INSTITUT FÜR INFORMATIK

Scheduling Malleable Tasks with Precedence Constraints

Klaus Jansen, Hu Zhang

Bericht Nr. 0906
März 2009

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Scheduling Malleable Tasks with Precedence Constraints

Klaus Jansen, Hu Zhang

Bericht Nr. 0906
März 2009

e-mail: kj@informatik.uni-kiel.de,
zhanghu@optserv.cas.mcmaster.ca

Scheduling Malleable Tasks with Precedence Constraints ⋆

Klaus Jansen

Institute of Computer Science and Applied Mathematics, University of Kiel
Olshausenstraße 40, D-24098 Kiel, Germany.

Hu Zhang *

Capital Market Risk Management, Canadian Imperial Bank of Commerce
161 Bay Street, 11th Floor, Toronto, ON M5J 2S8, Canada.

Abstract

In this paper we propose an approximation algorithm for scheduling malleable tasks with precedence constraints. Based on an interesting model for malleable tasks with continuous processor allotments by Prasanna and Musicus [22–24], we define two natural assumptions for malleable tasks: the processing time of any malleable task is non-increasing in the number of processors allotted, and the speedup is concave in the number of processors. We show that under these assumptions the work function of any malleable task is non-decreasing in the number of processors and is convex in the processing time.

Furthermore, we propose a two-phase approximation algorithm for the scheduling problem. In the first phase we solve a linear program to obtain a fractional allotment for all tasks. By rounding the fractional solution, each malleable task is assigned a number of processors. In the second phase a variant of the list scheduling algorithm is employed. By choosing appropriate values of the parameters, we show (via a nonlinear program) that the approximation ratio of our algorithm is at most $100/63 + 100(\sqrt{6469} + 13)/5481 \approx 3.291919$. We also show that our result is asymptotically tight.

Key words: approximation algorithm, scheduling, malleable tasks

1 Introduction

Large requirement of high performance computing arises with the significant development of information technology and its application in many areas in science and engineering. Parallel computer systems with large number of standard units play a key role in current high performance computing and have gradually replaced traditional super computers. In these systems, all units have some similar structure with a certain processing ability [4]. Algorithms with satisfactory performance guarantee are desired to coordinate the resources in such kind of complex systems. However, in general classical scheduling algorithms are not applicable in this case, mainly due to the large amount of communications between units. Many models have been proposed for this problem [3,9,14,27]. In this category scheduling *malleable tasks* proposed in [27] is an important and promising model. The processing time of a malleable task depends on the number of processors allotted to it. The communications between processors allotted to the same task, synchronization and other overhead are implicitly included in the processing time.

We assume that the malleable tasks are linked by *precedence constraints*, which are determined in advance by the data flow among tasks. Let $G = (V, E)$ be a directed acyclic graph, where $V = \{1, \ldots, n\}$ represents the set of malleable tasks, and $E \subseteq V \times V$ represents the set of precedence constraints among the tasks. If there is an arc $(i, j) \in E$, then task J_j cannot be processed before the completion of processing of task J_i. Task J_i is called a *predecessor* of J_j, while J_j a *successor* of J_i. We denote by $\Gamma^-(j)$ and by $\Gamma^+(j)$ the sets of the predecessors and of the successors of J_j, respectively. In addition, each task J_j can be executed on any integer number $l \in \{1, \ldots, m\}$ of homogeneous (identical) processors, and the corresponding discrete positive processing time is $p_j(l)$. The goal of the problem is to find a feasible schedule minimizing the

⋆ The preliminary version of this paper was published in Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2005), 86–95. This work was performed in part when the second author was with the University of Kiel and with McMaster University. This research was supported in part by the DFG - Graduiertenkolleg, Effiziente Algorithmen und Mehrskalenmethoden; by the EU Thematic Network APPOL I + II, Approximation and Online Algorithms, IST-1999-14084 and IST-2001-32007; by the EU Research Training Network ARACNE, Approximation and Randomized Algorithms in Communication Networks, HPRN-CT-1999-00112; by the EU Project CRESCCO, Critical Resource Sharing for Cooperation in Complex Systems, IST-2001-33135. The second author was also supported by an MITACS grant of Canada; by the NSERC Discovery Grant DG 5-48923; and by the NSERC Industrial Research and Development Fellowship.
* Corresponding author.
Email addresses: kj@informatik.uni-kiel.de (Klaus Jansen), hu.zhang@cibc.ca (Hu Zhang).

makespan $C_{\max}$ (maximum completion time).

In our model we do not allow preemption. There is also no release dates or due dates of the jobs, though the precedence constraints define the sequence of jobs linked by data flow. Furthermore, we define by a *critical path* in a schedule a path in G that has the largest total processing time of vertices along it.

Prasanna et al. [22–24] proposed a model of the malleable tasks. In their model, for each malleable task, the processing time is non-increasing in the number of processors allotted. In addition, a *speedup* function $s_j(l)$ for a malleable task J_j that is defined as the processing time $p_j(1)$ on one processor divided by the processing time $p_j(l)$ on l processors is concave in l. Their model has already been applied to the very massively parallel MIT Alewife machine [1,21]. However, their model allows non-integral numbers of processors. We use their model to obtain two natural assumptions for malleable tasks to obtain an approximation algorithm with a ratio 3.291919. The best know approximation ratio was 5.236 by Lepère et al. [17] and is recently improved to 4.730598 [13]. In our model we use a stronger assumption on the processing times than in [17,13].

In this paper, we consider the discrete model based on [22–24]. We assume that $p_j(0) = \infty$ as any task J_j cannot be executed if there is no processor available. For the processing time, we have the following assumptions:

Assumption 1 *The processing time $p_j(l)$ of a malleable task J_j is non-increasing in the number l of the processors allotted to it, i.e.,*

$$p_j(l) \leq p_j(l'), \text{for } l \geq l'; \tag{1}$$

Assumption 2 *The speedup function $s_j(l) = p_j(1)/p_j(l)$ of a malleable task J_j is concave in the number l of the processors allotted to it, i.e., for any $0 \leq l'' \leq l \leq l' \leq m$,*

$$\frac{p_j(1)}{p_j(l)} = s_j(l) \geq \frac{1}{l'-l''}[(l-l'')s_j(l') - (l-l')s_j(l'')] = \frac{p_j(1)}{l'-l''}\left[\frac{l-l''}{p_j(l')} - \frac{l-l'}{p_j(l'')}\right]. \tag{2}$$

It is worth noting that our model is realistic and practical. Assumption 1 indicates that more processing power results from more processors allotted such that the malleable task cannot be executed longer. Furthermore, Assumption 2 implies that the increase of processors allotted leads to increasing amount of communication, synchronization and scheduling overhead, such that the speedup effect cannot be linear. A typical example is that the processing time $p(l) = p(1)l^{-d_j}$, where l is the number of processors and $0 < d_j < 1$ (similar to the continuous case in [22–24]).

Another discrete model of malleable tasks was addressed in [2]. In their model,

there are also two assumptions for the malleable tasks. The first one is same as our Assumption 1, and their second assumption is as follows:

Assumption 2′ *The work function $W_j(l) = lp_j(l)$ of a malleable task J_j is non-decreasing in the number l of processors allotted to it, i.e.,*

$$W_j(l) \leq W_j(l'), \text{for } l \leq l'. \tag{3}$$

In Section 2 we show that our Assumption 2 implies Assumption 2′ for the work function. Furthermore, we also show that under our Assumption 2 the work function is convex in processing time.

A task J_j in any schedule is characterized by two values: the starting time τ_j and the number of processors l_j allotted to task J_j. A task J_j is called *active* during the time interval from its starting time τ_j to its completion time $C_j = \tau_j + p_j(l_j)$. A schedule is *feasible* if at any time t, the number of active processors does not exceed the total number of processors $\sum_{j:t\in[\tau_j,C_j)} l_j \leq m$, and if the precedence constraints $\tau_i + p_i(l_i) \leq \tau_j$, are fulfilled for all $i \in \Gamma^-(j)$, where $\Gamma^-(j)$ is the set of predecessors of task J_j.

Related work: The problem of scheduling independent malleable tasks (without precedence constraints) is strongly $\mathcal{NP}$-hard even for only 5 processors [5]. Approximation algorithms for the problem of scheduling independent malleable tasks with a ratio 2 was addressed in [7,18]. This was improved to $\sqrt{3}+\varepsilon$ by Mounié et al. [19], and further to $3/2+\varepsilon$ [20]. For the case of fixed m, Jansen and Porkolab proposed a PTAS [11]. If all $p(l) \leq 1$, for arbitrary m an AFPTAS was given by Jansen [10]. If the number of processors is polynomially bounded, recently Jansen and Thöle [12] presented a $(1+\varepsilon)$-approximation algorithm without monotonic assumption. If all processors assigned to a job must have contiguous addresses, the best-known approximation ratio is $3/2+\varepsilon$ [12].

Du and Leung [5] showed that scheduling malleable tasks with precedence constraints is strongly $\mathcal{NP}$-hard for $m = 3$. Furthermore, there is no polynomial time algorithm with approximation ratio less than $4/3$, unless $\mathcal{P}=\mathcal{NP}$[15]. If the precedence graph is a tree, a $(4+\varepsilon)$-approximation algorithm was developed in [16]. The idea of the two-phase algorithms was proposed initially in [16] and further used in [17] to obtain an improved approximation algorithm for general precedence constraints with a ratio $3+\sqrt{5} \approx 5.236$. In [17] the ratio was improved to $(3+\sqrt{5})/2 \approx 2.618$ when the precedence graph is a tree. Recently Jansen and Zhang [13] obtained the best known approximation ratio 4.730598 for general precedence constraints. The ratio 2.618 (4.730598, respectively) for tree precedence constraints (general precedence constraints, respectively) was shown tight in [26]. All above results are based on the model

under Assumption 1 and 2′. There is no result with our model under Assumption 1 and 2 yet. More details on the problem of scheduling independent or precedence constrained malleable tasks can be found in [6].

