Journal of Scheduling (2021) 24:507–521
https://doi.org/10.1007/s10951-019-00626-6

Scheduling divisible loads with time and cost constraints

M. Drozdowski[1] · **N. V. Shakhlevich**[2]

Published online: 20 November 2019
© The Author(s) 2019

Abstract
In distributed computing, divisible load theory provides an important system model for allocation of data-intensive computations to processing units working in parallel. The main task is to define how a computation job should be split into parts, to which processors those parts should be allocated and in which sequence. The model is characterized by multiple parameters describing processor availability in time, transfer times of job parts to processors, their computation times and processor usage costs. The main criteria are usually the schedule length and cost minimization. In this paper, we provide the generalized formulation of the problem, combining key features of divisible load models studied in the literature, and prove its NP-hardness even for unrestricted processor availability windows. We formulate a linear program for the version of the problem with a fixed number of processors. For the case with an arbitrary number of processors, we close the gaps in the study of special cases, developing efficient algorithms for single criterion and bicriteria versions of the problem, when transfer times are negligible.

Keywords Divisible load scheduling · Computational complexity · Linear programming

1 Introduction

Divisible load theory (DLT) is an important model of parallel computations. It is assumed that a big volume of data, conventionally referred to as *load*, can be divided continuously into parts which can be processed independently on distributed computers. DLT was proposed by Cheng and Robertazzi (1988) to represent computation in a chain of intelligent sensors. A very similar approach was proposed independently by Agrawal et al. (1988) to model performance of a network of workstations. DLT was successfully applied to scheduling data-intensive applications and to analyzing various aspects of their efficiency depending on communication sequences, load scattering algorithms, memory limitations and time-varying environments (Bharadwaj et al. 1996; Drozdowski 2009; Robertazzi 2003). Overall, DLT is widely recognized as an adequate and accurate model of real systems dealing with load distribution.

✉ M. Drozdowski
Maciej.Drozdowski@cs.put.poznan.pl

[1] Institute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznan, Poland

[2] School of Computing, University of Leeds, Leeds LS2 9JT, UK

In its general form, the problem of divisible load scheduling (DLS) can be formulated as follows. A computational load of volume V (measured in bytes) is initially held by a *master processor* P_0. The load must be distributed among *worker processors* from set $\mathcal{P} = \{P_1, \ldots, P_m\}$. In our model master processor P_0 only distributes load and does not do any computation. In some publications this assumption is waived so that P_0 performs computation after it completes all communications. The results discussed in the following sections can be easily adjusted for that case.

For the summary of the notation and explanation of the parameters used in this paper, see Table 1. Each processor P_i has its own availability interval $[r_i, d_i]$. The time required for sending x bytes of load to P_i is $s_i + c_i x$, for $i = 1, \ldots, m$. The communications are performed sequentially, i.e., only one processor at a time can receive its chunk of the load from the master. The transmission to P_i of the allocated chunk can start at any time, even before the processor's availability time r_i. The processing of the chunk can start only after the allocated chunk is received in full and no earlier than processor's availability time r_i. For the chunk of size x received by processor P_i, the computation time and the processor usage cost (computation cost) are $p_i + a_i x$ and $f_i + \ell_i x$, respectively.

It is required that P_i finishes computation by the end of its availability interval d_i, $d_i > r_i + p_i$. Due to the limited

Table 1 Problem parameters and notations

V	Total load size
T	Deadline (upper limit of the schedule length)
K	Budget (upper limit on cost)
x_i	Decision variable for the size of the load chunk assigned to processor P_i
B_i	The maximum load processor P_i can compute, due to memory limitations
C_i	Finishing time for computing load x_i on processor P_i
$C_{\max}$	Schedule length defined as $\max\{C_i \mid i = 1, \ldots, m\}$
$\mathcal{K}$	Overall cost
$[r_i, d_i]$	The time window when processor P_i is available
$p_i + a_i x_i$	The time for computing load x_i on processor P_i, where p_i is the setup time to start computation and a_i is the processing rate (or reciprocal of speed) of processor P_i
$s_i + c_i x_i$	The time for transferring load x_i to processor P_i, where s_i is communication start up time and c_i is the communication rate (or reciprocal of bandwidth) of the link to P_i
$f_i + \ell_i x_i$	The cost of computing load x_i by processor P_i, including the fixed cost f_i
$\mathcal{P}$	Set of the worker processors
$\mathcal{P}'$	Set of processors participating in computation, $\mathcal{P}' \subseteq \mathcal{P}$
m	Total number of processors, i.e., $m = \lvert\mathcal{P}\rvert$
m'	Number of processors in $\mathcal{P}'$, i.e., $m' = \lvert\mathcal{P}'\rvert$

memory size, there is an upper limit B_i on the maximum load that can be handled by processor P_i. A processor may be left unused if no load is sent to it. Such a processor does not incur any time or cost overheads.

Let C_i denote the time when processor P_i completes its chunk, and let $C_{\max}$ denote the length of the whole schedule. The cost of processing the load is denoted $\mathcal{K}$. Solving the DLS problem requires three decisions:

Decision 1 choosing the subset of processors $\mathcal{P}' \subseteq \mathcal{P}$ for performing computation; for any processor $P_i \in \mathcal{P}'$ the allocated chunk size is nonzero ($x_i > 0$) and such a processor is called *active*;
Decision 2 choosing the sequence in which the master processor P_0 sends parts of the load to the processors in $\mathcal{P}'$;
Decision 3 splitting the total load of size V into chunks x_i, one for each processor $P_i \in \mathcal{P}'$, such that the schedule length $C_{\max}$ and the total cost $\mathcal{K}$ are minimum,

$$C_{\max} = \max_{i \in \mathcal{P}'} \{C_i\},$$

$$\mathcal{K} = \sum_{i \in \mathcal{P}'} (f_i + \ell_i x_i).$$

With respect to decision 3, the most general version is bicriterion: finding the Pareto-front $(C_{\max}, \mathcal{K})$ of non-dominated solutions in criteria $C_{\max}$ and $\mathcal{K}$. We denote that problem by $\mathrm{DL}_{\mathrm{bicrit}}$. Its two counterparts deal with minimizing one objective subject to the bounded value of the second objective:

- in problem $\mathrm{DL}_{\mathrm{time}}(K)$ the objective is to minimize $C_{\max}$ subject to $\mathcal{K} \leq K$, where K is an upper limit of the available budget,
- in problem $\mathrm{DL}_{\mathrm{cost}}(T)$ the objective is to minimize the cost $\mathcal{K}$ subject to $C_{\max} \leq T$, where T is an upper limit of the acceptable schedule length.

There are three types of *overheads* for any processor $P_i \in \mathcal{P}'$: transfer time (also called communication time) $s_i + c_i x$, computation time $p_i + a_i x$ and computation cost $f_i + \ell_i x$. We refer to parameters s_i, p_i and f_i as *fixed overheads* as they define fixed amounts of time and cost incurred if a nonzero chunk of load is allocated to a processor. These amounts are independent of the chunk size.

In this paper, we perform the complexity study of the formulated DLS problem focusing on the most general case with arbitrary processors' availability windows $[r_i, d_i]$ and arbitrary restrictions on processors' maximum loads B_i. The results are summarized in Table 2. In the column "Objectives" we specify how the two objectives, $C_{\max}$ and $\mathcal{K}$, are handled: single criterion problems deal with either $C_{\max}$ or $\mathcal{K}$, while notation $(C_{\max}, \mathcal{K})$ is used for the bicriteria problem in the space of objectives $C_{\max}$, $\mathcal{K}$. If one of the objectives is bounded, then the corresponding inequality is stated in the column "Conditions".

In the presence of all three types of overheads, the DLS problem is NP-hard even if all fixed overheads are negligible,

