Scaling up Planning Systems using Parallel Hardware and Machine Learning

Diane J. Cook

University of Texas at Arlington
cook@centauri.uta.edu

Abstract

The objective of this project is to improve the applicability of artificial intelligence planning systems to large-scale manufacturing tasks. We present two approaches to improving planning systems: the first component develops parallel algorithms for improving the heuristic search underlying all planning systems, and the second component uses machine learning methods to improve the accuracy of generated plans. The success of this research can provide means for manufacturing applications to make effective use of AI planning systems.

One application of artificial intelligence techniques that captivates researchers is the development of automated planning techniques. A semi-automated or fully-automated agent can be used to perform tasks that are too tedious, hazardous, or sometimes complex for humans. Although a large variety of tasks can be assumed by automated mechanisms, several factors currently prevent effective use of planning systems. The first problem is the time required to compute plans. Even the problem of computing a plan to move a multi-jointed robot arm from one position to another can be computation-intensive, yet manufacturing tasks add several degrees of complexity to this problem. Second, the planning operators for a majority of manufacturing plans are supplied by human errors and are thus prone to problems with incompleteness and inability to adapt to change.

We are currently researching methods of improving AI planning systems by developing parallel search techniques to speed up the task and by using machine learning to automatically refine operator definitions. Parallel processing can considerably reduce time spent in search, and thereby speedup many AI techniques including planning. We parallelize IDA* search because this is an admissible search algorithm that requires only linear memory. IDA* consists of a series of depth-first searches, in which each pass through the space searches up to a specified cost limit. If a solution is not found on one iteration, the cost limit is increased for the next iteration.

Our parallelization of IDA* uses a MIMD architecture and employs both parallel window search (PWS) and distributed tree search (DTS). Using PWS, processors are given copies of the initial state and search the space to different limits simultaneously. Processors stop their search if the first discovered solution is acceptable, or wait for processors with lower thresholds to finish if an optimal solution is desired. PWS reduces serial search time because redundant search is performed in parallel, and PWS can find a close-to-optimal solution quickly because some processors will be considering alternatives beyond the optimal depth.

Using DTS, one processor expands the search tree until there are enough distinct nodes to distribute. Once a sufficient number of nodes have been expanded, the first processor passes a unique node from the search queue to each remaining processor. Each processor is thus responsible for the entire subtree rooted at the node it received. The processors perform IDA* on their unique subtrees simultaneously. All processors search to the same threshold. After all processors have finished a single iteration, they begin a new search pass through the same set of subtrees using a larger threshold. DTS offers distinct benefits from PWS because no processor is searching useless

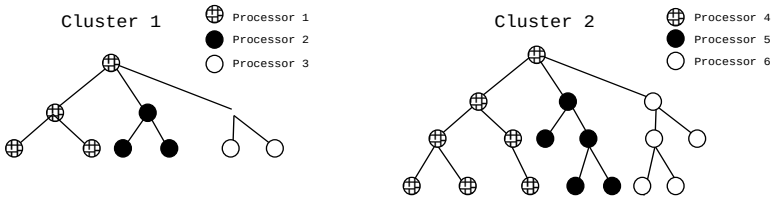

Figure 1: Space searched by two clusters, each with three processors

space beyond the optimal depth. On the other hand, if the tree is nonuniform, DTS will suffer from an abundance of idle processors.

Our system combines PWS and DTS by forming clusters of processors. Each cluster is assigned a unique cost limit (PWS), and work assigned to a single cluster is divided among the processors within the cluster (DTS). Figure 1 illustrates the process for six processors divided into two clusters. By varying the cluster size, the search can tend more toward parallel window search (#clusters = #processors) or distributed tree search (#clusters = 1).

This hybrid search technique is further improved by adding operator ordering and load balancing. Operator ordering gathers information from one iteration through the space to determine how to direct the search on the next iteration. Load balancing allows neighboring processors to share work when any of the processors become idle. Preliminary results have been gathered in two search domains using 64 processors on a nCUBE. These results indicate an average 34.6 speedup when searching for optimal solutions. We are currently developing methods for determining the optimal cluster size and integrating the parallel search into planning systems.

While parallel search can speed up planning systems, these systems are still sensitive to the accuracy of the supplied operator definitions. We are currently making use of machine learning techniques to learn and refine these definitions through experience. Typically, plans are generated by searching through a series of operator descriptions to find a sequence of operators that leads from the initial state of the problem to the goal state. Because operators are generated by humans, they are often subject to human error. One example of a common error is a missing precondition. A human expert may fail to note all the conditions which must truly exist for an operator to be successfully applied. Another source of error results from the dynamic nature of planning problems. Because planning domains change, tools change, processes change, and capabilities of agents change over time, a static set of operator rules which generated successful plans at one time may generate noisy plans later on.

Our approach uses machine learning techniques to automatically refine operator definitions. We allow successful plans to act as positive examples of how the operator should be used, and failed plans to act as negative examples of the operator application. From the example cases, errors in the operator can be pinpointed such as missing or erroneous preconditions, and missing or erroneous effects. Once this procedure is refined, it can be used to refine operator databases and to automatically adapt operator descriptions to a changing environment. Induction learning systems such as C4.5 can be further used to learn new features that best describe the environment in which an operator can be successfully applied. We are currently applying these techniques to the operator base employed by Texas Instruments for construction of rapid prototyping plans.

To date, few industrial applications of AI planning systems are in production. The limited use is due both to the computational complexity of the planning task and the inability of the systems to perform in the presence of incomplete or incorrect information. The improved performance obtained by using parallel hardware and machine learning techniques can bridge the gap between research and a wide variety of manufacturing applications.