Divisible Load Scheduling with Improved
Asymptotic Optimality

Reiji Suda *1

# Department of Computer Science, the University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
* JST, CREST
4-1-8 Hon-chou, Kawaguchi, Saitama, 332-0012, Japan
1 reiji@is.s.u-tokyo.ac.jp

Abstract—Divisible load model allows scheduling algorithms that give nearly optimal makespan with practical computational complexity. Beaumont et al. have shown that their algorithm produces a schedule whose makespan is within $1 + O(1/\sqrt{T})$ times larger than the optimal solution when the total amount of tasks $T$ scales up and the other conditions are fixed. We have proposed an extension of their algorithm for multiple masters with heterogeneous performance of processors but limited to uniform network performance. This paper analyzes the asymptotic performance of our algorithm, and shows that the asymptotic performance of our algorithm is either $1 + O(1/\sqrt{T})$, $1 + O(\log T/T)$ or $1 + O(1/T)$, depending on the problem. For the latter two cases, our algorithm asymptotically outperforms the algorithm by Beaumont et al.

I. INTRODUCTION

Scheduling problem on parallel processors is a discrete combinatorial problem. Sophisticated algorithms have the computational complexity of at least higher degree polynomial of the numbers of tasks and processors, and are impractical for very large scale problems. But simple algorithms with lower complexity may give schedules far from optimal. The problem is much more difficult when the processors are heterogeneous, while heterogeneous processors are more and more prevailing nowadays, from a PC with CPU and GPU to Roadrunner at LANL which attains 1 Piflops for the first time in the history with hybrid of Cell and Opteron processors [1].

Divisible load model [2] provides a simplified view of tasks in scheduling problems for parallel processors. In divisible load model, the tasks are considered as a “continuous matter” and we can divide them into an arbitrary size of positive real number. This is an extreme simplification of tasks and loses almost every characteristics of each task, but the gain is great: scheduling algorithms with low complexity and high quality. The divisible load model sometimes called “partitionable tasks” or “bag-of-tasks.”

Many works on divisible load theory assumes the master-worker model, where all the tasks are stored in the single master node at the initial state. Also early works on divisible load theory assume that the master sends a task to each worker only once (single installment or single round) [3], [4], however in many cases the makespan can be reduced by splitting the message into several parts and sending them separately (multi-installment or multi-round) [5]. Under the restriction of single round, the optimal solution can be represented in a closed form for some problems. If the multiple rounds are allowed, getting the optimal solution is much harder [6]. However, we can still obtain practically good schedules.

Among others, we found that the multi-round algorithm by Beaumont et al. [7] is noteworthy. Their algorithm can treat heterogeneous processors and heterogeneous networks in low computational complexity, while providing a performance guarantee in the following sense: The ratio of the makespans of the obtained schedule and the optimal schedule (asymptotic performance ratio hereafter) can be theoretically bounded. Specifically, they proved that the asymptotic performance ratio is $1 + O(1/\sqrt{T})$ under the assumption that the total amount of tasks $T$ is scaled up while the other components of the problems (the number of processors and their performance, etc.) are fixed.

We [8] have proposed an extension of their algorithm. The extension is on two points. First, the assumption that all the tasks are stored in a single master processor at the initial state is removed, and many (or even all) processors can have tasks at the initial state (cf. [9]). We called this assumption “multi-master,” but there is no predetermined classification of the processors into masters and workers. This multi-master extension allows us to use our algorithm for a wider class of dynamic load balancing problems. However we have to limit the network to be uniform. Some researchers have proposed algorithms for multi-master problems [10], [11], but Ooshita et al. [12] pointed out some difficulties with single port models. Our algorithm can be implemented with single ports.