$$s_i = p_i = f_i = 0 \text{ for all } P_i \in \mathcal{P}, \tag{1}$$

and processors' restrictions are relaxed,

$$r_i = 0,\ d_i = B_i = \infty \text{ for all } P_i \in \mathcal{P}. \tag{2}$$

If in addition to (1)–(2) per-unit transfer costs are equal,

Table 2 Summary of the results

Transfer time	Comput. time	Cost	Conditions	Objectives	Results
$c_i x_i$	$a_i x_i$	$\ell_i x_i$	$C_{\max} \le T$, $\mathcal{K} \le K$		NP-complete, Sect. 3 even if $r_i = 0$, $d_i = B_i = \infty$
cx_i (common c)	$a_i x_i$	$\ell_i x_i$	$r_i = 0$, $d_i = B_i = \infty$	$(C_{\max}, \mathcal{K})$	$O(m^3)$ Shakhlevich (2013)
					$O(m)$, Sect. 3.2 if $c_1 \le c_2 \le \cdots \le c_m$ and $\frac{\ell_1}{c_1} \le \frac{\ell_2}{c_2} \le \cdots \le \frac{\ell_m}{c_m}$
$s_i + c_i x_i$	$p_i + a_i x_i$	$f_i + \ell_i x_i$	arb. r_i, d_i, B_i	$(C_{\max}, \mathcal{K})$	FPT w.r.t. m, Sect. 2
$c_i x_i$	$a_i x_i$	0	$r_i = 0$, $d_i = B_i = \infty$	$C_{\max}$	$O(m \log m)$ Bharadwaj et al. (1994) Bharadwaj et al. (1996) Blazewicz and Drozdowski (1997) (Processor seq. $c_1 \le c_2 \le \cdots \le c_m$)
$s + cx_i$ (common s, c)	$a_i x_i$	0	$r_i = 0$, $d_i = B_i = \infty$	$C_{\max}$	$O(m \log m)$ Blazewicz and Drozdowski (1997) (Processor seq. $a_1 \le a_2 \le \cdots \le a_m$)
s_i	$p_i + a_i x_i$	0	$r_i = 0$, $d_i = B_i = \infty$	$C_{\max}$	NP-hard, even if processor sequence is fixed Drozdowski and Lawenda (2005)
s_i	$a_i x_i$	0	$r_i = 0$, $d_i = B_i = \infty$	$C_{\max}$	NP-hard Yang et al. (2007) $O(m \log(Vmas) \times \min\{\lfloor s + Va \rfloor, S\})^{*)}$
0	$p_i + a_i x_i$	$\ell_i x_i$	arb. r_i, d_i, B_i		
			$C_{\max} \le T$	$\mathcal{K}$	$O(m)$, Sect. 4.1
			$\mathcal{K} \le K$	$C_{\max}$	$O(m \log m)$, Sect. 4.1
				$(C_{\max}, \mathcal{K})$	$O(m \log m)$, Sect. 4.1
0	$a_i x_i$	f_i	$C_{\max} \le T$, $\mathcal{K} \le K$		NP-complete even if $r_i = 0$, $d_i = B_i = \infty$,Drozdowski and Lawenda (2005), Sect. 4.2
0	$p_i + a_i x_i$	$f_i + \ell_i x_i$	arb. r_i, d_i, B_i; fixed set of active processors		
			$C_{\max} \le T$	$\mathcal{K}$	$O(m)$, Sect. 4.2
			$\mathcal{K} \le K$	$C_{\max}$	$O(m \log m)$, Sect. 4.2
				$(C_{\max}, \mathcal{K})$	$O(m \log m)$, Sect. 4.2

$a = \max_{i=1,\ldots,n}\{a_i\}$, $s = \max_{i=1,\ldots,n}\{s_i\}$, $S = \Sigma_{i=1}^{m} s_i$

$$c_i = c \quad \text{for all} \quad P_i \in \mathcal{P},$$

the problem is solvable in $O(m^3)$ time even in the bicriteria setting (Shakhlevich 2013). As we show in this paper, the general case with arbitrary s_i, c_i, p_i, a_i, f_i, ℓ_i, r_i, d_i, B_i can be solved via linear programming under the condition that the number of worker processors m is fixed.

While the computation time overhead is at the center of the DLS problem and cannot be ignored, the two other types of overheads may become negligible in some scenarios. The version of the problem with zero cost overheads is well studied, see Bharadwaj et al. (1994), Bharadwaj et al. (1996), Blazewicz and Drozdowski (1997), Drozdowski and Lawenda (2005), Yang et al. (2007) and the summary of the results in the second part of Table 2. In this paper, we analyze the alternative version with zero transfer overheads; see the lower part of Table 2. It appears that if fixed cost overheads are negligible ($f_i = 0$ for all $P_i \in \mathcal{P}$), then the bicriteria version of the problem is solvable in $O(m \log m)$ time. Its single criterion counterpart of cost minimization subject to a bounded schedule length can be solved in $O(m)$ time. The version with nonzero fixed cost overheads f_i is NP-hard, but

can be solved in $O(m)$ time provided that the set of active processors is fixed.

Further organization of this paper is as follows. In Sect. 2 we study the general version of the problem, with arbitrary values of all parameters $s_i, c_i, p_i, a_i, f_i, \ell_i, r_i, d_i$ and B_i for all processors $P_i \subseteq \mathcal{P}$. In Sect. 3 we present our results for the case with zero fixed overheads, $s_i = p_i = f_i = 0$ for all processors $P_i \subseteq \mathcal{P}$. Section 4 is dedicated to the system with negligible transfer times, $s_i = c_i = 0$ for all $P_i \subseteq \mathcal{P}$. Conclusions are presented in Sect. 5.

2 Nonzero time/cost parameters—fixed set of active processors

In this section, we consider the DLS problem with arbitrary time/cost parameters $s_i, c_i, p_i, a_i, f_i, \ell_i$ and arbitrary processor availability parameters r_i, d_i, B_i. The number of worker processors m is fixed. We present linear programs for problems $\mathrm{DL}_{\mathrm{time}}(K)$ and $\mathrm{DL}_{\mathrm{cost}}(T)$, justifying that both problems are fixed parameter tractable (FPT) with respect to the parameter m. We then explain how problem $\mathrm{DL}_{\mathrm{bicrit}}$ can be solved in FPT time. Note that for an arbitrary m the problem is NP-hard, as we show in Sect. 3.

2.1 Limited cost *K*—schedule length minimization

Consider first problem $\mathrm{DL}_{\mathrm{time}}(K)$, assuming that the set of processors $\mathcal{P}' \subseteq \mathcal{P}$, which receive nonzero chunks of the load, is fixed, and their sequence is also fixed. At the end of the section, we discuss the case with a non-fixed processor sequence.

Let processors in $\mathcal{P}'$ be renumbered in the order of their activation so that P_1 receives the first chunk, P_2 receives the second one, etc., until $P_{m'}$ receives the last chunk of the load, where $m' = |\mathcal{P}'|$. Let $x_1, x_2, \ldots, x_{m'}$ represent the load distribution among processors $\mathcal{P}'$. Then, the completion time C_i of any processors P_i, $1 \le i \le m'$, can be calculated as

$$C_i = \max\left\{r_i, \sum_{k=1}^{i}(s_k + c_k x_k)\right\} + (p_i + a_i x_i)\,. \tag{3}$$

Note that the first term in (3) represents the earliest possible starting time of the computation: the release time of P_i or the total duration of the chain of communication times for the upstream processors $P_1, P_2, \ldots, P_i$, whichever is larger. The second term represents the computation time.

Using (3), we follow Drozdowski and Lawenda (2005) to formulate problem $\mathrm{DL}_{\mathrm{time}}(K)$ as a linear program $\mathrm{LP}_{\mathrm{time}}(K)$ of the form:

$$\mathrm{LP}_{\mathrm{time}}(K): \qquad \min T \tag{4}$$

s.t.

$$\sum_{k=1}^{i}(s_k + c_k x_k) + (p_i + a_i x_i) \le T, \quad i = 1, \ldots, m', \tag{5}$$

$$r_i + (p_i + a_i x_i) \le T, \quad i = 1, \ldots, m', \tag{6}$$

$$\sum_{k=1}^{i}(s_k + c_k x_k) + (p_i + a_i x_i) \le d_i, \quad i = 1, \ldots, m', \tag{7}$$

$$r_i + (p_i + a_i x_i) \le d_i, \quad i = 1, \ldots, m', \tag{8}$$

$$0 \le x_i \le B_i, \quad i = 1, \ldots, m', \tag{9}$$

$$\sum_{i=1}^{m'}(f_i + \ell_i x_i) \le K, \tag{10}$$

$$\sum_{i=1}^{m'} x_i = V. \tag{11}$$

Here schedule length T is the variable to be minimized. It is defined via inequalities (5)–(6), which model $T = \max_{1 \le i \le m'}\{C_i\}$ with C_i given by (3). Inequalities (7)– (8) guarantee that computation on machine P_i is completed by the end of the machine availability interval. Inequalities (9) guarantee that the load of each processor P_i does not exceed its memory size B_i. By (10) the total computation cost does not exceed K. The total size of the load allocated to m' processors is equal to V by (11).

There are $m' + 1$ variables and $5m' + 2$ constraints in $\mathrm{LP}_{\mathrm{time}}(K)$, not counting the nonnegativity constraints. The number of variables and constraints can be further reduced by one, using equation (11). The number of constraints can be further reduced by m' by combining conditions (8)– (9) for every $1 \le i \le m'$ into

$$0 \le x_i \le \min\left\{B_i, \tfrac{1}{a_i}(d_i - r_i - p_i)\right\}.$$

Thus, problem $\mathrm{DL}_{\mathrm{time}}(K)$ can be solved in $O(\mathrm{LP}(m', 4m' + 1))$ time, where $O(\mathrm{LP}(u, w))$ is the time complexity of solving an LP problem with u variables and w inequality constraints, see, e.g., Goldfarb and Todd (1989).

With $m' \le m$, there are at most 2^m possible selections for set $\mathcal{P}'$ and at most $m!$ processor sequences for each selection. Hence, problem $\mathrm{DL}_{\mathrm{time}}(K)$ can be solved in $O(\eta \times LP(m, 4m + 1))$ time, where

$$\eta = 2^m m! \tag{12}$$

which implies the FPT time complexity with respect to parameter m.

