Toward a Plan Steering Agent: Experiments with Schedule Maintenance

Tim Oates and Paul R. Cohen

Computer Science Technical Report 94-02

Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetts Amherst, MA 01003

Abstract

When a plan involves hundreds or thousands of events over time it can be difficult or impossible to tell whether those events are unfolding "according to plan" and to assess the impact of dynamic plan modifications. Pathological states may arise in which goals cannot be attained or are attained too slowly. Plan steering is an agent-based approach to this problem. The agent monitors an unfolding plan, detects and predicts pathological situations, and develops dynamic plan modifications that will steer the plan around the problem. We present results for such a system that performs the related task of schedule maintenance in the transportation planning domain. The agent uses limited domain knowledge and simple heuristics and is able to improve throughput significantly.

[0]This research is supported by ARPA-AFOSR contract F30602-91-C-0076.

1 Introduction

The world in which we operate is tremendously complex and unpredictable. These two factors conspire to make management of a wide variety of domains difficult for humans. Thus, plans formulated by humans to achieve goals in the real world will often fail. When they do, they may be repaired by humans manually, by computer programs automatically, or by both working in concert. The benefits of keeping both humans and computers in the loop are obvious: each has a relatively disjoint set of talents that augment the other's. Artificial intelligence technology currently exists to repair plans automatically and to recover from plan failures automatically. Some technology exists to automate prediction and avoidance of failures while plans execute. [3]

Plan steering addresses the problem of predicting and avoiding, in real time, pathological states that make it difficult or impossible to achieve goals. That is accomplished with an autonomous plan steering agent that assists humans in monitoring a plan as it unfolds, detecting and predicting pathological situations and suggesting beneficial plan modifications in real time. The agent helps the human to steer the plan out of or around trouble. To explore issues related to plan steering we have constructed a system for the related task of schedule maintenance. We are not in this paper testing any specific hypothesis about plan steering, but are instead attempting to develop a baseline system for schedule maintenance. We want to develop an appropriate architecture and determine which features of the task and environment affect the efficacy of the system.

Section 2 of the paper introduces the schedule maintenance agent, presenting its task and architecture. Section 3 outlines the results of several experiments that analyze the performance of various components of the agent as well as performance of the agent as a whole at its intended task. Analysis of the agent includes exploration of several factors that may affect its performance and comparison to a very simple but effective default rule for managing the domain. Section 4 presents conclusions and directions for future work.

2 An Agent for Schedule Maintenance

We have constructed an agent to perform schedule maintenance in the transportation planning domain. Its goal is to move cargo quickly through a

network starting with an initial schedule. This task is more difficult than simple load balancing due to the structure imposed by the initial schedule; we want to maintain as much of that structure as possible.

2.1 The Schedule Maintenance Task

Our agent manages a transportation simulation called TransSim. A TransSim scenario consists of a collection of ports, ships, cargo, and routes. The routes, called "simple movement requirements" or SMRs, specify when each piece of cargo is to begin its journey and through which ports it is to travel. For example:

> Cargo-12 should set sail from port-1 on day 10. From there it should stop at port-5 and port-3 before sailing for its final destination, port-20.

Ports and ships are limited resources. There may not be a ship available on day 10 at port-1. There may be a ship available, but the port may be clogged and the ship can't dock until day 15. Constraints govern which kinds of cargo can go on different kinds of ships. The simulation is nondeterministic in that ship travel time may vary.

SMRs are fully specified at the start of each scenario. The specific ship on which a piece of cargo will travel is determined at run time, based on availability. There may be a free ship locally or one may be requested from a nearby port. Therefore, the behavior observed during a simulation is largely determined by the scenario's SMRs. If many SMRs travel through one port, that port is likely to become clogged. If SMR destinations are evenly distributed in time over all ports then problems are unlikely to arise. The key point is that there is really no plan in this domain. Cargo must travel specific routes starting at certain times, but how that happens is determined dynamically at run time. This is why we characterize the task as schedule maintenance, and why our agent is but a first stab at the more general plan steering problem.

2.2 System Architecture

The architecture of a generic plan steering system is shown in Figure 1. The architecture for schedule maintenance is identical, only the task changes. A pathology demon monitors the environment as a plan unfolds, detecting and predicting pathological situations. The plan steering agent monitors the

demon's output and formulates plan modifications that address the problems found by the demon. A human user evaluates the agent's advice and effects plan changes when appropriate.