The second extension is the variable sizes of rounds, while the algorithm by Beaumont et al. repeats uniformly sized rounds. Here, a round represents a part of schedule in which the master sends tasks to each worker only once. The rounds in the algorithm by Beaumont et al. are exactly same as each other. In our algorithm, the rounds are similar but the sizes can be different. This resembles UMR by Yang et al. [13], but the sizes of the rounds are determined in a different way. Also, UMR is basically for homogeneous processors, while ours accepts heterogeneous processors. UMR needs resource selection (i.e. determine the optimum set of worker processors), but ours (and that by Beaumont et al.) computes the optimum selection of processors.
In our previous paper [8], we showed that the asymptotic performance ratio of our algorithm is $1 + O(1/\sqrt{T})$, which is the same asymptotic ratio as that of Beaumont et al. This paper will show that the analysis can be improved as following.

There are three classes of problems, where the asymptotic performance ratio of our algorithm is $1 + O(1/\sqrt{T})$, $1 + O(\log T/T)$, or $1 + O(1/T)$, respectively. Roughly speaking, the first class is for the schedules of uniformly sized rounds, the second class corresponds to the variable-sized rounds, and the third class is only for multi-master problems. Thus UMR-like variable-sized multi-round schedules are now proved to be asymptotically better than the uniformly sized multi-round schedules.

The rest of this paper is organized as follows. The next section presents necessary parts of our algorithm for the following discussion. Section 3 is the main part of this paper, where the asymptotic performance analysis will be presented. In section 4 we will show a few examples to confirm the analysis. Then a summary and comments will follow.

II. REVIEW OF OUR ALGORITHM

This section reviews our algorithm for a class of multi-master divisible load problems. In this paper we only discuss the basic case, but our algorithm can be easily extended to some classes of problems (as published in Japanese articles, not referred here).

A. Problem Statement

The tasks are divisible, uniform, and independent. There are $p$ processors, connected by a homogeneous network. The time to send a task of size $x$ from a processor to another is $\alpha + \beta x$, independent of the sender and the receiver. Each processor can do at most one communication at each time (single port), but there is no influence of any communication from a different sender to a different receiver (crossbar network).

The time taken for the $i$th processor to process a task of size $x$ is $\gamma_i x$ (heterogeneous processors). In this paper, we assume that each processor cannot process a task and communicate a task at the same time (no overlap of communication and computation). At the initial state, processor $i$ has a task of size $x_i$. Here $x_i$ can be zero or positive real number, so master-worker problems and multi-master problems are both included.

The objective is to process all tasks in as short time as possible. Collection of the computational results is not considered in this paper.

B. Overview of Our Algorithm

We do not know the optimal solution of the problem stated in the previous subsection in general. There are infinitely many schedules for each problem, since the task can be divided into an arbitrary number of pieces. To get an approximate solution in low computational complexity, we have introduced a few restrictions.

- There is no relay of tasks, that is, there is no processor that receives tasks and sends tasks.
- The schedule is composed of one or more rounds. Each round is derived from a common basic round.

From the first restriction, the processors are classified into three classes: masters who only send tasks, workers who only receive tasks, and independents who do not send or receive any task. Of course the independent processes need no scheduling (they just process the tasks they initially have).

From the second restriction, we need first determine the basic round, and then compose the schedule by modifying the basic round and arranging the modified rounds. In our algorithm, the concept of the round is somewhat simplified from that of Beaumont et al., where the starting times of a round were different on different processors, while in our algorithm, a round occupies the same time span on any processor (see Fig. 1).

Also we made a simplification of the treatment of the latency term of the communication time ($\alpha + \beta x$) over the algorithm by Beaumont et al. We just ignore that term, or equivalently assume $\alpha = 0$, except the last phase of the scheduling algorithm. By assuming $\alpha = 0$, the computation times and the communication times are proportional to the amount of tasks. Then any schedule of a round that is defined from the schedule of the basic round by scaling by a common factor (i.e. similar to the basic round) is valid.

Those two simplifications are crucial to the asymptotic performance analysis for variable-sized round scheduling.

Now we are ready to present overview of our algorithm.

1) Determine the basic rounds, assuming uniformly sized rounds and ignoring the constant term of the communication times.

2) Determine the number of rounds and the scaling factors of the rounds.