2.2 Limited schedule length *T*—cost minimization

Consider now problem $\mathrm{DL}_{\mathrm{cost}}(T)$ of finding the cheapest schedule for the common deadline T, assuming that the set of active processors $\mathcal{P}' \subseteq \mathcal{P}$ is fixed and their sequence

is $(1, 2, \ldots, m')$. Similarly to problem $\mathrm{DL}_{\mathrm{time}}(K)$, problem $\mathrm{DL}_{\mathrm{cost}}(T)$ can be solved by a linear program with objective $\sum_{k=1}^{m'}(f_i + \ell_i x_i)$, which is equivalent to minimizing $\sum_{k=1}^{m'} \ell_i x_i$:

$$\mathrm{LP}_{\mathrm{cost}}(T): \quad \min \sum_{i=1}^{m'} \ell_i x_i \tag{13}$$

s.t.

$$\sum_{k=1}^{i} (s_k + c_k x_k) + (p_i + a_i x_i) \le T, \quad i = 1, \ldots, m', \tag{14}$$

$$r_i + (p_i + a_i x_i) \le T, \quad i = 1, \ldots, m', \tag{15}$$

$$\sum_{k=1}^{i} (s_k + c_k x_k) + (p_i + a_i x_i) \le d_i, \quad i = 1, \ldots, m', \tag{16}$$

$$r_i + (p_i + a_i x_i) \le d_i, \quad i = 1, \ldots, m', \tag{17}$$

$$0 \le x_i \le B_i, \quad i = 1, \ldots, m', \tag{18}$$

$$\sum_{i=1}^{m'} x_i = V. \tag{19}$$

Simplifying the model, we combine (14) with (16) and also combine (15), (17), (18) to get

$$\mathrm{LP}_{\mathrm{cost}}(T): \quad \min \sum_{i=1}^{m'} \ell_i x_i \tag{20}$$

s.t.

$$\sum_{k=1}^{i} (s_k + c_k x_k) + (p_i + a_i x_i) \le \min\{T, d_i\},$$

$$i = 1, \ldots, m', \tag{21}$$

$$0 \le x_i \le \min\{B_i, \tfrac{1}{a_i}(\min\{T, d_i\} - r_i - p_i)\},$$

$$i = 1, \ldots, m', \tag{22}$$

$$\sum_{i=1}^{m'} x_i = V. \tag{23}$$

The number of variables and constraints can be further reduced by one, using Eq. (23). In the resulting LP, there are $m' - 1$ variables and $2m'$ constraints so that problem $\mathrm{DL}_{\mathrm{cost}}(T)$ can be solved in $O(\eta \times \mathrm{LP}(m - 1, 2m))$ time, where η is given by (12).

2.3 Time–cost trade-off

Consider now the bicriteria problem $\mathrm{DL}_{\mathrm{bicrit}}$. The approach described below constructs at most η trade-offs, one for each fixed processor activation sequence, and then takes the lower envelope out of them.

For a fixed sequence $\mathcal{S}_i$ with m' processors, $m' \le m$, the trade-off consists of linear pieces that connect breakpoints

$$(T^0, K^0), (T^1, K^1), \ldots, (T^q, K^q),$$

$$T^0 \le T^1 \le \cdots \le T^q,$$

$$K^0 \ge K^1 \ge \cdots \ge K^q.$$

The schedule corresponding to the extreme point (T^0, K^0) has the smallest length T^0, that can be found by solving $\mathrm{LP}_{\mathrm{time}}(\infty)$ with $K = \infty$ or equivalently with inequality (10) eliminated from the model. For the calculated T^0, the associated minimum cost K^0 can be found by solving $\mathrm{LP}_{\mathrm{cost}}(T^0)$.

Another extreme point (T^q, K^q) corresponds to the schedule with the smallest cost K^q. It can be found by solving $\mathrm{LP}_{\mathrm{cost}}(\infty)$ with $T = \infty$; then for the found value of K^q the associated minimum schedule length T^q can be found by solving $\mathrm{LP}_{\mathrm{time}}(K^q)$.

The remaining points (T^i, K^i), $1 \le i \le q - 1$, of the trade-off for the fixed processor activation sequence $\mathcal{S}_i$ can be found as the solution to the parametric linear programming problem $\mathrm{LP}_{\mathrm{cost}}(T)$ of type (13)–(19) with variable $T \in [T^0, T^q]$ in constraints (14)–(15) treated as a parameter. Again, the model can be simplified so that the resulting formulation has $m' - 1$ x-variables (after eliminating one variable using (19)) and $4m'$ constraints (after combining (17) with (18) and eliminating the equality constraint (19)). Using standard methods of parametric optimization, the breakpoints of the trade-off can be found in $O(q\mathrm{LP}(m - 1, 4m))$ time, see, e.g., Adler and Monteiro (1992). The number of breakpoints q is bounded by the number of basic feasible solutions, which does not exceed $\binom{4m}{m-1}$ for a linear program with $m - 1$ variables and $4m$ constraints. Using an upper bound

$$Q = 4^m m^m, \tag{24}$$

for the latter expression we conclude that the time complexity for constructing the trade-off for the fixed processor activation sequence $\mathcal{S}_i$ of m' processors, $m' \le m$, is $O(Q \times \mathrm{LP}(m - 1, 4m))$. With an upper bound η on the number of processor sequences given by (12), the overall time complexity for constructing η trade-off curves is $O(\eta Q \times \mathrm{LP}(m - 1, 4m)) = O(8^m m^m m! \times \mathrm{LP}(m - 1, 4m))$.

We have to construct a merged set of Pareto-optimum points in the (T, K)-space for all processor sequences $\mathcal{S}_i$. The resulting set of points $\mathcal{F}$ and the segments connecting them constitute a non-increasing function of cost K in time T. This function may contain convex and concave parts, and it can be discontinuous, see Fig. 1. A point of discontinuity $(t, K(t))$ corresponds to, e.g., the left end of a trade-off curve for some processor sequence $\mathcal{S}_i$, while another processor sequence $\mathcal{S}_j$ incurs a higher cost at t, or the right end of $\mathcal{S}_j$ trade-off curve is earlier than t.

A possible approach to finding the overall trade-off may consist of the following two steps.

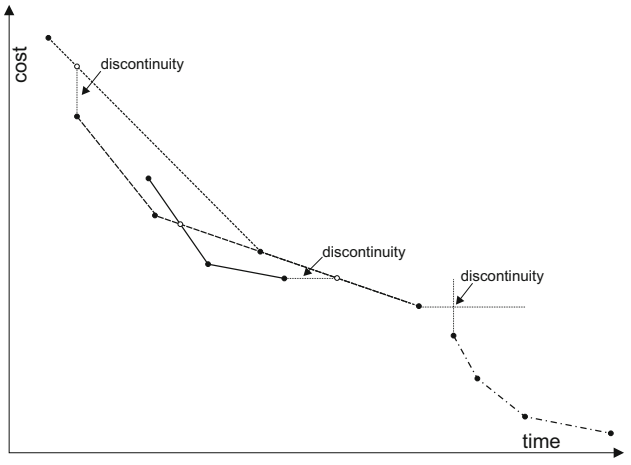

Fig. 1 Merging trade-off curves into the Pareto-front

(1) Find the intersection points of all pieces of trade-off curves defined for various processor sequences;
(2) Find the minimal layer of breakpoints and the intersection points, using, for example an approach outlined in Cormen et al. (2001) (p. 1045) for the symmetric problem of finding the maximal layer. Note that given z points in the plane, the minimal or maximal layer can be found in $O(z \log z)$ time.

To handle the discontinuity, introduce vertical lines from left end-points of individual trade-offs, horizontal lines from the right end-points, and include those pieces in Step (1). Then the overall number of linear pieces considered in Step (1) is not larger than $\eta\,(Q+2)$, their $O(\eta^2 Q^2)$ intersection points can be found in $O(\eta^2 Q^2)$ time, see, e.g., Cormen et al. (2001), and finally the minimal layer is constructed in $O(\eta^2 Q^2 \log(\eta Q))$ time. Taking into account expressions (12) and (24) for η and Q and the time complexity $O\,(8^m m^m m! \times \mathrm{LP}(m-1, 4m))$ for constructing all individual trade-offs, we conclude that the time complexity of solving problem $\mathrm{DL}_{\mathrm{bicrit}}$ is polynomially bounded for fixed m.

Statement 1 *Problems* $\mathrm{DL}_{time}(K)$, $\mathrm{DL}_{cost}(T)$, DL_{bicrit} *are fixed parameter tractable with respect to the number of machines* m.

3 Nonzero time/cost parameters—zero fixed overheads: $p_i = s_i = f_i = 0$

In this section, we assume that all fixed overheads are equal to zero, i.e., $p_i = s_i = f_i = 0$, for $i = 1, \ldots, m$, while the linear components of transfer time (c_i), computation time (a_i) and cost (ℓ_i) are not simultaneously equal to zero. In terms of the three types of decisions introduced in Sect. 1, decisions of type 3 imply decisions of type 1: any processor P_i which gets a chunk $x_i = 0$ does not contribute to any time or cost component because there are no fixed overheads. This implies that a processor receiving a 0-size chunk can be removed from the list $\mathcal{P}'$ of active processors. Thus, it is enough to make decisions 2 and 3. Our main result of this section is the **NP**-hardness proof of $\mathrm{DL}_{\mathrm{cost}}(T)$ and $\mathrm{DL}_{\mathrm{time}}(K)$.

3.1 Limited schedule length T and limited cost K

To prove **NP**-hardness of problem $\mathrm{DL}_{\mathrm{cost}}(T)$, let us introduce its decision version $\mathrm{DL}(T, K)$ which verifies whether there exists a feasible solution with the schedule length and the cost not exceeding the given thresholds T and K, respectively. We reduce the EVEN-ODD PARTITION to problem $\mathrm{DL}(T, K)$.

