International Journal of Foundations of Computer Science Vol. 13 No. 4 (2002) 613–627
© World Scientific Publishing Company

APPROXIMATION ALGORITHMS FOR SCHEDULING MALLEABLE TASKS UNDER PRECEDENCE CONSTRAINTS

RENAUD LEPÈRE
Laboratoire ID (Informatique et Distribution) - IMAG, 51 rue J.Kuntzmann
38330 Montbonnot St.Martin, France

and

DENIS TRYSTRAM
Laboratoire ID (Informatique et Distribution) - IMAG,
51 rue J.Kuntzmann, 38330 Montbonnot St.Martin, France
Denis.Trystram@imag.fr.

and

GERHARD J. WOEGINGER
Institut für Mathematik, Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria
Supported by the START program Y43-MAT of the Austrian Ministry of Science
gwoegi@opt.math.tu-graz.ac.at

Received 20 December 2000
Accepted 4 December 2001
Communicated by Albert Zomaya

ABSTRACT

This work presents approximation algorithms for scheduling the tasks of a parallel application that are subject to precedence constraints. The considered tasks are malleable which means that they may be executed on a varying number of processors in parallel. The considered objective criterion is the makespan, i.e., the largest task completion time. We demonstrate a close relationship between this scheduling problem and one of its subproblems, the allotment problem. By exploiting this relationship, we design a polynomial time approximation algorithm with performance guarantee arbitrarily close to $(3+\sqrt{5})/2 \approx 2.61803$ for the special case of series parallel precedence constraints and for the special case of precedence constraints of bounded width. These special cases cover the important situation of tree structured precedence constraints. For arbitrary precedence constraints, we give a polynomial time approximation algorithm with performance guarantee $3+\sqrt{5} \approx 5.23606$.

Keywords: parallel computing – scheduling – malleable tasks – precedence constraints – series parallel order – bounded width – approximation algorithm – project management – discrete time-cost tradeoff problem.

1. Introduction

Scheduling and load-balancing are central issues in the parallelization of large scale applications. One of the main problems in this area concerns efficient scheduling of the tasks of a parallel program. This problem asks to determine at what time and on which processor all the tasks should be executed. Among the various possible approaches, the most commonly used is to consider the tasks of the program at the finest level of granularity, and to apply some adequate clustering heuristics for reducing the relative communication overhead; see Gerasoulis & Yang [9]. Several models have been developed for modeling the communication and the parallelization overhead in these problems. In models with a finer communication representation (like in the LogP model [3]), the impact of the parallelization overhead is usually ignored.

Recently, a new computational model called *Malleable tasks* (MT) has been proposed by Turek, Wolf & Yu [19] as an alternative to the usual delay model. Under the MT model, the precedence task graph depicts a coarse grain vision of a parallel application and tasks are computational units which may be themselves executed in parallel. The execution time of a malleable tasks depends upon the number of processors alloted to execute it and is determined more or less precisely for each application. The influence of communications inside malleable tasks is taken into account implicitely by this execution time which includes a penalty due to the management of the parallelization (communications, synchronization, etc.). The communications between malleable tasks are usually neglected, since the granularity under the MT model is large. The MT model allows to exploit two levels of parallelism: inside and between malleable tasks.

The MT model is particulary well-suited for applications such as domain decomposition where a coarse grain description of the application is natural. Blayo, Debreu, Mounié & Trystram applied malleable tasks for solving an application for the simulation of the oceanographic circulation based on adaptive mesh refinement within finite differences [1]. The MT model has been also used for tree search for determining the polynomial roots using Descartes' method [20]. In this application, Decker & Krandick considered independant MT with identical computing times. We refer the reader to Lepère, Mounié, Robič & Trystram [13] for more details and motivations of the MT model.

MT are closely related to two other models, namely to the model of *multiprocessor tasks* (see e.g. Drozdowski [7]) and to the model of *divisible tasks* (Prasanna & Musicus [17]). The difference between these models lies in the freedom allowed to the task allotment, that is, the number of processors which execute each task: A multiprocessor task requires to be executed by a fixed integer number of processors, whereas divisible tasks share the processors as a continuously divisible resource.

1.1. The malleable tasks model

Throughout this paper we assume that the parallel program is represented by a set of generic malleable tasks, that is, computational units that may be parallelized and that are linked by precedence constraints. The precedence constraints are determined a priori by the analysis of the data flow between the tasks. More formally, let $G = (V, E)$ be a directed graph where $V = \{1, 2, \ldots, n\}$ represents the set of malleable tasks, and where $E \subseteq V \times V$ represents the set of precedence constraints among the tasks. If there is an arc from task i to task j in E, then task i must be processed completely before task j can begin its execution. This situation will be denoted by $i \rightarrow j$; i is called a *predecessor* of j, and j is called a *successor* of i. All tasks are available at time 0 for execution, and they are to be scheduled on an overall number of m processors. Every task j is specified by m positive integers $p_{j,q}$ $(1 \leq q \leq m)$ where $p_{j,q}$ denotes the execution time of task j when it is executed in parallel on q processors.

Motivated by the usual behavior of parallel programs (cf. Cosnard & Trystram [2]), we make the following assumptions on the task execution times. Blayo, Debreu, Mounié & Trystram [1] have shown these assumptions to be realistic while implementing actual parallel applications.

