Scheduling Malleable Tasks with Precedence Constraints *

Klaus Jansen
Institut für Informatik und Praktische Mathematik
Universität zu Kiel, Olshausenstraße 40
D-24098 Kiel, Germany
kj@informatik.uni-kiel.de

Hu Zhang
Department of Computing and Software
McMaster University, 1280 Main Street West
Hamilton, ON L8S 4K1, Canada
zhanghu@mcmaster.ca

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, 23, 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. In the phases we use two parameters $\mu \in \{1, \ldots \lfloor (m+1)/2 \rfloor\}$ and $\rho \in [0, 1]$ for the allotment and the rounding, respectively, where m is the number of processors. 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 very close to the best asymptotic one.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sequencing and scheduling*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms*

General Terms

Algorithms

Keywords

approximation algorithms, scheduling, malleable tasks, precedence constraints

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, 26]. In this category scheduling *malleable tasks* proposed in [26] 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 can not 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 makespan $C_{\max}$ (maximum completion time).

*Research supported in part by the DFG Graduiertenkolleg 357, Effiziente Algorithmen und Mehrskalenmethoden, by the EU Thematic Network APPOL II, Approximation and Online Algorithms for Optimization Problems, IST-2001-32007, by the EU Project CRESCCO, Critical Resource Sharing for Cooperation in Complex Systems, IST-2001-33135, by an MITACS grant of Canada, and by the NSERC Discovery Grant DG 5-48923. This research was performed in part when the second author was studying at the University of Kiel.

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

Prasanna et al. [22, 23, 24] proposed a model of the malleable tasks. In their model, for each malleable task, the processing time is non-decreasing 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 and to develop a 3.291919-approximation algorithm for the scheduling problem.

In this paper, we consider the discrete model based on [22, 23, 24]. We assume that $p_j(0) = \infty$ as any task J_j can not 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 $0 \leq l'' \leq l \leq l' \leq m$,*

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

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 can not 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 can not 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, 23, 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 will 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. Our model studied in this paper is essentially a special case of the model in [17, 13].

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 works: 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 addressed by Jansen [10].

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. 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, 27]. Then we develop an approximation algorithm for scheduling malleable tasks with precedence constraints for our model. Similar to [17, 13, 27], 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 length and the average work (total work divided by total number of processors) is minimized. In [17, 13, 27] 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 works 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 $100/63 + 100(\sqrt{6469} + 13)/5481 \approx 3.291919$. 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. Due to the limit of space, readers are referred to the full version [12] for details of the analysis.

2. PROPERTIES OF THE WORK FUNCTION

In this section, we will 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 monotonous penalty assumption on work functions in [2, 17, 13, 27]. 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

$$\frac{1}{p_j(1)} \geq \frac{1}{2} \quad \frac{1}{p_j(2)} + \frac{1}{p_j(0)} \quad .$$

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

$$\frac{1}{p_j(l)} \geq \frac{1}{2} \quad \frac{1}{p_j(l+1)} + \frac{1}{p_j(l-1)} \quad \geq \frac{1}{2} \quad \frac{1}{p_j(l+1)} + \frac{l-1}{lp_j(l)} \quad ,$$

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 series $W_j(1), \ldots, W_j(m)$, and the proof is complete. □

Theorem 2.1 in fact is Assumption $2'$ on work functions in [2, 17, 13, 27]. It also indicates that the communication overhead increases if more processors are allotted to one malleable task. The monotonous penalty assumption on work functions in [2, 17, 13, 27] is only a sequel of our Assumption 2. 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 Assumption 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):

$$\frac{p_j(1)}{p_j(l)} = s_j(l) \geq \bar{s}_j(l) = \frac{p_j(1)}{l' - l''} \quad \frac{l - l''}{p_j(l')} - \frac{l - l'}{p_j(l'')} \quad . \qquad (4)$$