EVEN-ODD PARTITION is defined as follows: given a set $E = \{e_1, \ldots, e_{2n}\}$ of positive integers, is there a subset $E_1 \subset E$ such that $\sum_{e_i \in E_1} e_i = \sum_{e_i \in E \setminus E_1} e_i = G$ and E_1 contains exactly one element from pair e_{2i-1}, e_{2i}, for $i = 1, \ldots, n$? For an instance of EVEN-ODD PARTITION, we construct an instance of $\mathrm{DL}(T, K)$ as follows. Let $T > 0$ be an arbitrary schedule length and $m = 2n$ be the number of processors of the set $\mathcal{P}$. The processor parameters are defined as $r_i = 0$, $d_i = B_i = \infty$ for $i = 1, \ldots, 2n$, and

$$\begin{aligned} a_{2i-1} = c_{2i-1} &= \frac{T}{2^{2i-1}\left(G^{n-i+2} + e_{2i-1}\right)}, \\ \ell_{2i-1} &= \frac{e_{2i-1}}{G^{n-i+2} + e_{2i-1}}, \\ a_{2i} \;=\; c_{2i} &= \frac{T}{2^{2i-1}\left(G^{n-i+2} + e_{2i}\right)}, \\ \ell_{2i} &= \frac{e_{2i}}{G^{n-i+2} + e_{2i}}, \end{aligned} \tag{25}$$

for $i = 1, \ldots, n$. The load size V and the cost limit K are given by

$$V = \frac{3}{2}\sum_{i=1}^{n+1} G^i, \quad K = \frac{3}{2}G.$$

It is easy to see that the reduction is polynomial.

Lemma 1 *If there exists a solution to an instance of* EVEN-ODD PARTITION, *then there exists a schedule* S *for the instance* $\mathrm{DL}(T, K)$ *which satisfies the following properties.*

(i) *The whole load of size* V *is fully processed.*
(ii) *Every processor is fully loaded completing its load chunk at time* T.
(iii) *The cost of the schedule is* K.
(iv) *Schedule* S *defines a solution to the instance of* $\mathrm{DL}(T, K)$.

Proof Let e_{i1} denote the element of the pair $\{e_{2i-1}, e_{2i}\}$ which belongs to E_1, and e_{i2} be the other element of the pair, $i = 1, \ldots, n$. Construct schedule S by selecting processor activating sequence $(P_{11}, P_{12}, \ldots, P_{u1}, P_{u2}, \ldots, P_{n1}, P_{n2})$, where processors P_{i1}, P_{i2} correspond to e_{i1}, e_{i2}, respectively, and define the load chunks as

$$x_{i1} = G^{n-i+2} + e_{i1}, \quad x_{i2} = \frac{1}{2}\left(G^{n-i+2} + e_{i2}\right) \; i = 1, \ldots, n. \tag{26}$$

We prove that properties (i)–(iv) hold for schedule S.
(i) The sum of all chunks allocated to processors $\mathcal{P}$ is as follows:

$$\begin{aligned}\text{Load} &= \sum_{i=1}^{n}(x_{i1} + x_{i2}) \\ &= \sum_{i=1}^{n}\left(G^{n-i+2} + e_{i1} + \frac{1}{2}G^{n-i+2} + \frac{1}{2}e_{i2}\right) \\ &= \frac{3}{2}\sum_{i=1}^{n} G^{n-i+2} + \frac{1}{2}\sum_{i=1}^{n}(e_{i1} + e_{i2}) + \frac{1}{2}\sum_{i=1}^{n} e_{i1} \\ &= \frac{3}{2}\sum_{i=1}^{n} G^{n-i+2} + G + \frac{1}{2}G = V. \end{aligned} \tag{27}$$

(ii) For any processor P_{i1}, $1 \le i \le n$, its communication time μ_{i1} and computation time ν_{i1} are given by

$$\begin{aligned}\mu_{i1} = \nu_{i1} &= \frac{T}{2^{2i-1}\left(G^{n-i+2} + e_{i1}\right)} \times \left(G^{n-i+2} + e_{i1}\right) \\ &= \frac{T}{2^{2i-1}}.\end{aligned}$$

Similarly, for any processor P_{i2} the associated values are

$$\begin{aligned}\mu_{i2} = \nu_{i2} &= \frac{T}{2^{2i-1}\left(G^{n-i+2} + e_{i2}\right)} \times \frac{1}{2}\left(G^{n-i+2} + e_{i2}\right) \\ &= \frac{T}{2^{2i}}.\end{aligned}$$

Thus, the schedule is of the form shown in Fig. 2, with every processor P_{ij} completing at time T, $i = 1, \ldots, n$, $j = 1, 2$.
(iii) The cost of S is as follows:

$$\begin{aligned}\text{Cost} &= \sum_{i=1}^{n}(\ell_{i1}x_{i1} + \ell_{i2}x_{i2}) \\ &= \sum_{i=1}^{n}\left(\frac{e_{i1}}{G^{n-i+2} + e_{i1}} \times (G^{n-i+2} + e_{i1})\right. \\ &\quad \left. + \frac{e_{i2}}{G^{n-i+2} + e_{i2}} \times \frac{1}{2}(G^{n-i+2} + e_{i2})\right) \\ &= \sum_{i=1}^{n}\left(e_{i1} + \frac{1}{2}e_{i2}\right) = \frac{3}{2}G = K. \end{aligned} \tag{28}$$

By properties (i)–(iii) the schedule is feasible and it defines a solution to DL(T, K) so that property **(iv)** holds. □

In the remaining part we prove that if there exists a solution to the instance of problem DL(T, K), then there exists a solution to the related instance of EVEN-ODD PARTITION. The lemma below starts with auxiliary properties of a feasible schedule and concludes with the main result.

Lemma 2 *A feasible schedule S for the instance of* DL(T, K) *satisfies the following properties.*

(1) *Schedule S can be transformed into a schedule with processor activating sequence* $(\{P_{11}, P_{12}\}, \ldots, \{P_{u1}, P_{u2}\}, \{P_{u+1,1}, P_{u+1,2}\}, \ldots, \{P_{n1}, P_{n2}\})$.
(2) *Consider a feasible schedule obtained by transformation (1). Renumber processors in the order they appear in the activating sequence and renumber the associated values e_{u1} and e_{u2} accordingly. For the resulting schedule, with processor activating sequence $(P_{11}, P_{12}, \ldots, P_{u1}, P_{u2}, \ldots, P_{n1}, P_{n2})$, the following inequality holds:*

$$\sum_{i=1}^{n} 2^{2i} G^{n-i+2} y_i \ge 3\sum_{i=2}^{n+1} G^i,$$

where

$$\begin{aligned} y_i &= \frac{1}{T}(c_{i1}x_{i1} + c_{i2}x_{i2}) \\ &= \frac{1}{2^{2i-1}}\left(\frac{x_{i1}}{G^{n-i+2} + e_{i1}} + \frac{x_{i2}}{G^{n-i+2} + e_{i2}}\right), \end{aligned} \tag{29}$$

for $i = 1, \ldots, n$.
(3) *If in a feasible schedule satisfying property 2) at least one processor P_{uk}, $1 \le u \le n$, $k = 1, 2$, is not fully loaded (i.e., $C_{uk} < T$ holds), then*

$$\sum_{i=1}^{n} 2^{2i} G^{n-i+2} y_i < 3\sum_{i=2}^{n+1} G^i.$$

(4) *Each of the $2n$ processors is fully loaded and has completion time T.*
(5) *Equality $\sum_{i=1}^{n} e_{i1} = G$ holds so that the set $\{e_{i1}\}_{i=1}^{n}$ defines a solution to* EVEN-ODD PARTITION.

Proof **(1)** Let schedule S be of the form $S = \left(P_{h_1}, P_{h_2}, \ldots, P_{h_{2n}}\right)$ and P_{h_z} be an out-of-order processor

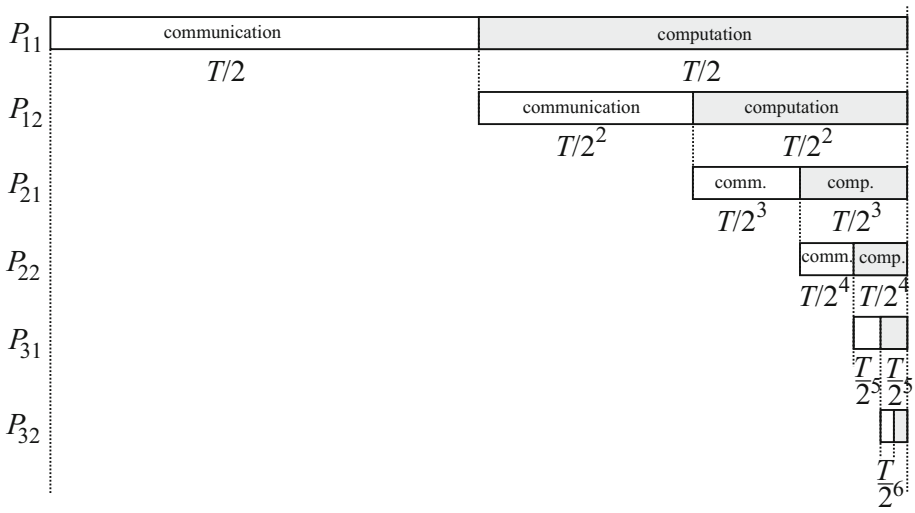

Fig. 2 A feasible schedule with processor sequence $(P_{11}, P_{12}, P_{21}, P_{22}, P_{31}, P_{32})$

such that it belongs to a pair $\{P_{u1}, P_{u2}\}$ with the smallest index u, and its predecessor in S is $P_{h_{z-1}} \in \{P_{v1}, P_{v2}\}$ with $v > u$. Notice that

$$c_{h_{z-1}} > 2c_{h_z} \tag{30}$$

since

$$\begin{aligned}
&\frac{c_{h_{z-1}}}{c_{h_z}} \\
&= \frac{T}{2^{2v-1}\left(G^{n-v+2} + e_{h_{z-1}}\right)} \Big/ \frac{T}{2^{2u-1}\left(G^{n-u+2} + e_{h_z}\right)} \\
&= \frac{G^{n-u+2} + e_{h_z}}{2^{2(v-u)}\left(G^{n-v+2} + e_{h_{z-1}}\right)} \\
&> \frac{G^{n-u+2}}{2^{2(v-u)+1}G^{n-v+2}} = \frac{G^{v-u}}{2^{2(v-u)+1}} > 2,
\end{aligned}$$

where the first inequality holds since $e_{h_z} > 0$ and $e_{h_{z-1}} < G^{n-v+2}$, and the second one is satisfied for a sufficiently large G, for example $G > 2^4$.