Assumption 1 *(Monotonous penalty assumptions)*

(a) The execution time $p_{j,q}$ of a malleable task j is a non-increasing function of the number q of processors executing the task.

(b) The work $w_{j,q} \doteq q \cdot p_{j,q}$ of a malleable task j is a non-decreasing function of the number q of processors executing the task.

Assumption (a) means that adding some processors for executing a malleable task cannot increase its execution time. In practice, the execution time even goes down in this situation, at least until a threshold from which onwards there is no more parallelism. Assumption (b) reflects that the total overhead for managing and administrating the parallelism usually increases with the number of processors.

A *schedule* σ is specified by two functions $start_\sigma : V \rightarrow \mathbb{N}$ and $allot_\sigma : V \rightarrow [1, m]$ where the function $start_\sigma$ associates to each task a date of execution (or starting time), and where the function $allot_\sigma$ specifies the number of processors to execute a task. In schedule σ, the task j completes at time $C_\sigma(j) = start_\sigma(j) + p_{j,allot_\sigma(j)}$. We say that task j is *active* during the time interval from $start_\sigma(j)$ to $C_\sigma(j)$, and we denote by $active(t)$ the set of all tasks that are active at time t. A schedule σ is a *feasible* schedule, if at any moment t in time at most m processors are engaged in the computation

$$\sum_{j \in active(t)} allot_\sigma(j) \leq m \qquad \text{for all } t \geq 0,$$

and if all the precedence constraints are respected:

$$start_\sigma(i) + p_{i,allot_\sigma(i)} \leq start_\sigma(j) \qquad \text{for all } i \rightarrow j.$$

The *makespan* $C_{\max}$ of a schedule σ is the maximum of all task completion times $C_\sigma(j)$. We now introduce the central problem of this paper.

Problem 2 *MAKESPAN PROBLEM FOR MALLEABLE TASKS (*MT-MAKESPAN*)*

INSTANCE: *A directed graph $G = (V, E)$ that represents a set of n precedence constrained malleable tasks; the number m of processors; positive integers $p_{j,q}$ with*

$1 \leq j \leq n$ and $1 \leq q \leq m$ that specify the task execution times.

GOAL: *Find a feasible schedule that minimizes the makespan $C_{\max}$.*

Consider an instance of MT-MAKESPAN, and assume that some processor allotment α has been prespecified for all tasks, and that task j is to be executed on exactly α_j processors. Then the execution time of task j is p_j, and its work is $\alpha_j p_j$. With every directed path through the precedence graph $G = (V, E)$, we associate the total execution time p_j of the vertices on this path. The longest path under this definition of length is called the *critical* path of the allotment α, and its length is denoted by L^α. Moreover, we denote by $W^\alpha = \sum_{j=1}^{n} \alpha_j p_j$ the overall work in allotment α. Clearly,

$$c(\alpha) \doteq \max\{L^\alpha, \frac{1}{m}W^\alpha\} \leq C_{\max} \tag{1}$$

holds for the makespan $C_{\max}$ of any feasible schedule σ under allotment α: Since the schedule must obey the precedence constraints, the tasks along the critical path form a chain that forces the makespan to at least L^α. Since the total work W^α can only be distributed across m processors, some processor will run for at least W^α/m time units. The value $c(\alpha)$ in equation (1) will be called the *cost* of allotment α. With this discussion, it is fairly natural to consider the following auxiliary problem.

Problem 3 *ALLOTMENT PROBLEM FOR MALLEABLE TASKS (*MT-ALLOTMENT*)*

INSTANCE: *A directed graph $G = (V, E)$ that represents a set of n precedence constrained malleable tasks; the number m of processors; positive integers $p_{j,q}$ with $1 \leq j \leq n$ and $1 \leq q \leq m$ that specify the task execution times.*

GOAL: *Find an allotment $\alpha : V \to [1, m]$ that minimizes the cost $c(\alpha)$.*

1.2. Known results

The complexity of the makespan problem for malleable tasks has been studied in the paper of Du and Leung [8]: The problem with arbitrary precedence constraints is strongly NP-hard for $m = 2$ processors, and the problem of scheduling independent malleable tasks is strongly NP-hard for $m = 5$ processors.

Only a few positive results are available for scheduling malleable tasks, and most of them concern independent task systems. Jansen & Porkolab [11] provide a polynomial time approximation scheme (PTAS) for the special case where the number m of processors is a fixed constant, and where the tasks are independent. For the case of independent tasks and an arbitrary number of machines, the best approximability result known has a performance guarantee of 2 (Ludwig & Tiwari [14]). For the slightly easier case where the execution times additionally satisfy the monotonous penalty Assumption 1, Mounié, Rapine & Trystram [16] gave a polynomial time approximation algorithm with a performance guarantee of $\sqrt{3} \approx 1.73205$.

Now let us turn to scheduling malleable tasks under precedence constraints. Prasanna & Musicus [17] proposed an algorithm for some specially structured precedence task graphs for the so-called continuous version of the problem; in the continuous version, a non-integer number of processors may be alloted to any task.

Moreover, they assume the same speed-up function for all tasks. The results of Lenstra & Rinnooy Kan [12] for makespan minimization of precedence constrained sequential tasks imply that unless $P{=}NP$, makespan minimization of precedence constrained malleable tasks cannot have a polynomial time approximation algorithm with worst case performance guarantee better than $4/3$.