![Fig. 1. Schedule with uniformly sized rounds ($k = 3$)](image)

In the first phase, we assume a schedule with uniformly sized rounds as shown in Fig. 1. The schedule is composed of $k + 1$ rounds ($k$ is an arbitrary positive integer). All the rounds, except the first and the last ones, are equal to each other (regular rounds). Each of the regular rounds does $1/k$ of the total computations and $1/k$ of the total communications. The first round (prologue) is determined by removing the computations from the regular round, and the last round (epilogue) is determined by removing the communications from the regular round.

On the above assumptions, we determine the optimal schedule for the regular rounds. Also we define the basic round by
scaling the regular rounds by \( k \) (note that each regular round does \( 1/k \) of the total work).

Detailed algorithm to determine the optimum regular round is presented in [8]. It is not reproduced here, because it is not needed for the asymptotic performance analysis. We just need to note the following property. Because we ignore the constant term, the optimum schedule for the regular rounds is independent of \( k \) (the number of rounds minus one). In addition, because each regular round do \( 1/k \) of the total computations and communications, \( k \) times the schedule length of a regular round must be a constant. Let \( T \) be that constant. Now remember that the schedule of a regular round is optimal for \( 1/k \) of the total work, so the optimum makespan cannot be less than \( T = k \times (1/k)T \). \( T \) is also the schedule length of the basic round.

The above discussion is very rough and must be unsatisfactory for most readers. A better derivation is in [8].

C. Determining the Number of Rounds and Scaling Factors

The second phase of our algorithm modifies the uniform-sized round schedule obtained in the first phase. First, computations are introduced into the prologue, and the distinction between it and the regular rounds is removed. Second, the sizes (or the scaling factors) of the rounds are now variable as Fig. 2, and determined in the following way.

The sizes of the rounds are not determined arbitrarily. The restriction is that a worker must receive the task before processing it. In the schedule in the first phase, this problem is solved by introducing the prologue, where no computation is done. The tasks to be processed in the first regular round were sent in the prologue, the tasks to be processed in the second regular round were sent in the first regular round, etc.

In the second phase, the restriction is solved by determining the scaling factor of the rounds appropriately. That is, if less tasks are available, then the next round will be smaller, and if more tasks are available, then the next round will be larger. In [8] we derived the optimal scaling factor of \( j \)th round \( r_j \) is determined as

\[
r_j = \min_{i} \left\{ \frac{x'_i + z_i}{w'_i} - R_j \frac{w_i}{w'_i} \right\}.
\]

where \( i \) runs for the indices of worker processors. \( R_j \) is defined as

\[
R_j = \sum_{i=0}^{j-1} r_i
\]

and \( R_0 = 0 \). Parameters \( w_i \) is, roughly speaking, the amount of tasks consumed (processed minus received) by worker \( i \) in the basic round, and \( w'_i \) is the amount of tasks that worker \( i \) must have before the basic round (mathematically clear definitions for them can be found in [8]). Actually \( w'_i \) and \( w_i \) can be zero or negative, but in this paper we assume \( w'_i > 0 \) for all workers for simplicity: Workers with \( w'_i \leq 0 \) pose no additional requirement on \( r_j \), and thus can be ignored. Parameter \( x'_i \) is defined as \( x'_i = x_i + t_i/\gamma_i \) where \( t_i \) is the idle times in the basic round on processor \( i \). Parameter \( z_i \) is defined by

\[
z_i = \zeta / \gamma_i,
\]

where \( \zeta \) is a parameter called “additional idle time” (see Fig. 2) whose value is determined later. In short, when the basic round and \( \zeta \) is fixed, then the sizes of the rounds are determined by Eq. (1). The schedule ends when \( r_j \) sums up 1, so the number of rounds is also determined by the basic round and \( \zeta \).

Remembering the constant term \( \alpha \) in the communication time model, we can bound the makespan produced by our algorithm by

\[
\bar{T} = T + kA + \zeta,
\]

where \( k \) is the number of rounds except epilogue, \( A \) is defined by \( A = \alpha c_{\text{max}} \) where \( c_{\text{max}} \) is the maximum number of communications by any processor in the basic round, and \( \zeta \) is the additional idle time (already introduced). Here \( k \) depends on \( \zeta \), so \( \bar{T} \) is a function of \( \zeta \), and we have proposed to minimize \( \bar{T}(\zeta) \), e.g. by using the golden section search.

