Enhancements to the Ground Processing Scheduling System *

Juan Jose Blanco and Lina Khatib
Computer Science
Florida Institute of Technology
Melbourne, Fl. 32901
{jblanco,lina}@cs.fit.edu

Abstract

This research statement describes work in progress to enhance the performance of the Ground Processing Scheduling System (GPSS). The GPSS is a constraint-based scheduler that, using an anytime algorithm, deals with three kinds of constraints: temporal, resource, and configuration. It starts with a complete schedule that may have some constraint violations (conflicts) and gradually improves the schedule by reducing the violations. The objective is to optimize the schedule in terms of constraint violations and resource utilization.

Introduction

The Ground Processing Scheduling System (GPSS) is a constraint-based scheduler. It is used for assigning *times* and *resources* to all the activities involved in the maintenance, repair and preparation of the fleet of Space Shuttles from the time one lands on the ground until it is next ready to launch. The problem domain is a very demanding one in which activities are constrained by resource limitations, configuration requirements, and temporal specifications.

Although the deployment of GPSS has been mainly a success, we have identified certain areas that could be enhanced. Through such enhancements, we provide a way to test techniques applicable towards solving general constraint satisfaction and scheduling problems.

Automatic scheduling in GPSS

Classical scheduling is the process of assigning times and resources to the activities of a plan. A scheduling problem can be naturally mapped into a constraint satisfaction problem (CSP), where the values (times) that the variables (activities) can take are constrained by temporal relations and by the finite capacity of the available resources. The complexity of classical scheduling is greatly increased when the problem is extended to include configuration requirements and effects, i.e., certain activities requesting an attribute of an object to be in a specified state and modifying such a state when they are completed.

The Ground Processing Scheduling System (GPSS) is a very large constraint-based scheduling assistant developed by Lockheed and NASA to aid in managing the maintenance, repair, and preparation of the Space Shuttle for a new mission. An orbiter processing flow between missions takes, on average, between 1000 and 2000 tasks to be scheduled, for about 10,000 to 16,000 individual shifts of work, and with around 100 scheduling changes made each day (Deale et. al. 94) 1994. The sheer size of the problem is further complicated by the need of accounting for configuration constraints, as well as by a demanding domain.

The temporal reasoning involved is a two-fold process involving, first, resolving *predecessor-successor* relations between tasks, and second, performing a specialization of classical scheduling called *fixed preemptive scheduling*. Fixed preemptive scheduling implies making activities comply with a work calendar which indicates when an activity can be scheduled.

All of the above account for the high degree of complexity of the scheduling domain faced by GPSS. Such complexity was the main obstacle the original designers of the system encountered when pursuing a solution based on constructive, or systematic, constraint-solving algorithms. These algorithms could require a prohibitive amount of time to solve easy instances of the problem (Zweben, Davis, and Deale 93). The intractability and other characteristics of the domain (such as high dynamism and requirement of rescheduling, over-constrainedness, search for optimality) led to the utilization of a repair-based, or stochastic approach.

GPSS Process

The GPSS provides a graphical interface for the user to specify all input. Then, it performs a first stage of tem-

[1] An extended version of this paper was published by IEEE as *Proceedings of TIME97*

poral reasoning by satisfying all temporal constraints among tasks. The Waltz algorithm (Waltz 75), was modified to propagate temporal constraints and achieve temporal constraint satisfaction with $O(N^2)$ complexity, where N is the number of tasks.

As a second step of temporal reasoning, each activity is broken into subtasks according to their work calendar. This setting of tasks serves as the initial assignment for the iterative algorithm to repair. Although the assignment is complete, it is not by any means valid since it contains resource and configuration conflicts. The GPSS provides an interactive mode that allows users to manually resolve constraint violations by moving tasks along the time line. Temporal consistency is maintained after each move by running the Waltz algorithm, and resource and attribute profiles are continuously updated and available to the user through graphical output. The goal here is to enhance the scheduling engine that performs automated conflict resolution and optimization.

GPSS uses a hill-climbing method coupled with a simulated annealing implementation to repair conflicts. The user can select relevant resources to focus on and determine the maximum time allotted for deconfliction. Also, the user can choose to fix tasks in their current time slot. During each iteration of the repair algorithm, the following steps are taken:

1. Select a conflict (resource or configuration) to be resolved taking into account specified priorities.

2. Build a table of tasks involved in the unsatisfied constraint (selected conflict). For each task, calculate ten heuristic values that will aid in determining which task to move for resolving the conflict.

3. The weighted sum of all heuristic values serves to select the task to move forward or backward (following a typical hill-climbing approach). The Waltz algorithm is run to satisfy all temporal constraints. Note that although the move resolves the conflict at hand, running Waltz to achieve temporal consistency again may cause new resource and/or configuration constraint violations to arise.

4. Simulated annealing (Kirkpatrick, Gelatt, and Vecchi 83) is then used to obtain the next state. If the move has caused an overall reduction in the number of conflicts, it is accepted and the new state becomes the initial state for the next iteration. But, unlike pure hill-climbing techniques which reject moves to higher-cost states, such moves are made in GPSS with a certain probability to avoid the tendency of stochastic CSP methods of becoming stuck in local minima.

Enhancement of the GPSS scheduling engine

This section describes the enhancements we propose for the GPSS which affects the efficiency of its automatic deconfliction process.

