The combinatorics in divisible load scheduling

M.Drozdowski[1], M.Lawenda[2]

Abstract

Divisible load scheduling problem is studied in this work. Though tractability of this problem in the practical cases is considered as its great advantage, we show that it has a hard combinatorial core. Computational hardness and polynomial time solvability of some special cases are shown.

Keywords: divisible loads, scheduling, computational complexity.

1 Introduction

Divisible load theory (DLT) is a new branch in the scheduling theory. DLT is used to represent communications and computations in distributed computer systems, or transportation and production systems. It is assumed in DLT, that the job (e.g. computation, production) can be divided into parts of arbitrary size. These parts can be processed in parallel by remote processing elements (computers, factories, etc.). The communication, or transportation, time must be taken into account. DLT was proved to be a versatile tool for modeling distributed computations, analyzing various communication topologies, and in performance evaluation. DLT predictions have been verified and confirmed experimentally. Surveys of DLT can be found in [2, 3, 5, 7, 10]. In the further discussion we will use distributed computing metaphor in divisible load processing. In the earlier literature, computational tractability of the divisible load model was considered as its great advantage. Though it is a justified observation for many practical cases, we will show that divisible load scheduling problems have hard combinatorial core.

[1]Institute of Computing Science, Poznań University of Technology, ul.Piotrowo 3A, 60-965 Poznań, Poland. This research was partially supported by a grant of Polish State Committee for the Scientific Research. Corresponding author. Email: `Maciej.Drozdowski@cs.put.poznan.pl`

[2]Poznań Supercomputing and Networking Center, ul.Noskowskiego 10, 61-794 Poznań, Poland.

In this work we consider star interconnection (a.k.a. a single level tree) of set $\mathcal{P}$ of processing elements. In the center of the star a computer P_0 called *originator* (or master, server) is located. Originator distributes the load to processing elements $P_1, \dots, P_m$ (slaves, workers, clients). The load is sent from the originator to a processing element in only one message. No other communications are performed. We assume that the time of returning the results can be neglected. The reasons for this assumption are twofold. First, we intend to consider the simplest, rudimentary cases of DLT. Second, this assumption is made for the simplicity of presentation. It is not limiting the generality or practicality of the considerations. It was shown in the earlier DLT literature [2, 3, 7, 10] that the process of result collection can be incorporated in the DLT models.

Assume that load chunk of size α_i is processed by P_i, where α_i is expressed in load units, e.g. bytes. The time of transferring this load to P_i is $S_i + \alpha_i C_i$. S_i is communication startup time, C_i is reciprocal of bandwidth. The computation time is $p_i + \alpha_i A_i$, where p_i denotes computation startup time which elapses before computations start, and A_i is processing rate (reciprocal of computing speed). Using processor P_i bears cost $f_i + \alpha_i l_i$. If memory size is limited to B_i load units, then load chunk must not exceed it, i.e. $\alpha_i \leq B_i$. Due to the existence of other more urgent computations, or maintenance periods, the availability of processor P_i may be restricted to some interval $[r_i, d_i]$. By such a restriction we mean that computations may take place only in interval $[r_i, d_i]$. A message with the load may arrive, or start arriving before r_i. We assume that computations start immediately after the later of two events: r_i, or the load arrival. The computation time $p_i + \alpha_i A_i$ must fit between the latter of the above two events, and d_i. Scheduling divisible load computations involve three decisions: selecting subset $\mathcal{P}'$ of the used processors from set $\mathcal{P}$, sequencing activation of processors in $\mathcal{P}'$, dividing total load V into chunks α_i for $P_i \in \mathcal{P}'$. The goal is to schedule divisible load computations such that schedule length $C_{max} = \max_{P_i \in \mathcal{P}'}\{t_i\}$ is minimum, where t_i is the completion time of the computations on P_i, and processing cost $G = \sum_{i \in \mathcal{P}'}(f_i + \alpha_i l_i)$ is bounded. An alternative formulation of the problem is to find a schedule with minimum cost G, such that its length C_{max} is limited. In the appendix we summarize the main notation used in this paper.

The combinatorial nature of DLT has been studied before. In [1] it has been shown that the sequence of processor activation in a star network affects schedule length. It was proved in [1, 2], and independently in [4], that when there are no communication startup times ($\forall_{P_i} S_i = 0$) the processors have to be activated according to the order of decreasing bandwidth of communication links. The case with non-negligible startup times ($\forall_{P_i} S_i > 0$) was studied in [4, 12]. It was determined in [4], and independently in [12], that if communication parameters are identical (i.e. $C_i = C, S_i = S$ for $i = 1, \dots, m$) then for the shortest schedule the order of decreasing processor speed should be the order of processor activation. This result was obtained under condition that all processors in $\mathcal{P}$ receive non-zero load and thus, can participate in processing. In [4] it was determined that the problem of divisible load scheduling on a system with startup times and multiple buses is **NP**-hard. The case of non-negligible startup times and limited memory buffers was shown to be **NP**-hard in [8]. The problem of optimizing the cost of a schedule has been studied in [6, 11]. Heuristic rules have been proposed in [6, 11] to select the set of used processors, and determine load assignment, efficiently in terms of cost and schedule length.