The ALLOTMENT PROBLEM FOR MALLEABLE TASKS is closely related to the *discrete time-cost tradeoff* problem, a well-known problem from the project management literature; see e.g. De, Dunne, Ghosh & Wells [4]. The discrete time-cost tradeoff problem is a bicriteria problem for projects, where a project essentially is a system of precedence constrained tasks. Every task may be executed according to several different alternatives, where each alternative takes a certain amount of time and costs a certain amount of money. By selecting one alternative for every task, one fixes the cost (= total cost of all tasks) and the duration (= length of the longest chain) of the project. In the budget variant of the discrete time-cost tradeoff problem, the instance consists of such a project together with a cost bound C. The goal is to select alternatives for all tasks such that the project duration is minimized subject to the condition that the project cost is at most C; the corresponding optimal duration is denoted by $D^*(C)$. By rounding the solutions of a linear programming relaxation, Skutella [18] derives a polynomial time algorithm for this budget variant that finds a solution with project cost at most $2C$ and project duration at most $2D^*(C)$.

Now let us discuss the connection between the allotment problem MT-ALLOTMENT and the discrete time-cost tradeoff problem. In the allotment problem MT-ALLOTMENT, every task j can be executed in m alternative ways by assigning α_j machines to it, where $1 \leq \alpha_j \leq m$. In the language of the discrete time-cost tradeoff problem, the resulting duration of task j is p_{j,α_j} and the resulting cost of task j is $\alpha_j p_{j,\alpha_j}/m$, i.e., its contribution to the value $\frac{1}{m}W^\alpha$. Then the corresponding project cost equals $\frac{1}{m}W^\alpha$, the corresponding project duration equals L^α, and the maximum of these two values equals the cost $c(\alpha)$ of allotment α. By combining the above mentioned result of Skutella [18] with a binary search procedure, we now get the following proposition.

Proposition 1 *The ALLOTMENT PROBLEM FOR MALLEABLE TASKS possesses a polynomial time 2-approximation algorithm.* ■

We furthermore note that the arguments of De, Dunne, Ghosh & Wells [5] imply that the ALLOTMENT PROBLEM FOR MALLEABLE TASKS is NP-complete in the strong sense.

2. Results and outline of the paper

We want to stress that all results in this paper are based on the monotonous penalty Assumption 1. In this paper, we derive polynomial time approximation algorithms for various cases of the MAKESPAN PROBLEM FOR MALLEABLE TASKS and of the ALLOTMENT PROBLEM FOR MALLEABLE TASKS. Let us first define for $m \geq 3$ the real numbers $r(m)$ by

$$r(m) = \min_{1 \leq \mu \leq (m+1)/2} \max \left\{ \frac{m}{\mu}, \frac{2m - \mu}{m - \mu + 1} \right\}. \tag{2}$$

Moreover, let $\mu(m)$ be the integer μ with $1 \leq \mu \leq (m+1)/2$ for which this minimum is attained. The following lemma provides the reader with some intuition on the (somewhat erratic) behaviour of the values $r(m)$ and $\mu(m)$. For small m, the values of $\mu(m)$ and $r(m)$ are listed in Figure 1.

Lemma 1 *The real numbers $r(m)$ and the integers $\mu(m)$ satisfy the following properties.*

(i) For all $m \geq 2$, we have $r(m) < (3 + \sqrt{5})/2 \approx 2.61803$.

(ii) As m tends to infinity, $r(m)$ tends to $(3 + \sqrt{5})/2$.

(iii) For every $m \geq 2$, the value $\mu(m)$ either equals the integer above or the integer below $\frac{1}{2}(3m - \sqrt{5m^2 + 4m})$.

(iv) For every $m \geq 2$ with $m \neq 3$ and $m \neq 5$, we have $\mu(m) \leq m/2$.

(v) As m tends to infinity, $\mu(m)/m$ tends to $(3 - \sqrt{5})/2 \approx 0.38196$. ■

m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$
2	1	2.0000	10	4	2.5000	18	8	2.5454	26	10	2.5625
3	2	2.0000	11	5	2.4285	19	8	2.5000	27	11	2.5294
4	2	2.0000	12	5	2.4000	20	8	2.5000	28	11	2.5454
5	3	2.3333	13	6	2.5000	21	9	2.5384	29	12	2.5555
6	3	2.2500	14	6	2.4444	22	9	2.5000	30	12	2.5263
7	3	2.3333	15	6	2.5000	23	9	2.5555	31	13	2.5789
8	4	2.4000	16	7	2.5000	24	10	2.5333	32	13	2.5500
9	4	2.3333	17	7	2.4545	25	10	2.5000	33	13	2.5384

Figure 1: A listing of the values $\mu(m)$ and $r(m)$ for $2 \leq m \leq 33$.

The straightforward proof of Lemma 1 is omitted. The following theorem summarizes our structural main result on the problems MT-MAKESPAN and MT-ALLOTMENT; its proofs can be found in Section 3. The theorem demonstrates that these two problems are strongly interlocked and interrelated. Moreover, up to some small constant factor it is sufficient to deal with the approximability of the – seemingly easier – problem MT-ALLOTMENT.

