Evaluation of a Mixed-Initiative Approach to Schedule Maintenance

Tim Oates and Paul R. Cohen
Experimental Knowledge Systems Laboratory
Computer Science Department, LGRC
University of Massachusetts
Box 34610
Amherst, MA 01003-4610
oates@cs.umass.edu, cohen@cs.umass.edu

1 Introduction

Plans and schedules formulated to run in the real world will often fail due to the complexity and unpredictability of the environment. An unexpected, high priority order may arrive at the job shop or a piece of equipment used to process orders may break down. Existing methods to deal with this problem include real time recovery from plan failures [1] [2] [7] and post-hoc plan repair based on failures observed while executing the plan [6]. Failure recovery mechanisms, such as replanning, can be expensive, and it may not be feasible to repair a plan by letting it repeatedly fail. An alternative strategy is to monitor the execution of the plan, attempting to predict pathological states that make it difficult or impossible to achieve goals [5]. Doing so admits the possibility of effecting plan modifications in real time to avoid pathological states.

Plan steering is a mixed-initiative approach to real time prediction and avoidance of plan failures [3]. A plan steering system comprises a pathology demon that monitors the execution environment to detect and predict pathological states, a plan steering agent that evaluates the demon's predictions and formulates plan modifications to avoid predicted pathologies, and a human user who monitors the environment, the demon, and the agent. The human and the agent work together to steer the plan away from potential problems by intervening before they develop. The benefits of keeping computers in the loop are clear. For large, complex plans, involving hundreds or thousands of events over time, determining whether events are unfolding according to plan and assessing the impact of dynamic plan modifications are impossible for humans.

As a first step toward plan steering, we built a demon and an agent for the related task of schedule maintenance in the transportation planning domain. The problems that we address in this domain are closely related to common problems in scheduling of manufacturing processes. Ships and cargo (orders) must flow through a series of ports (processing points) that have limited capacity in a specific order and in a timely manner. The primary difference is that the "schedule" for a ship orders the ports that must be visited but only specifies the time at which the ship should begin its journey. We experimentally assessed the performance of the system at its two primary tasks: predicting schedule pathologies and formulating schedule modifications to avoid those pathologies. We then assessed the performance of humans at the same task of schedule maintenance. We looked at performance in three conditions: the human acting alone, the agent acting alone, and the human and the agent working together.

2 The Schedule Maintenance System

The task for our system is management of schedules in a simulated shipping network called TransSim. A TransSim scenario consists of ships, ports, cargo, and simple movement require-

ments (SMRs) for each piece of cargo. An SMR specifies the route that a piece of cargo is to take through the network and when it is to begin its journey. The SMRs of a scenario constitute its schedule and largely determine the behavior of the simulation. Ports are limited resources and ships must queue for service when a port is being used to load or unload another ship's cargo. If many SMRs reference any one port then it is likely that a *bottleneck* will develop at that port. The schedule maintenance system attempts to maximize throughput by modifying SMRs while minimizing the number of changes to preserve as much of the structure imposed by the initial SMRs as possible. These two goals are often at odds with one another so an appropriate balance must be found.

The function of detecting and predicting pathologies (bottlenecks) is performed by a pathology demon that monitors the state of all ports in a scenario as it unfolds. The demon combines the current state of a port with information about ships that are en route to the port to project the port's state for each of several days into the future. Ship travel times are not deterministic and the demon's knowledge of its environment is imperfect, so there is an error component to its predictions. We experimentally evaluated the accuracy of the demon's predictions and the extent to which accuracy was affected by three environmental factors: the horizon into the future for which predictions are made, the amount of variance in the demon's ability to project ship arrival times, and a threshold that controlled how aggressive/conservative the demon is when determining that a given ship will be in port on a given day. We found significant effects of these factors on prediction performance and found that the demon's performance was good over a wide range of settings.

We have implemented a schedule maintenance agent that monitors the demon's predictions to identify potential bottlenecks. It applies a simple heuristic to convert predicted queue lengths for multiple future days into a boolean tag for each port: likely future bottleneck or unlikely future bottleneck. When a port is identified as a potential problem, the agent looks for an opportunity to modify the scenario's SMRs to avoid or alleviate the bottleneck. Currently, the only action the agent can take is to reroute cargo that is bound for the port in question. We ran several experiments to determine what effects the agent's rerouting decisions would have on throughput and how those effects changed with problem size and complexity. Each experiment involved recording various cost measures related to throughput for multiple simulations in which the demon is generating and following its own advice. We found that in a wide variety of conditions, the actions of the agent reduced most simulation costs. That effect scaled nicely with pathology intensity, problem size, and problem complexity. [1]

3 Bringing Humans into the Loop

Part of the motivation for plan steering is the belief that humans find it extremely difficult to perform tasks such as the one for which our agent was designed. Tracking hundreds of events over time and understanding primary and secondary effects of schedule modifications is not something that people do well. Therefore, we ran a series of experiments in which humans were asked to perform the same task at which the agent was previously evaluated [4]. We provided a set of graphical displays that gave the human user essentially the same information and rerouting capabilities available to the agent.

In one half of the trials the human works alone. In the other half the human has the aid of the schedule steering agent. We call these the *unassisted* and *assisted* conditions respectively and refer to this experimental factor as *trial type*. In the assisted condition the agent evaluates the state of the network and generates advice for the user. Advice identifies both a port that is thought to be

[1] The results described in this section are presented in detail in [3].

a potential bottleneck and a piece of cargo bound for that port, and suggests an alternative route. The human evaluates the agent's advice via the graphical interface and may decide to accept or reject the advice. In either case the human may implement a rerouting decision of his/her own construction.

