. The user can schedule both interactivly and automatically. The user can override any constraints by forcing an activity onto the schedule at a specific time. The user can validate the schedule using existing evaluation functionalities. And the user can print out reports in the desired format using PostScript.

References

[1] Fox, Barry R., *Mixed Initiative Scheduling*, AAAI - Spring Symposium on AI in Scheduling, Stanford, CA, March 1989.

[2] Fox, Barry R., *Non-Chronological Scheduling*, Proceedings AI, Simulation and Planning in High Autonomy Systems, University of Arizona, March 1990, IEEE Computer Society Press.

[3] Lund, Chet, *Expert System for Scheduling Simulation Lab Sessions*, Proceedings of the 1991 CLIPS Conference, pp 784-791.

[4] Wasserman, Philip D., Neural Computing Theory and Practice, pp 80-83.