The rest of this paper is organized as follows. In Section 2 we demonstrate that the problem of divisible load scheduling on a star network can be solved in polynomial time for G, and for C_{max} criteria, provided that the set of used processors and the sequence of their activation are given. In Section 3 we show that various special cases of these problems are **NP**-hard.

2 Fixed processor activation sequence

The problem we consider is a bi-criterial optimization problem. The criteria are schedule length C_{max}, and processor usage cost $G = \sum_{i \in \mathcal{P}'} (f_i + \alpha_i l_i)$, where $\mathcal{P}'$ is a set of the exploited processors. This bi-criterial problem can be relaxed to two simpler problems: (i) minimization of C_{max} on condition that $G \leq \overline{G}$, (ii) minimization of G on condition that $C_{max} \leq \overline{C_{max}}$, where $\overline{G}$ is a predetermined upper bound on the schedule cost, and $\overline{C_{max}}$ is a given upper bound on the schedule length. Both problems can be solved in polynomial time by use of linear programming, provided that the set $\mathcal{P}'$ of used processors and the sequence of their activation is known. Let us consider problem (i) first. We assume that $|\mathcal{P}'| = m'$, and without loss of generality, the sequence of processor activation is $P_1, P_2, \dots, P_{m'}$. Then, the linear program for (i) is as follows:

minimize C_{max}

subject to:

$$\sum_{k=1}^{i}(S_k+\alpha_k C_k)+p_i+\alpha_i A_i \leq C_{max} \qquad i=1,\dots,m' \tag{1}$$

$$\sum_{k=1}^{i}(S_k+\alpha_k C_k)+p_i+\alpha_i A_i \leq d_i \qquad i=1,\dots,m' \tag{2}$$

$$r_i+p_i+\alpha_i A_i \leq C_{max} \qquad i=1,\dots,m' \tag{3}$$

$$r_i+p_i+\alpha_i A_i \leq d_i \qquad i=1,\dots,m' \tag{4}$$

$$\sum_{j=1}^{m'}(f_j+\alpha_j l_j) \leq \overline{G} \tag{5}$$

$$0 \leq \alpha_j \leq B_j \qquad j=1,\dots,m' \tag{6}$$

$$\sum_{j=1}^{m'}\alpha_j = V \tag{7}$$

In the above formulation constraints (1)-(4) guarantee that computations are performed in an admissible interval. The left side of inequalities (1), (2) is the earliest possible completion time of the computations provided that they are started immediately after the end of the load transfer. The left side of inequalities (3), (4) is the earliest possible completion time of the computations provided that they are started immediately after processor release time. By inequality (5) total cost of the schedule does not exceed the limit $\overline{G}$. Constraints (6) ensure that memory buffer size is not exceeded, and by (7) all the load is processed. Consider an example.

Example. $m'=4, V=20$, parameters of the processor system are the following:

parameter \ processor	P_1	P_2	P_3	P_4
A_i	2	0.5	1	2
B_i	10	10	10	20
C_i	1	0.1	2	2
S_i	1	1	1	2
p_i	0	1	1	0
d_i	10	20	30	200
r_i	0	10	20	20
f_i	1	5	3	2
l_i	0.5	1	0.3	1

The solution for this instance depends on the value of cost limit $\overline{G}$. This is demonstrated for some example values of $\overline{G}$ in the following table:

$\overline{G}$	α_1	α_2	α_3	α_4	C_{max}
≥25.7669	3	10	5.3333	1.3333	26.333
24.25	3	5	7.5	4.5	41.5
24.1334	3	0.00285	7.6666	9.3306	60.656
<24.1334	infeasible				

Observe that schedule length increases as the limit put on the costs decreases. For $\overline{G} \geq 25.7669$ inequality (5) is ineffective. For $\overline{G} < 24.1334$ the problem is infeasible. The schedule for $\overline{G}$ ≥25.7669 is presented in Fig.1. The vertical arrows indicate the end of communication from the originator to a certain processor. insert Fig.1 here

Problem (ii) can be also solved in polynomial time by modifying linear program (1)-(7). Namely, the roles of the objective function and constraint (5) must be exchanged. Thus, to solve problem (ii) the minimized objective function should be $\sum_{j=1}^{m'}(f_j + \alpha_j l_j)$, while inequality (5) should be replaced by $C_{max} \leq \overline{C_{max}}$. Both problems can be solved provided that we know set $\mathcal{P}'$ of active processors and the sequence of their activation. In the next section we will demonstrate that determining them is computationally hard.