Our contribution: In this paper, we first analyze our model. We show that under Assumption 1 and 2, the work function is non-decreasing in the number of processors and is convex in the processing time. The first property is indeed the Assumption 2′ on work function in [17,13,28], which also shows that our model is a special case of the model in [17,13,28]. Then we develop an approximation algorithm for scheduling malleable tasks with precedence constraints for our model. Similar to [17,13,28], our algorithm is also a two-phase algorithm. In the first phase we solve an *allotment problem* approximately. For a given set of malleable tasks, the goal of the allotment problem is to find an allotment $\alpha : V \to \{1, \ldots, m\}$ deciding the numbers of processors allotted to execute the tasks such that the maximum between both opposite criteria of the critical path is minimized. In [17,13,28] the allotment problem is formulated as a bicriteria version of the discrete time-cost tradeoff problem, and the approximation algorithm in [25] is employed with a binary search procedure. In [17] a parameter $\rho = 1/2$ is fixed for the rounding strategy applied for the fractional solution, while in [13] ρ is not set as $1/2$ to obtain an improved result. However, here we do not apply their strategy of reducing the allotment problem to the discrete time-cost tradeoff problem. We just construct a piecewise linear work function according to the discrete values of work and processing times. With respect to the precedence constraints we are able to develop a piecewise linear program. Furthermore, since the work function is convex in the processing time, we are able to formulate the piecewise linear program as a linear program. We include also some additional constraints to avoid the binary search. Next we apply a new rounding technique for the (fractional) optimal solution to the linear program. The rounding procedure yields a feasible solution of the allotment problem with an approximation ratio depending on our rounding parameter $\rho \in [0, 1]$. In the second phase a variant of the list scheduling algorithm is employed to generate a new allotment and to schedule all tasks according to the precedence constraints. By studying the structure of the resulting schedule, we show that the approximation ratio is bounded by the optimal objective value of a min-max nonlinear program. Exploiting the solution to the min-max nonlinear program, we prove that the approximation ratio of our algorithm is not more than $100/63+100(\sqrt{6469}+13)/5481 \approx 3.291919$. This ratio is much better than all previous results for the general model in [17,13,28]. We also study the asymptotic behaviour of the solution to the min-max nonlinear program and show that the asymptotic best ratio is 3.291913.

The paper is organized as follows: The properties of the work function are studied in Section 2. In Section 3 our algorithm is presented, which is analyzed in Section 4.

2 Properties of the work function

In this section, we shall study the properties of work functions of the malleable task system according to the two assumptions in Section 1. We show that the assumptions lead to the second monotonic penalty assumption on work functions in [2,17,13,28]. Furthermore, we also show that the work functions are convex in the processing time.

Theorem 2.1 *For any malleable task J_j and m identical processors, if Assumption 2 for the processing time $p_j(l)$ of J_j holds for all $l = 0, \ldots, m$, then the work function $W_j(l) = lp_j(l)$ for task J_j is non-decreasing in l, i.e., $W_j(l') \geq W_j(l)$, for any integers $1 \leq l \leq l' \leq m$.*

Proof: We prove the theorem by induction. First, from the Assumption 2, we have that

$$\frac{1}{p_j(1)} \geq \frac{1}{2}\left[\frac{1}{p_j(2)} + \frac{1}{p_j(0)}\right].$$

Because $p_j(0) = \infty$, it holds that

$$\frac{1}{p_j(1)} \geq \frac{1}{2p_j(2)},$$

i.e., $2p_j(2) \geq p_j(1)$. Now we assume that the claimed inequality holds for $l-1$, i.e., $(l-1)p_j(l-1) \leq lp_j(l)$. Again, from Assumption 2, we have

$$\frac{1}{p_j(l)} \geq \frac{1}{2}\left[\frac{1}{p_j(l+1)} + \frac{1}{p_j(l-1)}\right] \geq \frac{1}{2}\left[\frac{1}{p_j(l+1)} + \frac{l-1}{lp_j(l)}\right],$$

i.e.,

$$\frac{l+1}{lp_j(l)} \geq \frac{1}{p_j(l+1)},$$

which is equivalent to $lp_j(l) \leq (l+1)p_j(l+1)$. Then we conclude that for any $l = 1, \ldots, m-1$, $W_j(l) = lp_j(l) \leq (l+1)p_j(l+1) = W_j(l+1)$, which leads to a non-decreasing sequence $W_j(1), \ldots, W_j(m)$, and the proof is complete. □

Theorem 2.1 in fact implies Assumption 2′ on work functions in [2,17,13,28]. It also indicates that the communication overhead increases if more processors are allotted to one malleable task. The monotonic penalty assumption on work functions in [2,17,13,28] is only a sequel of our Assumption 2. However, it is

worth noting that Assumption 2 is only a sufficient condition of Assumption 2′. There exist instances that Assumption 1 and Assumption 2′ hold for the processing time, but Assumption 2 does not. For example, let processing time $p_j(l) = 1/(1-\delta+\delta l^2)$ for some $\delta \in (0, 1/(m^2+1))$. It can be verified that in this case the work $W_j(l)$ is still increasing in l, while the speedup function $s(l)$ is convex in l. In fact, if for any (fractional) $l \in [1, m]$, the speedup function $s(l)$ satisfies $s'(l) \geq 0$, $s''(l) \geq 0$, $s(l) \geq ls'(l)$ and $s(1) = 1$, then the corresponding processing time leads to Assumption 2′ but not Assumption 2. In addition, there are also many instances such that the speedup function is locally convex in l.

Furthermore, if we regard the work functions as functions in the corresponding processing time, i.e., $w_j(p_j(l)) = W_j(l)$, the following theorem shows that Assumption 1 and 2 leads to a nice property of the work functions:

Theorem 2.2 *If Assumptions 1 and 2 hold for any malleable task J_j for any $l = 1, \ldots, m$, then the work function $w_j(p_j(l))$ is convex in the processing time $p_j(l)$.*

Proof: According to Assumption 2, the speedup function $s_j(l)$ is concave in the number l of processors. Therefore in the diagram of the speed function $s_j(l)$ versus l, $s_j(l) \geq \bar{s}_j(l)$, where $\bar{s}_j(l)$ is the vertical coordinate of the intersection point of the straight line connecting points $(l'', s_j(l''))$ and $(l', s_j(l'))$ and the vertical straight line passing through point $(l, s_j(l))$, where $1 \leq l'' \leq l \leq l' \leq m$. Then we obtain the following inequality by calculating the value of $\bar{s}_j(l)$ and also (2):

$$fp_j(1)p_j(l) = s_j(l) \geq \bar{s}_j(l) = \frac{p_j(1)}{l' - l''} \left[\frac{l - l''}{p_j(l')} - \frac{l - l'}{p_j(l'')} \right].$$

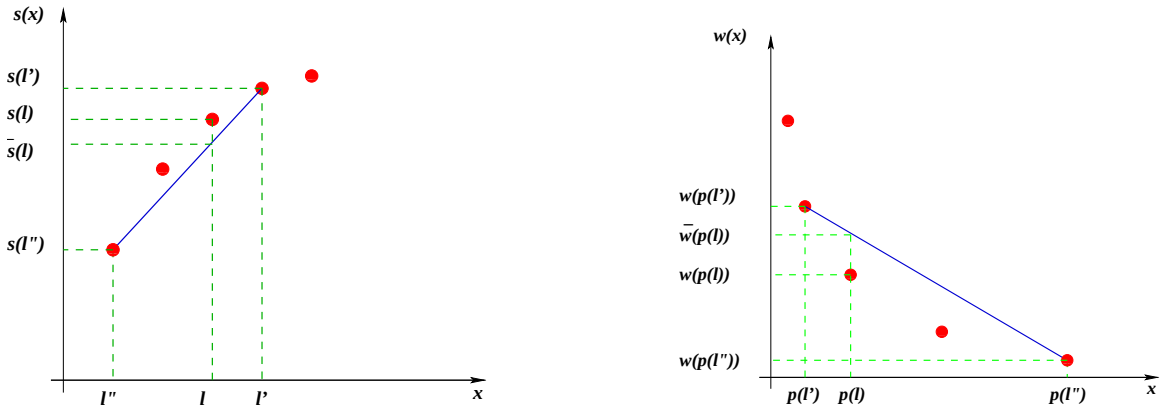
This is equivalent to

$$\frac{l}{p_j(l'')} - \frac{l}{p_j(l')} \geq \frac{l'}{p_j(l'')} - \frac{l''}{p_j(l')} - \frac{l' - l''}{p_j(l)}.$$

Multiplying both sides by the positive factor $p_j(l)p_j(l')p_j(l'')$ yields

$$lp_j(l)[p_j(l') - p_j(l'')] \geq p_j(l)[l'p_j(l') - l''p_j(l'')] - (l' - l'')p_j(l')p_j(l''). \quad (4)$$

We now consider the diagram of the work function $w_j(p_j(l))$ versus processing time $p_j(l)$. The straight line connecting points $(p_j(l''), w_j(p_j(l'')))$ and $(p_j(l'), w_j(p_j(l')))$ and the vertical straight line passing through point $(p_j(l), w_j(p_j(l)))$



$s_j(l)$ $w_j(p_j(l))$

Fig. 1. The diagrams of the speedup function $s_j(l)$ and the work function $w_j(p_j(l))$.

intersect at one point which has the vertical coordinate $\bar{w}_j(p_j(l))$ as follows:

$$\begin{aligned}
\bar{w}_j(p_j(l)) &= w_j(p_j(l'')) + \frac{p_j(l) - p_j(l'')}{p_j(l') - p_j(l'')}[w_j(p_j(l')) - w_j(p_j(l''))] \\
&= l''p_j(l'') + \frac{p_j(l) - p_j(l'')}{p_j(l') - p_j(l'')}[l'p_j(l') - l''p_j(l'')] \\
&= \frac{1}{p_j(l') - p_j(l'')}\{p_j(l)[l'p_j(l') - l''p_j(l'')] - (l' - l'')p_j(l')p_j(l'')\} \\
&\geq lp_j(l) = w_j(p_j(l)),
\end{aligned} \tag{5}$$

where (4) is applied to obtain the inequality. The inequality (5) shows that the work function $W_j(l) = w_j(p_j(l))$ is convex in processing time $p_j(l)$. □

A typical example of such a malleable task system is an instance with n malleable tasks as follows. Their processing times are $p_j(l) = p_j(1)l^{-d_j}$ for $j = 1, \ldots, n$, where $0 < d_j < 1$. This is similar to the example in [22–24]. However, it is worth noting that in their model the number of processors can be any real in $[0, m]$ and they only explored the fractional case. On the contrary, we study the different model with integral numbers of processors and discrete processing times.

3 Approximation algorithm

Different from the algorithms in [22–24], we here propose a two-phase approximation algorithm for scheduling malleable tasks with precedence constraints. Our algorithm is similar to those in [17,13]. But in the first phase, instead of solving a discrete time-cost tradeoff problem approximately, we solve a linear program. The fractional solution is then rounded to a feasible solution to the

LIST (J, m, α', μ) /*μ *defined in (24)*/*

initialization: allot $l_j = \min\{l'_j, \mu\}$ processors to task J_j, for $j \in \{1, \ldots, n\}$;

$SCHEDULED = \emptyset$;

if $SCHEDULED \neq J$ **then**

$READY = \{J_j | \Gamma^-(j) \subseteq SCHEDULED\}$;

compute the earliest possible starting time under α for all tasks in $READY$;

schedule the task $J_j \in READY$ with the smallest earliest starting time;

$SCHEDULED = SCHEDULED \cup \{J_j\}$;

end

Table 1
Algorithm **LIST**

allotment problem. In the second phase, we apply a variant of list scheduling algorithm to generate a feasible schedule. The algorithm is outlined as follows.