Let t be the starting time of communication of processor $P_{h_{z-1}}$ in S. Modify the fragment of schedule S, starting from t, by moving the full load $x_{h_{z-1}}$ from $P_{h_{z-1}}$ to P_{h_z}. In the new schedule, processor P_{h_z} finishes communication at time $t + c_{h_z}(x_{h_{z-1}} + x_{h_z})$, which is less than $t + c_{h_{z-1}}x_{h_{z-1}} + c_{h_z}x_{h_z}$, the communication finish time of P_{h_z} in S (since $c_{h_{z-1}} > c_{h_z}$). The same is true for the computation completion time: the new completion time of P_{h_z} is $t + 2c_{h_z}(x_{h_{z-1}} + x_{h_z})$, which is less than $t + c_{h_{z-1}}x_{h_{z-1}} + 2c_{h_z}x_{h_z}$, completion time of P_{h_z} in the original schedule (since $c_{h_{z-1}} > 2c_{h_z}$ by (30)). The cost of the modified schedule is less than that of the original one since each of the values ℓ_{v1} and ℓ_{v2} is greater than ℓ_{u1} and ℓ_{u1}.

As a result of the described transformation, processor P_{h_z} takes the full load of processor $P_{h_{z-1}}$, making $P_{h_{z-1}}$ idle. Modify the processor sequence by swapping $P_{h_{z-1}}$ and P_{h_z}. If P_{h_z} is still out of order, then perform a similar transformation: move the load from $P_{h_{z-2}}$ to P_{h_z}, making $P_{h_{z-2}}$ idle and swap the two processors. Continue shifting processor P_{h_z} upstream until it reaches the right position in the schedule, immediately after its partner from the pair $\{P_{u1}, P_{u2}\}$ or after a pair of processors $\{P_{u-1,1}, P_{u-1,2}\}$. Repeating the same transformation, we construct a schedule with no larger length and with a smaller cost.

(2) For a feasible schedule, the two inequalities Load $\geq V$ and Cost $\leq K$ hold so that

$$\text{Load} - \text{Cost} \geq V - K = \frac{3}{2}\sum_{i=2}^{n+1} G^i. \tag{31}$$

Using the expression

$$\begin{aligned}
&\text{Load} - \text{Cost} \\
&= \sum_{i=1}^{n}(x_{i1} + x_{i2}) - \sum_{i=1}^{n}(\ell_{i1}x_{i1} + \ell_{i2}x_{i2}) \\
&= \sum_{i=1}^{n}(x_{i1} + x_{i2}) \\
&- \sum_{i=1}^{n}\left(\frac{e_{i1}x_{i1}}{G^{n-i+2} + e_{i1}} + \frac{e_{i2}x_{i2}}{G^{n-i+2} + e_{i2}}\right) \\
&= \sum_{i=1}^{n} G^{n-i+2}\left(\frac{x_{i1}}{G^{n-i+2}+e_{i1}} + \frac{x_{i2}}{G^{n-i+2}+e_{i2}}\right)
\end{aligned}$$

and by applying variables y_i, $i = 1, \ldots, n$, defined by (29), the necessary condition (31) can be rewritten as

$$\sum_{i=1}^{n} 2^{2i-1}G^{n-i+2}y_i \geq \frac{3}{2}\sum_{i=2}^{n+1} G^i$$

so that property (2) holds. Let us observe that violating (31) means that insufficient load is processed or cost limit is exceeded.

(3) Consider a feasible schedule with processor activating sequence $(P_{11}, P_{12}, \ldots, P_{i1}, P_{i2}, \ldots, P_{n1}, P_{n2})$, where at least one condition from

$$C_{ik} \le T, \quad i = 1, \ldots, n, \quad k = 1, 2, \tag{32}$$

holds as a strict inequality. Taking a linear combination of these inequalities weighted by constants λ_i and $2\lambda_i$, we obtain:

$$\sum_{i=1}^{n} \lambda_i \left(C_{i1} + 2C_{i2}\right) < \sum_{i=1}^{n} 3\lambda_i T, \tag{33}$$

where

$$\lambda_i = 2^{2(i-1)} \left(G^{n-i+2} - 3 \sum_{u=2}^{n-i+1} G^u \right), \; i = 1, \ldots, n. \tag{34}$$

Notice that $\lambda_i > 0$ for $G > 4$ since $\lambda_i / 2^{2(i-1)} = G^{n-i+2} - 3\frac{G^{n-i+2}-G^2}{G-1} = \frac{G^{n-i+3}-4G^{n-i+2}+3G^2}{G-1} > 0$. In what follows, we show that the left-hand side of (33) is equal to $\left(\sum_{i=1}^{n} 2^{2i} G^{n-i+2} y_i\right) T$ and the right-hand side is equal to $\left(3 \sum_{i=2}^{n+1} G^i\right) T$ so that property (3) holds.

Starting with the left-hand side, we deduce:

$$\begin{aligned}
\text{LHS} &= \sum_{i=1}^{n} \lambda_i \left(\left[\sum_{u=1}^{i-1} (c_{u1}x_{u1} + c_{u2}x_{u2}) + 2c_{i1}x_{i1} \right] \right. \\
&\quad \left. + 2 \left[\sum_{u=1}^{i-1} (c_{u1}x_{u1} + c_{u2}x_{u2}) + c_{i1}x_{i1} + 2c_{i2}x_{i2} \right] \right) \\
&= \sum_{i=1}^{n} \lambda_i \left(3 \sum_{u=1}^{i-1} (c_{u1}x_{u1} + c_{u2}x_{u2}) \right. \\
&\quad \left. + 4(c_{i1}x_{i1} + c_{i2}x_{i2}) \right) \\
&= \sum_{i=1}^{n} \lambda_i \left(3 \sum_{u=1}^{i-1} y_u + 4y_i \right) T \\
&= \left(\sum_{i=1}^{n-1} \left(4\lambda_i + 3 \sum_{u=i+1}^{n} \lambda_i \right) y_i + 4\lambda_n y_n \right) T \\
&= \left(\sum_{i=1}^{n-1} b_i y_i + b_n y_n \right) T,
\end{aligned}$$

where

$$b_i = 4\lambda_i + 3 \sum_{u=i+1}^{n} \lambda_i, \quad i = 1, 2, \ldots, n-1, \tag{35}$$

$$b_n = 4\lambda_n. \tag{36}$$

It remains to prove that

$$b_i = 2^{2i} G^{n-i+2}, \quad 1 \le i \le n. \tag{37}$$

Indeed, by (34), $4\lambda_n = 2^{2n} G^2$ and (37) holds for $i = n$. If (37) holds for some i, $2 \le i \le n$, then it also holds for $i-1$ since by (35)

$$\begin{aligned}
b_{i-1} &= 4\lambda_{i-1} + 3 \sum_{u=i}^{n} \lambda_i = b_i + 4\lambda_{i-1} - \lambda_i \\
&= 2^{2i} G^{n-i+2} \\
&\quad + 4 \times 2^{2(i-2)} \left(G^{n-i+3} - 3 \sum_{u=2}^{n-i+2} G^u \right) \\
&\quad - 2^{2(i-1)} \left(G^{n-i+2} - 3 \sum_{u=2}^{n-i+1} G^u \right) \\
&= 2^{2i} G^{n-i+2} + 2^{2(i-1)} G^{n-i+3} \\
&\quad - 3 \times 2^{2(i-1)} \sum_{u=2}^{n-i+2} G^u - 2^{2(i-1)} G^{n-i+2} \\
&\quad + 3 \times 2^{2(i-1)} \sum_{u=2}^{n-i+1} G^u \\
&= 2^{2(i-1)} G^{n-i+3} \\
&\quad + 2^{2(i-1)} \left(4G^{n-i+2} - 3G^{n-i+2} - G^{n-i+2} \right) \\
&= 2^{2(i-1)} G^{n-i+3}.
\end{aligned}$$

Consider now the expression in the right-hand side of (33), divided by $3T$:

$$\begin{aligned}
\frac{\text{RHS}}{3T} &= \sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} 2^{2(i-1)} \left(G^{n-i+2} - 3 \sum_{u=2}^{n-i+1} G^u \right) \\
&= \sum_{i=1}^{n} 2^{2(i-1)} G^{n-i+2} - 3 \sum_{i=1}^{n-1} 2^{2(i-1)} \sum_{u=2}^{n-i+1} G^u \\
&= \sum_{u=2}^{n+1} 2^{2(n-u+1)} G^u - 3 \sum_{u=2}^{n} G^u \sum_{i=u}^{n} 2^{2(n-i)} \\
&= \sum_{u=2}^{n} \left(2^{2(n-u+1)} - 3 \sum_{i=u}^{n} 2^{2(n-i)} \right) G^u + G^{n+1} \\
&= \sum_{u=2}^{n} \left(2^{2(n-u+1)} - 3 \times \frac{4 \times 2^{2(n-u)} - 1}{4-1} \right) G^u \\
&\quad + G^{n+1} = \sum_{u=2}^{n+1} G^u.
\end{aligned}$$

Thus, inequality (33) is proved.