3 Complexity of divisible load scheduling

In this section we will demonstrate that even restricted cases of scheduling divisible load computations in star networks are computationally hard. All the cases we study are in class **NP** because it is enough to guess set $\mathcal{P}'$ of the used processors, and the sequence of their activation. Then, the load sizes can be calculated in polynomial time using the methods presented in Section 2. We will provide polynomial time transformations from an **NP**-complete problem PARTITION defined as follows [9]:

INSTANCE: A finite set $E = \{e_1, \ldots, e_q\}$ of positive integers.
QUESTION: Is there a subset $E' \subseteq E$ such that

$$\sum_{i \in E'} e_i = \sum_{i \in E-E'} e_i = \frac{1}{2}\sum_{i=1}^{q} e_i = F? \quad (8)$$

We will use DLS abbreviation for divisible load scheduling. Some parameters are not binding for some of the studied cases of DLS. We do not repeat definitions of such parameters, and unless specified otherwise, it is assumed that $B_i = d_i = \infty, C_i = f_i = l_i = p_i = r_i = 0$, for all $P_i \in \mathcal{P}$. In the following w present **NP**-hard cases of DLS problem.

DLS WITH PROCESSOR RELEASE TIMES (DLSPRT)
INSTANCE: Heterogeneous star $\mathcal{P}$, load size V, time interval T, non-zero processor release times $[r_1, \dots, r_m]$.
QUESTION: Can load V be processed on $\mathcal{P}$ in at most T units of time?

Theorem 1 *Problem DLSPRT is* **NP***-hard.*

Proof. The proof is based on the polynomial time transformation from the PARTITION problem. The instance of DLSPRT is constructed in time $O(q)$ as follows: $m = q; A_i = \frac{1}{e_i}, C_i = 0, S_i = e_i, r_i = F$ for $i = 1, \dots, q; T = F + 1, V = F$.

Suppose PARTITION has a positive answer. Then the processors corresponding to the elements in set E' receive the load in $\sum_{i \in E'} S_i = \sum_{i \in E'} e_i = F = T - 1$ units of time. Their total speed is $\sum_{i \in E'} \frac{1}{A_i} = \sum_{i \in E'} e_i = F$. Thus, $V = F$ units of load can be processed in the last time unit of the schedule (cf. Fig.2). On the other hand, when the answer to DLSPRT is positive then some set $\mathcal{P}'$ of processors is activated in at most $T = F + 1$ units of time, to process at least F units of the load. Note that all processors become available at $r_i = T - 1$. Since $\forall_{P_i \in \mathcal{P}} S_i \geq 1$, any processor activated in the last time unit of the schedule does not process any load. Thus, the duration of all communications to the processors in $\mathcal{P}'$ does not exceed $T-1$: $\sum_{P_i \in \mathcal{P}'} S_i = \sum_{P_i \in \mathcal{P}'} e_i \leq T-1 = F$. The whole load V is processed in the last time unit of the schedule because processors become available at $r_i = T - 1$. Hence, $V = \sum_{P_i \in \mathcal{P}'} \frac{1}{A_i} = \sum_{P_i \in \mathcal{P}'} e_i \geq F$. As $\frac{1}{A_i} = S_i = e_i$, for $i = 1, \dots, m$, the answer to PARTITION is also positive. □

insert Fig.2 here

Before proceeding to the next special case of DLS let us study the amount of load that can be distributed, and processed on a star network with $C_i = 0$, and processors available until finite time d_i, for $i = 1, \dots, m$. Let us assume that the sequence of processor activation is fixed, but the set of processors to be activated is yet to be decided. Without loss of generality, let the sequence be $P_1, \dots, P_m$. Let binary variable $x_i = 1$ denote that processor P_i has been activated in the sequence $P_1, \dots, P_m$, and $x_i = 0$ that processor P_i is not activated, for $i = 1, \dots, m$. The amount of load V that can be distributed, and processed in time T is

$$V = \sum_{i=1}^{m} \frac{x_i d_i}{A_i} - \sum_{1\le i\le j\le m}^{m} x_i x_j \frac{S_i}{A_j} \tag{9}$$