Please note that formal discussion of our algorithm can be found in [8] and here we have just extracted the minimum information that is needed in the asymptotic analysis.

III. ASYMPOTIC PERFORMANCE ANALYSIS

In their asymptotic analysis, Beaumont et al. increased the total amount of tasks at the only master processor, while the other parameters are fixed. To apply their analysis to our problem, let us introduce a scaling factor \( C \), and define problems by scaling \( x_i \) (the amount of tasks that processor \( i \) has at the initial state) by \( C \). The other parameters, number of processors \( p \), communication performance parameters \( \alpha \) and \( \beta \), processing performance parameters \( \gamma_i \), are fixed.

Since the basic round is defined with \( \alpha \) removed, the basic round for the scaled problem can be obtained just by scaling the original basic round by factor \( C \). Thus the schedule length of the basic round is \( CT \), and parameters \( x'_i, w_i \) and \( w'_i \) in Eq. (1) are also scaled by \( C \). Only \( z_i = \zeta / \gamma_i \) is not simply scaled, because \( \zeta \) is determined by minimization of \( \bar{T} \), which includes \( \alpha \). By introducing the scaling factor \( C \) into Eq. (1) we have

\[
r_j = \min_{i} \left\{ \frac{x'_i + \zeta/C}{\gamma_i w'_i} - R_j \frac{w_i}{w'_i} \right\}. \tag{2}
\]

Because the basic round of the scaled problem is a scaled schedule of the original basic round, the number of communications is independent of \( C \), and so \( A = \alpha c_{\text{max}} \) is also independent of \( C \). Thus \( \bar{T} \) of the scaled problem is

\[
\bar{T}(\zeta) = CT + k(\zeta, C)A + \zeta.
\]
Also note that $CT$ is a lower bound of the optimum makespan. As is already mentioned, there are three classes of problems in our asymptotic analysis.

A. Class 1

First assume that $x_i' = w_i = 0$ for some worker $i$. Roughly speaking, this $x_i' = w_i = 0$ means that worker $i$ has no task at the initial state and has no idle time in the basic round (it is always busy in receiving and processing tasks). In the algorithm by Beaumont et al., the first worker satisfies this condition, and thus Class 1 approximately corresponds to their analysis.

From Eq. (2) we have

$$r_j \geq \frac{\zeta / C}{\max_i \{\gamma_i w_i\}},$$

because (though we omit the proof) it holds that $x_i' - w_i \geq 0$ and $0 \leq R_j \leq 1$. Remember that the schedule ends when $r_j$s sum up 1, and then we can see that the number of the rounds $k$ satisfies

$$k \leq \frac{C \max_i \{\gamma_i w_i\}}{\zeta} + 1.$$

Thus we have

$$\bar{T} \leq CT + \frac{C \max_i \{\gamma_i w_i\}}{\zeta} A + A + \zeta.$$

The right-hand side is minimized by letting $\zeta = \sqrt{AC \max_i \{\gamma_i w_i\}}$ to

$$\bar{T}(T_{opt}) \leq CT + 2 \sqrt{AC \max_i \{\gamma_i w_i\}} + A.$$

Since we know $T_{opt} \geq CT$, we reach

$$\frac{\bar{T}(T_{opt})}{T_{opt}} \leq 1 + 2 \sqrt{\frac{A \max_i \{\gamma_i w_i\}}{CT}} + \frac{A}{CT} = 1 + O(\sqrt{C}).$$

This is the same asymptotic performance order as the algorithm by Beaumont et al. Note that we have never used the assumption $x_i' = w_i = 0$ for some worker $i$, and thus this is a proof of general asymptotic optimality.