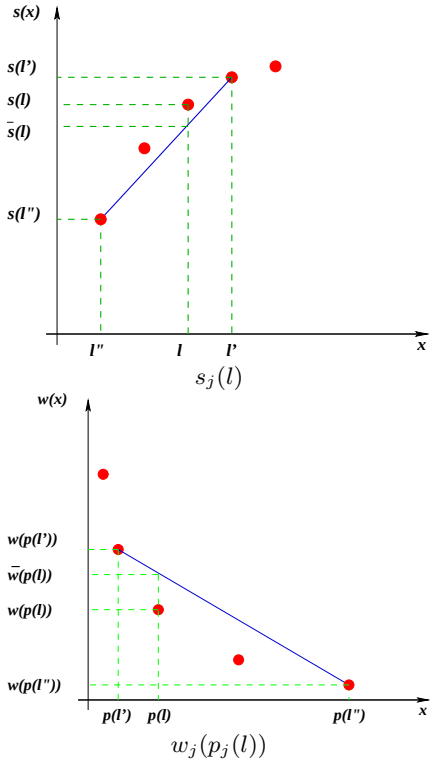

Figure 1: The diagrams of the speedup function $s_j(l)$ and the work function $w_j(p_j(l))$.

We now consider the diagram of the work function $w_j(p_j(l))$ versus processing time $p_j(l)$. The straight line connection 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 $(p_j(l), w_j(p_j(l)))$ intersect at one point which has the vertical coordinate $\bar{w}_j(p_j(l))$ as follows:

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

From (4) we have that

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

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

$$\begin{aligned} lp_j(l)[p_j(l')-p_j(l'')] &\geq p_j(l)[l'p_j(l')-l''p_j(l'')] \\ &\quad -(l'-l'')p_j(l')p_j(l''). \end{aligned}$$

By multiplying both sides by the negative factor $1/[p_j(l')-p_j(l'')]$ due to Assumption 1, together with (5), we immediately obtain that $w_j(p_j(l)) = lp_j(l) \leq \bar{w}_j(p_j(l))$, which show 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, 23, 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 realistic model with integral numbers of processors and discrete processing times.

3. APPROXIMATION ALGORITHM

Different from the algorithms in [22, 23, 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 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.

LIST (J, m, α', μ)
allot $l_j = \min\{l'_j, \mu\}$ processors to task J_j for all j;
$SCHEDULED = \emptyset$;
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

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: If $x = p_j(l)$ for $l = 1, \ldots, m$, then $w_j(x_j) = w_j(p_j(l))$. If $x \in (p_j(l+1), p_j(l))$ for $l = 1, \ldots, m-1$, then

$$\begin{aligned} w_j(x_j) &= \frac{w_j(p_j(l+1)) - w_j(p_j(l))}{p_j(l+1) - p_j(l)} x_j \\ &\quad + \frac{w_j(p_j(l))p_j(l+1) - w_j(p_j(l+1))p_j(l)}{p_j(l+1) - p_j(l)} \end{aligned} \tag{6}$$

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:

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

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 maximize the makespan C. Here both the total work W and the critical path length L are not constants but variables of the linear program (7).

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:

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

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:

$$
\begin{array}{lll}
\min & C & \\
\text{s.t.} & C_i + x_j \le C_j, & \forall i \in \Gamma^-(j), \forall j; \\
& C_j \le L, & \forall j; \\
& \dfrac{(l+1)p_j(l+1)-lp_j(l)}{p_j(l+1)-p_j(l)}x_j & \\
& \quad - \dfrac{p_j(l)p_j(l+1)}{p_j(l+1)-p_j(l)} \le w_j, & l = 1,\ldots,m-1, \forall j; \\
& L \le C; & \\
& W/m = \sum_{j=1}^{n} w_j/m \le C; & \\
& x_j \in [p_j(m), p_j(1)], & \forall j.
\end{array} \tag{9}
$$

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

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 given by $p_j(l_c) = p_j(l+1-\rho) = \rho p_j(l) + (1-\rho)p_j(l+1)$, and its work is $w_j(p_j(l_c)) = (1-\rho)w_j(p_j(l+1)) + \rho w_j(p_j(l)) = (1-\rho)(l+1)p_j(l+1) + \rho l p_j(l)$.