In the above equation term $\sum_{i=1}^{m} \frac{x_i d_i}{A_i}$ is the amount of load that would be processed by the selected processors provided that they were activated simultaneously at the beginning of the schedule (i.e. communication is timeless). Still, the communication is not timeless. Startup time S_i of the selected processor P_i delays the activation of all processors P_j for $j \ge i$. Therefore, S_i decreases the total load that could be processed by $x_i \sum_{j=i}^{m} x_j \frac{S_i}{A_j}$. Term $\sum_{1\le i\le j\le m}^{m} x_i x_j \frac{S_i}{A_j}$ in (9) is the amount of lost load that could not be processed due to the communication delays. Equation (9) has a graphical interpretation shown in Fig.3. The shaded area is the amount of lost load $\sum_{1\le i\le j\le m}^{m} x_i x_j \frac{S_i}{A_j}$. insert Fig.3 here

DLS WITH PROCESSOR DEADLINES (DLSPD)
INSTANCE: Heterogeneous star $\mathcal{P}$, load size V, finite processor deadlines $[d_1, \dots, d_m]$.
QUESTION: Can load V be processes on $\mathcal{P}$ before the deadlines $[d_1, \dots, d_m]$?

Theorem 2 *Problem DLSPD is* **NP***-hard even if the sequence of processor activation is known.*

Proof. We assume that the sequence of processor activation is given. Without loss of generality it is $P_1, \dots, P_m$. We will prove **NP**-hardness of DLSPD by a polynomial time transformation of PARTITION problem. The transformation is as follows: $S_i = 2e_i, A_i = \frac{1}{2e_i}, d_i = 2F + e_i$, for $i = 1, \dots, m$. $V = 2F^2$. By substituting these values in equation (9) we obtain:

$$\begin{aligned} &V = \\ &4F\sum_{i=1}^{m} x_i e_i + 2\sum_{i=1}^{m} x_i e_i^2 - 4\sum_{1\le i\le j\le m}^{m} x_i x_j e_i e_j = \\ &4F\sum_{i=1}^{m} x_i e_i + 2\sum_{i=1}^{m} x_i^2 e_i^2 - 4\sum_{i=1}^{m} x_i^2 e_i^2 - 4\sum_{1\le i<j\le m}^{m} x_i x_j e_i e_j = \\ &4F\sum_{i=1}^{m} x_i e_i - 2\sum_{i=1}^{m} x_i^2 e_i^2 - 4\sum_{1\le i<j\le m}^{m} x_i x_j e_i e_j = \\ &2F^2 - 2(\sum_{i=1}^{m} x_i e_i - F)^2 \end{aligned} \tag{10}$$

In the second line of the above equation we used the fact that $x_i = x_i^2$ for $x_i \in \{0, 1\}$.

By activating the processors corresponding to the elements in set E' in PARTITION problem we have $x_i = 1$ for $i \in E'$, and $x_i = 0$ otherwise, in formula (10). If there is a positive answer to PARTITION, then $\sum_{i=1}^{m} x_i e_i = \sum_{i \in E'} x_i e_i = F$. Therefore, $V = 2F^2$ units of load are distributed and processed before processor deadlines, as demonstrated in equation (10). And vice versa, when a feasible schedule exists in which $V = 2F^2$ units of the load is processed, then by inequality (10), it is possible only if $\sum_{i=1}^{m} x_i e_i = F$, and the answer to PARTITION is positive. □

DLS WITH PROCESSOR STARTUP TIMES (DLSPST)
INSTANCE: Heterogeneous star $\mathcal{P}$, load size V, time interval T, non-zero processor computation startup times $[p_1, \ldots, p_m]$.
QUESTION: Can load V be processed on $\mathcal{P}$ in time at most T?

Theorem 3 *Problem DLSPST is* **NP***-hard.*

Proof. This theorem can be proved by a modification of the proof of Theorem 2. In Theorem 2 the maximum computation time available on P_i, provided that communication is timeless, is d_i. In the case of problem DLSPST this amount of time is equal to $T - p_i$. By setting $T = 3F$, and $p_i = F - e_i > 0$ we obtain that $T - p_i = 2F + e_i > 0$. Note that $T - p_i$ here is equal to d_i in the proof of Theorem 2. If we set other parameters of $\mathcal{P}'$ as in the proof of Theorem 2, then the rest of this proof follows from the proof of Theorem 2. □

DLS WITH FIXED PROCESSOR CHARGES (DLSFPC)
INSTANCE: Heterogeneous star $\mathcal{P}$, load size V, time interval T, non-zero charges $[f_1, \ldots, f_m]$ for using the processors, total cost $\overline{G}$.
QUESTION: Can load V be processed on $\mathcal{P}$ in time at most T and cost at most $\overline{G}$?