In the initialization step, we compute the values of the rounding parameter ρ and the allotment parameter μ depending on the input m (See Section 4 for the formulae).

In the first phase, we develop a linear program. By rounding its fractional solution with the parameter $\rho \in [0, 1]$ we are able to obtain a feasible allotment α' such that each task J_j is allotted l'_j number of processors (See Subsection 3.1 for details).

In the second phase, with the resulting allotment α' and the pre-computed allotment parameter μ, the algorithm generates a new allotment α and runs **LIST**, a variant of the list scheduling algorithm, in Table 1 (as proposed in [8,17]) and a feasible scheduling is delivered for the instance.

3.1 The first phase of the algorithm

In this subsection, we describe the linear program formulation and the rounding technique in the first phase of our algorithm.

Denote by x_j the (fractional) processing time of task J_j. For the discrete work function $w_j(p_j(l)) = W_j(l) = lp_j(l)$ in processing time, we define a continuous

piecewise linear work function $w_j(x_j)$ as follows:

$$w_j(x_j) = \begin{cases} w_j(p_j(l)), & \text{if } x = p_j(l), l = 1, \ldots, m; \\ \dfrac{w_j(p_j(l+1)) - w_j(p_j(l))}{p_j(l+1) - p_j(l)} x_j & \text{if } x \in (p_j(l+1), p_j(l)), \\ + \dfrac{w_j(p_j(l))p_j(l+1) - w_j(p_j(l+1))p_j(l)}{p_j(l+1) - p_j(l)}, & \text{and } l = 1, \ldots, m-1. \end{cases} \tag{6}$$

In addition, in any schedule, we know that the makespan (maximum completion time) is an upper bound of the critical path length L and the total work W divided by m, i.e., $\max\{L, W/m\} \leq C_{\max}$. In the first phase of our algorithm, we solve the following piecewise linear program:

$$\begin{aligned} &\min\ C \\ &\text{s.t. } C_i + x_j \leq C_j, && \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\ &\quad 0 \leq C_j \leq L, && \text{for all } j; \\ &\quad L \leq C; \\ &\quad W/m = \textstyle\sum_{j=1}^{n} w_j(x_j)/m \leq C; \\ &\quad x_j \in [p_j(m), p_j(1)], && \text{for all } j. \end{aligned} \tag{7}$$

In (7) the first set of constraints come from the precedence constraints, while the second set of constraints indicate that all tasks finish by the length L of a critical path. The goal is to minimize the makespan C.

We notice that the functions $w_j(x_j)$ are piecewise linear and it is a crucial issue to replace them by linear functions. According to Theorem 2.2, the discrete work function $w_j(p_j(l))$ is convex in processing time $p_j(l)$. Therefore the continuous work function $w_j(x_j)$ is also convex in the fractional processing time x_j. Since $w_j(x_j)$ is piecewise linear, it can be written as follows:

$$\begin{aligned} w_j(x_j) &= \max_{l \in \{1,\ldots,m-1\}} \left\{ \frac{w_j(p_j(l+1)) - w_j(p_j(l))}{p_j(l+1) - p_j(l)} x_j \right. \\ &\qquad \left. + \frac{w_j(p_j(l))p_j(l+1) - w_j(p_j(l+1))p_j(l)}{p_j(l+1) - p_j(l)} \right\} \\ &= \max_{l \in \{1,\ldots,m-1\}} \left\{ \frac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} x_j - \frac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \right\}, \end{aligned} \tag{8}$$

for any $x_j \in [p_j(m), p_j(1)]$. Thus we are able to modify the piecewise linear

program (7) to following linear program:

$$
\begin{aligned}
&\min\ C \\
&\text{s.t. } C_i + x_j \le C_j, && \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\
&\quad 0 \le C_j \le L, && \text{for all } j; \\
&\quad \frac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} x_j - \frac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \le \bar{w}_j, && \text{for } l = 1, \ldots, m-1 \text{ and all } j; \\
&\quad L \le C; \\
&\quad W/m = \textstyle\sum_{j=1}^{n} \bar{w}_j/m \le C; \\
&\quad x_j \in [p_j(m), p_j(1)], && \text{for all } j.
\end{aligned} \tag{9}
$$

The size of (9) is polynomial in m and n. Thus it is polynomial time solvable.

Remark: Here the linear programs (7) and (9) are not of the most straightforward form in scheduling, which contains assignment variable $x_{j,l}$ indicating that task J_j is allotted with l processors. In this way, a linear programming relaxation of the scheduling problem is as follows:

$$
\begin{aligned}
&\min\ C \\
&\text{s.t. } C_i + \textstyle\sum_{l=1}^{m} x_{j,l} p_j(l) \le C_j, && \text{for all } i \in \Gamma^-(j) \text{ and all } j; \\
&\quad C_j \le C, && \text{for all } j; \\
&\quad \textstyle\sum_{j=1}^{n} \sum_{l=1}^{m} x_{j,l}(lp_j(l)) \le mC; \\
&\quad \textstyle\sum_{l=1}^{m} x_{j,l} = 1, && \text{for all } j; \\
&\quad x_{j,l} \ge 0, && \text{for all } j \text{ and all } l.
\end{aligned} \tag{10}
$$

For every task J_j, the optimal solution x_j^* to (7) will have the form $\xi p_j(l) + (1-\xi)p_j(l+1)$ for some $\xi \in [0,1]$. Setting $x_{j,l} = \xi$ and $x_{j,l+1} = 1-\xi$ yields a feasible solution to (10). Now consider an optimal solution to (10). We claim that for every job J_j, at most two $x_{j,l}$ are non-zero, and they are adjacent (i.e. some l and $l+1$). Otherwise, we first consider the case there are only two non-zero non-adjacent elements in an optimal solution for some jobs. Assume for task J_j there exist $k < l-1$ such that $x_{j,k} > 0$ and $x_{j,l} > 0$. Then there is a $\xi \in [0,1]$ such that $p_j(k+1) = \xi p_j(k) + (1-\xi)p_j(l)$ because of Assumption 1. In addition, according to Theorem 2.2, $(k+1)p_j(k+1) \le \xi k p_j(k) + (1-\xi) l p_j(l)$. On the other hand, there is a $\delta > 0$ such that one of $x_{j,k} - \delta\xi$ and $x_{j,l} - \delta(1-\xi)$ becomes 0 and the other one remains non-negative. We also increase $x_{j,k+1}$ by δ, and denote by $x'_{j,k} = x_{j,k} - \delta\xi$, by $x'_{j,k+1} = x_{j,k+1} + \delta$, and by $x'_{j,l} = x_{j,l} - \delta(1-\xi)$. Keeping all other assignment variables unchanged as in the optimal solution,

we have obtained a new solution fulfilling the last two constraints of (10). As for the processing time, we have

$$\begin{aligned}
&x_{j,k}p_j(k) + x_{j,k+1}p_j(k+1) + x_{j,l}p_j(l) \\
&= (x_{j,k} - \delta\xi)p_j(k) + \delta\xi p_j(k) + x_{j,k+1}p_j(k+1) + (x_{j,l} - \delta(1-\xi))p_j(l) + \delta(1-\xi)p_j(l) \\
&= x'_{j,k}p_j(k) + x'_{j,k+1}p_j(k) + x'_{j,l}p_j(l),
\end{aligned}$$

which shows that the total processing time for task J_j is the same as the original solution. As for the work of J_j, we have

$$\begin{aligned}
&x_{j,k}kp_j(k) + x_{j,k+1}(k+1)p_j(k+1) + x_{j,l}lp_j(l) \\
&= x'_{j,k}kp_j(k) + x_{j,k+1}(k+1)p_j(k+1) + x'_{j,l}p_j(l) + \delta(\xi kp_j(k) + (1-\xi)lp_j(l)) \\
&\geq x'_{j,k}kp_j(k) + x'_{j,k+1}(k+1)p_j(k+1) + x'_{j,l}p_j(l),
\end{aligned}$$

which shows that the total work will not increase in the new solution. If the total work of the original solution is greater than that of the new solution and the constraint that the total work is bounded by mC is tight, then this is a contradiction that the original solution is optimal. Otherwise, we consider the case that the total work does not change. If $x'_{j,l} = 0$ we are done because we have only $x'_{j,k} \geq 0$ and $x'_{j,k+1} \geq 0$. If $x'_{j,k} = 0$, we repeat the above procedure for $x'_{j,k+1}$ and $x'_{j,l}$ until we reach some solution y such that only $y_{j,k'} \geq 0$ and $y_{j,k'+1} \geq 0$ for some k'. Finally, if there are more than two non-zero elements in an optimal solution, we sort the indices of non-zero elements in the increasing order such that $k_1 < \ldots < k_\alpha$ where only $x_{j,k_1}, \ldots, x_{j,k_\alpha}$ are non-zero. Then we perform the above construction procedure for the pair of elements x_{j,k_1} and x_{j,k_α}. Repeating this procedure we will again obtain at most two adjacent non-zero elements in an optimal solution. It is worth noting that the above construction just needs at most mn iterations, each visit a group of three variables. Combining all cases completes the proof of our claim that for every job J_j, there are at most two non-zero $x_{j,l}$ and $x_{j,l+1}$ in an optimal solution to (10). Denote by $x^*_j = x_{j,l}p_j(l) + x_{j,l+1}p_j(l+1)$, which is a feasible solution to (7). Therefore, the new linear program (10) is equivalent to our linear program (7).

Now we consider the rounding strategy. For a task J_j, denote by x^*_j the corresponding optimal solution. Suppose that $x^*_j \in (p_j(l+1), p_j(l))$. In the interval $[p_j(l+1), p_j(l)]$, we define a critical point l_c such that $l_c = l + 1 - \rho$ for the rounding parameter $\rho \in [0, 1]$. The processing time $p_j(l_c)$ is defined as $p_j(l_c) = p_j(l+1-\rho) = \rho p_j(l) + (1-\rho)p_j(l+1)$, and its work is defined as $w_j(p_j(l_c)) = (1-\rho)w_j(p_j(l+1)) + \rho w_j(p_j(l)) = (1-\rho)(l+1)p_j(l+1) + \rho l p_j(l)$.

We apply the following rounding technique for the fractional solution to (9): If

$x_j^* \geq p_j(l_c)$ it will be rounded up to $p_j(l)$, and otherwise rounded down to $p_j(l+1)$. With the rounded solution of the processing time in $\{p_j(m), \ldots, p_j(1)\}$ we are able to identify an l'_j such that $p_j(l'_j)$ equals the rounded solution. Then we develop an allotment α' for all jobs where each job J_j is allotted a number l'_j of processors.

