# Scheduling with Duplication on $m$ Processors with Small Communication Delays

A. Goldman, [a] C. Rapine [b]

[a] *Department of Computer Science – IME – USP, e-mail:* goldman@ime.usp.br

[b] *ENSGI – Laboratoire GILCO – INPG, e-mail:* rapine@gilco.inpg.fr

**Abstract**

We consider the problem of scheduling a directed acyclic graph on a limited number of processors in presence of communication delays ($P|prec, c_{ij}|C_{\max}$). Under the small communication time assumption, meaning that the communication delays are smaller than the computation times of the tasks, we propose a new list scheduling algorithm. This algorithm takes advantage of the duplication to reduce the impact of the communications on the schedule makespan. We establish for this algorithm a worst case performance guarantee of 2 compared to an optimal solution.

*Key words:* Scheduling, communication delays, performance guarantee.

We consider the problem of scheduling a parallel application described by a directed acyclic graph on a limited number of processors allowing task duplication. The vertices of the graph represent tasks, and the arcs, precedence relations among them. Each task on the graph must be executed by at least one processor, without being interrupted. An arc $(i, j)$ on the graph means that task $j$ needs the data computed by task $i$ in order to start its computation. When not assigned to the same processors a communication delay must be assumed for data transfer between these two tasks. We are interesting in this paper in the minimization of the parallel time, also called *makespan*, of an application when communications are smaller than computations.

The central problem of scheduling theory on parallel machines $P|prec|C_{max}$ assuming no communication delay was first addressed by Graham [2] who proved that list scheduling algorithms lead to performance ratio $2 - \frac{1}{m}$. The performance ratio of an algorithm denotes here the worst instance value of $\omega/\omega_{opt}$, where $\omega$ is the makespan computed by the algorithm and $\omega_{opt}$ the one of an optimal schedule. Although slight improvements can be obtained choosing clever priority lists, no other algorithm has been proposed for general precedence graphs to reduce this gap.

When introducing communication delays, Veltman et al. [4] proved that unless $\mathcal{P} = \mathcal{NP}$ no polynomial algorithm can have a better performance ratio than $\frac{5}{4}$. Besides, list algorithms perform then less efficiently [6], leading to a performance ratio $2 - \frac{1}{m} + \rho$. The term $\rho$ will design in the following the granularity of the graph, defined as the ratio between the greatest communication delay and the smallest computation time. Instance problems with $\rho \leq 1$ are classically referred as *small communication time* (SCT) problems.

To reduce the impact of the communication delays on Graham's bound, a first approach consists in considering the relaxed resource constrains problem, referred as the *clustering problem*, allowing an unbounded number of processors. Indeed Konig and Munier [5] proved that an approximate solution with performance ratio $\lambda_\infty$ for the clustering problem can be used when $\rho \leq 1$ to build a polynomial pseudo list scheduling on $m$ processors with performance guarantee $\lambda_m = 1 + (1 - \frac{1}{m})\lambda_\infty$. Although the clustering problem remains "almost" as difficult as the initial scheduling problem since Veltman et al. established that there is no polynomial algorithm with better performance ratio than $\frac{7}{6}$ unless $\mathcal{P} = \mathcal{NP}$.

Another approach is to introduce task duplication to suppress a part of the communications. In case of an unlimited number of processors, Colin and Chrtienne [1] proved that duplication allows optimal polynomial solutions for $\rho \leq 1$ while Papadimitriou and Yannakakis [7] proposed a 2 performance ratio approximation for $\rho \geq 1$. However few authors dealt with the effective scheduling of task graph on $m$ processors, except Hanen and Munier [3] who addressed the case of unitary computation and communication costs (UET-UCT) and proposed an heuristic with performance ratio $2 - \frac{1}{m}$. Making analogy with the clustering problem and its reduction on $m$ processors in the non-duplication case, we could hope to obtain duplication algorithm with performance guarantee $1 + (1 - \frac{1}{m})$ for the SCT problem, that is to suppress the impact of the communication from Graham's bound. We present in the following a simple list scheduling heuristic using duplication, it has performance ratio of 2 for the small communication time problem. This problem is also known as $P|prec, c_{ij}, dup|C_{\max}$ under SCT assumption.

**The problem**

We consider the problem of scheduling a parallel application described by a directed acyclic graph $G = (V, E)$ on $m$ identical processors allowing task duplication. $p_i$ is the execution time of task $i$. Data dependence between tasks $i$ and $j$ is represented by an arc $(i, j) \in E$. Task $i$ is said to be an *immediate predecessor* of $j$. When task $i$ and $j$ are not assigned to the same processor a communication delay $c_{ij}$ is assumed for data transfer between these two tasks. We denote by $c = \max_{i,j}\{c_{ij}\}$ the maximum value of a communication delay, and by $\rho$ the ratio $c/\min_i\{p_i\}$. In the following of the paper we will focus on the SCT problem, where $\rho \leq 1$. Since duplication is allowed, a task $i$ may be executed in a schedule $\sigma$ more than once, by different processors and at different dates. Each execution of a task is called a *copy*. We particularize the copy (one of the copies) starting earliest as the *original copy* of the task.