Theorem 4 *If there exists a polynomial time ϱ-approximation algorithm A for problem* MT-ALLOTMENT *on m processors, then there exists a polynomial time $\varrho \cdot r(m)$-approximation algorithm B for problem* MT-MAKESPAN *on m processors.*

An immediate consequence of Proposition 1, Theorem 4, and Lemma 1(i) is the following corollary.

Corollary 1 *The MAKESPAN PROBLEM FOR MALLEABLE TASKS possesses a polynomial time approximation algorithm with performance guarantee* $3 + \sqrt{5} \approx 5.23606$. ■

The following Theorem 5 will be a strong and helpful tool for handling specially structured precedence constraints. Its proof can be found in Section 4.

Theorem 5 *Consider the decision version of problem* MT-ALLOTMENT *where for a given instance I of* MT-ALLOTMENT *and for a positive integer bound X, one must decide whether there exists an allocation of cost at most X. If there exists a pseudo-polynomial time exact algorithm for this decision version with running time polynomially bounded in the size of I and in the value of X, then there does exist a fully polynomial time approximation scheme for problem* MT-ALLOTMENT.

A directed precedence graph $G = (V, E)$ is *series parallel* (see e.g. Möhring [15]) if (i) it is a single vertex, (ii) it is the series composition of two series parallel graphs, or (iii) it is the parallel composition of two series parallel graphs. Only graphs that can be constructed via rules (i)–(iii) are series parallel. Here the *series composition* of two directed graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \cap V_2 = \emptyset$ is the graph that results from G_1 and G_2 by making all vertices in V_1 predecessors of all vertices in V_2, whereas the *parallel composition* of G_1 and G_2 simply is their disjoint union. Series parallel precedence constraints are a proper generalization of tree precedence constraints. We have the following result for series parallel precedence constraints.

Theorem 6 *There exists a pseudo-polynomial time exact algorithm for the decision version of the restriction of problem* MT-ALLOTMENT *to series parallel precedence graphs.*

Two tasks i and j are called *independent* if neither i is a predecessor of j nor j is a predecessor of i. A set of tasks is *independent*, if the tasks in it are pairwise independent. The *width* of the precedence graph G is the cardinality of its largest independent set. We have the following result for precedence graphs of bounded width.

Theorem 7 *There exists a pseudo-polynomial time exact algorithm for the decision version of the restriction of problem* MT-ALLOTMENT *to precedence graphs whose width is bounded by a constant d.*

The proofs of Theorems 6 and 7 can be found in Sections 5 and 6, respectively. Finally, by combining the statements in Theorems 4, 5, 6, and 7, we derive the following corollary.

Corollary 2 *For the restriction of the MAKESPAN PROBLEM FOR MALLEABLE TASKS to (a) series parallel precedence graphs and to (b) precedence graphs of bounded width, there exist polynomial time approximation algorithms whose performance guarantee can be made arbitrarily close to* $(3 + \sqrt{5})/2$. ■

3. From allotments to makespans

In this section we will prove Theorem 4. Consider an instance I of the malleable tasks problem as defined in Problems 2 and 3. Consider an optimal allotment α^+

and a ϱ-approximate allotment α^A for instance I with respect to problem MT-ALLOTMENT. Denote by W^+ and W^A the total work in these two allotments, and by L^+ and L^A the lengths of their critical paths, respectively. Since α^A is a ϱ-approximate allotment, we have

$$\max\{L^A,\ \frac{1}{m}W^A\} \ \le\ \varrho \cdot \max\{L^+,\ \frac{1}{m}W^+\}. \tag{3}$$

Moreover, consider an optimal feasible schedule for instance I with respect to problem MT-MAKESPAN, and let $C^*_{\max}$ denote the optimal makespan. By applying equation (1) to $C^*_{\max}$ and to the allotment induced by the optimal schedule, and by using the fact that α^+ minimizes the allotment cost, we get that

$$\max\{L^+,\ \frac{1}{m}W^+\} \ \le\ C^*_{\max}. \tag{4}$$

We will now define and analyze an approximation algorithm B for problem MT-MAKESPAN. This approximation algorithm is based on the value $\mu(m)$ with $1 \le \mu(m) \le (m+1)/2$ as we defined in the paragraph after equation (2). To simplify the presentation, we will from now on briefly write μ for $\mu(m)$, and omit the dependence on m. Algorithm B is a generalization of Graham's [10] well-known list scheduling algorithm for sequential tasks. The algorithm is described in Figure 2. The resulting schedule is denoted σ^B, the corresponding makespan is $C^B_{\max}$, the underlying allotment is α^B, the total work in α^B is W^B, and the length of the critical path in α^B is L^B. The only difference between allotments α^A and α^B is that the tasks using more than μ processors in α^A are compressed to μ processors in α^B. By the monotonous penalty Assumption 1(b), reducing the number of alloted processors cannot increase the work of a task. Together with inequalities (3) and (4) this yields