Theorem 4 *Problem DLSFPC is* **NP***-hard.*

Proof. The problem is based on the polynomial transformation of the PARTITION: $m = q, T = 1, \overline{G} = F, V = F, A_i = \frac{1}{e_i}, C_i = S_i = 0, f_i = e_i$, for $i = 1, \ldots, m$. Note that communications are timeless, and processors have one time unit for computations. Thus, the load processed is $V = \sum_{P_i \in \mathcal{P}'} \frac{1}{A_i} =$

$\sum_{P_i \in \mathcal{P}'} e_i$, where $P_i \in \mathcal{P}'$ is the set of activated processors. The cost of activating these processors is $\overline{G} = \sum_{P_i \in \mathcal{P}'} f_i = \sum_{P_i \in \mathcal{P}'} e_i$. Thus, if the cost is $\overline{G} \leq F$, and the size of processed load $V \geq F$, then a positive answer to PARTITION must exist. And vice versa, positive answer to PARTITION implies a positive answer to DLSFPC. □

MAXIMUM SPEED PROBLEM (MS)
INSTANCE: Heterogeneous star $\mathcal{P}$, time interval T, speed R.
QUESTION: Is there a subset $\mathcal{P}'$ of $\mathcal{P}$ with total speed at least R that can be activated in time at most T?

Theorem 5 *MS problem is* **NP***-hard.*

Proof. MS problem is in **NP** because NDTM must guess set $\mathcal{P}'$, of processors. Then it is enough to check if $\sum_{i \in \mathcal{P}'} S_i < T$, and $\sum_{i \in \mathcal{P}'} \frac{1}{A_i} > R$.

An instance of the MS Problem can be constructed on the basis of PARTITION instance in the following way: $m = q$; $A_i = \frac{1}{e_i}, C_i = 0, S_i = e_i$ for $i = 1, \ldots, q$. $R = F, T = F$. The instance can be constructed in polynomial time $O(q)$.

Suppose the answer to the PARTITION problem is positive. Then, there is set E' satisfying equation (8). If we activate the processors corresponding to the elements in set E', then their total speed is $\sum_{i \in E'} \frac{1}{A_i} = \sum_{i \in E'} e_i = F = R$. The time needed to activate these processors is $\sum_{i \in E'} S_i = \sum_{i \in E'} e_i = F = T$. Thus, the set of processors satisfying the conditions of MS exists.

On the other hand, let us assume that the answer to MS problem is positive. Hence, there is set $\mathcal{P}'$ such that $\sum_{i \in P'} \frac{1}{A_i} = \sum_{i \in P'} e_i \geq R = F$, and $\sum_{i \in P'} S_i = \sum_{i \in P'} e_i \leq T = F$. Consequently, $\sum_{i \in P'} e_i = F$ and the answer to the PARTITION problem is also positive. □

DLS WITH COMMUNICATION STARTUP TIMES (DLSCST)
INSTANCE: Heterogeneous star $\mathcal{P}$, load size V, time interval T, processing rates A_i, startup times S_i, are positive for all processors.
QUESTION: Can load V be processes on $\mathcal{P}$ in time at most T?

Conjecture 6 *Problem DLSCST is* **NP***-hard.*

We conjecture that problem DLSCST is **NP**-hard due to its similarity to MS problem: on one hand the activated processors must have sufficient speed to process given volume of load V, on the other hand their work time T is limited.

For the end of this section let us consider a special case of DLSCT. When $S_i = \frac{1}{A_i}, C_i = 0$, for $i = 1, \dots, m$, this problem can be solved in pseudopolynomial time. Though this case seems to be very peculiar from the practical point of view, but still it may give some insight into the combinatorial nature and the complexity of the problem.

Proposition 7 *DLSCT problem can be solved in pseudopolynomial time if* $S_i = \frac{1}{A_i}, C_i = 0$ *for* $i = 1, \dots, m$.

Proof. Consider formula (9). When $S_i = \frac{1}{A_i}$ for all i, then the load processed in time T is

$$
\begin{aligned}
&V = \\
&\sum_{i=1}^{m} \frac{x_i T}{A_i} - \sum_{1\le i\le j\le m}^{m} x_i x_j \frac{S_i}{A_j} = \\
&T\sum_{i=1}^{m} x_i S_i - \sum_{1\le i\le j\le m}^{m} x_i x_j S_i S_j = \\
&T\sum_{i=1}^{m} x_i S_i - \sum_{i=1}^{m} x_i S_i^2 - \sum_{1\le i<j\le m}^{m} x_i x_j S_i S_j = \\
&T\sum_{i=1}^{m} x_i S_i - \frac{1}{2}\sum_{i=1}^{m} x_i^2 S_i^2 - \sum_{1\le i<j\le m}^{m} x_i x_j S_i S_j - \frac{1}{2}\sum_{i=1}^{m} x_i S_i^2 - \frac{1}{2}T^2 + \frac{1}{2}T^2 = \\
&\frac{1}{2}T^2 - \frac{1}{2}(T - \sum_{i=1}^{m} x_i S_i)^2 - \frac{1}{2}\sum_{i=1}^{m} x_i S_i^2 \qquad (11)
\end{aligned}
$$

In deriving the above equation we used the observation that $x_i = x_i^2$ for $x_i \in \{0, 1\}$. Note that V does not depend on the sequence of the processor activations. It depends only on the set of used processors for which $x_i = 1$ because only these processors contribute to $\sum_{i=1}^{m} x_i S_i$, and $\sum_{i=1}^{m} x_i S_i^2$. The sequence of processor activation is immaterial for these sums.

