Controlling Anytime Scheduling of Observation Tasks

J W Baxter, J Hargreaves, N Hawes and R Stolkin

Abstract This paper describes how multiple independent observation tasks can be scheduled for an autonomous vehicle. Presented with large numbers of tasks, of differing reward levels, a vehicle has to evaluate the best schedule to execute given a limited time to both plan and act. A meta-management framework acting on top of an anytime scheduler analyses the problem and the progress made in generating solutions to identify when to stop planning and start executing. We compare a probabilistic management technique with active monitoring of the current execution reward and conclude that in this case detecting a local maxima in the predicted reward is the most effective policy.

1 Introduction

Anytime algorithms provide a useful technique for trading off planning time against execution. In situations where fast but well-supported decisions are needed they provide a way of providing bounded optimality [5]. This requires a stopping condition for the algorithm which provides the best response given the time and computation limits faced by an agent. Our problem domain consists of planning surveillance tours where a vehicle has to plan a tour of a number of locations making observations at each one. The observation tasks have a predicted duration and reward value and the aim is to maximise the expected reward from a tour. This is therefore a prize-gathering orienteering problem. These sorts of problems arise in many different situations such as unmanned aerial vehicle mission planning, robotic security patrols and the gathering of scientific data remotely. We have applied an anytime

J W Baxter
University of Birmingham, Poynting Institute and QinetiQ Ltd. BT15 2TT, e-mail: j.baxter@poynting.bham.ac.uk

J Hargreaves, N Hawes, R Stolkin
University of Birmingham, BT15 2TT e-mail: {jxh576, n.a.hawes , r.stolkin} @cs.bham.ac.uk

algorithm based upon weighted A* search [7] which can plan observation schedules in an anytime fashion and can provide the optimal solution if given sufficient time and memory. In initial trial work in a UAV mission planning domain we applied a simple fixed run time limit for this anytime scheduler. This paper compares the performance of different meta-management techniques for deciding when to stop planning and start executing.

2 Related Work

Team orienteering problems and the related prize gathering travelling salesman problem have been covered in the operations research literature [1, 2] where exact solutions for problems with up to 102 tasks have been found using integer programming. Some problems could be solved very rapidly but others took minutes or hours of CPU time, especially as the length of tours increased (roughly equivalent to longer total execution time in our problem formulation). Shi et al. [6] propose an ant colony optimisation method for solving these problems but don't give data on the computational cost. We are not aware of other work describing an anytime planning approach to these problems. Anytime planners have a long history and anytime heuristic search is analysed in depth in [3]. More recent work [7] has looked in detail at the effect of using different heuristic weights on the speed with which bounded optimal solutions can be found and the effect of combining admissible and non-admissible heuristics. Zilberstein and Hansen [4] propose a range of different, and increasingly complex, meta-management schemes for anytime algorithms. They focus on a specific problem type (the 20 city travelling salesman problem) without considering if this data generalises to different sizes of problem.

3 Meta-Management

In our UAV planning domain we compared two different techniques for stopping an anytime algorithm in order to get the best result: the 'myopic' stopping criteria proposed in [4] which uses statistics gathered over previous problem instances, and a 'loss limiting' policy which executes the plan as soon as the benefit of executing the current plan falls below a set threshold of the best (predicted) execution reward seen so far.

Developing the meta-management system consist of two steps, gathering data to characterise generic performance and analysing a specific problem instance to predict how it is likely to perform. Unlike the approach in [4] we assert that the cost of time is not independent of the problem instance. In our case we represent time as a lost chance to gain rewards. Therefore we need to be able to estimate from a problem what the expected reward is and how much reward a delay is likely to cost.

3.1 Characterising performance

To characterise the performance of the scheduler we used a random problem generator and ran a series of 100 test runs recording the reward of the best plan found against time. The reward level was compared against the best result found after 60s of runtime (on a 2.3Ghz Intel 2 Core Duo with 2Gb of RAM) or the optimal if the algorithm terminated before that time having proved its solution was optimal. In order to better represent real world problems we used both fully random positioning of tasks and a clustering generator where tasks were more likely to be close to other tasks than spread uniformly through the space. The algorithm is highly effective at finding high quality solutions very quickly in single vehicle cases and the curve showing the average plan quality (as a proportion of the best found in 60s) can be seen in Figure 1. This data is used to predict future improvements for the 'myopic' scheme and to predict how close a solution is likely to be to the optimal. The reward lost by executing a plan later is estimated by calculating a linear expected reward per unit time from the estimated optimal reward divided by the time allowed for execution.