For this class of problems, we cannot expect better asymptotic performance. Substituting $x_i' = w_i = 0$ in Eq. (2), we have

$$r_j \leq \frac{\zeta / C}{\gamma_i w_i}.$$

From this we have

$$k \geq \frac{C \gamma_i w_i}{\zeta},$$

and

$$\bar{T} \geq CT + \frac{C \gamma_i w_i}{\zeta} A + \zeta.$$

The right-hand side is minimized when $\zeta = \sqrt{AC \gamma_i w_i}$ to

$$\bar{T} \geq CT + 2 \sqrt{AC \gamma_i w_i}.$$

Thus we have

$$\frac{\bar{T}(T_{opt})}{CT} = 1 + \Omega(1/\sqrt{C}),$$

where $\Omega$ means an order with a lower bound.

Let us summarize the results as follows.

**Theorem 1:** If there is any worker with $x_i' = w_i = 0$ and $w_i' > 0$, then the makespan of our algorithm $T_{our}$ satisfies

$$\frac{T_{our}}{T_{opt}} = 1 + O \left( \frac{1}{\sqrt{C}} \right).$$

We cannot expect more than this, in the following sense:

$$\frac{T}{CT} = 1 + \Omega \left( \frac{1}{\sqrt{C}} \right).$$

B. Class 2

Next assume that there is no worker $i$ with $x_i' = w_i = 0$, but there is some worker $i$ with $x_i' = 0$ or $x_i' - w_i = 0$. Since $x_i'$ is the amount of tasks that processor $i$ initially has, the master-worker problems are classified in this class if it is not in Class 1.

Let us assume both conditions of $x_i' = 0$ and $x_i' - w_i = 0$, because then we can analyze the other cases similarly. Note that $0 \leq R_j \leq 1$ thus it holds that

$$\frac{x_i' - w_i}{w_i'} + \frac{\zeta / C}{\gamma_i w_i} \leq \frac{x_i' - w_i}{\gamma_i w_i'} - R_j \frac{w_i}{w_i'}.$$

Let us define the following parameters:

$$a_{min} = \min_i \left\{ \frac{-w_i}{w_i'} \mid x_i' = 0 \right\},$$

$$b_{min} = \min_i \left\{ \frac{x_i' - w_i}{w_i'} \mid x_i' > 0, x_i' - w_i > 0 \right\},$$

$$c_{min} = \min_i \left\{ \frac{x_i'}{w_i'} \mid x_i' - w_i = 0 \right\},$$

$$d_{min} = \min_i \left\{ \frac{1}{\gamma_i w_i'} \right\},$$

which are all positive. Then we can bound $r_j$ as

$$r_j = \min \left\{ \frac{x_i'}{w_i'} + \frac{\zeta / C}{\gamma_i w_i'} - R_j \frac{w_i}{w_i'} \right\} \geq \min \{ R_j a_{min}, b_{min}, (1 - R_j)c_{min} \} + d_{min} \zeta / C$$

from which we can obtain an upper bound of the number of rounds $k$.

In the following discussion, we assume that all three terms of $\min \{ R_j a_{min}, b_{min}, (1 - R_j)c_{min} \}$ are chosen for some $R_j$. It is easy to modify the analysis for other cases.

First, let us consider the term $R_j a_{min}$, which is chosen when $R_j < b_{min}/a_{min}$, and then it holds that $r_j \geq R_j a_{min} + d_{min} \zeta / C$. Then we have $r_0 \geq d_{min} \zeta / C$ and

$$R_{j+1} = R_j + r_j \geq R_j (1 + a_{min}) + d_{min} \zeta / C.$$

From this we have

$$R_{j+k} \geq (R_0 + d') (1 + a_{min})^k + d'.$$
where \( k \) is a non-negative integer and \( d' = (d_{\min}/a_{\min})(\zeta/C) \). So by letting

\[
k_0 = \left\lfloor \log \left( \frac{b_{\min}C}{d_{\min}c_{\min} - 1} \right) \right\rfloor / \log(1 + a_{\min}),
\]
we have \( R_{k_0} \geq b_{\min}/a_{\min} \). Therefore with at most \( k_0 = O(\log(C/\zeta)) \) rounds, the sum of \( r_j \)'s reaches \( b_{\min}/a_{\min} \).