Property **(4)** immediately follows from properties (2) and (3). It remains to prove property **(5)**.

By property (4), $C_{11} = T$ so that

$$C_{11} = 2c_{11}x_{11} = 2 \times \frac{T}{2\left(G^{n+1} + e_{11}\right)} x_{11} = T$$

and $x_{11} = G^{n+1} + e_{11}$. By the same property, $C_{12} = T$ so that

$$C_{12} = c_{11}x_{11} + 2c_{12}x_{12} = \tfrac{T}{2} + 2 \times \tfrac{T}{2(G^{n+1}+e_{12})} x_{12} = T$$

and $x_{12} = \frac{1}{2}\left(G^{n+1} + e_{12}\right)$. Proceeding in a similar way it is easy to verify that in a feasible schedule variables x_{i1}, x_{i2} are defined by (26).

In order to prove the equality $\sum_{i=1}^{n} e_{i1} = G$, consider conditions Load $\geq V$ and Cost $\leq K$ which hold for any feasible schedule. Repeating calculations (27) and (28), we obtain:

$$\begin{aligned}
\text{Load} &= \frac{3}{2}\sum_{i=1}^{n} G^{n-i+2} + \frac{1}{2}\sum_{i=1}^{n}(e_{i1} + e_{i2}) + \frac{1}{2}\sum_{i=1}^{n} e_{i1} \\
&= \frac{3}{2}\sum_{i=2}^{n+1} G^{i} + G + \frac{1}{2}\sum_{i=1}^{n} e_{i1} \geq V = \frac{3}{2}\sum_{i=1}^{n+1} G^{i}, \\
\text{Cost} &= \sum_{i=1}^{n}\left(e_{i1} + \frac{1}{2}e_{i2}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}(e_{i1} + e_{i2}) + \frac{1}{2}\sum_{i=1}^{n} e_{i1} \\
&= G + \frac{1}{2}\sum_{i=1}^{n} e_{i1} \leq K = \frac{3}{2}G.
\end{aligned}$$

The latter two inequalities imply

$$\frac{1}{2}\sum_{i=1}^{n} e_{i1} \geq \frac{1}{2}G,$$
$$\frac{1}{2}\sum_{i=1}^{n} e_{i1} \leq \frac{1}{2}G,$$

which together imply property (5). □

We conclude with the main result which follows from Lemmas 1 and 2.

Theorem 1 *Problem* $\mathrm{DL}(T, K)$ *is* ***NP*** *-complete, problems* $\mathrm{DL}_{cost}(T)$ *and* $\mathrm{DL}_{time}(K)$ *are* ***NP****-hard, even if computation time, communication time and cost have no fixed overheads, and* $r_i = 0, d_i = B_i = \infty$*, for* $i = 1, \ldots, m$.

3.2 Time–cost trade-off

For zero overheads, the arguments from Section 2 can be simplified. MIP formulations (13)–(19) and (20)–(23) do hold, but the number of different sequences can be reduced from $\eta = 2^m m!$, given by (12), to $\eta = m!$. Notice that due to zero overheads there is no need to make a selection of the set of active processors $\mathcal{P}'$ since an idle processor can be kept in any place of the sequence. The smaller value of η results in a slightly lower time complexity for enumerating all trade-offs, namely $O\left(4^m m^m m! \times LP(m-1, 4m)\right)$.

The problem of finding extreme points in the (T, K)-space, with the shortest schedule or with the smallest cost, was addressed in the prior research for the special case when all processors are available simultaneously and have no deadline and capacity restrictions, $r_i = 0$, $d_i = B_i = \infty$ for all $1 \leq i \leq m$. As shown in Bharadwaj et al. (1994, 1996), Blazewicz and Drozdowski (1997), the shortest schedule is provided if processors are sequenced in the non-decreasing order of c_i and complete all tasks simultaneously. For the same special case, the cheapest solution is constructed if the whole load is processed by the cheapest processor, i.e., $P_i : \ell_i = \min_{j=1}^{m}\{\ell_j\}$. Hence, in the bicriteria problem $\mathrm{DL}_{\mathrm{bicrit}}$ end-points (T^0, K^0), (T^q, K^q) of the time–cost trade-off can be found in, respectively, $O(m \log m)$ and $O(m)$ time.

For the general case of arbitrary r_i, d_i, B_i, finding the solution (T^q, K^q) with the lowest cost, i.e., the rightmost point (T^q, K^q) in the time–cost trade-off, is computationally hard by Theorem 1, because even though schedule length may be arbitrary to find the lowest cost schedule, processor availability constraints r_i, d_i, B_i may impose limits equivalent to schedule length. We conjecture that finding the solution $\left(T^0, K^0\right)$ with the shortest schedule is also computationally hard.

Conjecture 1 For arbitrary r_i, d_i, B_i, for all processors $P_i \in \mathcal{P}$, problem $\mathrm{DL}_{\mathrm{time}}(\infty)$ is **NP**-hard, even if computation time, communication time and cost have no fixed overheads.

In the remaining part of this section, we consider the case of *agreeable* processors. In that case, processors can be renumbered so that the two conditions hold:

$$\begin{aligned}
&c_1 \leq c_2 \leq \cdots \leq c_m, \\
&\tfrac{\ell_1}{c_1} \leq \tfrac{\ell_2}{c_2} \leq \cdots \leq \tfrac{\ell_m}{c_m}.
\end{aligned} \tag{38}$$

Theorem 2 *If processors* $\mathcal{P}$ *are agreeable and have no availability and capacity restrictions, i.e.,* $r_i = 0$*,* $d_i = B_i = \infty$*,* $1 \leq i \leq m$ *, then an optimum solution can be found in polynomial time.*

Proof Assume that processors are numbered in accordance with (38) and they are activated in the order of their numbering. The processor sequence corresponding to $c_1 \leq c_2 \leq$

$\cdots \le c_m$ guarantees the shortest schedule (Bharadwaj et al. 1994, 1996; Blazewicz and Drozdowski 1997). It is also known (Yang et al. 2007) that in the shortest schedule there are no idle times between communications and all processors finish computation simultaneously.

We show, by interchange argument, that under the agreeable condition (38) the total cost is also minimum. Consider pair P_i, P_{i+1} in the sequence. Let us assume that communication to P_i starts τ units of time before the end of the schedule. The load processed by P_i is $x_i = \tau/(a_i + c_i)$. Since there are no idle times in the schedule and P_{i+1} receives its load and processes it while P_i is computing, the load processed by P_{i+1} is $x_{i+1} = a_i x_i/(a_{i+1} + c_{i+1}) = (\tau a_i)/((a_i + c_i)(a_{i+1} + c_{i+1}))$. The cost of processing on P_i, P_{i+1} is

$$K_1 = \ell_i x_i + \ell_{i+1} x_{i+1} = \frac{\tau(\ell_i a_{i+1} + \ell_i c_{i+1} + \ell_{i+1} a_i)}{(a_i + c_i)(a_{i+1} + c_{i+1})}$$

If the order of communications were P_{i+1}, P_i, the cost would be

$$K_2 = \frac{\tau(\ell_{i+1} a_i + \ell_{i+1} c_i + \ell_i a_{i+1})}{(a_i + c_i)(a_{i+1} + c_{i+1})}.$$

The difference between the two costs is

$$K_1 - K_2 = \frac{\tau(\ell_i c_{i+1} - \ell_{i+1} c_i)}{(a_i + c_i)(a_{i+1} + c_{i+1})},$$

The processor sequence (P_i, P_{i+1}) results in a cheaper solution, if $\ell_i/\ell_{i+1} \le c_i/c_{i+1}$. Thus, for agreeable processors the shortest schedule is also the cheapest and the Pareto-front is reduced to a point in the time×cost space. It is possible to check whether processors are agreeable in $O(m \log m)$ time and calculate load sizes x_i in $O(m)$ time (Blazewicz and Drozdowski 1997). □

4 Zero transfer overheads: $s_i = c_i = 0$

The main model studied in this section is characterized by zero transfer times and zero fixed cost overheads for all processors P_i: $s_i = c_i = 0$, $f_i = 0$, $1 \le i \le m$ (Sect. 4.1). In terms of the three types of decisions introduced in Sect. 1, only decisions of type 1 and 3 should be considered: any processor P_i which gets a zero-size chunk $x_i = 0$ should be removed from the list $\mathcal{P}'$ of active processors, and all processors in $\mathcal{P}'$ can be sequenced arbitrarily. We also discuss how the proposed methods can be adjusted for the case with arbitrary cost overheads f_i (Sect. 4.2) and their applicability to the related models with nonzero transfer times (Sect. 4.3).

4.1 Zero fixed cost overheads: $f_i = 0$

In this section, we study the version of the main problem with the cost function $F = \sum_{i=1}^{n} \ell_i x_i$, i.e., $f_i = 0$ for $1 \le i \le m$.