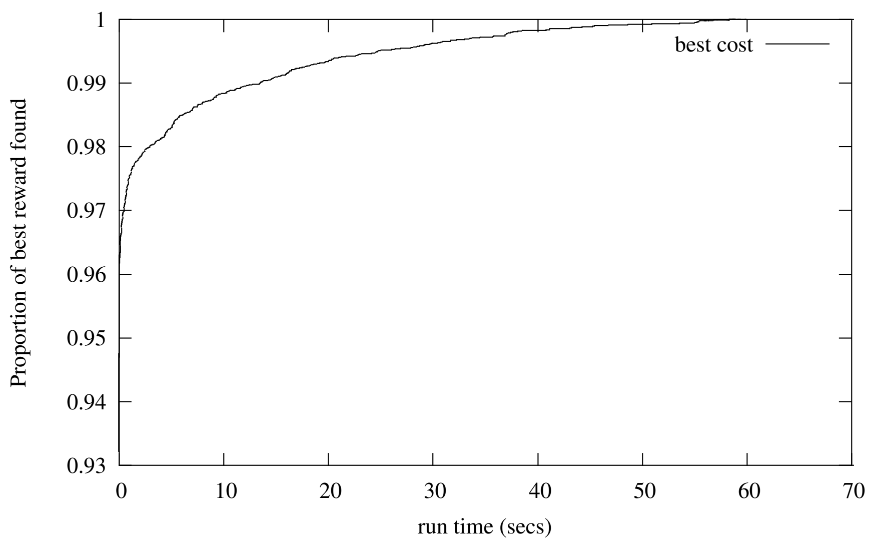

Fig. 1 Reward level of best plan found against time (averaged over 100 runs)

3.2 Online decision making

The 'myopic' management policy uses the (estimated) current quality level of a solution and a probability table derived by sampling 100 test runs to predict if the expected gain in reward through planning will exceed the reward lost by delaying execution. The 'loss limiting' policy attempts to execute just after the estimated execution reward has peaked (at the maxima of the curves shown in Figure 2). It does this by recording the maximum execution reward found so far and executing if the current estimated reward ever falls below 99.5% of this maximum. The tolerance allows for short periods in which no improvement is found.

4 Results

To analyse the effect of the meta-management layer we examined the effect of *time pressure*, i.e. how the system responds to changes in the balance between the speed of computation and the speed of execution. To do this we vary the relative rates of computation and execution so that the meta-management layer has to adjust the execution decision point. Figure 2 shows the result of scaling the times in the problem by rates between 1 and 0.1. This is equivalent to either reducing the computational speed or reducing the travel time, task durations and total execution time by the rate factor. To keep the same time axis, the graph uses the scaled task times, giving the same computation time but with the task times and distances between them reduced. The actual execution reward is calculated by assuming the output schedule is run but only those tasks which can be fully completed provide a reward. As more time is spent running the scheduler it is more likely that one or more tasks at the end of the schedule will not be completed reducing the actual reward. The graph shows the proportion of the maximal reward obtainable after 60 seconds of planning (at the base rate) if execution is started at the time on the x axis (averaged over 100 test instances). As can be seen at the base level of time pressure planning can continue for 20 seconds before the benefit of improved plans is outweighed by the lost rewards due to the delay. As the time pressure increases, the total reward peaks much earlier and decreases swiftly, showing that the benefit of additional planning is rapidly lost.

To identify the performance of the meta-management layer, Table 1 shows the average result of 100 runs using different policies to decide when to stop planning and start executing. The values are the proportion of the reward of the best possible plan found after 60 seconds when stopping planning according to the policy and starting to execute. The first three policies use a fixed planning time with no meta-management. The final policy, hindsight, shows the maximum possible result which could be obtained given the scheduler's performance by looking back over every decision point of a 60 second run for each problem to find the maximum.