Second, let us consider the term \( b_{\min} \), which is chosen when \( b_{\min}/a_{\min} \leq R_j < 1 - b_{\min}/c_{\min} \). In this region we have \( r_j \geq b_{\min} \), and thus with no more than

\[
k_1 = 1/b_{\min} - 1/a_{\min} - 1/c_{\min}
\]
rounds, sum of \( r_j \)'s reach \( 1 - a_{\min}/c_{\min} \). Note that \( k_1 \) is independent of \( C \) and of \( \zeta \).

Last, let us consider \( (1 - R_j)c_{\min} \), which is chosen when \( R_j \geq 1 - b_{\min}/c_{\min} \). Here it holds that \( r_j \geq d_{\min}\zeta/C + (1 - R_j)c_{\min} \). We have

\[
1 - R_{j+1} = 1 - R_j - r_j \leq (1 - R_j)(1 - c_{\min}) - d_{\min}\zeta/C,
\]
and thus

\[
1 - R_{j+k} \leq (1 - R_j + b')(1 - c_{\min})^{k - b'}
\]
for any integer \( k \geq 1 \), where \( b' = (d_{\min}/c_{\min})(\zeta/C) \). From this we can conclude that \( R_{j+k} \) reaches 1 with \( k \leq k_2 \), where \( k_2 \) is

\[
k_2 = \frac{\log b' + \log(1 - R_j + b') - \log(1 - c_{\min})}{\log(1 - b'/(1 - c_{\min}))}.
\]

Note that \( k_2 = O(\log(C/\zeta)) \).

Summing up the above, we showed that the number of rounds \( k \) satisfies \( k \leq k_0 + k_1 + k_2 \). Here \( k_0 = O(\log(C/\zeta)) \), \( k_1 \) is a constant, and \( k_2 = O(\log(C/C)) \), when \( C/\zeta \to \infty \). Thus it holds that

\[
\bar{T} \leq CT + \zeta + A_1 \log(C/\zeta) + A_2
\]
where \( A_1 \) and \( A_2 \) are constants. The right-hand side is minimized by \( \zeta = A_1 \) as

\[
\bar{T} \leq CT + A_1(\log C - \log A_1 + 1) + A_2,
\]
and thus

\[
\frac{\bar{T}}{T_{\text{opt}}} = 1 + O(\log(C/C)).
\]

This asymptotic performance is better than that of Class 1.

We can also obtain a lower bound of the makespan. By letting

\[
a_{\max} = \max_i \left\{ \frac{-w_i}{x_i'} \mid x_i' = 0 \right\},
\]

\[
b_{\max} = \max_i \left\{ \frac{x_i'}{w_i'} \mid x_i' > 0, x_i' - w_i > 0 \right\},
\]

\[
c_{\max} = \max_i \left\{ \frac{x_i'}{w_i'} \mid x_i' - w_i = 0 \right\},
\]

\[
d_{\max} = \max_i \left\{ \frac{1}{\gamma_i w_i'} \right\},
\]
we have

\[
r_j \leq \min \{ R_j a_{\max}, b_{\max}, (1 - R_j)c_{\max} \} + d_{\max}\zeta/C.
\]

By a reasoning almost parallel to the previous one, we can obtain

\[
\frac{\bar{T}}{C T} = 1 + \Omega(\log(C/\zeta)).
\]

Again let us summarize the results as follows.

**Theorem 2:** If there is no worker with \( x_i' = w_i = 0 \) and \( w_i' > 0 \), but there is some worker with \( x_i' = 0 \) or \( x_i' - w_i = 0 \) and \( w_i' > 0 \), then the makespan of our algorithm \( T_{\text{opt}} \) satisfies

\[
\frac{T_{\text{opt}}}{T_{\text{our}}} = 1 + O\left( \frac{\log C}{C} \right).
\]

We cannot expect more than this, in the following sense:

\[
\frac{\bar{T}}{CT} = 1 + \Omega\left( \frac{\log C}{C} \right).
\]

**C. Class 3**

Last, we assume that \( x_i' > 0 \) and \( x_i' - w_i > 0 \) for all workers.

This condition is not satisfied for master-worker problems. Roughly speaking, this assumption means all the processors have their own initial set of tasks but need redistribution for better performance.