Remark: In the first phase of the algorithms in [17,13], the allotment problem is approximately solved by employing the approximation algorithm for the discrete time-cost tradeoff problem in [25]. Here we avoid the complicated procedure to reduce the problem to a large number of instances of linear time-cost tradeoff problem with only two durations in [25]. Furthermore, we here directly embed the critical path length L and the total work W into (9) to avoid the binary search procedure in [17].

4 Analysis of the algorithm

We shall show the approximation ratio of our algorithm. Denote by L, W, $C_{\max}$ and by L', W', $C'_{\max}$ the critical path lengths, the total works and the makespans of the final schedule delivered by our algorithm and the schedule corresponding to the allotment α' generated in the first phase, respectively. Furthermore, we denote by $C^*_{\max}$ the optimal objective value of (9), and L^*, W^* the (fractional) optimal critical path length and the (fractional) optimal total work in (9). It is worth noting that here $W' = \sum_{j=1}^{n} l'_j p_j(l'_j)$. Denote by OPT the overall optimal makespan (over all feasible schedules with integral number of processors allotted to all tasks). It is obvious that

$$\max\{L^*, W^*/m\} \leq C^*_{\max} \leq OPT. \tag{11}$$

In allotments α and α', a task J_j is allotted l_j and l'_j processors, and their processing times are $p_j(l_j)$ and $p_j(l'_j)$, respectively. In the optimal (fractional) solution to (9), each task J_j has a fractional processing time x_j^*. We define the fractional number of processors allotted as follows:

$$l_j^* = w_j(x_j^*)/x_j^*. \tag{12}$$

According to this definition, l_j^* has the following property:

Lemma 4.1 *For any malleable task J_j, if $p_j(l+1) \leq x_j^* \leq p_j(l)$ for some $l \in \{1, \ldots, m-1\}$, then $l \leq l_j^* \leq l+1$.*

Proof: Suppose that $p_j(l+1) \le x_j^* \le p_j(l)$, according to (6) or (8), we can calculate the value of l_j^* as follows:

$$\begin{aligned} l_j^* = \frac{w_j(x_j^*)}{x_j^*} &= \frac{(l+1)p_j(l+1) - lp_j(l)}{p_j(l+1) - p_j(l)} - \frac{p_j(l)p_j(l+1)}{p_j(l+1) - p_j(l)} \frac{1}{x_j^*} \\ &= l + \frac{p_j(l+1)}{p_j(l) - p_j(l+1)} \left[\frac{p_j(l)}{x_j^*} - 1 \right]. \end{aligned} \tag{13}$$

Since $p_j(l) \ge x_j^*$, we have $p_j(l)/x_j^* \ge 1$ and $l_j^* \ge l$. From $p_j(l+1) \le x_j^*$, by multiplying both sides by $p_j(l)$ and subtracting $x_j^* p_j(l+1)$ from both sides, we obtain that $[p_j(l) - x_j^*]p_j(l+1) \le x_j^*[p_j(l) - p_j(l+1)]$, i.e.,

$$\frac{p_j(l+1)}{p_j(l) - p_j(l+1)} \left[\frac{p_j(l)}{x_j^*} - 1 \right] \le 1.$$

Thus $l_j^* \le l+1$ and the lemma is proved. □

Lemma 4.1 shows that l_j^* is well defined. It is worth noting that the rounded integral number of processors $l_j' \in [l, l+1] = [\lfloor l_j^* \rfloor, \lceil l_j^* \rceil]$ according to the rounding approach in our algorithm.

We now consider the influence of the rounding procedure in the first phase to the change of the duration and the work of any malleable task.

Lemma 4.2 *For any job J_j, in the allotment α' its processing time $p_j(l_j') \le 2x_j^*/(1+\rho)$ and the its work $w_j(p_j(l_j')) = l_j' p_j(l_j') \le 2l_j^* x_j^*/(2-\rho) = 2w_j(x_j^*)/(2-\rho)$.*

Proof: Suppose $x_j^* \in (p_j(k+1), p_j(k))$. In the interval $[p_j(k+1), p_j(k)]$, the critical point is $k_c = k + 1 - \rho$. Its processing time is $p_j(k_c) = p_j(k+1-\rho) = \rho p_j(k) + (1-\rho)p_j(k+1)$ and its work is $w_j(p_j(k_c)) = w_j(p_j(k+1-\rho)) = (1-\rho)(k+1)p_j(k+1) + \rho k p_j(k)$. We consider the following two cases.

In the first case, $x_j^* \ge p_j(k_c)$. In the rounding procedure the processing time is rounded up to $p_j(k)$, and the fractional number of processors is reduced to $l_j' = k$. Therefore the work is also reduced due to Theorem 2.1. However, the increase of the processing time is

$$\begin{aligned} \frac{p_j(l_j')}{x_j^*} &\le \frac{p_j(k)}{p_j(k_c)} = \frac{p_j(k)}{\rho p_j(k) + (1-\rho)p_j(k+1)} \\ &\le \frac{p_j(k)}{\rho p_j(k) + (1-\rho)k p_k(k)/(k+1)} = \frac{k+1}{k+\rho}. \end{aligned}$$

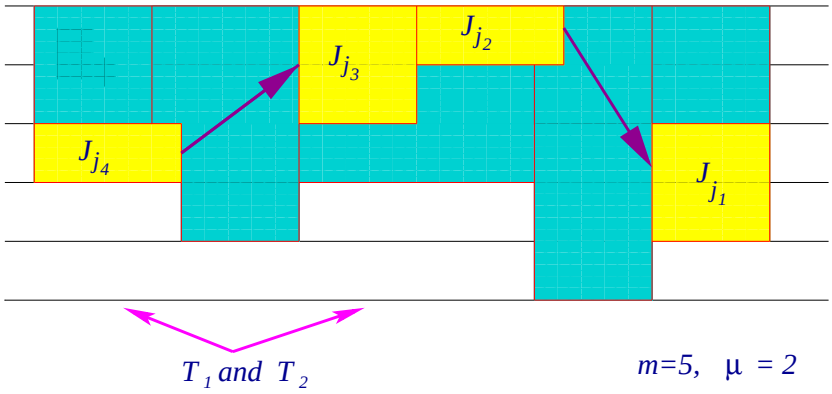


Fig. 2. An example of the "heavy" path.

The second inequality holds also from Theorem 2.1, $kp_j(k) \leq (k+1)p_j(k+1)$. In the second case, $x_j^* < p_j(k_c)$. In the rounding the processing time is rounded down to $p_j(k+1)$, and the fractional number of processors is increased to $l'_j = k+1$. In addition, it is also easy to verify that $w_j(x)$ is non-increasing in x. So $w_j(x_j^*) \geq w_j(p_j(k_c))$. Since more processors are allotted, according to Theorem 2.1 the work increases by the following factor:

$$\frac{w_j(p_j(l'_j))}{w_j(x_j^*)} \leq \frac{(k+1)p_j(k+1)}{w_j(p_j(k_c))} = \frac{(k+1)p_j(k+1)}{(1-\rho)(k+1)p_j(k+1)+\rho k p_j(k)}$$
$$\leq \frac{(k+1)p_j(k+1)}{(1-\rho)(k+1)p_j(k+1)+\rho k p_j(k+1)} = \frac{k+1}{k+1-\rho}.$$

Since k is an integer, the above two factors can be further bounded by $(k+1)/(k+1-\rho) \leq 2/(2-\rho)$ and $(k+1)/(k+\rho) \leq 2/(1+\rho)$. This means that after the first phase, for each task J_j, the processing time increases by at most a factor of $2/(1+\rho)$ and the work increases by at most $2/(2-\rho)$. □

Same as [17,13,28], in the final schedule, the time interval $[0, C_{\max}]$ consists of three types of time slots. In the first type of time slots, at most $\mu - 1$ processors are busy. In the second type of time slots, at least μ and at most $m - \mu$ processors are busy. In the third type at least $m - \mu + 1$ processors are busy. Denote the sets of the three types time slots by T_1, T_2 and T_3, and $|T_i|$ the overall lengths for $i \in \{1, 2, 3\}$. In the case that $\mu = (m+1)/2$ for m odd, $T_2 = \emptyset$. In other cases all three types of time slots may exist. Then we have the following bound on $|T_1|$ and $|T_2|$:

Lemma 4.3 $(1+\rho)|T_1|/2 + \min\{\mu/m, (1+\rho)/2\}|T_2| \leq C_{\max}^*$.

Proof: We construct a "heavy" directed path $\mathcal{P}$ in the final schedule, similar to [17,13,28]. The last task in the path $\mathcal{P}$ is any multiprocessor task J_{j_1} that completes at time $C_{\max}$ (the makespan of the final schedule). After we have

defined the last $i \geq 1$ tasks $J_{j_i} \to J_{j_{i-1}} \to \cdots \to J_{j_2} \to J_{j_1}$ on the path $\mathcal{P}$, we can determine the next task $J_{j_{i+1}}$ as follows: Consider the latest time slot t in $T_1 \cup T_2$ that is before the starting time of task J_{j_i} in the final schedule. Let V' be the set of task J_{j_i} and its predecessor tasks that start after time t in the schedule. Since during time slot t at most $m - \mu$ processors are busy, and since at most μ processors are allotted to any task in V', no task in V' is ready for execution during the time slot t. Therefore for every task in V' some predecessor is being executed during the time slot t. Then we select any predecessor of task J_{j_i} that is running during slot t as the next task $J_{j_{i+1}}$ on the path $\mathcal{P}$. This search procedure stops when $\mathcal{P}$ contains a task that starts before any time slot in $T_1 \cup T_2$. An example of the "heavy" path is illustrated in Figure 2. Now we examine the stretch of processing time for all jobs in $\mathcal{P}$ in the rounding procedure of the first phase and in the new allotment α of the second phase.

For any job J_j in $T_1 \cap \mathcal{P}$, the processing time of the fractional solution to (9) increases by at most a factor $2/(1+\rho)$. The processing time does not change in the second phase as in α' the job J_j is only allotted a number $l'_j \leq \mu$ of processors such that it can be in the time slot of T_1. Therefore for such kind of jobs we have $p_j(l_j) = p_j(l'_j) \leq 2x^*_j/(1+\rho)$.

