Parallel Tabu Search Message-Passing
Synchronous Strategies for Task Scheduling
Under Precedence Constraints

STELLA C.S. PORTO
Department of Telecommunication Engineering, Universidade Federal Fluminense, Rua Passos da Pátria 156, Niterói 24210, RJ, Brazil, email: stella@caa.uff.br

CELSO C. RIBEIRO
Department of Computer Science, Pontifícia Universidade Católica, Rua Marquês de São Vicente 225, Rio de Janeiro 22453-970, RJ, Brazil, email: celso@inf.puc-rio.br

Abstract
This paper presents parallelization strategies for a tabu search algorithm for the task scheduling problem on heterogeneous processors under task precedence constraints. Parallelization relies exclusively on the decomposition of the solution space exploration. Four different parallel strategies are proposed and implemented on an asynchronous parallel machine under PVM: the master-slave model, with two different schemes for improved load balancing, and the single-program-multiple-data model, with single-token and multiple-token message passing schemes. The comparative analysis of these strategies shows that the tabu search approach for this problem is very suitable to the parallelization of the neighborhood search, with efficiency results almost always close to one for problems over a certain size.

Key Words: task scheduling, tabu search, parallel algorithms, master-slave, SPMD

1. Introduction

When parallel application programs are executed on MIMD machines, the parallel portion of the application can be speeded up according to the number of processors allocated to it. In a homogeneous architecture, where all processors are identical, the sequential portion of the application will have to be executed in one of the processors, considerably degrading the execution time. A faster processor tightly coupled to smaller ones and responsible for executing the serial portion of the parallel application, may lead to higher performance (Menascé and Almeida, 1990). In a homogeneous multiprocessor environment, one has to be able to determine the optimum number of processors to be allocated to an application (processor allocation), as well as which tasks will be assigned to each processor (processor assignment). In a heterogeneous setting, one has to determine not only how many but also which processors should be allocated to an application, as well as which processors are going to be assigned to each task. Deciding the processor on which a certain task will be executed is a more complex procedure in the homogeneous case, where all processors have distinct processing speeds.

Given a parallel application defined by a task precedence graph, task scheduling (or processor assignment) may be performed either statically (before execution) or dynamically
(during execution). In the former case, there is no scheduling overhead to be considered during execution, but decisions are usually based on estimated values about the parallel application and the multiprocessor system. The work of each processor is defined at the time of compilation. More accurate information is used in a dynamic scheduling scheme. Each processor does not know a priori which tasks it will execute because processors are assigned to tasks during the execution of the application. To avoid the overhead due to the scheduling procedure, processor assignment should be done very fast by a simple algorithm, although this has the evident disadvantage of causing the quality of the resulting solution eventually to deteriorate. By contrast, in the case of static scheduling, although less information is available, more sophisticated algorithms may be used since the compiler will be in charge of the assignment. The compilation time will certainly be longer, but the cost of task management should be smaller, since each processor will be ready in advance.

Dynamic processor assignment is justified when the processors allocated to an application are not known beforehand or when the execution times cannot be accurately estimated at the time of compilation. If the task-precedence graph that characterizes the parallel application can be accurately estimated a priori, then a static approach is more attractive. Moreover, increasing compilation times is entirely justified—for example, for large scientific programs, where the execution times are much more relevant. These applications present reliable estimates for task execution times, due to their known behavior and regularity. Thus, scheduling will be performed only once, and the program itself will be repeatedly executed, relying on few parameters whose management will not interfere with its processing time. Consequently, even if the scheduling effort is costly, its cost will be amortized over the many times the schedule will be re-applied.

The task-scheduling problem in a heterogeneous multiprocessor environment with applications represented by task-precedence graphs was first considered by Porto and Menascé in (Menascé and Porto, 1992; Porto, 1991). The focus of this work was on the processor assignment problem, assuming that processor allocation had already been performed. Greedy algorithms for processor assignment of parallel applications modeled by task-precedence graphs in heterogeneous multiprocessor architectures have been proposed and compared. More recently, Porto and Ribeiro (1995) applied the tabu search metaheuristic to the static task-scheduling problem under precedence constraints in a heterogeneous multiprocessor environment. The results obtained by tabu search improved by approximately 25 percent the makespan (that is, the completion time) of the parallel applications, with respect to the schedule generated by the best greedy algorithm.

Tabu search is a local search adaptive procedure for solving combinatorial optimization problems, which guides a hill-descending heuristic to continue exploration without becoming confounded by an absence of improving moves and without falling back into local optima from which it previously emerged (Porto and Ribeiro, 1995). In the case of the task scheduling problem considered in this work, where the precedence constraints determine a cost function whose evaluation takes $O(n^2)$ time (where $n$ stands for the number of tasks to be scheduled), the computational times required by any method that relies on such evaluations may be very large. This motivates the development of a parallel tabu search implementation, considering the promising results of applying this approach to other combinatorial problems already reported in the literature (Chakrapani and Skorin-Kapov, 1992; Crainic, Toulouse, and Gendreau, 1993a, 1993b; Fiechter, 1994; Garcia and Toulouse, 1994).