The maximum load V can be found by calculating function $H(j, \tau)$ which is the minimum sum of $\sum_{i=1}^{j} x_i S_i^2$ such that $\sum_{i=1}^{j} x_i S_i = \tau$. Function $H(j, \tau)$ can be calculated using the following recursive equations:

$$
H(j,\tau) = \begin{cases} H(j-1,\tau) & \text{for} \quad \tau < S_j \\ \min \begin{cases} H(j-1,\tau), \\ H(j-1,\tau - S_j) + S_j^2 \end{cases} & \text{for} \quad \tau \ge S_j \end{cases} \qquad (12)
$$

for $j = 1, \ldots, m, \tau = 1, \ldots, T$. $H(0, \tau) = \infty$, for $\tau = 1, \ldots, T$, $H(j, 0) = 0$ for $j = 0, \ldots, m$. Then, the load processed for particular values of j, τ is $V(j, \tau) = \min\{0, \frac{1}{2}(T^2 - H(j, \tau) - (T - \tau)^2)\}$. The optimum load is found for $\tau' \in \{1, \ldots, T\}$ such that $V(m, \tau')$ is maximum. The set of processors to be exploited can be found by backtracking using equation (12), from the value of $H(m, \tau')$ corresponding with the optimum $V(m, \tau')$. A processor is used in computation if $H(j, \tau) = H(j - 1, \tau - S_j) + S_j^2$, then we backtrack recursively to $H(j - 1, \tau - S_j)$, and so on until locating $H(j', \tau') \in \{0, \infty\}$. This method can be implemented to run in time $O(mT)$. □

Based on formula (11) we can draw one more observation. The problem of maximization of $T^2 - (T - \sum_{i=1}^{m} x_i S_i)^2 - \sum_{i=1}^{m} x_i S_i^2$ has a geometric interpretation (see Fig.4). Suppose a square Y of area T^2 is given. The diagonal sequence of squares in Fig.4 is equivalent to $\sum_{i=1}^{m} x_i S_i^2$. These squares must fit in rectangle Y. The last square X has area $(T - \sum_{i=1}^{m} x_i S_i)^2$. Maximization of $T^2 - (T - \sum_{i=1}^{m} x_i S_i)^2 - \sum_{i=1}^{m} x_i S_i^2$ is equivalent to determining a subset of $\{S_1, \ldots, S_m\}$ such that the sum of the areas of the squares along the diagonal is minimal. To our best knowledge, the complexity of this problem, remains unknown.

insert Fig.4 here

4 Conclusions

In this paper we studied the problem of divisible load scheduling on a star network for the schedule length and the schedule cost criteria. It has been demonstrated that the optimum load distribution can be found in polynomial time by using linear programming, on condition that the set of used processors and the sequence of their activation are known. However, in many cases determining this set and the activation order is computationally hard.

References

[1] V.Bharadwaj, D.Ghose, V.Mani, Optimal Sequencing and Arrangement in Distributed Single-Level Tree Networks with Communication Delays, IEEE Transactions on Parallel and Distributed Systems, vol. 5, No. 9, 1994, 968-976.

[2] V.Bharadwaj, D.Ghose, V.Mani, T.Robertazzi, *Scheduling divisible loads in parallel and distributed systems*, IEEE Computer Society Press, Los Alamitos CA (1996)

[3] V.Bharadwaj, D.Ghose, T.Robertazzi: Divisible load theory: A new paradigm for load scheduling in distributed systems. *Cluster Computing*, vol.6, No.1, 2003, 7-17.

[4] J.Błażewicz, M.Drozdowski, Distributed processing of divisible jobs with communication startup costs, *Discrete Applied Mathematics*, vol. 76, Issue 1-3, 13 June 1997, 21-41.

[5] J.Błażewicz, M.Drozdowski, K.Ecker, Management of Resources in Parallel Systems, in: J.Błażewicz, K. Ecker, B. Plateau, D. Trystram, *Handbook on Parallel and Distributed Processing*, Springer, Heidelberg, 2000, 263-341.