For any job J_j in $T_2 \cap \mathcal{P}$, there are two cases. In the first case, in α' a job J_j is allotted $l'_j \leq \mu$ processors. This is same as the case before and we also have $p_j(l_j) \leq 2x^*_j/(1+\rho)$. In the second case, in α' a job J_j is allotted $l'_j > \mu$ processors, and $l_j = \mu$. Then there are two subcases according to the solution to (9). In the first subcase, in the fractional solution to (9) there are $l^*_j \geq \mu$ processors allotted. Since μ is an integer, we have $l^*_j \geq \lfloor l^*_j \rfloor \geq \mu \geq l_j$. Then $l_j p_j(l_j) = W_j(l_j) \leq W_j(\mu) \leq W_j(\lfloor l^*_j \rfloor) \leq w_j(x^*_j) = l^*_j x^*_j \leq W_j(\lceil l^*_j \rceil)$ due to Theorem 2.1 and (12). Because $l^*_j \leq m$, and $w_j(x^*_j) = x^*_j l^*_j \geq p_j(l_j) l_j = W_j(l_j)$, it holds that $p_j(l_j) \leq x^*_j l^*_j / l_j \leq x^*_j m/\mu$. In the second subcase, in the fractional solution there are $l^*_j < \mu$ processors allotted to J_j. Then in the rounding procedure of the first phase the processing time must be rounded down from x^*_j to $p_j(l'_j)$ as only in this way the assumption that $l'_j > \mu$ of this case can be satisfied. Then in the second phase, J_j is allotted μ processors and from Theorem 2.1, $p_j(l_j) l_j \leq p_j(l'_j) l'_j$. Since there are at most m processors allotted to J_j in α', we have $p_j(l_j) \leq p_j(l'_j) l'_j / l_j \leq p_j(l'_j) m/\mu \leq x^*_j m/\mu$. Therefore for any job J_j in $T_2 \cap \mathcal{P}$, $p_j(l_j) \leq x^*_j \max\{2/(1+\rho), m/\mu\}$.

With the construction of the direct path $\mathcal{P}$, it covers all time slots in $T_1 \cup T_2$ in the final schedule. In addition, because of Lemma 4.2, in the schedule resulted from the fractional solution to (9), the jobs processed in T_1 in the final schedule contribute a total length of at least $(1+\rho)|T_1|/2$ to $L^*(\mathcal{P})$, the length of the critical path $\mathcal{P}$ in the optimal solution. In addition, the tasks processed in T_2 contribute a total length of at least $|T_2| \min\{(1+\rho)/2, \mu/m\}$ to $L^*(\mathcal{P})$. Since the critical path $L^*(\mathcal{P})$ is not more than the makespan $C^*_{\max}$ according to (11),

we have proved the claimed inequality. □

In addition, we have the following bound on the makespan of the final schedule:

Lemma 4.4 *The makespan of the schedule delivered by our algorithm is bounded as follows:*

$$(m - \mu + 1)C_{\max} \leq 2mC^*_{\max}/(2 - \rho) + (m - \mu)|T_1| + (m - 2\mu + 1)|T_2|.$$

Proof: According to the definitions, all time slots in T_1, T_2 and T_3 in the final schedule cover the whole interval $[0, C_{\max}]$. Therefore

$$C_{\max} = |T_1| + |T_2| + |T_3|. \tag{14}$$

In addition, as during the time slots of the first (respectively the second and the third) type at least one (respectively μ and $m-\mu+1$) processors are busy, a lower bound on the total work in the final schedule is:

$$W \geq |T_1| + \mu|T_2| + (m - \mu + 1)|T_3|. \tag{15}$$

Multiplying (14) by $m - \mu + 1$ and subtracting (15) from it yield

$$(m - \mu + 1)C_{\max} \leq W + (m - \mu)|T_1| + (m - 2\mu + 1)|T_2|. \tag{16}$$

In the second phase, for any job J_j, the allotted number of processors l_j is not more than l'_j, the number of processors in the first phase. Therefore according to Theorem 2.1 the total work is non-increasing, i.e., $W' \geq W$. According to Lemma 4.2, in the rounding procedure of the first phase, the total work only increases by at most a factor of $2/(2-\rho)$ from the total work of the fractional solution to (9). In this case we have that $W' \leq 2W^*/(2 - \rho)$. Furthermore, from (11), $W \leq 2W^*/(2-\rho) \leq 2mC^*_{\max}/(2-\rho)$. Substituting it to the bound on $C_{\max}$ in (16) we obtain the claimed inequality. □

Define the normalized overall length of the i-th type of time slots by $x_i = |T_i|/C^*_{\max}$ for $i = 1, 2, 3$. Thus we are able to obtain a min-max nonlinear program as follows where its optimal value is an upper bound on the approximation ratio r:

Lemma 4.5 *The approximation ratio of our algorithm is bounded by the op-*

timal objective value of the following min-max nonlinear program

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \ & \frac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} \\ s.t. \ & (1+\rho)x_1/2 + \min\{\mu/m, (1+\rho)/2\}x_2 \le 1; \\ & x_1, x_2 \ge 0; \\ & \rho \in [0,1]; \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{17}$$

Proof: Dividing the inequalities in Lemma 4.3 and Lemma 4.4 by $C^*_{\max}$, together with inequality (11), the definitions of x_i and the definition of the approximation ratio r, we have the first constraint in (17) and the following inequality:

$$r = \sup \frac{C_{\max}}{OPT} \le \sup \frac{C_{\max}}{C^*_{\max}} = \max_{x_1,x_2} \frac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1}.$$

On the other hand, we can select appropriate μ and ρ to minimize the ratio r. Hence, by combining them together with the other constraints for the variables according to their definitions, the approximation ratio is the objective value of (17). □

In the following we shall solve the min-max nonlinear program (17) to get the best approximation ratio of our algorithm.

4.1 Analysis of the min-max nonlinear program (17)

In order to solve (17), we need to consider two cases that either $\rho \le 2\mu/m - 1$ or $\rho > 2\mu/m - 1$ to simplify the first constraint.

For the analysis we need the following properties of polynomials:

Proposition 4.1 *Suppose that x_1 is the largest real root to the equation $f(x) = \sum_{i=1}^{n} a_i x^i = 0$ ($a_n \neq 0$). If $a_n > 0$, then $f(x) > 0$ for $x \in (x_1, \infty)$. If $a_n < 0$, then $f(x) < 0$ for $x \in (x_1, \infty)$.*

Recall that the notation C^d means the class of all d-th order continuously differentiable functions. We need the following lemma for our analysis:

Lemma 4.6 *For two functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ defined on $[a, b]$ and $f(x)$, $g(x) \in C^1$, if one of the following two properties holds:*

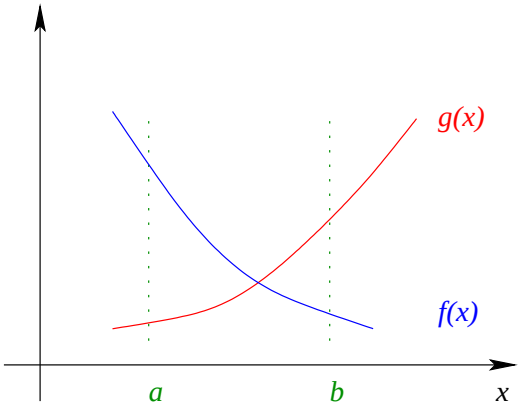


Fig. 3. An example of functions with property Ω_1 in Lemma 4.6.

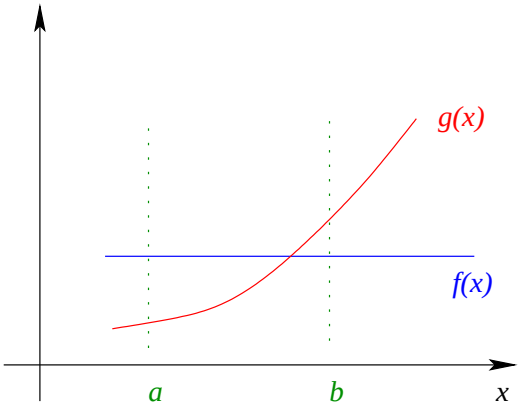


Fig. 4. An example of functions with property Ω_2 in Lemma 4.6.

Ω_1: $f'(x) \cdot g'(x) < 0$ *for all* $x \in [a, b]$;
Ω_2: $f'(x) = 0$ *and* $g'(x) \neq 0$ *for all* $x \in [a, b]$,

and the equation $f(x) = g(x)$ *has a solution in interval* $[a, b]$*, then the root* x_0 *is unique and it minimizes the function* $h(x) = \max\{f(x), g(x)\}$.

The proof of Lemma 4.6 is straightforward and is omitted here. Examples for properties Ω_1 and Ω_2 are illustrated in Figure 3 and 4, respectively.

4.1.1 Solve (17) for the case $\rho \leq 2\mu/m - 1$

In this case, to guarantee that $\rho \geq 0$ it is required that $m/2 \leq \mu \leq (m+1)/2$. Because μ is an integer, we have $\mu = m/2$ for m even and $\mu = (m+1)/2$ for m odd. In addition, as $(1+\rho)/2 \leq \mu/m$, it holds that $\rho = 0$ for m even and $\rho \leq 1/m$ for m odd. Therefore, we need to consider two cases depending on m.

- **CASE 1**: m even.

In this case, we need to solve the following linear program to determine the approximation ratio:

$$\begin{aligned} \max_{x_1,x_2} \ & \frac{2m + mx_1 + 2x_2}{m+2} \\ \text{s.t. } & x_1 + x_2 \leq 2; \\ & x_1, x_2 \geq 0. \end{aligned} \tag{18}$$

It is easy to see the following bound

$$\frac{2m + mx_1 + 2x_2}{m+2} \leq \frac{2m + (m-2)x_1 + 4}{m+2} \leq \frac{4m}{m+2}.$$

So the approximation ratio in this case is bounded by $4m/(m+2)$.

- **CASE 2**: m odd.

Thus we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_\rho \max_{x_1,x_2} \ & \frac{4m+(2-\rho)(m-1)x_1}{(2-\rho)(m+1)} \\ \text{s.t. } & (1+\rho)x_1/2+(1+\rho)x_2/2 \le 1; \\ & x_1, x_2 \ge 0; \\ & \rho \in [0, 1/m]. \end{aligned} \tag{19}$$

It is obvious that the objective function of (19) is bounded by the a function $A(\rho)$ as follows:

$$A(\rho) = \frac{2[(m+1)\rho+4m+2]}{(m+1)(1+\rho)(2-\rho)}.$$

Now we just need to minimize $A(\rho)$ for all $\rho \in [0, 1/m]$. The first order partial derivative of $A(\rho)$ with respect to ρ is:

$$A(\rho)'_\rho = \frac{2}{m+1}\frac{(m+1)\rho^2+4(2m-1)\rho-2(m-2)}{(1+\rho)^2(2-\rho)^2}.$$

We shall examine whether $A(\rho)'_\rho$ is nonnegative or not. It is obvious that the denominator is positive. The only positive root to equation $A(\rho)'_\rho = 0$ is

$$\bar{\rho} = \frac{3\sqrt{2m^2-2m}-2(2m-1)}{m+1}.$$

When $\rho < \bar{\rho}$, $A(\rho)$ is decreasing in ρ. Otherwise $A(\rho)$ is increasing in ρ. Furthermore, it can be verified that when $m = 3$ and $m = 5$, $\bar{\rho} < 1/m$. In this way we need to check two subcases as follows:

- **SUBCASE 1**: $m = 3$ or $m = 5$.