Here \( r_j \) in Eq. (2) is always positive even for \( \zeta = 0 \), and thus the number of round is bounded by a constant \( K \). So it holds that

\[
\bar{T} \leq CT + AK
\]
and

\[
\frac{\bar{T}}{T_{\text{opt}}} \leq 1 + O(1/C).
\]

The reverse inequality is also clear because we need at least one round and thus we have

\[
\bar{T} \geq CT + A
\]
and

\[
\frac{\bar{T}}{CT} \geq 1 + O(1/C).
\]

Again let us summarize the results as follows.

**Theorem 3:** If there is no worker with \( x_i' = w_i = 0 \) nor \( x_i' - w_i = 0 \) and \( w_i' > 0 \), then the makespan of our algorithm \( T_{\text{opt}} \) satisfies

\[
\frac{T_{\text{opt}}}{T_{\text{our}}} = 1 + O\left( \frac{1}{C} \right).
\]

We cannot expect more than this, in the following sense:

\[
\frac{\bar{T}}{CT} = 1 + \Omega\left( \frac{1}{C} \right).
\]

Note that the performance ratios are written with \( T \) instead of \( C \) in the abstract and section 1. This is justified, because there \( T \) is said the total amount of tasks, which is proportional to \( C \). This is a little abuse of notation, but we follow that of Beaumont et al.
IV. EXPERIMENTS

In this section, a few experimental results are shown to confirm the above asymptotic performance analysis. Our algorithm is implemented in C language, and the problems shown in Table I are applied. Note that the following results are schedule lengths and not execution times on real machines.

Case 1 is a master-worker problem, for which the network is too slow that some workers remain idle. This is of Class 1. In case 2a, the network is fast enough, and in case 2b, every worker has small amount of tasks at the initial state. Those are of Class 2. In case 3, most workers have significant amount of tasks initially. This is of Class 3.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>EXAMPLE PROBLEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>case1</td>
</tr>
<tr>
<td>α</td>
<td>0.1</td>
</tr>
<tr>
<td>β⁻¹</td>
<td>300</td>
</tr>
<tr>
<td>γ⁻¹</td>
<td>80</td>
</tr>
<tr>
<td>x₁</td>
<td>600</td>
</tr>
<tr>
<td>x₂</td>
<td>0</td>
</tr>
<tr>
<td>x₃</td>
<td>0</td>
</tr>
<tr>
<td>x₄</td>
<td>0</td>
</tr>
<tr>
<td>x₅</td>
<td>0</td>
</tr>
<tr>
<td>x₆</td>
<td>0</td>
</tr>
<tr>
<td>x₇</td>
<td>0</td>
</tr>
<tr>
<td>x₈</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3 plots $T_{our}(C)/(CT) - 1$ against $C$, where $T_{our}(C)$ is the makespan obtained by our algorithm for a problem scaled by a factor $C$, and $CT$ is a lower bound of the makespan. By subtracting 1 from the performance ratio, we get the ratios that must be $O(1/\sqrt{C})$, $O(\log C/C)$, and $O(1/C)$.

For case 1 and case 3, the plots are almost straight lines, and for case 2a and case 2b, the plots curve slightly. Those results confirm the asymptotic analysis in the previous section.

V. SUMMARY

This paper has discussed an asymptotic performance analysis of our algorithm for multi-master divisible load problems proposed in [8]. We have shown that there are three classes of problems, and the asymptotic performance ratios are $1 + O(1/\sqrt{T})$, $1 + O(\log T/T)$, and $1 + O(1/T)$, respectively, and two of which are better than the analysis by Beaumont et al. Class 2 gives performance ratio of $1 + O(\log T/T)$, which shows the effectiveness of the variable sized rounds in asymptotic performance, and Class 3 gives $1 + O(1/T)$, which shows that multi-master systems can be more efficient than single master ones.

We hope that our results can be somewhat extended, e.g. for overlapped communication and computation, and collection of the results of the computations to the source processor of the tasks.

ACKNOWLEDGMENTS

This research is supported by Core Project of IJARC of Microsoft Research, Grant-in-aid for scientific research of MEXT Japan, and JST, CREST.

REFERENCES