UC San Diego
Technical Reports

Title
NP-Completeness of the Divisible Load Scheduling Problem on Heterogeneous Star Platforms with Affine Costs

Permalink
https://escholarship.org/uc/item/2bn997rb

Authors
Legrand, Arnaud
Yang, Yang
Casanova, Henri

Publication Date
2005-03-10

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

NP-Completeness of the Divisible Load Scheduling Problem on Heterogeneous Star Platforms with Affine Costs

Arnaud Legrand[2] Yang Yang[1] Henri Casanova [1,3]
[1] Dept. of Computer Science and Engineering
University of California, San Diego
[2] Laboratoire Informatique et Distribution,
Institut d'Informatique et de Mathématiques Appliquées de Grenoble
[3] San Diego Supercomputer Center
University of California, San Diego

March 23, 2005

Abstract

In this paper we prove that the Divisible Load Scheduling (DLS) problem is NP-complete when the underlying distributed computing platform is a heterogeneous star network and when the costs for communicating and computing load chunks are affine functions of the chunk size. While the complexity of the DLS problem was known in the case of linear communication and computation costs (polynomial), or in the case of homogeneous platforms (polynomial), to the best of our knowledge it was still open for heterogeneous platforms with affine cost. We prove (weak) NP-completeness by reducing the DLS problem from 2-Partition, i.e., partition of a set of integer values into two sets whose sums are equal.

1 Introduction

Recent years have seen an increase in interest for the study of the *Divisible Load Scheduling* (DLS) problem [6] and its complexity [3], with a particular emphasis on master-worker platforms with a star topology. Consider an application that consists of an amount of work, or *load*, which can be arbitrarily divided into any number of "chunks", where each chunk consists of some amount of input data and some computation to perform on this data. The objective of the DLS problem is to assign load chunks to workers, which are accessible from a master over a network, so that the *makespan*, that is the overall application execution time, is minimized. The entire load initially resides at the master. It is known that the DLS problem is polynomial-time if the times to communicate and to compute a chunk are linear in the chunk size [3]. It is also known that if the computation and communication times, or "costs", are affine in the chunk size (thereby modeling start-up costs) and the platform is homogeneous, then the problem is polynomial-time [6]. In this paper we show that in the case of affine costs and of a heterogeneous platform the problem is NP-complete, which to the best of our knowledge is a new result. Furthermore the affine heterogeneous case corresponds to a more realistic and general platform model.

We define the DLS problem with AFfine costs (DLSAF) as follows:

Problem 1. *Consider a set of N worker processors and a master processor. The master must send out W_0 units of load to workers to be computed. Each worker can compute s_i, $i \in [\![1, N]\!]$, load units per second, and each transfer to a worker takes a fixed amount of time, β_i seconds. Is it possible to compute all W_0 load units within T_0 seconds after the master starts sending out the first load unit?* *(T_0, W_0, β_i, s_i are all* rational numbers*, due to the divisible nature of the load.)*

Note that in the problem definition above we use a linear model for the computation time, and a fixed model for the communication time. For the sake of simplicity we use infinite bandwidths and zero computation start-up costs, but it would be a purely technical matter to rewrite our proof with "very large" bandwidths and "very small" computation start-up costs. At any rate, Problem 1 as it is defined above is a special case of the more general problem with truly affine communication and computation costs, and we prove that it is NP-complete, thereby making the more general problem NP-complete. Intuitively, Problem 1 is difficult because, due to the affine costs, the total communication start-up times, $\sum_{1 \leqslant i \leqslant N} \beta_i$, may be larger than T_0. Therefore, we must use only a carefully chosen *subset* of the workers, giving the problem a combinatorial flavor.

We note two related previous works that have made some contribution for determining the complexity of the DLS problem on a heterogeneous star platform with affine costs. The work in [4] shows that the difficult combinatorial resource selection problem can be bypassed if one assumes that the total load is "sufficiently large". In this case, all the start-up costs are much smaller than the overall makespan, and all workers should be used. The authors in [4] state that when the "sufficiently large" assumption is removed, then the complexity of DLS problem is open. In [1], the author studied the DLS problem with added "buffer constraints", i.e., for each worker P_i at most b_i load units can be stored on that worker. This limitation essentially provides one more condition, which helps when reducing from known NP-complete problems, and NP-completeness is proven in [1]. In [4], the authors strengthen the result by proving that the DLS problem with buffer constraints is NP-complete in the strong sense. In this paper we consider the more general problem without the "sufficiently large load" assumptions and without buffer constraints, and we prove its NP-completeness.