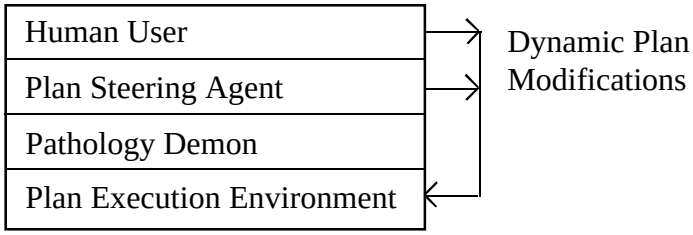

Figure 1: Plan Steering Architecture

A pathology demon has been implemented to monitor TransSim as cargo is shipped about. It predicts when and where in the network a *bottleneck* is likely to arise. Bottlenecks result when too many ships attempt to travel through a port, causing reduced throughput. The demon uses simple domain information to make its predictions. It looks at the current state of each port and ships that are traveling in channels toward the port and assigns a probability to each possible state of the port on each day out to a horizon. That is, the demon only uses information local to a given port. Resource-based schedule revision with local information was used successfully in [4].

The schedule maintenance agent combines the demon's predictions for multiple days in the future to determine which ports are likely to become substantial problems. The agent then uses simple heuristics to generate advice that, when implemented, will either alleviate or avoid the predicted bottleneck. This may be contrasted with reactive approaches, such as [2], that respond to unexpected events at the time that they occur. Currently, the only advice the agent offers is based on a simple rerouting heuristic. If a piece of cargo is being loaded onto a ship bound for a potential bottleneck, the agent changes the cargo's SMR so that it travels to the port closest to the original destination that is not problematic. This seems to be a reasonable approach in that rerouted cargo continues to make progress toward its destination. We assume that ports that are close geographically are "equivalent" in some sense.

3 Performance Assessment

We now focus on assessing the ability of our system to manage the TransSim domain. First, we will investigate performance of the pathology demon as influenced by environmental factors. Then, we will determine whether the agent's advice is helpful, harmful, or neither. We will also explore how much of the benefit is due to the intelligence built into the agent and how much is due to other factors. Finally, we want to ascertain to what extent environmental factors, such as pathology severity and problem complexity, affect the agent's performance. The general procedure is to run several simulations in each of a variety of experimental conditions, taking several dependent cost measures which are then compared to determine the effect of the condition.

3.1 Pathology Demon

The pathology prediction demon models each ship as a probability distribution of arrival times. Combining this distribution with the current state of each port, the demon arrives at a predicted docking queue length. The model is similar to that used in [1] for exploring the effects of resource allocation decisions. Long docking queues indicate the presence of a bottleneck. Several factors affect the accuracy of the demon's predictions. They are (1) the distance into the future for which predictions are made, (2) the certainty of the demon's knowledge about ship arrivals, and (3) a prediction threshold that controls how aggressive the demon is at making predictions (low values are aggressive, high values are conservative). We expect predictions to be less accurate when made far into the future or when there is a lot of variance in ship arrivals. There should be an optimal prediction threshold, at least for a given level of the other two factors.

The accuracy of the demon was explored by running a fully factorial experiment with 10 simulations for each of three levels of the factors (a total of 270 trials). The demon's predictions for each port as well as the actual state of each port were recorded and average squared prediction error was calculated for each trial. For predictions 2, 4, and 6 days into the future, average squared error was 0.137, 0.244, and 0.214; increasing but not monotonically. Though the effect of variance in the demon's knowledge about ship arrivals was not a significant main effect, it did interact significantly with prediction distance. High variance had an increasingly adverse effect on error as prediction distance was increased. Finally, plotting error at several prediction

threshold values resulted in a bowl shaped curve with 0.2 being the optimal threshold.

3.2 Measures of Cost

Having established some influences of demon accuracy, we turned next to the schedule maintenance agent and its performance. We defined several measures of cost associated with a single run of TransSim. These were used to evaluate the effect of moving from one experimental condition to another.

Bottleneck Predictions The sum over all days of the number of ports marked as potential trouble spots by the demon for a given day.

Cargo transit The sum over all pieces of cargo of the number of days from when the item first came on line until it reached its final destination.

Idle Cargo Cargo is considered idle if it is ready to be loaded onto a ship but none is available or if it is sitting in a ship queued for docking. This measure is the sum over all days of the number of pieces of idle cargo.

Queue Length Each port has a limited number of docks. Ships waiting to dock are placed in the docking queue. This measure is the sum over all days and all ports of the number of ships queued for docking.

Simulated days The number of simulated days required to ship all cargo to its final destination.

Ship utility Ships may travel empty when called from one port to another to service a piece of cargo. This measure is the sum over all simulated days of the number of ships traveling empty on a given day.