[6] S.Charcranoon, T.Robertazzi, S.Luryi, Load sequencing for a parallel processing utility, Journal of Parallel and Distributed Computing 64, 2004, 29-35.

[7] M.Drozdowski: Selected problems of scheduling tasks in multiprocessor computer systems. Poznań University of Technology Press, Series: Monographs, No.321, Poznań (1997). Also: `http://www.cs.put.poznan.pl/~maciejd/txt/h.ps`

[8] M.Drozdowski, P.Wolniewicz, Optimum divisible load scheduling on heterogeneous stars with limited memory, 2002, submitted.

[9] M.R.Garey, D.S.Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman, San Francisco, 1979.

[10] T.Robertazzi, Ten reasons to use divisible load theory, *IEEE Computer*, vol. 36, No.5, 2003, 63-68.

[11] J.Sohn, T.Robertazzi, S.Luryi, Optimizing computing costs using divisible load analysis, *IEEE Transactions on Parallel and Distribute Systems*, vol 9, No. 3, 1998, 225-234.

[12] L.Xiaolin, Studies on Divisible Load Scheduling Strategies in Distributed Computing Systems: Design, Analysis and Experiments, PhD thesis, National University of Singapore, 2001.

Appendix. Notation

A_i - processing rate (reciprocal of speed) of P_i,
α_i - load assigned to P_i,
B_i - memory size of P_i,
C_i - communication rate (reciprocal of bandwidth) of the link to P_i,
$\underline{C_{max}} = \max\{t_i\}$ - schedule length,
$\overline{C_{max}}$ - an upper limit on schedule length,
d_i - deadline of P_i, upper limit of P_i availability for computations,
E - set of integers in PARTITION problem,
e_i - value of element i in PARTITION problem,
$F = \frac{1}{2}\sum_{i=1}^{q} e_i$ - a number defined for PARTITION problem,
f_i - fixed part of the cost of using processor P_i,
$\underline{G} = \sum_{i\in\mathcal{P}'}(f_i + \alpha_i l_i)$ - total cost of the schedule on processors in set $\mathcal{P}'$,
$\overline{G}$ - an upper limit on cost G
l_i - coefficient of the linear part of the cost of using P_i,
m - number of processing nodes,
$\mathcal{P}$ - set of available processing nodes,
$\mathcal{P}'$ - set of nodes participating in the computations,
P_i - processing element i,
p_i - computation startup time on processor P_i,
q - the number of elements in partition problem,
r_i - release time of P_i, lower limit of P_i availability for computations,
S_i - communication startup time of the link to P_i,
T - upper limit of the schedule length,
t_i - completion of the computations on P_i,
V - total load size.

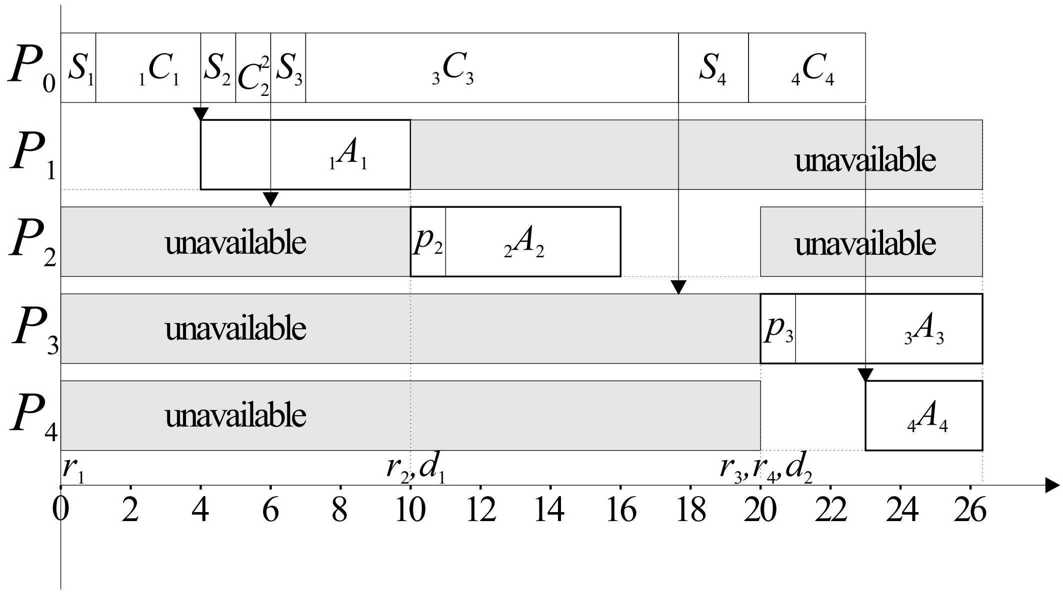

Figure 1: Schedule for the example with cost limit $G \geq 25.7669$.

