Baselines for On-Line Search Methods

Joseph C. Pemberton
pemberto@dc.isx.com
ISX Corporation
2000 North 15th Street, Suite 1000
Arlington, VA 22201

Abstract

On-line search algorithms have become increasingly important in a wide variety of application domains. Our position is that research in on-line search algorithms should at a minimum include experimental results that compare the algorithm's performance to a reasonable set of baselines. We suggest a general approach to baselines, and then present examples for a particular domain of on-line search problems.

Introduction

On-line search is often necessary when solving real-world decision problems, particularly when there is a measurable time, solution quality or opportunity cost associated with the deliberation needed to find a solution. The general conjecture that underlies on-line search algorithm research is that algorithms that are *aware* of the on-line conditions can exploit them to produce better quality decisions. In this paper, we suggest several baseline algorithms that can be used to help identify whether or not a particular on-line decision problem can benefit from a customized on-line search algorithm.

For this paper, we have focussed on on-line decision problems where the task is to find a sequence of decisions that is optimal for some objective function (*e.g.*, minimum cost, maximum solution quality, etc.). The complete solution is thus a sequence of decisions that transform the initial state into the goal state. In addition, we assume any partial solution can be extended to a complete solution (*i.e.*, there are no dead ends in the search space), and that there exists some function for estimating the value of a complete decision sequence given a partial decision sequence.

One example of this type of on-line decision problem is on-line scheduling optimization. Scheduling problems consist of a set of tasks and a set of resources with which to perform the tasks, and the objective is to determine the order in which to process the tasks on the given resources while minimizing the solution cost. Scheduling problems become difficult when there are not enough resources to perform all of the tasks at once, and when there are ordering constraints on the tasks (*e.g.*, a part must be milled before it is sanded, a painted part must be dry before it can be installed). There are many different variations on the general scheduling problem (*e.g.*, flowshop scheduling, job-shop scheduling, transportation scheduling). For our purposes, these scheduling optimization problems become on-line decision problems when the cost of an executed schedule includes the cost of the time spent deciding what order to perform which tasks.

General baseline recommendations

Any on-line search algorithm will ultimately have to trade-off the cost of finding a solution with the cost of executing the solution found. The first step toward providing a baseline for evaluating an on-line search algorithm is to consider the on-line cost of finding an optimal complete decision sequence using the best known algorithm. We call this approach the solve-then-execute algorithm. For a scheduling problem, this consists of solving for an optimal schedule and then executing it. When computation is inexpensive (*i.e.*, free) relative to the cost of executing the solution found, solve-then-execute is a reasonable competitor. For the more interesting case where computation is not free, the solve-then execute solution cost provides a baseline with which to measure the potential value of an on-line decision algorithm.

The second step toward a general purpose baseline is to consider very simple (*i.e.*, low-cost) heuristics for making a sequence of greedy on-line decisions. For the scheduling problem, one example is to schedule the tasks in increasing order of their processing time (subject to the ordering constraints). It makes sense to spend some time considering possible greedy-decision heuristics because they can also be used to direct the problem-space exploration in more complicated on-line search algorithms. Other obvious greedy heuristics to

consider include longest-first, most-constrained-first, and least-constrained-first.

The solve-then-execute and simple greedy algorithms together define a spectrum of on-line search algorithm performance. In particular, if the cost of the computation to find the decisions is counted separately from the cost of executing the solution found, then it is possible to parameterize the complete solution cost (deliberation and execution) as a function of the ratio of deliberation to execution cost. When the cost of deliberation is small, solve-then-execute is the algorithm to beat. Alternatively, when the cost of deliberation very expensive, an on-line algorithm that makes greedy decisions is likely to be the algorithm to beat. The more interesting cases from an on-line algorithm design perspective are in between these two extremes.

Iterative-Deepening Minimin

The main characteristics of on-line decision-making problems are that there is a limit on the amount of time available before a decision must be made, and that the opportunity exists to interleave planning and execution. In general, on-line decision making can be broken down into three sub-problems. The first is deciding what information to gather (or generate) to support the decision process given the limited time available. The second is deciding when to stop collecting information (or evaluating the effect of future decisions) and commit to a decision. The third is determining what action to execute given the information that has been gathered (or generated and evaluated) to support the decision. In this section, we briefly discuss baselines for each of these three subproblems. The combination of the three subproblem baselines is iterative-deepening minimin (Pemberton 1995b).

The choice of which information gathering method to employ normally boils down to the following question: Is a sophisticated information gathering method that is more particular about which information to gather better than a simple information gathering method that can explore a larger percentage of the search space due to its smaller overhead cost? The key observation here is that the time spent figuring out where to search will naturally take away from the amount of searching that can be done. The possibilities for information gathering range from brute force search, to simple heuristic exploration, to full-blown value-of-information calculations. Given a limited amount of time with which to explore the problem space, a reasonable baseline for information gathering is iterative-deepening branch-and-bound, where the bound is based on a simple (*i.e.*, low cost) admissible heuristic evaluation of the complete decision sequence cost given a partial decision sequence. This baseline provides the decision-making component of the algorithm with a reasonable set of information given the time constraint.

Perhaps the simplest way to decide when to make a decision is to delay the decision until the time for the decision is at hand. The advantage of delaying a decision is that it makes it possible for additional information to change the current decision. The disadvantage is that committing to a decision at the last minute may waste exploration resources that otherwise could have been saved for later decisions. In many on-line decision problems, the natural decision point is determined by the problem itself (*e.g.*, when a resource becomes available in a scheduling problem). For single-agent on-line search problems, we recommend that the decision be delayed as long as additional cost is not being incurred. We refer to this as last-minute decision making.

Given a partially explored problem space, we recommend that the baseline method for action choice be the minimin decision method (Korf 1990). A minimin decision is to move one step toward the lowest cost (or best) frontier node of the currently explored search space. This simple decision-making method has been shown to be very effective in both randomly generated search trees and randomly generated flow-shop scheduling problems (Pemberton 1995b). The added advantage of the minimin decision method is that the minimin decision is the same for an admissible branch-and-bound exploration and a full-width fixed-depth exploration.

These three baselines are the basis of iterative-deepening minimin. Prior work (Pemberton 1995b) has shown that this on-line search algorithm is competitive and thus a reasonable first comparison point for more complicated search algorithms.

Summary

We have presented three baseline algorithms to consider when evaluating on-line search algorithms for optimization decision problems: search-then-execute, greedy, and iterative-deepening minimin. These algorithms provide an initial baseline against which to measure more sophisticated on-line search methods.

The baseline recommendations presented here do not apply to on-line search problems for which there does not exist a function for evaluating intermediate nodes in the search space (*e.g.*, robot navigation in unknown space, scheduling with hard deadlines). For a scheduling problem where the complete decision sequence is constrained to finish before a preset deadline, a partial decision sequence cannot be evaluated without first de-

termining if there is a feasible completion. For this and other on-line decision problems we recommend that other baseline algorithms be developed and adopted in order to provide an experimental comparison point for newly developed on-line search methods

References

Boddy, M., and Dean, T. L. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence* 67:245–285.

Dean, T. 1991. Planning under uncertainty and time pressure. In *Proceedings of the 1990 Workshop on Innovative Approaches to Planning, Scheduling and Control*, 390–395.

Goodwin, R. 1994. Reasoning about when to start acting. In Hammond, K., ed., *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, 86–91.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2–3):189–211.

Pemberton, J. C. 1995a. k-best: A new method for real-time decision making. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Pemberton, J. C. 1995b. *Incremental Search Methods for Real-Time Decision Making.* Ph.D. Dissertation, University of California, Los Angeles.