When the agent is not offering advice, we expect a positive correlation between Bottleneck Predictions and many of the other measures such as Queue Length. If the demon is predicting many bottlenecks, and the agent is not doing anything to avoid them, then if the demon is accurate our other cost measures will rise accordingly. The agent was designed to predict and avoid large docking queues so we expect its actions to reduce Queue Length. One extreme way to reduce Queue Length is to let there be only one ship traveling at any time. Both Cargo Transit and Simulated Days provide another view into the agent's performance on a global scale so we may ensure that it is not adopting a similarly pathological strategy.

3.3 Agent Advice vs No Advice

Several experiments were run to evaluate the agent's performance. They typically consisted of two conditions. In the first condition the agent monitors a running simulation but offers no advice. In the second condition the agent monitors a running simulation and implements any advice that it may have. The goal in each of these experiments is to determine how the agent's performance compares to doing nothing. We assess the impact of the agent's actions by performing an analysis of variance (ANOVA) of each cost measure on whether or not agent advice is used. If the result is significant then the actions of the agent affect the cost measure and inspection of cell means will indicate if it is a beneficial effect. A total of 10 trials (simulations) were run in each condition. Also, we allowed any type cargo to be carried by any type of ship. Experiments with more constraints are described in Section 3.5.

By varying the number of SMRs for a simulation we have some crude control over the frequency and duration of bottlenecks. Few SMRs results in low traffic intensity and few bottlenecks. Many SMRs has the opposite effect. Therefore, we ran an advice vs. no advice experiments at each of three levels of the number of SMRs in the scenario. The results for all three experiments are shown below.

Cost	p Value	No Advice Mean	Advice Mean	% Reduction
BP	0.0172	119.9	104.8	12.6
CT	0.0	1512.8	1365.2	9.8
IC	0.0	747.7	594.4	20.5
QL	0.0	581.7	433.3	25.5
SD	0.0333	125.1	116.8	6.6
SU	0.4820	47.7	56.6	-18.7

Table 1: Effects of Agent Advice - 25 SMRs

There are several interesting items in these tables. First, increasing the number of SMRs did in fact increase the number or severity of bottlenecks. This may be seen by noting that the mean queue length (an objective measure of bottleneck severity) in the "no advice" condition is positively correlated with the number of SMRs in the scenario. Next, the agent was beneficial regardless of pathology intensity. In fact, the percent reduction

Cost	p Value	No Advice Mean	Advice Mean	% Reduction
BP	0.0	216.4	175.8	18.8
CT	0.0	2170.5	1998.8	7.8
IC	0.0	1112.3	942.6	15.3
QL	0.0	793.9	640.6	19.3
SD	0.6961	135.9	134.5	1.0
SU	0.6535	126.9	130.9	-3.2

Table 2: Effects of Agent Advice - 30 SMRs

Cost	p Value	No Advice Mean	Advice Mean	% Reduction
BP	0.0	232	169.9	26.8
CT	0.0	2659.9	2261.8	15.0
IC	0.0	1542.9	1208.2	21.7
QL	0.0	911.1	598.5	34.3
SD	0.0325	168.2	149	11.4
SU	0.018	181.1	216.1	-19.3

Table 3: Effects of Agent Advice - 35 SMRs

in all cost measures was largest in the most pathological 35 SMR case. The agent achieved its design goal of reducing queue length. In doing so, it reduced the amount of time cargo spends sitting idle and actually increased the speed with which cargo travels to its destination locally (decreased Cargo Transit) and globally (decreased Simulated Days).

To investigate the effects of increasing the complexity of the task on the agent's ability to perform, a similar experiment was run with a larger scenario. The number of ports and ships were doubled (to 10 and 40 respectively) and the number of SMRs was set at 60. Again, 10 trials per condition were run. We obtain significant reductions in all cost measures except Ship Utility. Comparing percentage reductions with all three of the previous tables we see that increasing the complexity of the task increases the extent to which the agent is able to help.

The obvious conclusion is that in a wide variety of conditions, the agent is able to reduce the costs associated with a simulation. Neither pathology

Cost	p Value	No Advice Mean	Advice Mean	% Reduction
BP	0.0	283.5	204.6	27.8
CT	0.0	4284.6	3523.7	17.7
IC	0.0	2222.8	1577.6	29.0
QL	0.0	1406.7	817.4	41.9
SD	0.0	189.2	157.5	16.8
SU	0.5178	320.35	330.6	-3.2

Table 4: Effects of Agent Advice - Large Scenario

intensity nor problem complexity seem to nullify its ability to perform. In fact, the agent shines most brightly in precisely those situations where it is needed most, highly pathological and large/complex scenarios.