We apply the following rounding technique for the fractional solution to (9): If $x_j^* \ge p_j(l_c)$ 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 to the rounded solution. Then we develop an allotment α' for all jobs where each job J_j is allotted a number l_j' 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 $W^* = P + \sum_{j=1}^{n} \hat{w}_j(x_j^*)$ and $W' = \sum_{j=1}^{n} l_j' p_j(l_j')$. Denote by OPT the overall optimal makespan (over all feasible schedules with integral number of processors allotted to all tasks). It is obvious that

$$
\max\{L^*, W^*/m\} \le C^*_{\max} \le OPT. \tag{10}
$$

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:

$$
l_j^* = w_j(x_j^*)/x_j^*. \tag{11}
$$

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

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

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:

$$
\begin{aligned}
l_j^* \frac{w_j(x_j^*)}{x_j^*} &= \frac{(l+1)p_j(l+1)-lp_j(l)}{p_j(l+1)-p_j(l)} \\
&\quad - \frac{p_j(l)p_j(l+1)}{p_j(l+1)-p_j(l)} \frac{1}{x_j^*} \\
&= l + \frac{p_j(l+1)}{p_j(l)-p_j(l+1)} \quad \frac{p_j(l)}{x_j^*} - 1 \quad .
\end{aligned} \tag{12}
$$

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.,

$$
\frac{p_j(l+1)}{p_j(l)-p_j(l+1)} \quad \frac{p_j(l)}{x_j^*} - 1 \quad \le 1.
$$

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

Lemma 4.1 shows that l_j^* is well defined. We notice that l_j^* is just a notation and we only have knowledge that the real fractional number of processors corresponding to x_j^* should be in the interval $[l, l+1]$ if $p_j(l+1) \le x_j^* \le p_j(l)$. The notation l_j^* here fulfils this property and is convenient for our following analysis. 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 $p_j(l_j') \le \frac{2}{1+\rho}x_j^*$, and the its work $w_j(p_j(l_j')) = l_j' p_j(l_j') \le \frac{2}{2-\rho}l_j^* x_j^* = \frac{2}{2-\rho}w_j(x_j^*)$, where x_j^* is the optimal solution to (9). l_j^* and $w_j(x_j^*)$ are the fractional number of processors and work of J_j corresponding to the optimal solution x_j^* as defined in (11) and (8).*

PROOF. Suppose $x_j^* \in (p_j(k+1), p_j(k))$. In the domain $[p_j(k+1), p_j(k)]$, the critical point $k_c = k+1-\rho$. 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.

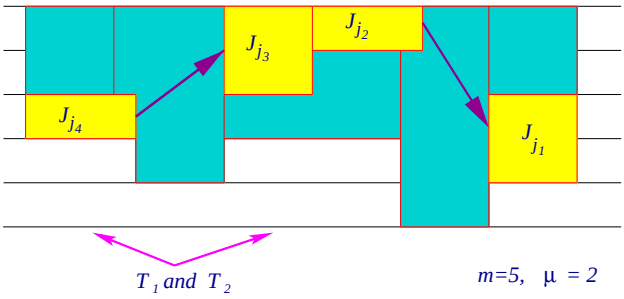

Figure 2: An example of the "heavy" path.