In this case, we set $\rho^* = \bar{\rho}$, and the corresponding minimum value $\min_\rho A(\rho) = A(\rho^*)$. For $m = 3$, $\min_\rho A(\rho) = 2(2+\sqrt{3})/3$, and for $m = 5$, $\min_\rho A(\rho) = 2(7+2\sqrt{10})/9$.

- **SUBCASE 2**: $m \ge 7$.

In this case, we set $\rho^* = 1/m$. Then the corresponding minimum value $\min_\rho A(\rho) = 2m(4m^2 - m + 1)/[(m+1)^2(2m-1)]$.

Combining all above cases we have the following lemma:

Lemma 4.7 *In the case that $\rho \leq 2\mu/m - 1$, the approximation ratio of our algorithm has the following bound:*

$$r \leq \begin{cases} 2(2+\sqrt{3})/3, & \text{if } m = 3; \\ 2(7+2\sqrt{10})/9, & \text{if } m = 5; \\ 2m(4m^2 - m + 1)/[(m+1)^2(2m-1)], & \text{if } m \geq 7 \text{ and } m \text{ odd}; \\ 4m/(m+2), & \text{otherwise.} \end{cases}$$

4.1.2 Solve (17) for the case $\rho > 2\mu/m - 1$

In this case we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} \\ \text{s.t.} \quad & (1+\rho)x_1/2 + \mu x_2/m \leq 1; \\ & x_1, x_2 \geq 0; \\ & \rho \in (\max\{2\mu/m - 1, 0\}, 1]; \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{20}$$

In this case we need to solve the following min-max nonlinear program:

$$\begin{aligned} \min_{\mu,\rho} \max_{x_1,x_2} \quad & \frac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} \\ \text{s.t.} \quad & (1+\rho)x_1/2 + \mu x_2/m \leq 1; \\ & x_1, x_2 \geq 0; \\ & \rho \in (\max\{2\mu/m - 1, 0\}, 1]; \\ & \mu \in \{1, \ldots, \lfloor (m+1)/2 \rfloor\}. \end{aligned} \tag{21}$$

We notice that the constraints on x_1 and x_2 in (21) form a triangle, and the extreme points are $E_1 : (x_1, x_2) = (2/(1+\rho), 0)$, $E_2 : (x_1, x_2) = (0, m/\mu))$, and $E_3 : (x_1, x_2) = (0, 0)$. Since (21) is linear in x_1 and x_2, for a fixed pair of ρ and μ, the maximum value of the objective function exists at one of the extreme points. It is clear that the objective function cannot attain the

maximum value at E_3. So we just consider E_1 and E_2. Denote by $A(\mu,\rho)$ and $B(\mu,\rho)$ the objective values at the E_1 and E_2, respectively. Then we have:

$$A(\mu,\rho) = \frac{2(1+\rho)m + 2(2-\rho)(m-\mu)}{(1+\rho)(2-\rho)(m-\mu+1)};$$
$$B(\mu,\rho) = \frac{2m\mu + (2-\rho)m(m-2\mu+1)}{\mu(2-\rho)(m-\mu+1)}.$$

The first order partial derivative of $A(\mu,\rho)$ with respect to μ is

$$A(\mu,\rho)'_\mu = \frac{2((1+\rho)m-(2-\rho))}{(1+\rho)(2-\rho)(m-\mu+1)^2}.$$

It is obvious that the denominator is always positive. The numerator is nonnegative when $(1+\rho)m-(2-\rho) \geq 0$, i.e., $\rho \geq (2-m)/(m+1)$. This inequality is always true because $\rho \geq 0$ and $m \geq 2$. Thus $A(\rho)'_\mu$ is always nonnegative. So for any $m \geq 2$, $A(\mu,\rho)$ is increasing in μ.

Furthermore, the first order partial derivative of $B(\mu,\rho)$ with respect to μ is

$$B(\mu,\rho)'_\mu = \frac{-2(1-\rho)m\mu^2 + 2(2-\rho)m(m+1)\mu - (2-\rho)m(m+1)^2}{(2-\rho)\mu^2(m-\mu+1)^2}.$$

When $\rho = 1$, $B(\mu,1)'_\mu = m(m+1)[2\mu-(m+1)]/\mu^2(m-\mu+1)^2 \leq 0$ because $\mu \leq (m+1)/2$. So $B(\mu,\rho)$ is decreasing in μ. Now we consider the case that $\rho < 1$. Solving the quadratic equation $B(\mu,\rho)'_\mu = 0$ we obtain that $\mu = (2-\rho \pm \sqrt{\rho(2-\rho)})(m+1)/2(1-\rho)$. Since $\rho < 1$, we have $1 > \rho(2-\rho)$. Because both sides are positive, we can take the square roots of both sides to obtain $1 > \sqrt{\rho(2-\rho)}$. Thus $2-\rho-\sqrt{\rho(2-\rho)} > 1-\rho$. So we obtain that $(2-\rho+\sqrt{\rho(2-\rho)})/(1-\rho) > (2-\rho-\sqrt{\rho(2-\rho)})/(1-\rho) > 1$. Therefore the roots of the equation $B(\mu,\rho)'_\mu = 0$ violate the constraint $\mu \leq (m+1)/2$. So there is no feasible root for this equation. Since in the numerator of $B(\mu,\rho)'_\mu$ the coefficient of the term of μ^2 is negative, we have $B(\mu,\rho)'_\mu < 0$ for all feasible pair ρ and μ from Proposition 4.1.

According to Lemma 4.6, if we find a solution to the equation $A(\mu,\rho) = B(\mu,\rho)$, the optimal value is attained. The equation $A(\mu,\rho) = B(\mu,\rho)$ can be simplified to $2\mu^2 - (4+2\rho)m\mu + (1+\rho)m(m+1) = 0$. The roots to this equation are $\mu = ((2+\rho)m \pm \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m})/2$. Since $m \geq 1$, we have $(2+\rho)m + \sqrt{(\rho^2+2\rho+2)m^2 - 2(1+\rho)m} \geq 2m + 2\rho m > m+1$, which violates the constraint that $\mu \leq (m+1)/2$. So the only feasible root to

the equation $A(\mu, \rho) = B(\mu, \rho)$ is

$$\mu^* = ((2+\rho)m - \sqrt{(\rho^2 + 2\rho + 2)m^2 - 2(1+\rho)m})/2. \tag{22}$$

Then we have the following lemma:

Lemma 4.8 *For a fixed $\rho > 2\mu/m - 1$, the optimal objective value is attained when $\mu = ((2+\rho)m - \sqrt{(\rho^2 + 2\rho + 2)m^2 - 2(1+\rho)m})/2$.*

4.2 Approximation ratio

According to the analysis in Subsubsection 4.1.1, when $\rho \leq 2\mu/m - 1$, if $m \geq 6$, we just need to set $\rho^* = 0$ and $\mu^* = \lceil m/2 \rceil$ to obtain the optimal value of (17), i.e., the approximation ratio of our algorithm. For the special cases of $m = 2, 3, 4, 5$, optimal ρ^* and μ^* can be chosen according to Subsubsection 4.1.1. The ratio is listed in Lemma 4.7.

Now we investigate the case $\rho > 2\mu/m - 1$ based on Subsubsection 4.1.2. Unfortunately, we shall show in Subsection 4.3 that we are not able to use the technique in analysis for μ is Subsubsection 4.1.2 to obtain the optimal value of (17) over ρ. Thus, in this case we still can fix the value of ρ to obtain an improved approximation ratio. We also show that asymptotically it is almost the optimal choice. The value of ρ in our algorithm is set as follows:

$$\hat{\rho}^* = 0.26. \tag{23}$$

By substituting it to (4.8) we set

$$\hat{\mu}^* = \frac{113m - \sqrt{6469m^2 - 6300m}}{100}. \tag{24}$$

We need to examine whether $\hat{\rho}^*$ and $\hat{\mu}^*$ in (23) and (24) satisfy the assumption that $\hat{\rho}^* \geq 2\hat{\mu}^*/m - 1$. Since $m \geq 2 > 6300/3969$, $2500 < 6469 - 6300/m$. Because both sides are positive, taking the square root we have $50 < \sqrt{6369 - 6300/m}$. Therefore $\hat{\rho}^* = 13/50 > (63 - \sqrt{6369 - 6300/m})/50 = 2\hat{\mu}^*/m - 1$.

Lemma 4.9 *In the case that $\rho \geq 2\mu/m - 1$, our algorithm has an approximation ratio r at most*

$$\frac{100}{63} + \frac{100}{345303}\frac{(63m - 87)(\sqrt{6469m^2 - 6300m} + 13m)}{m^2 - m}.$$

Proof: It is worth noting that the $\hat{\mu}^*$ in (24) can be a fractional number. Therefore we need to consider $\lceil \hat{\mu}^* \rceil$ and $\lfloor \hat{\mu}^* \rfloor$. Since we should minimize the objective function over μ, the approximation ratio with integer value of μ is bounded as follows:

$$r \leq \min\{\max\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*)\}, \max\{A(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}\}.$$

According to the analysis in Subsubsection 4.1.2, here $A(\mu, \rho)$ is increasing in μ and $B(\mu, \rho)$ is decreasing in μ for a fixed ρ. Thus the bound on approximation ratio is

$$r \leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\}$$

Furthermore, $\lceil \hat{\mu}^* \rceil \leq \hat{\mu}^* + 1$ and $\lfloor \hat{\mu}^* \rfloor \geq \hat{\mu}^* - 1$. Again, because $A(\mu, \rho)$ is increasing and $B(\mu, \rho)$ is decreasing, we have

$$A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*) \leq A(\hat{\mu}^* + 1, \hat{\rho}^*); \qquad B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*) \leq B(\hat{\mu}^* - 1, \hat{\rho}^*).$$

Thus we have the following bound on the ratio r:

$$\begin{aligned} r &\leq \min\{A(\lceil \hat{\mu}^* \rceil, \hat{\rho}^*), B(\lfloor \hat{\mu}^* \rfloor, \hat{\rho}^*)\} \leq \min\{A(\hat{\mu}^* + 1, \hat{\rho}^*), B(\hat{\mu}^* - 1, \hat{\rho}^*)\} \\ &\leq A(\hat{\mu}^* + 1, \hat{\rho}^*). \end{aligned}$$

