Efficient Planning by Graph Rewriting

José Luis Ambite and **Craig A. Knoblock**
Information Sciences Institute and Department of Computer Science
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{ambite, knoblock}@isi.edu

Planning involves the generation of a network of actions that achieves a desired goal given an initial state of the world. There has been significant progress in the analysis of planning algorithms, particularly in partial-order and in hierarchical task network (HTN) planning (Kambhampati 95; Erol et al. 94). In this abstract we propose a more general framework in which planning is seen as a graph rewriting process. This approach subsumes previous work and offers new opportunities for efficient planning.

As motivation, we will look at two domains: query processing in a distributed environment and manufacturing operation planning. Distributed query processing involves generating an efficient plan to satisfy a user query. This plan is composed of data retrieval actions at diverse information sources and operations on this data (such as join, selection, etc). Some systems use a general-purpose planner to solve this problem (Knoblock 95). We have observed that, in this domain, it is relatively easy to construct an initial plan, and then transform it using a hill-climbing search to reduce its cost. The plan transformations exploit the commutative and associative properties of the (relational algebra) operators, and the fact that when a group of operators can be executed together at a remote information source it is generally more efficient to do so. Some sample rules are: join-swap, $get(q1, db1) \bowtie (get(q2, db2) \bowtie get(q3, db3)) \Leftrightarrow get(q2, db2) \bowtie (get(q1, db1) \bowtie get(q3, db3))$; and remote-eval, $get(R, db) \bowtie get(S, db) \Leftrightarrow get(R \bowtie S, db)$. In centralized databases, some domain-specific planners exploited a similar idea (Graefe and DeWitt 87).

In manufacturing, the problem is to find an economical plan of machining operations that implement the desired features of a design. In a feature-based approach (Nau et al. 95), it is possible to enumerate the possible actions involved in building a piece by analyzing its CAD model. It is more difficult to find an ordering of the operations and the setups that optimize the machining cost. However, similar to query planning, it is possible to incrementally transform a (possibly inefficient) initial plan. Often, the order of actions does not affect the design goal, only the quality of the plan, thus actions can commute. Also, it is important to minimize the number of setups because fixing a piece on the machine is rather time consuming. Such grouping of machining operations on a setup is analogous to evaluating a subquery at a remote information source.

A (partial) plan is a labelled graph whose nodes are actions and whose edges express constraints (ordering, causal links, etc). In planning by graph rewriting we allow the substitution of an arbitrary partial plan by another partial plan. This subsumes the main transformations present in partial-order planners (adding a new node or linking to a previous one for goal establishment, adding ordering edges for threat resolution), and in HTN planners (substituting a non-primitive action by a partial plan). We have seen several planning domains that benefit from expressive plan transformations (as the query evaluation rules above). We expect that hill-climbing from a (possibly suboptimal but easily constructed) initial plan using such transformations will be efficient for many domains, similarly to (Minton et al. 92). Moreover, this more expressive planning language should give users more control over the kind of solutions they prefer (Kambhampati 95).

References

S. Kambhampati. 1995. A Comparative Analysis of Partial-Order Planning and Task Reduction Planning. SIGART 6(1).

K. Erol, D. Nau, and J. Hendler. 1994. UMCP: A Sound and Complete Planning Procedure for Hierarchical Task-Network Planning. AIPS'94.

C. A. Knoblock. 1995. Planning, Executing, Sensing, and Replanning for Information Gathering. IJCAI'95.

G. Graefe and D. J. DeWitt. 1987. The EXODUS Optimizer Generator. SIGMOD'87.

D. S. Nau, S. K. Gupta, and W. C. Regli. 1995. AI Planning versus Manufacturing-Operation Planning: A Case Study. IJCAI'95.

S. Minton, M. D. Johnston, A. B. Philips & P. Laird. 1992. Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems. Artificial Intelligence, 58(1–3).