In the first case, $x_j^* \geq p_j(k_c)$. 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

$$\begin{aligned}\frac{p_j(l'_j)}{x_j^*} &\leq \frac{p_j(k)}{p_j(k_c)} = \frac{p_j(k)}{\rho p_j(k) + (1-\rho)p_j(k+1)} \\ &\leq \frac{p_j(k)}{\rho p_j(k) + (1-\rho)kp_k(k)/(k+1)} = \frac{k+1}{k+\rho}.\end{aligned}$$

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 $l'_j = k+1$. Since more processors are allotted, according to Theorem 2.1 the work increases by the following factor:

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

Since k is an integer, the above two factors can be further bounded by $(k+1)/(k+1-\rho) \leq 2/(2-\rho)$ and $(k+1)/(1+\rho) \leq 2/(1+\rho)$. 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, 27], 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 while 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. $\frac{1+\rho}{2}|T_1| + \min\left\{\frac{\mu}{m}, \frac{1+\rho}{2}\right\}|T_2| \leq C_{\max}^*$.

PROOF. We construct a "heavy" directed path $\mathcal{P}$ in the final schedule, similar to [17, 13, 27]. 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', all the tasks in V' can not be 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 (11). 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, 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 $\rho|T_1|$ to $L^*(\mathcal{P})$. 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 (10), 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:*

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

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

$$C_{\max} = |T_1| + |T_2| + |T_3|. \tag{13}$$

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:

$$W \ge |T_1| + \mu|T_2| + (m-\mu+1)|T_3|. \tag{14}$$

Multiplying (13) by $m-\mu+1$ and subtracting (14) from it yield

$$(m-\mu+1)C_{\max} \le W + (m-\mu)|T_1| + (m-2\mu+1)|T_2|. \tag{15}$$

Since 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., $W' \ge W$. 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 $W' \le 2W^*/(2-\rho)$. Furthermore, from (10), $W \le 2W^*/(2-\rho) \le 2mC^*_{\max}/(2-\rho)$. Substituting it to the bound on $C_{\max}$ in (15) 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 optimal approximation ratio of our algorithm is bounded by the optimal objective value of the following min-max nonlinear program*

$$\begin{array}{lll} \min_{\mu,\rho} \max_{x_1,x_2} & \dfrac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} & \\ s.t. & \dfrac{1+\rho}{2}x_1 + \min\left\{\dfrac{\mu}{m}, \dfrac{1+\rho}{2}\right\} x_2 \le 1; & \\ & x_1, x_2 \ge 0; & \\ & \rho \in [0,1]; & \\ & \mu \in \{1, \ldots, \dfrac{m+1}{2}\}. & \end{array} \tag{16}$$

PROOF. Dividing the inequalities in Lemma 4.3 and Lemma 4.4 by $C^*_{\max}$, together with inequality (10), the definitions of x_i and the definition of the approximation ratio r, we have the first constraint in (16) and the following inequality:

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

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 (16). □

According to Lemma 4.5, we just need to solve the min-max nonlinear program (16) to obtain an upper bound of the approximation ratio of our algorithm. To solve (16), we need the following technical lemma:

LEMMA 4.6. *For two functions $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ defined on $[a, b]$ and $f(x)$, $g(x)$ are continuously differentiable, if one of the following two properties holds:*

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

Ω_2: $f'(x) = 0$ and $g'(x) \ne 0$ 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 $h(x) = \max\{f(x), g(x)\}$.

We leave the proof in our full version [12]. Therefore the following theorem holds:

THEOREM 4.1. *There exists an algorithm for the problem of scheduling malleable tasks under precedence constraints under Assumption 1 and 2 with an approximation ratio as follows: for $m \le 5$,*

$$r \le \begin{cases} 2, & \text{if } m = 2; \\ 2(2+\sqrt{3})/3, & \text{if } m = 3; \\ 4(1+2\sqrt{2}/3)/3, & \text{if } m = 4; \\ 2(7+2\sqrt{10})/9, & \text{if } m = 5. \end{cases}$$

For $m \ge 6$,

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

PROOF. Due to the limit of space, we only give a sketch of the proof here. The details of proof of this theorem can be found in the full version [12].