3.4 Control Condition - Random Advice

An experiment was run to determine the effects of demon accuracy on agent performance. The factors that affect demon accuracy (see Section 3.1) were varied with the surprising result that there was no significant impact on the efficacy of the agent. If the agent performs equally well with good and bad predictions of bottlenecks, then perhaps its ability to reduce costs is due to shuffling of routes, not to its ability to predict. Random rerouting, periodically picking a piece of cargo and sending it on a newly chosen random route, may be as good as the agent. Random rerouting has the advantage of tending to evenly distribute cargo over the network, minimizing contention for any one port. The disadvantage is that it destroys the structure inherent in the initial schedule.

To investigate the utility of random advice, we ran an experiment in which the agent, with varying frequency, rerouted a randomly selected piece of cargo. This was done with no regard for bottleneck predictions. We varied the probability of performing a reroute on each day over four values: 5%, 15%, 25%, 35%. As with the advice vs. no advice experiments, the number of SMRs in the simulation was varied to get a feel for how these effects changed with pathology intensity. ANOVA was used to identify significant effects. The tables below show the mean number of reroutes performed at each level of advice. In addition, the means of the cost measures that showed a significant main effect of the level of random advice are given.

When a Scheffe' test[1] determines that one of the means at a random level is significantly different from the mean at the agent advice level, an '*' is used to mark the mean at the random level.

Level	Reroutes	CT	IC	QL
Real Advice	7.5	1365.2	594.4	433.3
Random 5	4	1466.87	669.12	493.12
Random 15	9.125	1290.87	578.75	430.5
Random 25	15.25 *	1203.5 *	503.62	372
Random 35	18 *	1191.37 *	504.12	375

Table 5: Real vs. Random Advice - 25 SMRs

Level	Reroutes	CT	IC	QL
Real Advice	12	1998.8	942.6	640.6
Random 5	5.7 *	2080	1040 *	726.8 *
Random 15	13.8	2038	1050.8 *	771.1 *
Random 25	20.1 *	1950.8	956.6	688.5
Random 35	27.5 *	1855.1 *	879.5	638.3

Table 6: Real vs. Random Advice - 30 SMRs

Level	Reroutes	CT	IC	QL
Real Advice	21.3	2261.8	1208.2	598.5
Random 5	7.4 *	2449.2 *	1385.6 *	764.9 *
Random 15	17.5	2292	1229.3	647.3
Random 25	27.5 *	2099.1 *	1084.6	571.8
Random 35	37.1 *	2141.5	1119.6	590.5

Table 7: Real vs. Random Advice - 35 SMRs

As expected, cost measures decrease with increasing frequency of random

[1] A Scheffe' test is a t-test for multiple pairwise comparisons that preserves a specified experimentwise error.

rerouting. For the 25 SMR case the decrease tends to flatten out with the two highest levels of randomness being nearly equivalent. For the 30 SMR case the decrease continued monotonically whereas for the 35 SMR case the cost measures actually spiked back up at the highest level of randomness. It appears that for highly pathological scenarios, there is a limit beyond which random shuffling hurts more than it helps.

It is apparent that in the 25 SMR case any amount of randomness is indistinguishable from the agent's advice when performance is measured in terms of Idle Cargo and Queue Length, and virtually identical when measured by Cargo Transit. The situation is somewhat better in the 30 SMR case. With an average of 12 reroutes, the agent is able to equal the performance of randomly rerouting 20 times in the Idle Cargo and Queue Length columns. (Note that a Scheffe' test indicates that the amount of real advice given is significantly lower than the amount given in the random 25 case.) This is an important point. Remember that there is no "plan" is this domain, only the structure imposed by the initial SMRs. Improving performance with the minimal number of rerouting decisions is key to maintaining that structure. The results in the 35 SMR case are somewhat inconclusive. The agent performed better than the random 5 level but only matched performance of the other random levels. The two highest levels of randomness (which had significantly higher mean numbers of reroutes) were statistically equivalent to the agent in terms of simulation costs. Again we see that the demon is able to achieve good results with a few well directed rerouting decisions rather than with large numbers of random ones.

The same large scenario described previously was used to investigate the effects of problem size and complexity on the efficacy of random advice. The results here are striking. Increasing the amount of random advice seems to help monotonically. The amount of rerouting performed by the agent was statistically equivalent to the random 25 condition only. In that condition, Scheffe' tests show that the agent's performance is significantly better than random advice; its domain knowledge is paying great rewards. Looking at the data another way, the agent performed equally as well as the random 35 condition with 34% fewer reroutes.

