Research Summary of Investigations Into Optimal Design-to-time Scheduling*

Alan Garvey and Victor Lesser
Computer Science Department
University of Massachusetts
Amherst, MA 01003

Abstract

Design-to-time scheduling problems are real-time problems where multiple methods are available for many subproblems. This paper briefly summarizes recent work empirically investigating the design-to-time search space. The effects of several parameters on the difficulty of the scheduling are mentioned and discussed.

Introduction

We are interested in the scheduling of real-time tasks in environments where multiple methods exist for solving many of the tasks. Our approach is known as *design-to-time real-time scheduling* [Garvey *et al.*, 1993; Garvey and Lesser, 1993] and involves designing solutions at runtime that take advantage of all available time to generate the best answers we can find. We are particularly interested in the effects of hard and soft interactions among subtasks. This research summary is an abbreviated version of a technical report that discusses our investigations into optimal design-to-time scheduling [Garvey and Lesser, 1994].

We are interested in research on experimental evaluation of reasoning and search methods for two reasons. One reason is to help us understand how difficult our scheduling problem is for typically-sized tasks. The other reason is to help us understand what aspects of the problem make it particularly difficult, which may be useful knowledge when designing heuristic solutions.

Our Problem

The design-to-time scheduling problem is to schedule the execution of executable *methods* in TÆMS task structures [Decker and Lesser, 1993]. The leaves in a TÆMS task structure are executable computations (known as methods); other elements are tasks that represent ways that methods can be combined together to achieve *quality*, the value of the overall computation. Tasks achieve quality as a function of the quality of their subtasks. In the general case there is a deadline associated with each element of a task structure, although in most of our recent work we have assumed that there is a single deadline for each *task group* (independent set of tasks and methods). Figure 1 shows a very simple example of a single task group to be scheduled. Different methods are often available for achieving quality for a task (e.g., Method B1 or Method B2 could be executed to achieve quality for Task B). Some methods and tasks interact with one another (e.g., Method A1 facilitates[1] Tasks B and C, and Task A enables[2] Task B). Figure 2 is an example of the kind of output expected from the scheduler. The evaluation criteria we use prefers schedules that produce the highest possible quality with a secondary criteria of minimizing schedule duration.

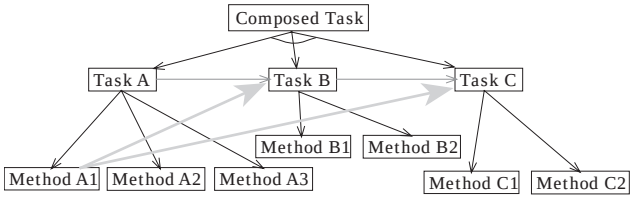

Figure 1: An example of a very simple task structure to be scheduled. The black lines indicate subtasks relationships (forming an and/or graph). The thick gray lines indicate facilitates relationships. The thin gray lines indicate enables relationships.

Previous work on design-to-time scheduling has focused on heuristic algorithms because of the inherent complexity of the problem. More recently we have been looking at algorithms for finding optimal design-to-time schedules. In the worst case this problem is $O(m!)$ where m is the number of executable methods. In many cases pruning can reduce

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9208920, NSF contract CDA 8922572, DARPA under ONR contract N00014-92-J-1698 and ONR contract N00014-92-J-1450. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

[1] A facilitates relationship from Task X to Method Y means that, if Task X executes before Method Y, then Method Y can achieve greater quality and/or have reduced duration. For example, facilitates can occur when one method generates a result that reduces the work required for another method.

[2] An enables relationship is just a precedence constraint. Task A must have quality greater than a threshold before Task B can begin execution.

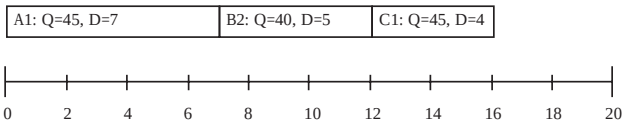

Figure 2: An example of a schedule for Figure 1 produced by the scheduler.

this complexity significantly. For example, if all methods are strictly ordered by precedence constraints (i.e., there is exactly one acceptable ordering for any group of methods) then the complexity is reduced to 2^m (the size of the power set of m). In general pruning is very difficult for this problem, because it is difficult to take a partial schedule and get a tight bound on the highest quality that can be expected to be produced by schedules built on that partial schedule.