To solve (16), we need to consider two cases: either $\rho \le 2\mu/m - 1$ or $\rho < 2\mu/m - 1$. For the first case, we need to study the following min-max nonlinear program:

$$\begin{array}{lll} \min_{\mu,\rho} \max_{x_1,x_2} & \dfrac{2m/(2-\rho) + (m-\mu)x_1 + (m-2\mu+1)x_2}{m-\mu+1} & \\ \text{s.t.} & \dfrac{1+\rho}{2}x_1 + \dfrac{1+\rho}{2}x_2 \le 1; & \\ & x_1, x_2 \ge 0; & \\ & \rho \in [0,1]; & \\ & \mu \in \{1, \ldots, \dfrac{m+1}{2}\}. & \end{array}$$

We can prove that in this case, the approximation ratio has the following bound (details in the full version [12]):

$$r = \begin{cases} 2, & \text{if } m = 2; \\ 2(2+\sqrt{3})/3, & \text{if } m = 3; \\ 4(1+2\sqrt{2}/3)/3, & \text{if } m = 4; \\ 2(7+2\sqrt{10})/9, & \text{if } m = 5; \\ 4m/(m+2), & \text{otherwise.} \end{cases}$$

For the second case we need to consider the following min-max nonlinear program:

$$\begin{array}{rl} \min_{\mu,\rho}\max_{x_1,x_2} & \dfrac{\dfrac{2m}{2-\rho}+(m-\mu)x_1+(m-2\mu+1)x_2}{m-\mu+1} \\ \text{s.t.} & \dfrac{1+\rho}{2}x_1+\dfrac{\mu}{m}x_2\le 1; \\ & x_1,x_2\ge 0; \\ & \rho\in \quad \max \quad \dfrac{2\mu}{m}-1,0 \quad ,1 \quad ; \\ & \mu\in \quad 1,\ldots, \quad \dfrac{m+1}{2} \quad . \end{array} \tag{17}$$

We notice that the constraints on x_1 and x_2 in (17) forms 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 (17) 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 can not 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:

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

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 if $(1+\rho)m-(2-\rho)\ge 0$, i.e., $\rho\ge(2-m)/(m+1)$. This inequality is always true as $\rho\ge 0$ and $m\ge 2$. Thus $A(\rho)'_\mu$ is always nonnegative. So for any $m\ge 2$, $A(\mu,\rho)$ is increasing in μ. Furthermore, the first order partial derivative of $B(\mu,\rho)$ with respect to μ is

$$\begin{aligned} B(\mu,\rho)'_\mu &= [-2(1-\rho)m\mu^2+2(2-\rho)m(m+1)\mu \\ &\quad -(2-\rho)m(m+1)^2]/[(2-\rho)\mu^2(m-\mu+1)^2]. \end{aligned}$$

If $\rho=1$, then $B(\mu,\rho)'_\mu=\dfrac{m(m+1)[2\mu-(m+1)]}{\mu^2(m-\mu+1)^2}\le 0$ as $\mu\le(m+1)/2$. 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 the following roots:

$$\mu=\frac{2-\rho\pm\sqrt{\rho(2-\rho)}}{1-\rho}\cdot\frac{m+1}{2}.$$

We can prove that both roots violate the constraint $\mu\le(m+1)/2$. 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. According to Lemma 4.6, if we find a solution to the equation $A(\mu,\rho)=B(\mu,\rho)$, then the value attains the optimum. The equation $A(\mu,\rho)=B(\mu,\rho)$ is as follows:

$$2\mu^2-(4+2\rho)m\mu+(1+\rho)m(m+1)=0.$$

The only feasible root is as follows (the other one violates the constraint $\mu\le(m+1)/2$):