Our initial proposition that random rerouting would help to lower the various cost measures was borne out. In fact, it seems that more random rerouting is better than less, except perhaps in highly bottlenecked scenarios. There existed some level of randomness that equaled the performance of the agent for each of the previous experiments. However, the agent typically rerouted many fewer pieces of cargo than the equivalent level of randomness,

Level	Reroutes	CT	IC	QL
Real Advice	28.25	3543.35	1606.05	826.85
Random 5	8.2 *	4057.05 *	2028.8 *	1224.3 *
Random 15	19.15 *	3851.6 *	1884.5 *	1134.7 *
Random 25	29.2	3696 *	1753.8 *	1042.3 *
Random 35	42.85 *	3454.85	1523.6	852.15

Table 8: Real vs. Random Advice - Large Scenario

thereby preserving more of the structure of the simulation. Finally, it appears that for large/complex scenarios the difference between randomness and the agent is more pronounced.

3.5 Highly Constrained Scenarios

To increase the realism of the agent's task, several constraints were added to the scenarios. There are three types of cargo: CONT, GENL, and RORO. Rather than using a single cargo type, we used multiple types and limited the cargo handling capabilities of both ships and docks. Now for cargo to flow through the network, it must match in type with any ship that is to carry it and any dock where it will be handled. We ran agent advice vs. random advice experiments under these conditions after modifying the agent to consider the additional constraints. Whenever random advice generated an incompatible routing assignment, it was penalized with a short delay for the offending piece of cargo. The results are presented below.

Level	Reroutes	CT	IC	QL
Real Advice	9.6	1833.0	990.9	488.5
Random 5	4.6 *	2010.2 *	1135.6 *	617.4 *
Random 15	16.6 *	1770.4	961.3	534.5
Random 25	22.4 *	1697.0 *	877.9	443.0
Random 35	31.8 *	1651.0 *	860.7 *	452.8

Table 9: Real vs. Random Advice - 30 SMR Constrained Scenario

Again we see that the agent is able to match the performance of random

rerouting with many fewer changes to the schedule. The fact that Queue Length for the agent is different only from the Random 5 condition and that Cargo Transit for the agent is different only from the highest and lowest levels of random advice points to a result of constraining the scenario: the performance of random advice in any one condition is highly variable. The variance associated with Cargo Transit for random advice was on average 3.5 times higher than for agent advice. Likewise, the variance associated with Idle Cargo was 4.1 times higher and the variance associated with Queue Length was 3.2 times higher. The agent is able to achieve good average case performance with much higher consistency when compared to random rerouting by making a few appropriate rerouting decisions.

4 Conclusions and Future Work

We have seen that an intelligent agent may be constructed that takes advantage of minimal domain knowledge and that uses simple heuristics to manage a complex problem domain. Such an agent was constructed for schedule maintenance in a simulated transportation network and its performance was evaluated. The presence of the agent significantly reduced cost measures when compared to doing nothing. In addition, the benefit of the agent grew with problem size and complexity. A default rule for managing the domain with random rerouting was explored and shown to have substantial utility in reducing cost. Unfortunately, random rerouting tended to destroy the structure of the scenarios. At a given level of simulation cost, the agent used fewer rerouting actions than the default rule. As problem size and complexity grew this difference became more pronounced.

In the near term we want to improve the utility of the agent by investigating pathologies other than bottlenecks, advice other than rerouting, and non-local effects for pathology prediction. The next step with the system as a whole is to run experiments with human users in the loop. Can a human acting with the help of the agent manage TransSim better than either the human or the agent acting alone? We hope to arrive at a generalizable plan steering architecture that will allow us to replace TransSim with the real world and to achieve similar results.

Acknowledgements

This research is supported by ARPA under contract F30602-91-C-0076. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copright notation hereon.

References

[1] N. Muscetolla and S.F. Smith, "A Probabilistic Framework for Resource-Constrained Multi-Agent Planning", in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 1063-1066, Morgan Kaufmann, 1987.

[2] P.S. Ow, S.F. Smith, and A. Thiriez, "Reactive Plan Revision", in *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 77-82, Morgan Kaufmann, 1988.

[3] N. Sadeh, "Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler", to appear in *Intelligent Scheduling*, edited by M. Zweben and M. Fox, Morgan Kaufmann, 1993.

[4] S.F. Smith, P.S. Ow, N. Muscetolla, J. Potvin, and D.C. Matthys, "An Integrated Framework for Generating and Revising Factory Schedules", *Journal of the Operational Research Society*, Vol. 41. No. 6, 1990.