$$W^B \ \le\ W^A \ \le\ m\varrho C^*_{\max}. \tag{5}$$

The time interval from 0 to $C^B_{\max}$ is partitioned into three types of time slots: During the first type of time slot, at most $\mu - 1$ processors are busy. During the second type, at least μ and at most $m - \mu$ processors are busy, and during the third type at least $m - \mu + 1$ processors are busy. The corresponding sets of time slots are denoted by T_1, T_2, and T_3, respectively. The overall length of the time slots in set T_i, $1 \le i \le 3$, is denoted by $|T_i|$. If $\mu \le m/2$, then every time slot from 0 to $C^B_{\max}$ belongs to exactly one of the three types, and all three types may actually occur. In the boundary case where $\mu = (m+1)/2$ every time slot from 0 to $C^B_{\max}$ either belongs to the first or to the third type. In this boundary case there are no time slots of second type, since this would require that at least $(m+1)/2$ and at most $(m-1)/2$ processors are busy, which clearly is impossible. Since in either case these three types of time slots cover the whole interval from 0 to $C^B_{\max}$, we get that

$$C^B_{\max} \ =\ |T_1| + |T_2| + |T_3|. \tag{6}$$

Since during time slots of the first (respectively second and third) type at least one (respectively μ and $m - \mu + 1$) processors are busy, we get that

$$W^B \ \ge\ |T_1| + \mu|T_2| + (m - \mu + 1)|T_3|. \tag{7}$$

Lemma 2 *The sets T_1 and T_2 of time slots satisfy the following inequality with*

1. Initialization.
– Allot to task j ($j = 1, \dots, n$) exactly $\alpha_j^B = \min\{\alpha_j^A, \mu\}$ processors.
– This fixes the execution time p_j^B and the work $w_j^B = \alpha_j^B \cdot p_j^B$ of every task j.

2. Repeat the following step until all tasks have been scheduled.
– Let READY denote the set of tasks whose predecessors all have already been scheduled.
– Compute for each task $j \in$READY the earliest possible start time under the allotment α^B.
– Schedule the task in READY with the smallest computed earliest start time (ties are broken in favor of tasks with smaller indices).

Figure 2: Approximation algorithm B for problem MT-MAKESPAN.

respect to the length L^A of the critical path in allotment α^A.

$$|T_1| + \frac{\mu}{m}|T_2| \leq L^A. \tag{8}$$

Proof. T □

he idea is to construct a 'heavy' directed path $\mathcal{P}$ in the transitive closure of the graph $G = (V, E)$. The last task in the path $\mathcal{P}$ is any multiprocessor task j_1 that completes at time $C_{\max}^B$ in the schedule σ^B. After we have defined the last $i \geq 1$ tasks $j_i \to j_{i-1} \to \cdots \to j_2 \to j_1$ on the path $\mathcal{P}$, we find the next task j_{i+1} as follows: Consider the latest time slot t in $T_1 \cup T_2$ that lies before the starting time of task j_i in σ^B. Consider the set V' of tasks that consists of task j_i and of all its predecessor tasks that start after time t in σ^B. Since during time slot t at most $m - \mu$ processors are busy, and since σ^B allots at most μ processors to any task in V', all the tasks in V' cannot be ready for execution during the time slot t. Hence, for every task in V' some predecessor is being executed during the time slot t. As the next task j_{i+1} on path $\mathcal{P}$, we select any predecessor of task j_i that is running during slot t. This procedure terminates when $\mathcal{P}$ contains a task that starts before all time slots in $T_1 \cup T_2$.

Now consider a task j on the resulting path $\mathcal{P}$. If α^B allots less than μ processors to task j, then α^A and α^B both allot the same number of processors to j. In this case the execution times of j in α^A and α^B are identical. In schedule σ_B such a task j may be executed during any time slot in $T_1 \cup T_2$. If α^B allots exactly μ processors to task j, then α^A may allot any number k of processors to j, where $\mu \leq k \leq m$. By the monotonous penalty Assumption 1(b), the work $\mu \cdot p_j^B$ in α^B is less or equal to the work $k \cdot p_j^A$ in α^A. Therefore, the execution time p_j^A of task j in allotment α^A is at least $\mu/k \geq \mu/m$ times the execution time p_j^B of j in allotment α^B. In schedule σ_B such a task j may be executed during any time slot in T_2, but not during a time slot in T_1.

By our construction, the tasks on the directed path $\mathcal{P}$ cover all time slots in

$T_1 \cup T_2$ in schedule σ_B. Let us estimate the length $L^A(\mathcal{P})$ of the path $\mathcal{P}$ under the allotment α^A. The tasks that are executed during time slots in T_1 contribute a total length of at least $|T_1|$ to $L^A(\mathcal{P})$. The tasks that are executed during time slots in T_2 contribute a total length of at least $|T_2|\mu/m$ to $L^A(\mathcal{P})$. Since the length L^A of the critical path in α^A is an upper bound on $L^A(\mathcal{P})$, our proof is complete. ■

Now let us complete the proof of Theorem 4. Multiplying (6) by $m-\mu+1$ and subtracting (7) from it yields

$$(m-\mu+1)C^B_{\max} \le W^B + (m-\mu)|T_1| + (m-2\mu+1)|T_2|. \qquad (9)$$

We distinguish two cases. In the first case we assume that $m/\mu \le (2m-\mu)/(m-\mu+1)$. Then (2) yields $r(m) = (2m-\mu)/(m-\mu+1)$. Moreover, the assumed inequality is equivalent to $(m-2\mu+1) \le \mu(m-\mu)/m$. Plugging this into (9), using (8) to bound $|T_1| + \mu|T_2|/m$, using (5) to bound W^B, and using (3) and (4) to bound L^A by $\varrho C^*_{\max}$ alltogether yields that

$$\begin{aligned}(m-\mu+1)C^B_{\max} &\le W^B + (m-\mu)|T_1| + \mu(m-\mu)|T_2|/m \le W^B + (m-\mu)L^A \\ &\le m\varrho C^*_{\max} + (m-\mu)\varrho C^*_{\max} = (2m-\mu)\varrho C^*_{\max}.\end{aligned}$$