$$\mu^*=\frac{(2+\rho)m-\sqrt{(\rho^2+2\rho+2)m^2-2(1+\rho)m}}{2}. \tag{18}$$

We can substitute this μ^* to either $A(\mu,\rho)$ or $B(\mu,\rho)$ to obtain functions $A(\rho)$ or $B(\rho)$ in ρ. Solving the equation $A(\rho)'_\rho=0$ or $B(\rho)'_\rho=0$ we are able to find the optimal ρ. Unfortunately, as shown in the Subsection 4.1, we need to find analytical roots of a polynomial of degree 6, which is impossible in general. Thus we fix $\hat\rho^*=0.26$ and $\hat\mu^*$ becomes $\hat\mu^*=\dfrac{113m-\sqrt{6469m^2-6300m}}{100}$ according to (18). Furthermore, we need to consider the integer value of $\hat\mu^*$. Thus we have the following bound:

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

Finally, we need to compare the minimum objective values in both cases $\rho\le 2\mu/m-1$ and $\rho>2\mu/m-1$. Therefore we can obtain the claimed bounds. Details of the proof can be found in our full version [12]. □

Moreover, the following corollary holds:

COROLLARY 4.1. *For all $m\in\mathbb{N}$ and $m\ge 2$, the approximation ratio r is at most $100/63+100(\sqrt{6469}+13)/5481\approx 3.291919$. Furthermore, when $m\to\infty$, the upper bound in Theorem 4.1 tends to the above value.*

PROOF. We also only give a proof sketch here. It is obvious that if $m\le 6$, then the approximation ratios fulfils the inequality. We now consider the case that $m\ge 6$. Here we need to show that

$$\frac{63m-87}{63}\frac{\sqrt{6469m^2-6300m}+13m}{m^2-m}\le\sqrt{6469}+13.$$

It is equivalent to the following inequality:

$$\begin{aligned} &(2475942+2184\sqrt{6469})m^2 \\ +\ &(315337-2184\sqrt{6469})m-2649150\ge 0. \end{aligned} \tag{19}$$

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 $m\ge 1$ the inequality (19) holds. Therefor we have proved the claimed bound. Details are also in the full version [12]. □

The values in Theorem 4.1 for $m=2,\ldots,33$ are listed as follows:

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.500	2.0000	18	7	0.260	3.1792
3	2	0.098	2.4880	19	7	0.260	3.1451
4	2	0.243	2.5904	20	7	0.260	3.1160
5	2	0.260	2.6868	21	8	0.260	3.1981
6	3	0.260	2.9146	22	8	0.260	3.1673
7	3	0.260	2.8790	23	8	0.260	3.1404
8	3	0.260	2.8659	24	8	0.260	3.2110
9	4	0.260	3.0469	25	9	0.260	3.1843
10	4	0.260	3.0026	26	9	0.260	3.1594
11	4	0.260	2.9693	27	9	0.260	3.2123
12	5	0.260	3.1130	28	10	0.260	3.1976
13	5	0.260	3.0712	29	10	0.260	3.1746
14	5	0.260	3.0378	30	10	0.260	3.2135
15	6	0.260	3.1527	31	11	0.260	3.2085
16	6	0.260	3.1149	32	11	0.260	3.1870
17	6	0.260	3.0834	33	11	0.260	3.2144

4.1 Asymptotic behaviour of approximation ratio of our algorithm

In our algorithm we set $\hat{\rho}^* = 0.26$. However, the approximation ratio r could be improved by choosing the value of ρ^* depending on m. In this subsection we shall study it. Readers are referred to the full version [12] for details.

Recall that μ^* in (18) is the minimizer of the objective function in (16). 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 $A(\rho)$ or $B(\rho)$ over all ρ, we need to solve the equation $A(\rho)'_\rho = 0$ or $B(\rho)'_\rho = 0$. 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

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

Combine the two terms together and the denominator is positive. So the equation $A(\rho)'_\rho = 0$ can be simplified as follows:

$$-(2-\rho)^2(\mu^*)^2 + [(\rho^2-10\rho+7)m + (2-\rho)^2]\mu^* + (1+\rho) \cdot(2-\rho)[(1+\rho)m - (2-\rho)](\mu^*)'_\rho + 3(2\rho-1)m(m+1) = 0.$$