In the cost minimization problem $\mathrm{DL}_{\mathrm{cost}}(T)$, given the schedule length limit T, the upper bounds u_i on the chunks x_i allocated to each processor i can be found from (6) combined with (9) and (11):

$$u_i(T) = \min\left\{\widetilde{B}_i, \frac{1}{a_i}[T - r_i - p_i]^+\right\} \tag{39}$$

where $[f]^+ = \max\{f, 0\}$ and

$$\widetilde{B}_i = \min\left\{B_i, \frac{1}{a_i}(d_i - r_i - p_i), V\right\}.$$

We assume that $\sum_{i=1}^{m} u_i(T) \ge V$ so that a feasible solution exists. For the given T, the cost minimization problem can be modeled as the following linear program:

$$\mathrm{LP}_{\mathrm{cost}}(T): \quad \min \sum_{i=1}^{m} \ell_i x_i \quad \text{s.t.} \tag{40}$$

$$\sum_{i=1}^{m} x_i = V, \tag{41}$$

$$0 \le x_i \le u_i(T), \quad i = 1, \ldots, m. \tag{42}$$

Note that (40)–(42) is the continuous knapsack problem, solvable in $O(m)$ time by the algorithm due to Balas and Zemel (1980), which implies the following result.

Statement 2 *If there are no transfer overheads and the cost function is* $F = \sum_{i=1}^{n} \ell_i x_i$, *then problem* $\mathrm{DL}_{\mathrm{cost}}(T)$ *is solvable in* $O(m)$ *time.*

For the counterpart $\mathrm{DL}_{\mathrm{time}}(K)$ of problem $\mathrm{DL}_{\mathrm{cost}}(T)$, we can only propose an $O(m \log m)$-time algorithm. As we show next, the bicriteria problem $\mathrm{DL}_{\mathrm{bicrit}}$ is solvable in $O(m \log m)$ time as well. Thus, in what follows we focus on $\mathrm{DL}_{\mathrm{bicrit}}$; a solution to $\mathrm{DL}_{\mathrm{time}}(K)$ can be found from a solution to $\mathrm{DL}_{\mathrm{bicrit}}$ without increasing the $O(m \log m)$ time complexity.

In order to find the Pareto-front for $\mathrm{DL}_{\mathrm{bicrit}}$, consider $\mathrm{LP}_{\mathrm{cost}}(T)$ as the underlying model and treat it in a parametric way, with parameter T that varies in $\left[\min_{1 \le i \le m}\{d_i\}, \max_{1 \le i \le m}\{d_i\}\right]$. Notice that for small values of T from that interval the problem $\mathrm{LP}_{\mathrm{cost}}(T)$ may be infeasible.

Since $\mathrm{LP}_{\mathrm{cost}}(T)$ is the continuous knapsack problem for each fixed T, an optimal load distribution is defined by the formulae:

$$x_1(T) = u_1(T), \tag{43}$$

$$x_i(T) = \min\left\{u_i(T), \left[V - \sum_{k=1}^{i-1} u_k(T)\right]^+\right\}, \quad i = 2, \ldots, m, \tag{44}$$

assuming that processors are numbered so that

$$\ell_1 \le \ell_2 \le \cdots \le \ell_m. \tag{45}$$

Informally, the cheapest processor P_1 gets the highest possible load, and every subsequent processor P_i gets the highest possible load, after all cheaper processors are loaded to their maximum capacity without violating the given T.

We start with the rightmost point of the trade-off, that corresponds to a solution with the largest length and minimum cost. It can be found in $O(m)$ time by solving problem $\mathrm{LP}_{\mathrm{cost}}(T)$ with $T = \max_{1\le i\le m}\{d_i\}$. The resulting point (T, K) has $K = \sum_{i=1}^{m} \ell_i x_i(T)$.

Consider a cost-optimum schedule of some length T. Let P_s be a processor with the smallest index whose load can be increased without increasing the length T. Such a processor satisfies a strict inequality

$$x_s < u_s(T). \tag{46}$$

We call P_s a *split processor* as it corresponds to the so-called *split item* s in the solution to the underlying continuous knapsack problem,

$$\begin{aligned} &x_i(T) = u_i(T), && 1 \le i \le s-1,\\ &x_s(T) = V - \sum_{i=1}^{s-1} u_i(T), &&\\ &x_j(T) = 0, && s+1 \le j \le m. \end{aligned}$$

In accordance with definition (39) of $u_i(T)$, we divide processors $\{P_1, \ldots, P_{s-1}\}$ into three subsets:

critical processors $\mathcal{P}_c$, with $x_i(T) = (T - r_i - p_i)/a_i$ so that $C_i = T$,
non-critical processors $\mathcal{P}_n$, with $x_i(T) = \widetilde{B}_i$ so that $C_i < T$,
excluded processors $\mathcal{P}_e$, with $x_i(T) = 0$.

Note that each excluded processor $P_i \in \mathcal{P}_e$ defined for some schedule length T remains excluded for any smaller schedule length, since

$$r_i + p_i \ge T.$$

The approach described below moves from a current breakpoint, denoted by (T', K'), to the next one (T'', K''), repeating similar steps: it simultaneously decreases the load values of processors $P_i \in \mathcal{P}_c$ and compensates that change by increasing the load of the split processor P_s, keeping the total processed volume equal to V at all stages. For an efficient implementation, we define two auxiliary values:

$$h(\mathcal{P}_c) = \sum_{P_i \in \mathcal{P}_c} \frac{1}{a_i}$$

the combined speed of processors $\mathcal{P}_c$, and

$$k(\mathcal{P}_c) = \sum_{P_i \in \mathcal{P}_c} \frac{\ell_i}{a_i}$$

the combined cost of occupying processors $\mathcal{P}_c$ for 1 time unit.

Consider a transition from a solution with load values x_i', corresponding to (T', K'), to a solution with load values x_i'', corresponding to (T'', K''). For any $P_i \in \mathcal{P}_c$, the T-value changes from $T' = r_i + p_i + a_i x_i'$ to $T' - \Delta = r_i + p_i + a_i x_i''$ so that

$$x_i'' = x_i' - \frac{1}{a_i}\Delta, \quad P_i \in \mathcal{P}_c.$$

For the split processor P_s, the increase in its load should be equal to the cumulative decrease in the load of processors $\mathcal{P}_c$:

$$x_s'' = x_s' + h(\mathcal{P}_c)\,\Delta. \tag{47}$$

The next breakpoint (T'', K'') is triggered by one of the events (a)–(d), whichever is reached first. It corresponds to the smallest value of Δ, defined in the event descriptions below.

(a) The set $\mathcal{P}_c$ is adjusted to exclude a processor whose load reduces to 0. This happens if the decreased value $T'' = T' - \Delta$ reaches $r_i + p_i$ for some $P_i \in \mathcal{P}_c$. In this case

$$\Delta = T' - \max\{r_i + p_i \,|\, P_i \in \mathcal{P}_c\}. \tag{48}$$

(b) The set $\mathcal{P}_c$ is adjusted to include a non-critical processor which becomes critical. This happens if $T'' = T' - \Delta$ reaches an absolute deadline $\widetilde{d}_i$ for some non-critical processor $P_i \in \mathcal{P}_n$,

$$\widetilde{d}_i = \min\{d_i, r_i + p_i + a_i B_i\}. \tag{49}$$

In this case

$$\Delta = T' - \max\left\{\widetilde{d}_i \,|\, P_i \in \mathcal{P}_n\right\}. \tag{50}$$

(c) The split processor P_s can no longer get any additional load since its increased load $x_s'' = x_s' + h(\mathcal{P}_c)\,\Delta$ reaches an absolute upper bound $\widetilde{B}_s$, which implies

$$\Delta = \frac{\widetilde{B}_s - x_s'}{h(\mathcal{P}_c)}.$$

(d) The split processor P_s can no longer get any additional load since processing its increased load x_s'' reaches T'' so that processor P_s becomes a critical processor. This happens if the completion time $r_s + p_s + a_s\left(x_s' + h\left(\mathcal{P}_c\right)\Delta\right)$ of P_s becomes equal to $T'' = T' - \Delta$, which implies

$$\Delta = \frac{T' - \left(r_s + p_s + a_s x_s'\right)}{a_s h\left(\mathcal{P}_c\right) + 1}.$$

Thus, it suffices to calculate Δ-values for each of the events (a)–(d), select the smallest one and compute the characteristics of the next breakpoint:

$$T'' = T' - \Delta, \qquad K'' = K' - k\left(\mathcal{P}_c\right)\Delta + \ell_s h\left(\mathcal{P}_c\right)\Delta,$$

Note that the term $-k\left(\mathcal{P}_c\right)\Delta = -\sum_{P_i\in\mathcal{P}_c}\frac{\ell_i}{a_i}\Delta = = -\sum_{P_i\in\mathcal{P}_c}\ell_i\left(x_i' - x_i''\right)$ defines the cost change due to the decrease in the loads of processors $\mathcal{P}_c$, while the term $\ell_s h\left(\mathcal{P}_c\right)\Delta = \ell_s\left(x_s'' - x_s'\right)$ defines the cost change due to the increase in the load of the split processor P_s. To complete the transition from $\left(T', S'\right)$ to $\left(T'', S''\right)$, perform the following updates.