Of course, all of these complexity measures show that, even with simplifications, our problem is intractable in the general case. In practice we find it necessary to place a cutoff on the number of nodes expanded in the search for an optimal schedule (usually 50,000 in the experiments described here). This makes it more difficult to interpret our experimental results.

Results

We are interested in looking at the effects of parameters such as tightness of deadline, likelihood of facilitates or enables relationships, and optimality criteria on the size of the search space for the optimal scheduling algorithm. Detailed results of our experiments are described in a longer technical report [Garvey and Lesser, 1994]. Here we provide a brief overview of the main results.

As an example of the kind of experimental results we are hoping to produce, Figure 3 shows the effect of reducing the optimality criteria on the size of the search space (number of nodes that need to be expanded to find an optimal schedule). In this experiment we calculated an upper bound on the quality that could be expected to be achieved by extensions to a partial schedule and pruned partial schedules that could not exceed the quality of the current best schedule by more than an experimentally-varied percent. Figure 3 shows that as the percent increases from 0 to 25% the size of the search space expanded is reduced by nearly two-thirds.

At this point we have qualitative results that address many of the questions we would like to answer. These results suggest what the answers will probably look like, but have not been verified to our satisfaction.

- Effect of deadline tightness – As deadlines get tighter, the number of schedules that do not violate deadlines gets smaller. For this reason we would expect more pruning to be possible with tighter deadlines. In practice this seems to be the case, with an approximately linear decrease in nodes searched as the deadlines get tighter.
- Effect of likelihood of soft effects (e.g., facilitates) – This is the likelihood that a soft relationship exists between a pair of elements in the task structure. Theoretically as the likelihood increases, the size of the search space should increase. This seems to be the empirical result, although the effect seems to be small.
- Effect of likelihood of hard effects (e.g., enables) – Similarly, this is the likelihood of a hard relationship (meaning one that cannot be violated) in the task structure. As mentioned briefly above, as the likelihood of precedence constraints increases, the number of valid schedules should decrease. In practice we have not found this to be the case, but that may be a shortcoming of the pruning in our optimal algorithm.

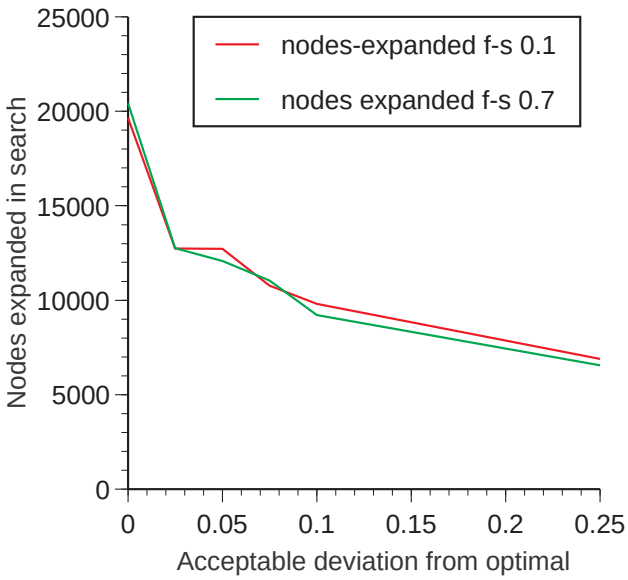

Figure 3: Nodes expanded to find optimal schedule versus the acceptable deviation from optimal in schedule quality. The two lines are for facilitates strengths of 0.1 and 0.7. Facilitates strengths determines how much the duration and quality of the facilitated method are affected.

Other issues that we are investigating include a comparison of the performance of our heuristic scheduler with the optimal scheduler, and the usefulness of priming the optimal scheduler with a good schedule (perhaps produced by first running the heuristic scheduler).

References

Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of AAAI-93*, pages 217–224, Washington, D.C.

Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1491–1502, 1993.

Alan Garvey and Victor Lesser. Investigations into optimal design-to-time scheduling. CS technical report, University of Massachusetts, 1994. To appear.

Alan Garvey, Marty Humphrey, and Victor Lesser. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of AAAI-93*, pages 580–585, Washington, D.C.