Hence, in this case schedule σ^B indeed yields a $\varrho \cdot r(m)$-approximation for $C^*_{\max}$. In the second case we assume that the inequality $m/\mu \ge (2m-\mu)/(m-\mu+1)$ holds. Then (2) yields $r(m) = m/\mu$. Moreover, the assumed inequality is equivalent to $(m-\mu) \le (m-2\mu+1)m/\mu$. By plugging this into (9) and by using similar arguments as in the first case, we conclude that

$$\begin{aligned}(m-\mu+1)C^B_{\max} &\le W^B + (m-2\mu+1)m|T_1|/\mu + (m-2\mu+1)|T_2| \\ &\le W^B + (m-2\mu+1)mL^A/\mu \\ &\le m\varrho C^*_{\max} + (m-2\mu+1)m\varrho C^*_{\max}/\mu = (m-\mu+1)m\varrho C^*_{\max}/\mu.\end{aligned}$$

Hence, also in the second case schedule σ^B yields a $\varrho \cdot r(m)$-approximation for $C^*_{\max}$. Since it is straightforward to implement algorithm B in polynomial time, the proof of Theorem 4 is complete.

4. From a pseudo-polynomial time algorithm to an FPTAS

In this section we will prove Theorem 5. Our first goal is to get a fast algorithm for the following auxiliary allotment problem MT-ALLOTMENT on series parallel precedence graphs: We assume that we are given an instance I of MT-ALLOTMENT, a positive real ε, and an a priori bound X such that there exists an allotment for I with cost at most X. Our goal is to find within polynomial time an allotment α that satisfies $c(\alpha) \le (1+\varepsilon)X$.

Define $Z = \varepsilon X/n$. Furthermore, define a scaled instance I' by setting $p'_{j,q} = \lfloor p_{j,q}/Z \rfloor$ for all tasks j and all $1 \le q \le m$ while keeping the same precedence constraints as in instance I. Note that $p_{j,q} \le Z(p'_{j,q}+1)$. Moreover, note that instance I' must have an allotment of cost at most X/Z, since the original instance I had some allotment of cost at most X. Take the pseudo-polynomial time algorithm that exists according to the assumption of Theorem 5, and apply it to the scaled instance I' with bound $\lfloor X/Z \rfloor$. Denote the resulting allotment by α with $c(\alpha) \le$

$\lfloor X/Z \rfloor$, and interprete allotment α for I' as an allotment β for the original instance I. Consider an arbitrary path $\mathcal{P}$ with $|\mathcal{P}|$ tasks in allotment β. Then

$$\sum_{j\in\mathcal{P}} p_{j,\beta(j)} \;\leq\; \sum_{j\in\mathcal{P}} Z(p'_{j,\beta(j)}+1) \;=\; Z|\mathcal{P}|+Z\sum_{j\in\mathcal{P}} p'_{j,\alpha(j)} \;\leq\; Z\,n+Z\,L^{\alpha}. \quad (10)$$

This implies $L^{\beta} \leq Z\,n + Z\,L^{\alpha}$. Moreover,

$$\sum_{j\in V} \beta(j)\cdot p_{j,\beta(j)} \;\leq\; \sum_{j\in V} Z\cdot\alpha(j)\cdot(p'_{j,\alpha(j)}+1) \;\leq\; Z\,mn+Z\,W^{\alpha}. \quad (11)$$

This implies $W^{\beta} \leq Z\,mn + Z\,W^{\alpha}$. Putting things together we conclude that

$$\begin{aligned} c(\beta) &= \max\{L^{\beta},\ \frac{1}{m}W^{\beta}\} \;\leq\; \max\{Z\,n+Z\,L^{\alpha},\ Z\,n+Z\,\frac{1}{m}W^{\beta}\} \\ &= Z\,n+Z\,c(\alpha) \;\leq\; \varepsilon X+Z\,(X/Z) \;=\; (1+\varepsilon)X. \end{aligned}$$

Hence, the cost of allotment β for I is at most $(1{+}\varepsilon)X$ as desired. By the assumption of Theorem 5, the time to find β is polynomially bounded in the size of I and in $X/Z = n\varepsilon$. To summarize, we can solve our auxiliary problem and find the desired allotment within a running time that is polynomially bounded in the size of I and in $1/\varepsilon$.

It remains to get rid of the assumption that we do have an a priori knowledge of the bound X. Let $P = \sum_{j=1}^{n} p_{j,1}$ denote the total execution time of all tasks in I when they are executed on a single processor. By the monotonous penalty Assumption 1, every critical path in every allotment for I has length at most P, and also the average work of every allotment is at most P. Therefore, the cost of the optimal allotment is at most P, and we can find an $(1+\varepsilon)$-approximation by performing a binary search over the interval from 1 to P. This completes the proof of Theorem 5.