2 Intuition Behind the Proof

In this section we present the intuition behind our proof, which is detailed in the next section. For an instance to satisfy DLSAF, it has to meet two broad requirements: (i) have a makespan lower than T_0, meaning that the sum of communication start-up costs of the selected workers must be *small enough*; and (ii) compute more than W_0 units of load, meaning that the compute speeds of the selected workers must be *large enough*. Those two requirements suggest that we should probably attempt a reduction from the 2-Partition problem. In the reduction from 2-Partition to DLSAF, we have the following variables at our disposal, which can be set freely to "force" the selection of workers: W_0, T_0, s_i, β_i. What we need to do is then to carefully choose a *small enough* T_0, and a *large enough* W_0. We also got some ideas from the proof of a set of related problems in [2].

3 NP-Completeness Proof

First it is easy to see that DLSAF is in NP: given a solution to an instance of DLSAF, we can verify in polynomial time that the subset of workers complete workload W_0 within time T_0 [5].

Then we prove that DLSAF is NP-hard via a reduction from the NP-complete 2-Partition problem. The 2-Partition problem [7] is defined as follows:

Problem 2. *Given a finite set A of integers α_i, $1 \leqslant i \leqslant 2m$, is there a subset $A' \subset A$ such that $|A'| = m$ and*

$$\sum_{\alpha \in A - A'} \alpha = \sum_{\alpha \in A'} \alpha = L?$$

Given an instance of 2-Partition, we construct an instance of DLSAF as follows. For each α_i we create a worker P_i, whose fixed communication cost is β_i, and whose computing speed is s_i, with $\beta_i = s_i = M + \alpha_i$. So together we have $N = 2m$ workers. We then choose $T_0 = mM + L + 1/2$, and $W_0 = \frac{m}{2}(m-1)M^2 + (m-1)LM + \frac{m}{2}M$. We must chose M above as a "large" number, and in our case it turns out that choosing $M = 8m^2L^2$ is sufficient.

Fig. 1 depicts a schedule with four workers with time on the horizontal axis (from time 0 to time T_0) and the four workers on the vertical axis. For each worker we show a communication phase (in white), followed by a computation phase (in various shades of gray, whose meaning will be explained shortly). Note that all workers finish computing at the same time. Indeed, there is no reason why a worker should stop working before time T_0. Furthermore, since we have a fixed cost for sending chunks over the network, one can easily send enough load to each worker to keep it busy until time T_0. Clearly, the number of load units that a worker contributes to the computation is the product of its computational speed, s_i, and the duration of the time interval from the end of communication to the overall application finish time. To make the proof simpler to follow, we

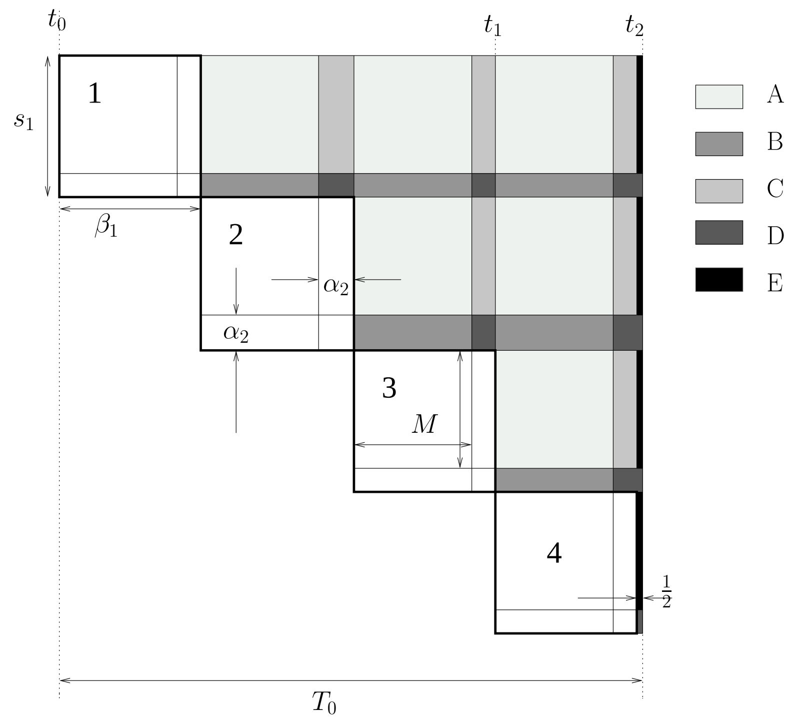

Figure 1: Illustration of proof.

let the width of each worker slot in the figure be s_i, so that the worker's contribution to the overall computation number of work units is given by **area** of its slot. In the proof, we will compute such areas in order to estimate numbers of computed load units. We prove the reduction with the usual two steps.