Therefore here we shall find an upper bound on $A(\hat{\mu}^* + 1, \hat{\rho}^*)$, which is also an upper bound on the approximation ratio r. Substituting $\hat{\mu}^*$ in (24) and $\hat{\rho}^*$ in (23) in $A(\hat{\mu}^* + 1, \hat{\rho}^*)$ gives:

$$\begin{aligned} r \leq A(\hat{\mu}^* + 1, \hat{\rho}^*) &= \frac{2}{1 + \hat{\rho}^*} + \frac{2}{1 + \hat{\rho}^*} \frac{(1 + \hat{\rho}^*)m - (2 - \hat{\rho}^*)}{(2 - \hat{\rho}^*)(m - \hat{\mu}^*)} \\ &= \frac{100}{63} + \frac{100}{345303} \frac{(63m - 87)(\sqrt{6469m^2 - 6300m} + 13m)}{m^2 - m}. \end{aligned}$$

This is the claimed bound in the lemma. □

Combine Lemma 4.7 and Lemma 4.9 we have the following theorem of the approximation ratio of our algorithm:

Theorem 4.1 *There exists an algorithm for the problem of scheduling malleable tasks with precedence constraints under Assumption 1 and 2 with an*

approximation ratio

$$r \leq \begin{cases} 2, & \text{if } m = 2; \\ 2(2+\sqrt{3})/3, & \text{if } m = 3; \\ 8/3, & \text{if } m = 4; \\ 2(7+2\sqrt{10})/9, & \text{if } m = 5; \\ \dfrac{100}{63} + \dfrac{100}{345303}\dfrac{(63m-87)(\sqrt{6469m^2-6300m}+13m)}{m^2-m}, & \text{otherwise.} \end{cases}$$

Proof: We need to compare the minimum objective values in both cases $\rho \leq 2\mu/m - 1$ and $\rho > 2\mu/m - 1$. Thus for $m \geq 6$ we need to compare $4m/(m+2)$ and the value in Lemma 4.9. Suppose that

$$\frac{4m}{m+2} \geq \frac{100}{63} + \frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{63m^2-37m-100}.$$

By moving the right hand side to the left hand side and simplification, we obtain

$$\frac{62601m^3 - 112501m^2 + 142700m - 25(63m^2+13m-58)\sqrt{6469m^2-6300m}}{(m+2)(m^2-m)} \geq 0.$$

Denote by NUM the numerator, and by DEN the denominator, of the left hand side of the above inequality, respectively. It is obvious that $DEN > 0$ for any $m \geq 2$. Now we consider NUM. Solving equation $NUM = 0$ numerically we obtain the following roots: $m_1 = 1.35285$, $m_{2,3} = 2.27502 \mp 1.68612i$, $m_{4,5} = -0.230259 \mp 0.779709i$. It means that when $m \geq 2 > m_1$, both NUM and DEN are positive according to Proposition 4.1. Therefore for any integer $m \geq 6$ the $\hat{\rho}^*$ and $\hat{\mu}^*$ should be taken by (23) and (24) to obtain the approximation ratio bounded in Lemma 4.9. Then we need to compare the ratios according to Lemma 4.7 and Lemma 4.9 for $m = 2, 3, 4, 5$. When $m = 2$, by Lemma 4.9, $r \leq 2.384810$, which is greater than the bound in Lemma 4.7. So we should take $\rho^* = 3\sqrt{2} - 4$ and $\mu^* = 1$, with an approximation ratio $r \leq 1 + 2\sqrt{2}/3$. When $m = 3$, by Lemma 4.9, $r \leq 2.755556$, which is greater than the bound in Lemma 4.7. So we should take $\rho^* = (3\sqrt{3} - 5)/2$ and $\mu^* = 2$, with an approximation ratio $r \leq 2(2 + \sqrt{3})/3$. When $m = 4$, by Lemma 4.9, $r \leq 2.908646$, which is greater than the bound in Lemma 4.7. So we should take $\rho^* = 3\sqrt{2} - 4$ and $\mu^* = 2$, with an approximation ratio $r \leq 4(1 + 2\sqrt{2})/3$. When $m = 4$, by Lemma 4.9, $r \leq 2.993280$, which is greater than the bound in Lemma 4.7. So we should take $\rho^* = \sqrt{10} - 3$ and $\mu^* = 3$, with an approximation ratio $r \leq 2(7 + 2\sqrt{10})/9$. This complete the proof. $\square$

Then the following corollary holds for the upper bound on the approximation ratios:

Corollary 4.1 *For all $m \in \mathbb{N}$ and $m \geq 2$, the approximation ratio*

$$r \leq \frac{100}{63} + \frac{100(\sqrt{6469}+13)}{5481}.$$

Furthermore, when $m \to \infty$, the upper bound in Theorem 4.1 tends to

$$\frac{100}{63} + \frac{100(\sqrt{6469}+13)}{5481} \approx 3.291919.$$

Proof: It is obvious that when $m \leq 6$, the approximation ratios fulfils the inequality. We now consider the case that $m \geq 6$.

Now we need to show that

$$\frac{100}{63} + \frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{m^2-m} \leq \frac{100}{63} + \frac{100(\sqrt{6469}+13)}{5481},$$

which is equivalent to $(63m-87)(\sqrt{6469m^2-6300m}+13m)/63(m^2-m) \leq \sqrt{6469}+13$. Since $m^2-m > 0$ for all $m \geq 2$, we can multiply both sides by $63(m^2-m)$. Then simplification gives $(21m-29)\sqrt{6469m^2-6300m} \leq 21\sqrt{6469}m^2 - 21\sqrt{6469}m + 104m$. When $m \geq 2$, $21m-29 > 0$ and $m^2-m > 0$, so both sides are positive. We can take square of both sides and obtain the following inequality:

$$(2475942 + 2184\sqrt{6469})m^2 + (315337 - 2184\sqrt{6469})m - 2649150 \geq 0. \quad (25)$$

It is easy to verify that $2475942 + 2184\sqrt{6469} \approx 2651601.325 > 2649150$ and $315337 - 2184\sqrt{6469} \approx 139677.675 > 0$. Therefore for any $m \geq 1$ the inequality (25) holds. So the bound in the corollary holds.

When $m \to \infty$, the bound

$$\begin{aligned} &\frac{100}{63} + \frac{100}{5481}\frac{(63m-87)(\sqrt{6469m^2-6300m}+13m-100)}{63m^2-37m-100} \\ \to\ &\frac{100}{63} + \frac{100(\sqrt{6469}+13)}{5481} \approx 3.291919. \end{aligned}$$

□

We give the list of values of approximation ratios for our algorithm for $m = 2, \ldots, 33$ in Table 2. Here it is worth noting that we still take $\hat{\rho}^* = 0.26$ for

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0	2	10	4	0.260	3.0026	18	7	0.260	3.1792	26	9	0.260	3.1594
3	2	0.098	2.4880	11	4	0.260	2.9693	19	7	0.260	3.1451	27	9	0.260	3.2123
4	2	0	2.6667	12	5	0.260	3.1130	20	7	0.260	3.1160	28	10	0.260	3.1976
5	2	0.260	2.6868	13	5	0.260	3.0712	21	8	0.260	3.1981	29	10	0.260	3.1746
6	3	0.260	2.9146	14	5	0.260	3.0378	22	8	0.260	3.1673	30	10	0.260	3.2135
7	3	0.260	2.8790	15	6	0.260	3.1527	23	8	0.260	3.1404	31	11	0.260	3.2085
8	3	0.260	2.8659	16	6	0.260	3.1149	24	8	0.260	3.2110	32	11	0.260	3.1870
9	4	0.260	3.0469	17	6	0.260	3.0834	25	9	0.260	3.1843	33	11	0.260	3.2144

Table 2
Listing of bounds on approximation ratios for our algorithm.

m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$	m	$\mu(m)$	$r(m)$
2	1	4.0000	10	4	5.0000	18	8	5.0908	26	10	5.1250
3	2	4.0000	11	5	4.8570	19	8	5.0000	27	11	5.0588
4	2	4.0000	12	5	4.8000	20	8	5.0000	28	11	5.0908
5	3	4.6667	13	6	5.0000	21	9	5.0768	29	12	5.1111
6	3	4.5000	14	6	4.8889	22	9	5.0000	30	12	5.0526
7	3	4.6667	15	6	5.0000	23	9	5.1111	31	13	5.1578
8	4	4.8000	16	7	5.0000	24	10	5.0667	32	13	5.1000
9	4	4.6667	17	7	4.9091	25	10	5.0000	33	13	5.0768

Table 3
Listing of bounds on approximation ratios for the algorithm in [17].

$m = 5$, because the bound in Lemma 4.9 is not a tight upper bound on the objective value of (17) with $\rho = \hat{\rho}^* = 0.26$. In fact with the rounded value of $\lfloor \hat{\mu}^* \rfloor$ or $\lceil \hat{\mu}^* \rceil$ the objective values are lower than the bound in Lemma 4.9 as listed above. The list is to be compared with the values of approximation ratios for the algorithm in [17] (Table 3). Our algorithm leads to a visible improvement for all m for our model (which is a special case of the model in [17]).

4.3 Asymptotic behaviour of approximation ratio

In our algorithm we set $\hat{\rho}^* = 0.26$. However, the bound of the approximation ratio r can be improved by choosing the value of ρ^* depending on m. In this subsection we shall study it.

Recall that μ^* in (4.8) is the minimizer of the objective function in (17). By substituting μ^* to $A(\mu, \rho)$ or $B(\mu, \rho)$ we can obtain two functions $A(\rho)$ or $B(\rho)$. Since our goal is to find the minimum value of $A(\rho)$ and $B(\rho)$ over all ρ, we need to solve the equation $A(\rho)'_\rho = 0$ and $B(\rho)'_\rho = 0$. Because $A(\rho) = B(\rho)$, we just need to consider one of them, say, $A(\rho)$. The first order partial derivative of $A(\rho)$ with respect to ρ is

$$A(\rho)'_\rho = \frac{2m((m-\mu^*+1)+(2-\rho)(\mu^*)'_\rho)}{(2-\rho)^2(m-\mu^*+1)^2} - \frac{2}{(1+\rho)^2} + \frac{2((m-\mu^*+1)-(1+\rho)(\mu^*)'_\rho)}{(1+\rho)^2(m-\mu^*+1)^2}$$