5. Allotments for series parallel graphs

In this section we will prove Theorem 6. Hence, we are given an instance I of MT-ALLOTMENT where the precedence graph $G = (V, E)$ is series parallel, together with a positive integer bound X. Our goal is to decide within pseudo-polnomial time, whether there exists an allotment α with cost $c(\alpha) \leq X$.

It is well known that a series parallel graph can be decomposed in polynomial time into its atomic parts according to the series and parallel compositions (see e.g. Möhring [15]). Essentially, such a decomposition corresponds to a rooted, ordered, binary tree where all interior vertices are labeled by s or p (series or parallel composition) and where all leaves correspond to single vertices of the precedence graph G. We associate with every interior vertex v of the decomposition tree the series parallel graph $G(v)$ induced by the leaves of the subtree below v. Note that for the root vertex we have $G(\mathit{root}) = G$.

For a vertex v in the decomposition tree, and for an integer ℓ with $1 \leq \ell \leq X$, we denote by $F[v,\ell]$ the smallest possible value w with the following property: There exists an allotment α for the tasks in $G(v)$ with $L^{\alpha} \leq \ell$ and $W^{\alpha} \leq w$. It is easy to compute all such values $F[v,\ell]$ by a dynamic programming approach that starts in the leaves of the decomposition tree, and then moves upwards towards the root. This algorithm is sketched in Figure 3. The time complexity of this dynamic

```
1. Initialization of leaf vertices.
   – For every leaves $v$ of the decomposition tree and for every $\ell$ with $0 \le \ell \le X$, set
     $F[v,\ell] := \min_{1\le q\le m}\{q \cdot p_{v,q} \mid p_{v,q} \le \ell\}$.
2. Handling interior vertices of the decomposition tree.
   – For every interior vertex $v$ with left child $v_1$ and right child $v_2$ and for every $\ell$ with $0 \le \ell \le X$ do the following:
   – If $v$ is a $p$ vertex, then $F[v,\ell] := F[v_1,\ell] + F[v_2,\ell]$
   – If $v$ is an $s$ vertex, then $F[v,\ell] := \min_{1\le k\le \ell-1} F[v_1,k] + F[v_2,\ell-k]$
3. Termination.
   – Answer YES if there exists some $1 \le \ell \le X$ with $F[root,\ell]/m \le X$.
     Otherwise, answer NO.
```

Figure 3: A dynamic programming algorithm for computing $F[v,\ell]$.

programming algorithm is $O(nmX^2)$ which is pseudo-polynomially bounded in the input size as desired. By storing appropriate auxiliary information and by performing some backtracking, one can also explicitly compute the corresponding allotment with cost at most X while increasing the running time only by a constant factor. Since these are standard techniques, we do not elaborate on them.

6. Allotments for graphs of bounded width

In this section we will prove Theorem 7. We are given an instance I of MT-ALLOTMENT where the width of the precedence graph $G = (V,E)$ is some fixed constant d, together with a positive integer bound X. Our goal is to decide within pseudo-polnomial time, whether there exists an allotment α with cost $c(\alpha) \le X$.

A well-known theorem of Dilworth [6] states that if the width of a precedence graph equals d then the set V of tasks can be partitioned into d totally ordered chains $V^{(1)},\dots,V^{(d)}$. Moreover, it is straightforward to compute such a chain partition in $O(n^d)$ time. Now consider a maximal set U of independent tasks in G, and let $G(U)$ be the graph that is induced by all the tasks in U together with all their predecessors. For appropriate integers $1 \le i_1^U,\dots,i_d^U \le n$ the task set of graph $G(U)$ consists of the first i_j^U tasks from every chain $V^{(j)}$.

For a maximal independent set U and for d integers $\ell_1,\dots,\ell_d$ with $1 \le \ell_j \le X$, we denote by $F[U,\ell_1,\dots,\ell_d]$ the smallest possible value w with the following property: There exists an allotment α for the tasks in $G(U)$ with total work $W^\alpha \le w$, such that the total execution time on every directed path ending in the i_j^Uth task in the jth chain $V^{(j)}$ is bounded by ℓ_j. It is easy to compute all values $F[U,\ell_1,\dots,\ell_d]$ by a dynamic programming approach, as long as set U is handled before set U' whenever $G(U)$ is a subgraph of $G(U')$. This algorithm is sketched in Figure 4. The time complexity of this dynamic programming algorithm is $O(nmX^{2d})$ which is pseudo-polynomially bounded in the input size. This completes the proof of

1. Initialization.
– If U is the set of tasks without predecessors, then $i_j^U \equiv 1$ and $G(U)$ contains the first task from every chain. For all $\ell_1, \ldots, \ell_d$ with $1 \leq \ell_j \leq X$ compute the value $F[U, \ell_1, \ldots, \ell_d]$ by enumerating all possible allotments for these d tasks.