2-PARTITION ⇒ DLSAF If we have a solution to the 2-PARTITION problem, we show that we also have a solution to the DLSAF problem. Pick all m workers P_i such that $\alpha_i \in A$. For simpler notations, and without loss of generality, we assume that $i = 1, \ldots, m$. First, we note that the sum of the communication start-up costs is

$$\sum_{i=1}^{m} \beta_i = \sum_{i=1}^{m} (M + \alpha_i) = mM + L < T_0.$$

Consequently, all m workers can participate in the computation and appear in the schedule.

Given the set of workers participating in the schedule, we now estimate the number of load units that are computed before time T_0, which corresponds to the shaded area shown in Fig. 1. The figure shows the shaded area partitioned in five different types of rectangular zones. Zone A consists of $m(m-1)/2$ squares of dimension $M \times M$. Zone B consists of $m - i$ rectangles of dimension $M \times \alpha_i$ for $i = 1, \dots, m-1$, for a total of $m(m-1)/2$ such rectangles. Zone C is similar to zone B, but with $i-1$ rectangles of dimension $\alpha_i \times M$ for $i = 2 \dots, m$. Zone D, which visually corresponds to the intersections between rectangles in zones B and C, consists of $m(m-1)/2$ rectangles of dimensions $\alpha_j \times \alpha_i$, for $i = 1, \dots, m-1$, and $j \in [2, \dots, m]$ with $j > i$, and of m rectangles of dimension $\frac{1}{2}\alpha_i$, for $i = 1, \dots, m$. Finally, zone E consists of m rectangles of dimension $\frac{1}{2} \times M$.

We compute the sum of the areas of zones A, B, C, and E. The total area of zone A is clearly $M^2 m(m-1)/2$. The total area of zone B is:

$$M \times ((m-1)\alpha_1 + (m-2)\alpha_2 + \ldots + \alpha_{m-1}).$$

The total area of zone C is:

$$M \times (\alpha_2 + 2\alpha_3 + 3\alpha_4 + \ldots + (m-1)\alpha_m).$$

Finally the total area of zone E is $\frac{m}{2}M$. Because we have not counted the area due to rectangles in zone D, we can bound below the total number of computed load units, W, as follows:

$$\begin{aligned}
W &\geqslant \frac{m}{2}(m-1)M^2 + M\sum_{i=1}^{m-1} i\alpha_{i+1} + M\sum_{i=1}^{m-1}(m-i)\alpha_i + \frac{m}{2}M \\
&= \frac{m}{2}(m-1)M^2 + (m-1)M\sum_{i=1}^{m}\alpha_i + \frac{m}{2}M \\
&= \frac{m}{2}(m-1)M^2 + (m-1)ML + \frac{m}{2}M \\
&= W_0.
\end{aligned}$$

We conclude that we have a solution to the DLSAF problem.

$\boxed{\textbf{DLSAF} \Rightarrow \textbf{2-Partition}}$ If we have a solution to the DLSAF problem, we now show that we also have a solution to the 2-Partition problem. First, we know that in the solution to DLSAF we can not have more than m workers. Otherwise, the β_i startup costs would add up to a time larger than T_0. We can also see that we need at least $m-1$ workers. Indeed, a smaller number of workers will not suffice because the overall number of computed load units will be strictly lower than W_0. (Intuitively, we waste the opportunity to compute at least M^2 load units when using $m-2$ workers.) Therefore we must use either $m-1$ or m workers. However, we now prove that we cannot use $m-1$ workers.

Let us assume that the solution of DLSAF uses $m-1$ workers, and let us compute the total number of load units computed before time T_0. The intuition is that by not having the mth worker we miss its contribution by zone E, which makes the overall number of computed load units strictly lower than W_0. We count the area in two parts, the area before worker $m-1$ finishes communication (left of time instant t_1 in Fig. 1), and the area after that. For the first part, the squares in zone A sum up to $M^2(m-1)(m-2)/2$. Those in zones B and C sum up to $M(m-2)\sum_{i=1}^{m-1}\alpha_i$ as before. Rectangles in zone D left of t_1 take up $\sum_{1\leqslant i<j\leqslant m-1}\alpha_i\alpha_j < \frac{1}{2}(m-1)(m-2)(2L)^2 < 2m^2L^2$, because $\alpha_i \leqslant \sum_{i=1}^{m-1}\alpha_i \leqslant 2L$. The second part is easily computed as the area between t_1 and t_2:

$$\left(\sum_{i=1}^{m-1} s_i\right)\left(T_0 - \sum_{i=1}^{m-1}\beta_i\right).$$

We can add the two parts and obtain the total number of computed load units, W, as follows:

$$\begin{aligned}
W &< \left\{\frac{(m-1)(m-2)}{2}M^2 + M(m-2)\sum_{i=1}^{m-1}\alpha_i + 2m^2L^2\right\} + \\
&\quad \left\{\left[(m-1)M + \sum_{i=1}^{m-1}\alpha_i\right]\left[T_0 - \left((m-1)M + \sum_{i=1}^{m-1}\alpha_i\right)\right]\right\} \\
&= \frac{(m-1)m}{2}M^2 + L(m-1)M + 2m^2L^2 + \\
&\quad (L+\frac{1}{2})\sum_{i=1}^{m-1}\alpha_i - (\sum_{i=1}^{m-1}\alpha_i)^2 + \frac{1}{2}mM - \frac{1}{2}M \\
&< \left[\frac{(m-1)m}{2}M^2 + L(m-1)M + \frac{1}{2}mM\right] + \left[\left(2m^2L^2 + (L+\frac{1}{2})\sum_{i=1}^{m-1}\alpha_i\right) - \frac{1}{2}M\right] \\
&< \frac{(m-1)m}{2}M^2 + L(m-1)M + \frac{1}{2}mM \\
&= W_0.
\end{aligned}$$

Therefore, we cannot have a solution with $m-1$ workers, and we must use exactly m workers.

For the solution of DLSAF with m workers we can write that the makespan is lower than T_0 (otherwise it would not be a solution). The makespan is equal to the sum of the β_i series and T_0 is equal to $mM+L+\frac{1}{2}$. Therefore, we have:

$$\sum_{i=1}^{m}\beta_i \leqslant mM + L + \frac{1}{2}$$

Replacing β_i by its value, we obtain:

$$\sum_{i=1}^{m}\alpha_i \leqslant L + \frac{1}{2}.$$

Because $\sum_{i=1}^{m} \alpha_i$ is integer, we have

$$\sum_{i=1}^{m} \alpha_i \leqslant L. \tag{1}$$

We now estimate the total number of load units computed. The small area of zone D, denoted by $\mathcal{D}$, is

$$\sum_{1 \leqslant i < j \leqslant m} \alpha_i \alpha_j + \frac{1}{2} \sum_{1 \leqslant i \leqslant m} \alpha_i \leqslant \frac{1}{2} m(m-1)L^2 + \frac{1}{2} L,$$

because of Eq. 1. Then the total load is:

$$\begin{aligned} W &= \frac{m}{2}(m-1)M^2 + M(m-1)\sum_{i=1}^{m} \alpha_i + \mathcal{D} + \frac{1}{2} mM \\ &\leqslant \frac{m}{2}(m-1)M^2 + M(m-1)\sum_{i=1}^{m} \alpha_i + \frac{1}{2} mM + \frac{1}{2} m(m-1)L^2 + \frac{1}{2} L. \end{aligned}$$

Since the schedule is a solution of DLSAF, we also have:

$$W \geqslant W_0 = \frac{m}{2}(m-1)M^2 + (m-1)LM + \frac{1}{2} mM.$$

Therefore

$$M(m-1)\sum_{i=1}^{m} \alpha_i + \frac{1}{2} m(m-1)L^2 + \frac{1}{2} L \geqslant (m-1)LM,$$

which, given that $M = 8m^2L^2$, implies that:

$$\sum_{i=1}^{m} \alpha_i \geqslant L - \frac{1}{16m} - \frac{1}{(m-1)8m^2L} > L - 1.$$

Because $\sum_{i=1}^{m} \alpha_i$ is integer, we have:

$$\sum_{i=1}^{m} \alpha_i \geqslant L. \tag{2}$$

Eqs. (1) and (2) show that $\sum_{i=1}^{m} \alpha_i = L$. Therefore, there is a solution to the 2-Partition problem, which is obtained by picking all the α_i values that correspond to the workers participating in the computation in the solution for the DLSAF problem.