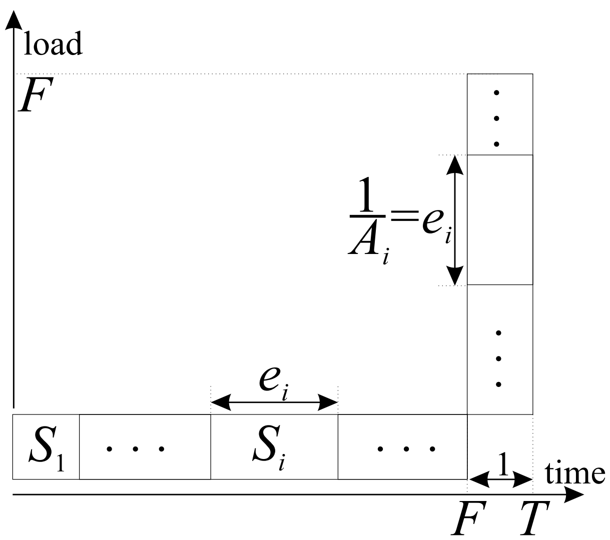

Figure 2: Illustration to the proof of Theorem 1

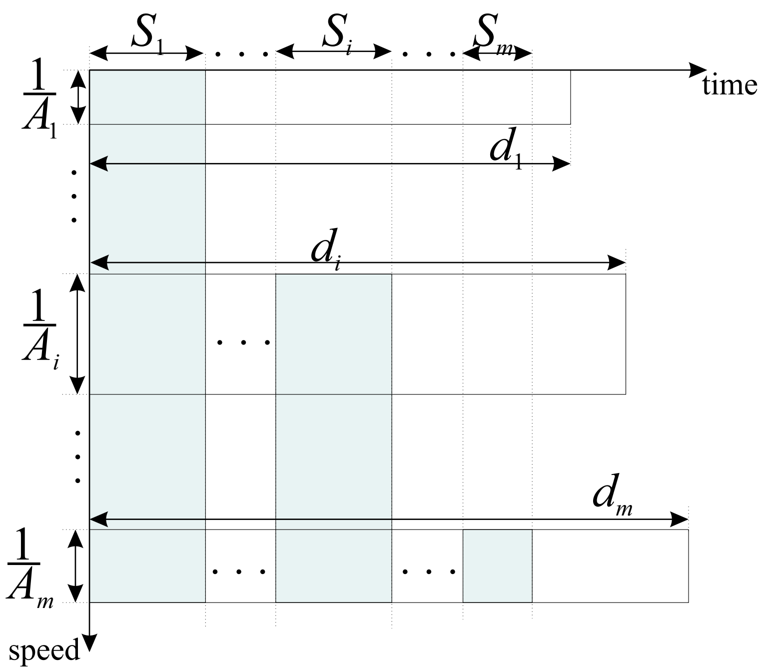

Figure 3: Illustration to the proof to Theorem 2

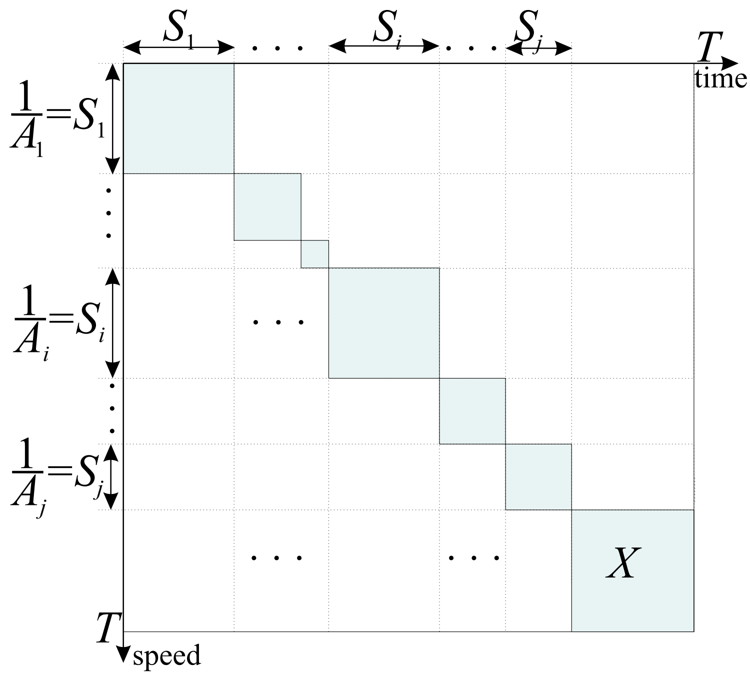

Figure 4: Illustration to Proposition 7