Combine the two terms together and the denominator is positive. So the equation $A(\rho)'_\rho = 0$ can be simplified as $-(2-\rho)^2(\mu^*)^2 + [(\rho^2-10\rho+7)m + (2-\rho)^2]\mu^* + (1+\rho)(2-\rho)[(1+\rho)m-(2-\rho)](\mu^*)'_\rho + 3(2\rho-1)m(m+1) = 0$, where

$$\begin{aligned}
\mu^* &= m(2+\rho)/2 - \sqrt{\Delta}/2; \\
(\mu^*)^2 &= ((\rho^2+3\rho+3)m^2 - (\rho+1)m)/2 - (2+\rho)m\sqrt{\Delta}/2; \\
(\mu^*)'_\rho &= m/2 - ((\rho+1)m^2 - m)/2\sqrt{\Delta},
\end{aligned}$$

and $\Delta = (\rho^2+2\rho+2)m^2 - 2(1+\rho)m$. Substituting them to the equation, we obtain the following equation: $A_1\Delta + A_2\sqrt{\Delta} + A_3 = 0$, where the coefficients are as follows:

$$\begin{aligned}
A_1 &= m\rho^3 + (-3m-1)\rho^2 + (6m+4)\rho + (m-4); \\
A_2 &= m[-m\rho^4 + (m+1)\rho^3 + (-3m-2)\rho^2 + (2m+8)\rho + (-2m+2)]; \\
A_3 &= m[(m^2+m)\rho^4 + (m^2-3m-1)\rho^3 + (-3m^2-3m+3)\rho^2 \\
&\quad +(-5m^2+7m)\rho + (-2m^2+6m-4)].
\end{aligned}$$

To remove the square root, we can simplify the equation to an equivalent equation $(A_1\Delta + A_3)^2 - A_2^2\Delta = 0$. After simplification, it can be written as the following form:

$$m^2(1+m)(1+\rho)^2 \sum_{i=0}^{6} c_i \rho^i = 0, \tag{26}$$

where the coefficients are as follows:

$$
\begin{aligned}
c_0 &= -8(m-1)^2(m-2);\\
c_1 &= 8(m-1)(m-2)(3m-2);\\
c_2 &= 21m^3 - 59m^2 + 16m + 24;\\
c_3 &= 2(m+1)(7m^2 - 7m - 4);\\
c_4 &= 3m^3 - 7m^2 + 15m + 1;\\
c_5 &= 2m(3m^2 - 4m - 1);\\
c_6 &= m^2(m+1).
\end{aligned}
$$

Eliminating the common factor $(\rho+1)^2$ we are able to obtain an equation with degree of 6. Unfortunately in general there are no analytic roots for polynomial with degree more than 4. So we are not able to solve (26) to obtain the optimal ρ^* depending on m like in Subsubsection 4.1.1.

However, we can estimate the asymptotic behaviour of the approximation ratio. When $m \to \infty$, equation (26) tends to: $m^3(\rho^6 + 6\rho^5 + 3\rho^4 + 14\rho^3 + 21\rho^2 + 24\rho - 8) = 0$. Thus we just need to consider the equation $\rho^6 + 6\rho^5 + 3\rho^4 + 14\rho^3 + 21\rho^2 + 24\rho - 8 = 0$. Solving it by numerical methods, we have the following roots: $\rho_1 = -5.8353$, $\rho_{2,3} = -0.949632 \pm 0.89448i$, $\rho_4 = 0.261917$, $\rho_{5,6} = 0.72544 \pm 1.60027i$. The only feasible root here in the interval $\rho \in (0, 1)$ is $\rho^* = 0.261917$. Substituting it to (4.8) the optimal $\mu^* \to 0.325907m$. With these data, from either A or B one has that $r \to 3.291913$.

In our algorithm we fix $\hat{\rho}^* = 0.26$ just because it is close to the asymptotic optimal ρ^*. The ratio of our algorithm could be further improved by fix $\hat{\rho}^*$ to a better approximation to ρ^*. In this way we conjecture that there exists a 3.291913-approximation algorithm for scheduling malleable tasks with precedence constraints. However, the analysis is too complicated and our algorithm has a ratio 3.291919, which is already very close to this asymptotic ratio.

We can also use numerical method to solve the min-max nonlinear program (21). We can construct a grid of ρ in the interval $[0, 1]$, and μ in $[1, \lfloor (m+1)/2 \rfloor]$. The grid size for ρ is $\delta\rho$ and for μ is 1 as μ is an integer. We can compute the values of $A(\mu, \rho)$ and $B(\mu, \rho)$ on each grid point, and search for the minimum over all grid points to decide the optimal objective values depending on m. The results by setting $\delta\rho = 0.0001$ and $m = 2, \ldots, 33$ are in Table 4. Compared them with the results in Table 2 we can see that the solutions of our algorithm are already very close to the optimum.

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.000	2.0000	10	4	0.310	2.9992	18	6	0.143	3.1065	26	9	0.308	3.1515
3	2	0.098	2.4880	11	4	0.273	2.9671	19	7	0.328	3.1384	27	9	0.200	3.1579
4	2	0.243	2.5904	12	4	0.067	3.0460	20	7	0.300	3.1092	28	10	0.335	3.1895
5	2	0.200	2.6389	13	5	0.318	3.0664	21	7	0.167	3.1273	29	10	0.310	3.1663
6	3	0.243	2.9142	14	5	0.286	3.0333	22	8	0.331	3.1600	30	10	0.212	3.1695
7	3	0.292	2.8777	15	5	0.111	3.0802	23	8	0.304	3.1330	31	10	0.129	3.1972
8	3	0.250	2.8571	16	6	0.325	3.1090	24	8	0.185	3.1441	32	11	0.312	3.1785
9	3	0.000	3.0000	17	6	0.294	3.0776	25	9	0.333	3.1765	33	11	0.222	3.1794

Table 4
Numerical results of min-max nonlinear program (21).

5 Conclusion

We have presented a 3.291919-approximation algorithm for scheduling malleable tasks with precedence constraints for the discrete version of the malleable task model initiated by Prasanna et al. [22–24]. This improves the previous results for the scheduling problem. Since our model has already been applied for real parallel computers, our algorithm has large potential for further application in practice. It is also worth noting that we can generalize our model to the case where the work function is convex in the processing times and Assumption 1 holds. Our algorithm and analysis are both still valid in this generalized model.

Acknowledgments: The authors thank the anonymous referees for their insightful and constructive comments to significantly improve this paper, and thank Ulrich M. Schwarz for checking the equivalence between (7) and (10).

References

[1] A. Agarwal, D. Chaiken, K. Johnson, D. Kranz, J. Kubiatowicz, K. Kurihara, B.-H. Lim, G. Maa, and D. Nussbaum, The MIT Alewife machine: a large-scale distributed-memory multiprocessor, *Proceedings of the 1st Workshop on Scalable Shared Memory Multiprocessors*, 1991.

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

[3] D. E. Culler, R. Karp, D. Patterson, A. Sahay, E.Santos, K. Schauser, R. Subramonian and T. von Eicken, LogP: A practical model of parallel computation, *Communications of the ACM*, 39 (11) (1996), 78-85.

[4] D. E. Culler, J. P. Singh and A. Gupta, *Parallel computer architecture: A hardware/software approach*, Morgan Kaufmann Publishers, San Francisco, 1999.

[5] J. Du and J. Leung, Complexity of scheduling parallel task systems, *SIAM Journal on Discrete Mathematics*, 2 (1989), 473-487.

[6] P.-F. Dutot, G. Mounié and D. Trystram, Scheduling parallel tasks – approximation algorithms, *in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, J. Y.-T. Leung (Eds.)*, CRC Press, Boca Raton, 2004.

[7] M. Garey and R. Graham, Bounds for multiprocessor scheduling with resource constraints, *SIAM Journal on Computing*, 4 (1975), 187-200.

[8] R. L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal*, 45 (1966), 1563-1581.

[9] J. J. Hwang, Y. C. Chow, F. D. Anger and C. Y. Lee, Scheduling precedence graphs in systems with interprocessor communication times, *SIAM Journal on Computing*, 18(2) (1989), 244-257.

[10] K. Jansen, Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme, *Algorithmica* 39(1) (2004), 59-81.

[11] K. Jansen and L. Porkolab, Linear-time approximation schemes for scheduling malleable parallel tasks, *Algorithmica* 32(3) (2002), 507-520.

[12] K. Jansen and R. Thöle, Approximation algorithms for scheduling parallel jobs: breaking the approximation ratio of 2, *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, ICALP 2008, LNCS 5125,
234-245. Full version: *Technical Report 0808*, University of Kiel, Germany, 2008.
`http://www.informatik.uni-kiel.de/en/ifi/research/technical-reports/`

[13] K. Jansen and H. Zhang, An approximation algorithm for scheduling malleable tasks under general precedence constraints, *ACM Transactions on Algorithms*, 2(3) (2006), 416-434.

[14] T. Kalinowski, I. Kort and D. Trystram, List scheduling of general task graphs under LogP, *Parallel Computing*, 26(9) (2000), 1109-1128.

[15] J. K. Lenstra and A. H. G. Rinnooy Kan, Complexity of scheduling under precedence constraints, *Operations Research*, 26 (1978), 22-35.

[16] R. Lepère, G. Mounié and D. Trystram, An approximation algorithm for scheduling trees of malleable tasks, *European Journal of Operational Research*, 142(2) (2002), 242-249.

[17] R. Lepère, D. Trystram and G. J. Woeginger, Approximation algorithms for scheduling malleable tasks under precedence constraints, *International Journal of Foundations of Computer Science*, 13(4) (2002), 613-627.

[18] W. Ludwig and P. Tiwari, Scheduling malleable and nonmalleable parallel tasks, *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms*, SODA 1994, 167-176.

[19] G. Mounié, C. Rapine and D. Trystram, Efficient approximation algorithms for scheduling malleable tasks, *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA 1999, 23-32.

[20] G. Mounié, C. Rapine and D. Trystram, A 3/2-dual approximation algorithm for scheduling independent monotonic malleable tasks, *SIAM Journal on Computing*, 37(2) (2007), 401-412.

[21] G. N. S. Prasanna, Structure Driven Multiprocessor Compilation of Numeric Problems, *Technical Report MIT/LCS/TR-502*, Laboratory for Computer Science, MIT, April 1991.

[22] G. N. S. Prasanna and B. R. Musicus, Generalised multiprocessor scheduling using optimal control, *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA 1991, 216-228.

[23] G. N. S. Prasanna and B. R. Musicus, Generalized multiprocessor scheduling for directed acyclic graphs, *Proceedings of Supercomputing 1994*, 237-246.

[24] G. N. S. Prasanna and B. R. Musicus, The Optimal Control Approach to Generalized Multiprocessor Scheduling, *Algorithmica*, 15(1) (1996), 17-49.

[25] M. Skutella, Approximation algorithms for the discrete time-cost tradeoff problem, *Mathematics of Operations Research*, 23 (1998), 909-929.

[26] U. M. Schwarz, Tightness results for malleable task scheduling algorithms, *Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics*, PPAM 2007, LNCS 4967, 1059-1067.

[27] J. Turel, J. Wolf and P. Yu, Approximate algorithms for scheduling parallelizable tasks, *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, SPAA 1992, 323-332.

[28] H. Zhang, Approximation Algorithms for Min-Max Resource Sharing and Malleable Tasks Scheduling, *PhD thesis*, University of Kiel, Germany, 2004. `http://e-diss.uni-kiel.de/diss_1408/`