4 Pseudo-Polynomial Time Algorithm

Drozdowski and Lawenda [8] gave a pseudo-polynomial time algorithm for the DLSAF problem, thus showing that DLSAF is weakly NP-Complete. We also independently found a pseudo-polynomial time algorithm, given below as algorithm 3. Without loss of generality, we assume all quantities involved are integers. For a schedule depicted in Fig. 1, the total work-units computed in a schedule is

$$\langle \text{shaded area between } t_0 \text{ and } t_1 \rangle + \langle \text{shaded area between } t_1 \text{ and } t_2 \rangle. \tag{3}$$

The first term of Eq. 3 can be calculated in two steps: (1) First we find out the subset of workers whose sum of start-up costs is exactly the interval from t_0 to t_1. This can be solved by using the pseudo-polynomial time algorithm 1 for SUBSETSUM [7] (page 90). (This algorithm was originally given to solve PARTITION, but also works for SUBSETSUM.) We run algorithm 1 for every possible value of t_1. (2) Then, given a valid set of chosen workers, we calculate the work-units completed before t_1 by using algorithm 2, DSLAF_Given_Workers, from [5].

Then we calculate the second term as:

$$\left(\sum_{\substack{i|P_i \text{ is chosen} \\ \text{by SUBSETSUM}}} s_i \right) \cdot \left(T_0 - \sum_{\substack{i|P_i \text{ is chosen} \\ \text{by SUBSETSUM}}} \beta_i \right).$$

Algorithm 3 solves the DLSAF problem. It returns True if it finds a schedule that computes more than W_0, otherwise it returns False. The algorithm is correct because we have tried all valid schedules.

```
SUBSETSUM($t, \boldsymbol{\beta}$)
1: calculate $\boldsymbol{\beta}' \subset \boldsymbol{\beta}$ so that $\sum_{\beta_i \in \boldsymbol{\beta}'} \beta_i = t$
```

Algorithm 1: SUBSETSUM algorithm

```
DLSAF_GIVEN_WORKERS($t, \boldsymbol{\beta}$)
1: given a set of workers, whose start-up costs are $\boldsymbol{\beta}$, and $\sum_{\beta_i \in \boldsymbol{\beta}'} \beta_i \leqslant t$,
   calculate the work units completed within $t$ time units.
```

Algorithm 2: DLSAF_Given_Workers algorithm

5 Conclusion

Our proof has several implications. We now know that the resource selection problem for a heterogeneous star network with affine costs is a hard problem.

```
DLSAF($T_0, \boldsymbol{\beta}$)
1: for $t$=0 to $T_0$ do
2:   $(feasible, \boldsymbol{\beta}') \leftarrow$ SubsetSum $(t, \boldsymbol{\beta})$        $\{\boldsymbol{\beta}'$ is a sublist of $\boldsymbol{\beta}\}$
3:   if $feasible$ then
4:     $W \leftarrow$DLSAF_Given_Worker($t, \boldsymbol{\beta}'$)
           $+ \left(T_0 - \left(\sum_{\beta_i \in \boldsymbol{\beta}'} \beta_i\right)\right) \cdot \left(\sum_{\beta_i \in \boldsymbol{\beta}'} s_i\right)$
5:     if $W > W_0$ then
6:
7:       return True
8:
9: return False
```

Algorithm 3: Pseudo-polynomial algorithm to solve DLSAF

This justifies the use of heuristics such as the ones employed in [9]. It is important to note that our proof not only works for one-round schedules, but also for multi-round schedules. Indeed, our proof does not make any assumptions about the number of rounds (although in our particular instance using multiple rounds would not be effective). Finally, although we have proved that when bandwidth is infinite, DLSAF is weakly NP-Complete, it would be interesting to determine whether the more general problem, i.e. with finite bandwidth, is only weakly NP-hard or strongly NP-hard.

References

[1] M. Drozdowski and P. Wolfniewicz. On the Complexity of Divisible Job Scheduling with Limited Memory Buffers. Technical Report RA-001-2001, Institute of Computing Science, Poznań University of Technology, 2001.

[2] Maciej Drozdowski and M. Lawenda. The Combinatorics in Divisible Load Scheduling. Technical Report RA-012/04, Institute of Computing Science, Poznań University of Technology University of Technology, 04.

[3] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang. Scheduling Divisible Loads on Star and Tree Networks: Results and Open Problems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 16(3):207–218, 2005.

[4] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Independent and Divisible Task Scheduling on Heterogeneous Star-Shaped Platforms with Limited Memory. Technical Report RR2004-22, Ecole Normale Superieure de Lyon, April 2004.

[5] O. Beaumont, A. Legrand, and Y. Robert. Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems. Technical Report 2002-36, Ecole Normale Superieure de Lyon, October 2002.

[6] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, 1996.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman, 1991.

[8] Maciej Drozdowski and M. Lawenda. Pseudopolynomial Algorithms for Divisible Load Scheduling on Heterogeneous Stars, 2005. personal communication.

[9] Y. Yang and H. Casanova. UMR: a Multi-Round Algorithm for Scheduling Divisible Workloads. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2003)*, April 2003.