For a given schedule $\sigma$, we denote by $\omega$ its makespan, defined as the maximal completion time of a task, and by $t_i$ and $\pi_i$ the starting time and processor assignment of task $i$. To be *valid* a schedule $\sigma$ must satisfy the following conditions: 1) Each task $i$ is executed without interruption during time $p_i$ by at least one processor; 2) At any time a processor executes at most one task; 3) If arc $(j, i) \in E$, either there exits a copy of $j$ finishing before date $t_i$ on $\pi_i$, or $j$ completes before date $t_i - c_{ji}$. The last property stands that the communication delay between task $j$ and $i$ can be (partially) reduced executing a copy of $j$ on the same processor than $i$.

We introduce now some definitions for the ease of the presentation; recall that for each definition we address to the original copy of a task. The following definitions refer to a constructive schedule. The underlying meaning captured by these definitions is the existence of a particular date for the scheduling of a task $i$, namely the *free date* $\tilde{d}_i$, when the communication from all its predecessors has time to take place. This date reflects the constraints upon the processor allocation of the task. Indeed, while the task can be scheduled indifferently on any processor after its *free date*, an earlier execution means the existence of a particular predecessor, called *favorite*, and the allocation of the task on the same processor than it (or one of its copies). We define the *available date* of $i$, $d_i$, to be its earliest possible scheduling time according to its (original) predecessors. We characterize such an allocation introducing the delay between the completion time of the favorite predecessor and the starting time of the task $i$, $\Delta_i$. Figure 1 illustrates these definitions.

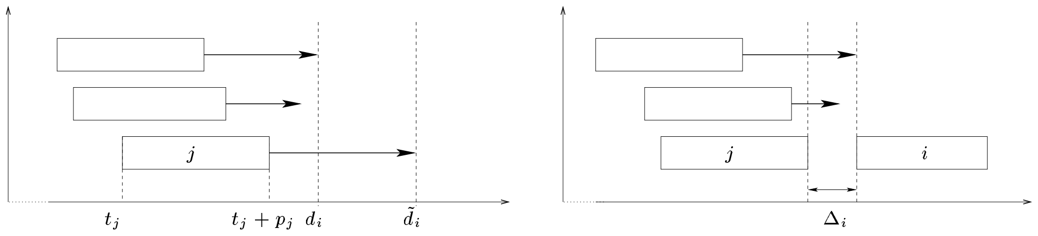


Fig. 1. The set of predecessors of a task $i$ (on the left) and its possible delayed allocation (on the right). Task $j$ is the favorite predecessor of $i$. The different quantities $\tilde{d}_i$ (free date), $d_i$ (available date) and $\Delta_i$ are depicted on the figure.

The favorite predecessor definition makes appear a particular subset of the immediate predecessors of a task $i$ such that allocating $i$ to the same processor than one of these tasks minimizes its starting time (equal by definition to $d_i$). We can easily show that a task $i$ admits at most one favorite predecessor $j$.

EDTF (Earliest Duplication Task First) is a list scheduling algorithm based on the feasible starting time of a task. The computation of a feasible starting time $\tau_i$ for a task $i$, when duplication is allowed, is based on the notion of duplicative path similarly to [3]. The main idea is to construct a path where at each step the favorite predecessor of the last added node is included. The construction stops when the delays $\Delta_i$ on the path sums up to $\Delta \geq c$ or when a task with no favorite predecessor (free task) is found. Using this path, and duplicating all, some, or none of its tasks, the feasible starting time of each ready task (task whose predecessors are already scheduled) on each processor can be computed. The principle of the EDTF algorithm is to allocate at each step the task with the smallest feasible starting time.

It is interesting to observe that under UET, UCT hypothesis [3] the duplicative path does not have any delay and it always starts on a free task. In this case, to compute the feasible starting time, the duplicative path path is duplicated either from its beginning or from one of its already allocated tasks.

To prove that the EDTF algorithm has a performance guarantee of 2 we use a technique similar to Hanen & Munier [3]. By bounding the total idle time of the schedule $\sigma$ in function of the makespan of $\sigma^\infty$, an optimal schedule on an unlimited number of processors.

This paper can be viewed as a natural continuation of Hanen and Munier's work on the UET-UCT problem. Under SCT assumption, tasks duplication strategies lead to a simple list algorithm which almost get ride of the factor added by the communication costs to the Graham's bound.

## References


[1] J.-Y. Colin and P. Chrétienne. CPM scheduling with small interprocessor communication delays. *Oper. Res.*, 39(3), 1991.

[2] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17(2):416–429, March 1969.

[3] C. Hanen and A. Munier. Using duplication for scheduling unitary tasks on $m$ processors with communication delays. *Theo. Comp. Science*, 178:119–127, 1997.

[4] B. Veltman J. Hoogeveen, J. Lenstra. Three, four, five, six, or the complexity of scheduling with communication delays. *Oper. Res. Lett.*, 16:129–137, 1994.

[5] J.C. Konig and A. Munier. A heuristic for a scheduling problem with communications delays. *Oper. Res.*, 45(1), 1997.

[6] Z. Liu. A note on Graham's bound. *Info. Proc. Lett.*, 36:1–5, October 1990.

[7] C. Papadimitriou and M. Yannakakis. Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comp.*, 19(2):322–328, 1990.