2. Handling the other independent sets U.
– Assume the following (and otherwise, proceed in a symmetric way): $i_1^U \geq 2$ holds, and the i_1^Uth task in chain $V^{(1)}$ is task v. For $1 \leq i \leq k$, the i_j^Uth task in chain $V^{(j)}$ is a predecessor of v. For $k < i \leq d$, the i_j^Uth task in chain $V^{(j)}$ is not a predecessor of v. Let Z be the maximal independent set for which $G(Z)$ equals $G(U) - \{v\}$.
– For all $\ell_1, \ldots, \ell_d$ with $1 \leq \ell_j \leq X$ set

$$F[U, \ell_1, \ldots, \ell_d] := \min\{q \cdot p_{v,q} + F[Z, \ell'_1, \ldots, \ell'_d]\}$$

where the minimum is taken over all values q and $\ell'_1, \ldots, \ell'_d$ such that $1 \leq q \leq m$, such that $\ell'_i + p_{v,q} = \ell_i$ for $1 \leq i \leq k$, and such that $\ell'_i = \ell_i$ for $k < i \leq d$.

3. Termination.
– Let U^* be the set of tasks without successors in G. Answer YES if there exist some $1 \leq \ell_1, \ldots, \ell_d \leq X$ with $F[U^*, \ell_1, \ldots, \ell_d]/m \leq X$. Otherwise, answer NO.

Figure 4: A dynamic programming algorithm for computing $F[U, \ell_1, \ldots, \ell_d]$.

Theorem 7.

7. Conclusions

In this paper, we have studied the problem of scheduling malleable tasks in the presence of precedence constraints. We designed a polynomial time approximation algorithm with performance guarantee arbitrarily close to $(3+\sqrt{5})/2$ for the special case of series parallel precedence constraints and for the special case of precedence constraints of bounded width. Series parallel precedence constraints contain tree structured precedence constraints as a proper special case. For arbitrary precedence graphs of malleable tasks, we exploited a relationship to the discrete time-cost tradeoff problem and thus derived a polynomial time approximation algorithm with performance guarantee $3 + \sqrt{5}$. We hope that these preliminary theoretical results may open a way to obtain good practical approximation algorithms for scheduling malleable tasks under precedence constraints.

References

1. E. Blayo, L. Debreu, G. Mounié, and D. Trystram [1999]. Dynamic load balancing for ocean circulation with adaptive meshing. *Proceedings of the 5th European Conference on Parallel Computing (Euro-Par'99)*, Springer LNCS 1685, 303–312.

2. M. Cosnard and D. Trystram [1995]. *Parallel Algorithms and Architectures.* International Thomson Publishing.
3. D. Culler, R.Karp, D.Patterson, A.Sahay, E. Santos, K. Schauser, R. Subramanian, and T. von Eicken [1996]. LogP: A practical model of parallel computation. *Communications of the ACM 39*, 78–85.
4. P. De, E.J. Dunne, J.B. Gosh, and C.E. Wells [1995]. The discrete time-cost tradeoff problem revisited. *European Journal of Operational Research 81*, 225–238.
5. P. De, E.J. Dunne, J.B. Gosh, and C.E. Wells [1997]. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research 45*, 302–306.
6. R.P. Dilworth [1950]. A decomposition theorem for partially ordered sets. *Annals of Mathematics 51*, 161–166.
7. M. Drozdowski [1996]. Scheduling multiprocessor tasks – An overview. *European Journal of Operational Research 94*, 215–230.
8. J. Du and J.Y.-T. Leung [1989]. Complexity of scheduling parallel task systems. *SIAM Journal on Discrete Mathematics 2*, 473–487.
9. A. Gerasoulis and T. Yang [1992]. PYRROS: Static scheduling and code generation for message passing multiprocessors. *Proceedings of the 6th ACM International Conference on Supercomputing*, 428–437.
10. R.L. Graham [1966]. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal 45*, 1563–1581.
11. K. Jansen and L. Porkolab [1999]. Linear time approximation schemes for scheduling malleable parallel tasks. *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, 490–498.
12. J.K. Lenstra and A.H.G. Rinnooy Kan [1978]. Complexity of scheduling under precedence constraints. *Operations Research 26*, 22–35.
13. R. Lepère, G. Mounié, B. Robič, and D. Trystram [1999]. Malleable tasks: An electromagnetic efficient model for solving actual parallel applications. *Proceedings of the International Conference on Parallel Computing 99 (Parco'99)*, Imperial College Press, 598–605.
14. W. Ludwig and P. Tiwari [1994]. Scheduling malleable and non malleable parallel tasks. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, 167–176.
15. R.H. Möhring [1989]. Computationally tractable classes of ordered sets. In: I. Rival (ed.) *Algorithms and Order*, Kluwer Academic Publishers, 105–193.
16. G. Mounié, Ch. Rapine, and D. Trystram [1999]. Efficient approximation algorithms for scheduling malleable tasks. *Proceedings of the 11th Annual Symposium on Parallel Algorithms and Architectures (SPAA'99)*, 23–32.
17. G.N.S. Prasanna and B.R. Musicus [1991]. Generalized multiprocessor scheduling using optimal control. *Proceedings of the 3rd Annual Symposium on Parallel Algorithms and Architectures (SPAA'91)*, 216–228.
18. M. Skutella [1998]. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research 23*, 909–929.
19. J. Turek, J. Wolf, and P. Yu [1992]. Approximate algorithms for scheduling parallelizable tasks. *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures (SPAA'92)*, 323–332.
20. Th. Decker and W. Krandick [1999]. Parallel real root isolation using the

Descartes method. *Proceedings of the 6th High Performance Conference (HiPC99)*, 261–268.