We found that humans working with the help of the agent are able to obtain better performance than humans working alone. All cost measures were lower in the assisted condition compared to the unassisted condition, with many being significantly so. However, this improved performance comes at the expense of disrupting the scenario to a greater extent: on average, about six pieces of cargo were rerouted without assistance, compared to about twelve pieces rerouted with assistance. Since performance is better in the assisted condition, it is not the case that the agent's advice makes things worse and therefore more intervention is required. Apparently, the agent is bringing pathological states to the attention of the human user that they would otherwise have missed and that the human believes require attention. The agent is serving its intended purpose of helping the human track large numbers of events as they occur in a complex environment.

How does the human's performance in either condition compare to the agent's? The unassisted human performs significantly worse than the agent in all cost measures. However, the agent implements almost three times as many changes to the scenario. Neither seems to be striking a good balance between maximizing throughput and minimizing schedule disruption. The story is quite different in the assisted condition. The performance of the assisted human is indistinguishable from the agent's performance; none of the cost measures are significantly different. This result alone is interesting since the agent performs quite well. The difference is that the assisted human is able to achieve this feat with significantly fewer changes to the scenario: twelve reroutes for the assisted human compared to more than eighteen for the agent. Apparently our mixed-initiative approach to schedule maintenance in this particular domain is working. As noted before, the agent is probably flagging potential pathologies that the human would have otherwise missed. However, the human is selectively filtering the agent's suggestions to implement only those that seem most crucial and that are not wasteful.

With indications in hand as to the utility of our approach, we attempted to determine why performance in the assisted condition was so good. During an assisted trial, the user is constantly evaluating the state of the network and deciding whether or not to act. We focus on three specific decision points to assign credit for the assisted human's performance. They are (1) the agent offers advice and it is accepted, (2) the agent offers advice and it is rejected, and (3) the human makes a rerouting decision independent of the agent. Is good performance due to the intelligence of the agent? Is it due to the human's ability to differentiate between good and bad advice? Or is it due to the human's ability to formulate schedule modifications independently? The metric we have chosen for this credit assignment task is daily queue length summed over all ports. Every time during the course of a single simulation that the human makes one of the three decisions, we look at total queue length over a window of fixed size in the future to determine if the decision was good or bad. This is complicated by trend in queue length over the course of a simulation. We compensate for trend by computing an expected queue length curve and comparing actual performance to expectations.

We analyzed one of the experimental scenarios and found that accepting the agent's advice results in smaller than expected queue lengths, but the result is not significant. Rejecting the agent's advice led to significantly larger than expected queues. It appears that in this scenario, the agent's advice tends to stave off potential pathologies and ignoring its advice is detrimental. In terms of making beneficial schedule modifications, the human fares quite well. When compared to expectations, the results of the human's rerouting decisions are significantly better. With the tools that we provided, the human was able to evaluate the state of the transportation network, identify potential trouble spots, and formulate a preventative plan. Therefore, poor human performance in

the unassisted trials was not due to an inability to understand and manipulate the domain. [2]

4 Future Work

We presented a mixed-initiative system for schedule maintenance in a simulated shipping network. Simultaneously achieving the two goals of maximizing throughput and minimizing the number of changes to the initial schedule proved to be difficult for both the human and the agent. The human rerouted few pieces of cargo at the expense of high simulation costs. The agent's simulation costs were quite low but the number of pieces of cargo rerouted was high. In this domain, the optimal balance was struck by the agent and the human working together.

The goal of this research is to arrive at a generalizable architecture for plan steering. We want to be able to replace TransSim with the real world and have agents working with humans to avoid pathologies in plans and schedules. To that end, we will continue to push on this system by investigating pathologies other than bottlenecks, advice other than rerouting, and methods for increasing predictive accuracy. We then hope to study other problem domains to understand how they are different from transportation planning and how those differences impact the efficacy of our architecture.

References

[1] Ambros-Ingerson, Jose A. and Steel, Sam. Integrating planning, execution and monitoring. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 83-88, Minneapolis, Minnesota, 1988.

[2] Lopez-Mellado, Ernesto and Alami, Rachid. A failure recovery scheme for assembly workcells. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 702-707, 1990.

[3] Oates, T. and Cohen, P.R. Toward a plan steering agent: experiments with schedule maintenance. To appear in *Proceedings of the Second International Conference on AI Planning Systems*, 1994. Also Computer Science Department Technical Report 94-02, University of Massachusetts at Amherst.

[4] Oates, T. and Cohen, Paul R. Mixed-Initiative Schedule Maintenance: a First Step Toward Plan Steering. To appear in *Proceedings of the ARPA/Rome Lab Planning Initiative Workshop*, Tucson, AZ, Feb. 1994. Also Computer Science Department Technical Report 94-31, University of Massachusetts at Amherst.

[5] Sadeh, N. Micro-opportunistic scheduling: the Micro-Boss factory scheduler, to appear in *Intelligent Scheduling*, edited by M. Zweben and M. Fox, Morgan Kaufmann, 1994.

[6] Simmons, Reid G. A theory of debugging plans and interpretations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 94-99, Minneapolis, Minnesota.

[7] Wilkins, David E. Recovering from execution errors in SIPE. Technical Report 346, Artificial Intelligence Center, Computer Science and Technology Center, SRI International, 1985.

[2] The results described in this section are presented in detail in [4].