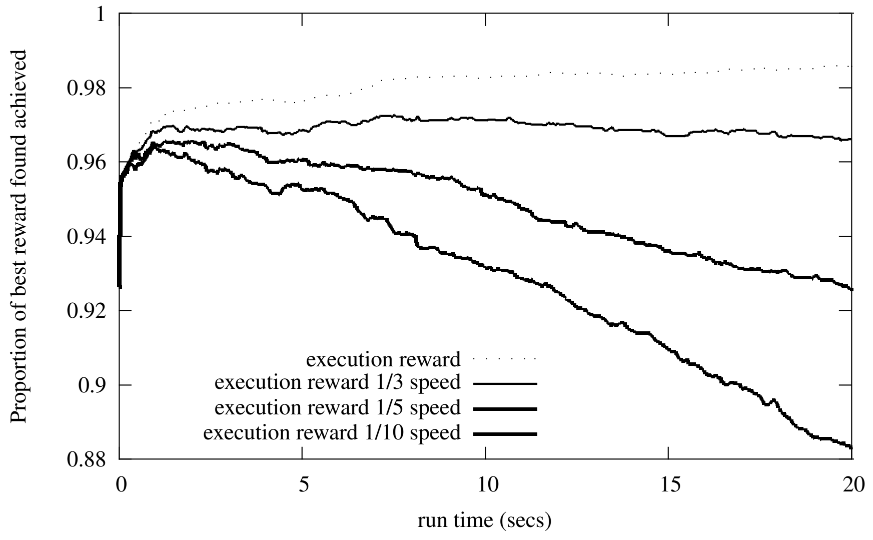

Fig. 2 Actual reward achieved at execution for increasing levels of time pressure (averaged over 100 runs)

Table 1 The resulting reward as a proportion of the maximum possible averaged over 100 runs for each policy with varying processing speeds

	Rate			
Policy	1	0.33	0.2	0.1
fixed 1s	96.7679	96.3957	**96.2939**	95.5168
fixed 10s	**97.0843**	95.1745	93.1818	88.1618
fixed 20s	96.5668	92.5464	88.1677	77.0558
myopic	96.6338	96.2636	95.987	95.5899
loss limiting	96.9283	**96.5873**	96.2759	**95.7427**
hindsight	98.8795	97.8987	97.6254	97.2579

5 Conclusions

The anytime scheduler we are using is highly efficient for these problems, getting on average 93% of the reward found after a minute within the first step (which typically takes 33ms) and rapidly improving the solution.

The best result over all policies is highlighted in bold. This shows that the fixed time policies produce good results if they are matched to the problem but that they cannot match the performance of the managed policies across the different levels of time pressure. The ‘loss limiting’ policy outperforms the ‘myopic’ forward looking policy and is the best or second best in all cases. The ‘myopic’ forward looking strategy is limited because its performance is dependent on the quality of the improvement probability table as well as the estimates of the maximal reward and cost of time. As its future value prediction is based on averages over large numbers of runs it tends to perform badly in cases where the current problem is non typical and big improvements are made much earlier or later than the average. The 'loss limiting' policy is more robust to early gains (which skew the estimates of the optimal reward) but still terminates early in cases where the scheduler finds large improvements relatively late. The 99.5% loss limiting parameter was found by trial and error, improved versions could make use of information about the variability of previous problems to set this more appropriately.

6 Further Work

We hope to extend this work to looking at problems with multiple vehicles and time windows on tasks. The cost of time could be better estimated by examining the slack in the schedule. A schedule with a few seconds of 'spare' time at the end has effectively zero cost of time and this could be used to continue planning. In the longer term we are interested in applying the meta-management to more complex problems, and planning and execution strategies. In particular we would like to analyse the ability of meta-management to cope with more dynamic environments where tasks and travel times may change and multiple decisions on planning and re-planning are necessary.

References

1. de Arago, M.P., Viana, H., Uchoa, E.: The team orienteering problem: Formulations and branch-cut and price. In: T. Erlebach, M.E. Lbbecke (eds.) ATMOS, *OASICS*, vol. 14, pp. 142–155. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2010). URL http://dblp.uni-trier.de/db/conf/atmos/atmos2010.html
2. Boussier, S., Feillet, D., Gendreau, M.: An exact algorithm for team orienteering problems. 4OR: A Quarterly Journal of Operations Research **5**, 211–230 (2007). URL http://dx.doi.org/10.1007/s10288-006-0009-1. 10.1007/s10288-006-0009-1
3. Hansen, E.A., Zhou, R.: Anytime heuristic search. Journal of Artificial Intelligence Research (JAIR **28**, 267–297 (2007)
4. Hansen, E.A., Zilberstein, S.: Monitoring and control of anytime algorithms: A dynamic programming approach. Artificial Intelligence **126**, 139–157 (2001)
5. Russell, S., Wefald, E.: Do the right thing. In: Studies in Limited Rationality. MIT Press (1991)
6. Shi, X., Wang, L., Zhou, Y., Liang, Y.: An ant colony optimization method for prize-collecting traveling salesman problem with time windows. In: Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 07, ICNC '08, pp. 480–484. IEEE Computer Society, Washington, DC, USA (2008). DOI 10.1109/ICNC.2008.470. URL http://dx.doi.org/10.1109/ICNC.2008.470
7. Thayer, J.T., Ruml, W.: Faster than weighted a*: An optimistic approach to bounded suboptimal search. In: proceedings of the international conference on planning and scheduling (2008)