The deployment of GPSS into its operational environment at KSC has been mainly a success since it was first used in March 1992 for space shuttle Columbia's STS-50 flight. However, as its user community has expanded and has relied on its results for delivering daily schedules for the shuttle ground processing, there has been a clear trend toward utilizing GPSS's interactive mode rather than the automatic scheduling engine. The main reasons for this behavior follow.

1. GPSS global improvement of schedules deviates from the way human schedulers perform their work. Rather than being interested in an overall good schedule extending over the approximately 60 to 80 days a typical flow lasts, the high complexity and dynamism of the environment makes a human expert want a conflict-free schedule for a selected period of time (e.g., schedules are currently updated and delivered to the OPF to account for the next 48 hours of work). GPSS does not allow such preference. Instead, the automatic deconfliction process selects conflicting tasks randomly, from within the whole time-line, and resolves conflicts in each iteration. Hence, the algorithm may be spending a lot of time to resolve constraint violations at the tail end of the schedule which is currently of much less interest to the user. A side effect of the difficulty for human users to appreciate the "goodness" of such complex and global schedules is the loss of confidence in the computer's result.

2. The automatic scheduling algorithm is quite slow; resolving conflicts in such a complex environment is a time consuming process for any algorithm, but we believe some improvements can be made to the scheduling engine to increase its efficiency.

Following the intention to resolve the above problems, we propose a series of enhancements to the current implementation of GPSS.

Fencing capability

Adding a feature that allows the user to specify a temporal window of interest on which to focus will greatly increase the applicability of GPSS. Fencing techniques are useful and applicable to any other scheduling system with similar characteristics. The user could select the time period for which he/she desires to obtain a

conflict-free schedule and GPSS should then concentrate on removing the conflicts occurring during that period.

Improvement of activity selection for deconfliction

GPSS resolves constraint violations (either resource or configuration) by moving a task or a set of tasks involved in the conflict to a new time period where the conflict does not arise. There are basically two trends in performing the selection of the task to move:

1. Use local heuristics. This is computationally cheap but uninformed in the sense that they are mere heuristics. This type of selection is used by GPSS.

2. Use a greater depth of repair or look-ahead knowledge. For instance, the actual use of heuristics in GPSS does not take into account the fact that moving a task will generally require moving several other temporally dependent tasks when the Waltz algorithm is re-applied. This may cause an increase in constraint violations which has not been previously calculated, and which may be worse than for other tasks that scored lower in the heuristic evaluation.

It is not clear which of the methodologies is better, since performing the more informed look-ahead involves a computational overhead that may degrade the scheduler if the cost is overly expensive. However, this more knowledge- intensive technique assures that each of the moves performed will actually yield the best possible improvement in cost (Minton et. al. 92). We intend to run tests in both ends of the scale. Based on these studies, we will determine the optimal technique for activity selection when deconflicting in the Space Shuttle domain, extracting general conclusions about the tradeoff between informedness versus speed in CSP and scheduling problems.

Enhancement of the stochastic search method

The main advantage of simulated annealing over other stochastic search methods (pure Hill-climbing, Min-conflicts, GSAT) is its ability to escape local minima while searching, instead of having to re-start from a new initial assignment. However, precisely the way simulated annealing performs this jump-out has been lately questioned by several authors. For instance, in (Selman and Kautz 93), a performance comparison between a greedy GSAT and the annealing algorithm led to the conclusion: "much of the effort expended by simulated annealing in the initial high temperature part of the schedule is wasted".

Given that the benefits that simulated annealing brings are doubtful and there is no formal guide to its application, we propose to apply in GPSS a totally greedy algorithm to improve the cost of the current solution when possible, and to rely on a procedure to escape local minima only when one is reached. A variant of the breakout method (Morris 93) is one such technique we intend to experiment with.

Summary

In this research statement we put forth several enhancements to the current version of the Ground Processing Scheduling System (GPSS). Such enhancements will significantly improve its efficiency and usability by human schedulers. The enhancement is done by improving currently used algorithms and heuristics and by adding the extra feature of fencing. The fencing capability adds the flexibility of obtaining an optimal sub-schedule within a specified period of time.

References

M. Deale, M. Yvanovich, D. Schnitzius, D. Kautz, M. Carpenter, M. Zweben, G. Davis, and B. Daun, 1994. The Space Shuttle Ground Processing Scheduling System. *Intelligent Scheduling*, edited by M. Zweben and M. Fox, Morgan Kaufmann Publisher, pp. 423-449.

S. Kirkpatrick, C. Gelatt, and M. Vecchi, 1983. Optimization by Simulated Annealing. *Science*, vol. 220 #4598.

S. Minton, M. Johnston, A. Philips, and P. Laird, 1992. Minimizing conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, vol. 58, pp. 161-205.

P. Morris, 1993. The breakout method for escaping from local minima. *Proceedings of AAAI-93*, pp. 40-45.

P. Selman and H. Kautz, 1993. An Empirical Study of Greedy Local Search for Satisfiability Testing. *Proceedings of AAAI-93*, pp. 46-51.

D. Waltz, 1975. Understanding line drawings of scenes with shading. *The Psychology of Computer Vision*, edited by P. Winston, McGraw-Hill.

M. Zweben, E. Davis, and M. Deale, 1993. Iterative repair for scheduling and rescheduling. *IEEE Systems, Man, and Cybernetics*, Special issue on Planning, Scheduling, and Control.