Here

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

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

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

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:

$$(1+\rho)^2 \sum_{i=0}^{6} c_i \rho^i = 0, \tag{20}$$

where the coefficients are as follows:

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

Unfortunately, we are not able to solve (20) to obtain the optimal ρ^* depending on m because in general there is no analytic root to polynomials of degrees more than 4. Therefore we investigate the asymptotic behaviour of the optimal value, i.e., the optimal value when $m \to \infty$. In that case, equation (20) tends to $\rho^6+6\rho^5+3\rho^4+14\rho^3+21\rho^2+24\rho-8 = 0$. Solving it by numerical methods, we have the following roots:

$$\begin{aligned} \rho_1 &= -5.8353; \quad \rho_{2,3} = -0.949632 \pm 0.89448i; \\ \rho_4 &= 0.261917; \quad \rho_{5,6} = 0.72544 \pm 1.60027i. \end{aligned}$$

The only feasible root here in the interval $\rho \in (0,1)$ is $\rho^* = 0.261917$. Substituting it to (18) the optimal $\mu^* \to 0.325907m$. With these data, from either A or B one have 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 could exist a 3.291913-approximation algorithm for the problem of scheduling malleable tasks with precedence constraints. However, the analysis is 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 (17). 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 as follows:

m	$\mu(m)$	$\rho(m)$	$r(m)$	m	$\mu(m)$	$\rho(m)$	$r(m)$
2	1	0.500	2.0000	18	6	0.143	3.1065
3	2	0.098	2.4880	19	7	0.328	3.1384
4	2	0.243	2.5904	20	7	0.300	3.1092
5	2	0.200	2.6389	21	7	0.167	3.1273
6	3	0.243	2.9142	22	8	0.331	3.1600
7	3	0.292	2.8777	23	8	0.304	3.1330
8	3	0.250	2.8571	24	8	0.185	3.1441
9	3	0.000	3.0000	25	9	0.333	3.1765
10	4	0.310	2.9992	26	9	0.308	3.1515
11	4	0.273	2.9671	27	9	0.200	3.1579
12	4	0.067	3.0460	28	10	0.335	3.1895
13	5	0.318	3.0664	29	10	0.310	3.1663
14	5	0.286	3.0333	30	10	0.212	3.1695
15	5	0.111	3.0802	31	10	0.129	3.1972
16	6	0.325	3.1090	32	11	0.312	3.1785
17	6	0.294	3.0776	33	11	0.222	3.1794

Compared with the above results, we find that the solutions of our algorithm are already very close to the best possible.

5. CONCLUSION

We have presented a 3.291919-approximation algorithm for scheduling malleable tasks with precedence constraints for our model with strongly realistic background. 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 model under Assumption 1 and $2'$ with an additional assumption that the work function is convex in processing time. Our algorithm and analysis are still valid in this generalized model.

6. ACKNOWLEDGEMENTS

The authors thank Denis Trystram for informing us about the work by Prasanna and Musicus [22, 23, 24].

7. 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, *Proceedings of the 10th European Symposium on Algorithms*, ESA 2002, LNCS 2461, 562-573.

[11] K. Jansen and L. Porkolab, Linear-time approximation schemes for scheduling malleable parallel tasks, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 1999, 490-498.

[12] K. Jansen and H. Zhang, Improved approximation algorithms for scheduling malleable tasks with precedence constraints, Technical Report, http://www.informatik.uni-kiel.de/~hzh/malle.ps.

[13] K. Jansen and H. Zhang, An approximation algorithm for scheduling malleable tasks under general precedence constraints, *manuscript*.

[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, *manuscript*.

[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] 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.

[27] H. Zhang, Approximation Algorithms for Min-Max Resource Sharing and Malleable Tasks Scheduling, *PhD thesis*, University of Kiel, Germany, 2004.