- In the case of event (a) triggered by processor $P_i \in \mathcal{P}_c$, calculate x_s'' by (47) and set

$$\mathcal{P}_c := \mathcal{P}_c \backslash \{P_i\},\ \mathcal{P}_e := \mathcal{P}_e \cup \{P_i\},$$

$$h\left(\mathcal{P}_c\right) := h\left(\mathcal{P}_c\right) - \frac{1}{a_i},\ k\left(\mathcal{P}_c\right) := k\left(\mathcal{P}_c\right) - \frac{\ell_i}{a_i}.$$

- In the case of event (b) triggered by processor $P_i \in \mathcal{P}_n$, calculate x_s'' by (47) and set

$$\mathcal{P}_c := \mathcal{P}_c \cup \{P_i\},\ \mathcal{P}_n := \mathcal{P}_n \backslash \{P_i\},$$

$$h\left(\mathcal{P}_c\right) := h\left(\mathcal{P}_c\right) + \frac{1}{a_i},\ k\left(\mathcal{P}_c\right) := k\left(\mathcal{P}_c\right) + \frac{\ell_i}{a_i}.$$

- In the case of event (c), set

$$\mathcal{P}_n := \mathcal{P}_n \cup \{P_s\},$$

and define a new split processor by considering processors P_i, $i = s+1, s+2, \ldots$ one by one: if $r_i + p_i < T$, then P_i becomes a new split processor (note that its load is 0); otherwise, P_i joins the set of excluded processors $\mathcal{P}_e$, and the next processor is examined. If no processor from $\{P_{s+1}, \ldots, P_m\}$ becomes a split processor, the algorithm stops.
- In the case of event (d), set

$$\mathcal{P}_c := \mathcal{P}_c \cup \{P_s\},\ h\left(\mathcal{P}_c\right) := h\left(\mathcal{P}_c\right) + \frac{1}{a_s},$$

$$k\left(\mathcal{P}_c\right) := k\left(\mathcal{P}_c\right) + \frac{\ell_s}{a_s},$$

and find the next split processor P_s as in the case of event (c).
- If both events (c) and (d) happen simultaneously, proceed as in the case of event (d).
- We can now treat the found breakpoint $\left(T'', K''\right)$ as the current one and proceed similarly to finding the next breakpoint. The algorithm stops if $s = m$ and event (c) or (d) happens.

In order to implement all calculations efficiently, we maintain sets $\mathcal{P}_c$ and $\mathcal{P}_n$ as priority queues so that each calculation (48) or (50) can be done in $O(1)$ time. Since an element is added to $\mathcal{P}_c$ or removed from $\mathcal{P}_c$ at most once, all add/remove operations require $O(m \log m)$ time. The same holds for add/remove operations on $\mathcal{P}_n$. Similarly, $\mathcal{P}_e$ is updated no more than m times.

Initialization involves renumbering processors in accordance with (45), finding the first solution $\left(x_1', x_2', \ldots, x_m'\right)$ for $T' = \max\left\{\widetilde{d}_i | 1 \le i \le m\right\}$, computing K' and auxiliary values $h\left(\mathcal{P}_c\right)$ and $k\left(\mathcal{P}_c\right)$. All required steps can be done in $O(m \log m)$ time.

A transition from one breakpoint to the next one requires updating the two priority queues, which takes $O(\log m)$ time, and updating the five parameters, x_s'', T'', K'', $h\left(\mathcal{P}_c\right)$ and $k\left(\mathcal{P}_c\right)$, which can be done in $O(1)$ time. Note that x-values for $i \neq s$ are not maintained. Since there are at most m events of each type, the total number of breakpoints is no larger than $4m$, and the overall time complexity is $O(m \log m)$. Thus, the following statement holds.

Statement 3 *Problem* DL_{bicrit} *has a trade-off with at most* $4m$ *breakpoints which can be computed in* $O(m \log m)$ *time.*

Let us finish this section with an example to illustrate calculation of the time–cost trade-off. Assume that load size is $V = 30$ and the number of processors is $m = 8$. Their parameters are given in Table 3. The breakpoints (T, K) are presented in the first two rows of Table 4. The boxed elements represent the load of the split processor P_s. The loads of the remaining processors are not maintained by the described algorithm, in order to achieve the $O(m \log m)$ time complexity. For completeness, we present the optimal x-values for all processors.

4.2 Arbitrary cost overheads f_i, ℓ_i

In the general case, with nonzero overheads f_i in the cost function $K = \sum_{i=1}^{m}\left(f_i + \ell_i x_i\right)$, both problems $\mathrm{DL}_{\mathrm{time}}$ and $\mathrm{DL}_{\mathrm{cost}}$ are **NP**-hard, see Drozdowski and Lawenda (2005).

Table 3 Example data for time–cost trade-off calculation

i	a_i	B_i	r_i	d_i	p_i	ℓ_i	$r_i + p_i$	$\widetilde{d}_i$	$\widetilde{B}_i$
1	1	10	80	100	1	1	81	91	10
2	4	40	30	110	2	2	32	110	19.5
3	8	10	20	40	5	3	25	40	1.875
4	4	20	20	70	4	5	24	70	11.5
5	5	10	10	80	2	8	12	62	10
6	6	10	40	100	2	10	42	100	≈ 9.667
7	3	30	5	50	1	20	6	50	≈ 14.667
8	2	50	10	60	3	40	13	60	23.5

As we show in this section, the problem can be solved efficiently if we limit our search to a class of solutions with a fixed set of active processors $\mathcal{P}' \subseteq \mathcal{P}$. The associated problem is of the form: given a set of active processors $\mathcal{P}'$, it is required to allocate a positive load to each active processor minimizing the objective function T or K. Note that some processors may get an infinitely low load $\varepsilon > 0$; such a processor then has a completion time $r_i + p_i + a_i\varepsilon$, which should be taken into account when calculating the length T of the schedule.

There are two common properties that hold for any problem, $\mathrm{DL}_{\mathrm{time}}$, $\mathrm{DL}_{\mathrm{cost}}$ or $\mathrm{DL}_{\mathrm{bicrit}}$, under the assumption that $\mathcal{P}'$ is fixed:

Property 1: component $\sum_{i\in\mathcal{P}'} f_i$ is constant and can be excluded from K;
Property 2: the schedule length T satisfies $T > \rho$, where

$$\rho = \max_{1\le i\le m'} \{r_i + p_i\}.$$

Based on these two properties, the results from Sect. 4.1 can be adjusted to handle the case with a fixed set $\mathcal{P}'$.

For problem $\mathrm{DL}_{\mathrm{cost}}(T)$ with a given set $\mathcal{P}'$ and $T > \rho$, consider the continuous knapsack formulation (40) defined over $\mathcal{P}'$. If in an optimal knapsack solution $x_i = 0$ for some $P_i \in \mathcal{P}'$, then such a solution is adjusted by replacing 0-values by ε. The overall time complexity remains the same, $O(m)$.

For problem $\mathrm{DL}_{\mathrm{bicrit}}$, apply the approach from Sect. 4.1 for the processor set $\mathcal{P}'$ and output the part of the trade-off that satisfies $T > \rho$. Treat all found solutions as if each idle processor gets an ε-load. This assumption does not affect the values of T and K, assuming that ε is infinitely small. There is one case that needs a special attention. It occurs if all breakpoints (T, K) satisfy $T \le \rho$. In that case consider the rightmost point (T^*, K^*) of the trade-off and output the unique solution obtained from (T^*, K^*) by allocating ε-loads to all idle processors. The described adjustments do not affect the $O(m \log m)$ time complexity derived for problem $\mathrm{DL}_{\mathrm{bicrit}}$ in Sect. 4.1.

Treating problem $\mathrm{DL}_{\mathrm{time}}(K)$ as a special case of problem $\mathrm{DL}_{\mathrm{bicrit}}$, we conclude that it is solvable in $O(m \log m)$ time.

Statement 4 *If there are arbitrary cost overheads f_i, ℓ_i in the cost function $F = \sum_{i=1}^{n} (f_i + \ell_i x_i)$, and a set of active processors is fixed, then problem $\mathrm{DL}_{\mathrm{cost}}(T)$ is solvable in $O(m)$ time, while problems $\mathrm{DL}_{\mathrm{time}}(K)$ and $\mathrm{DL}_{\mathrm{bicrit}}$ are solvable in $O(m \log m)$ time.*

4.3 Arbitrary transfer overheads s_i, c_i

The results from Sect. 4.1 can be applied to special scenarios with nonzero transfer overheads s_i, c_i.

One scenario arises in parallel communication with simultaneous start mode, see Kim (2003), Robertazzi (2003), with computation speeds slower than transfer speeds. For the simultaneous start mode, load transfer to all worker processors starts at the same time. Worker processors start computing as soon as the first grain of the load is received. Due to the slower computation speeds compared to transfer speeds, computation time of any grain is higher than the transfer time of any subsequent grain. Thus, in parallel communication with simultaneous start, communication does not affect the overall schedule length and cost. Consequently,

Table 4 Load allocations x_i and total costs in time–cost trade-off calculation

T	34.99	40	42	44.60	59.45	62	64.25	70	81	84.5	91	104.5	110
K	688.83	508.76	446.13	351.55	168.68	159.38	152.06	139.13	122.63	92.88	62	51.88	50.5
x_1	0	0	0	0	0	0	0	0	0	3.5	10	10	10
x_2	0.748	2	2.5	4.5	6.863	7.5	8.063	9.5	12.25	13.125	14.75	18.125	19.5
x_3	1.249	1.875	1.875	1.875	1.875	1.875	1.875	1.875	1.875	1.875	1.875	1.875	0.5
x_4	2.748	4	4.5	5.151	8.863	9.5	10.063	11.5	11.5	11.5	3.375	0	0
x_5	4.598	5.6	6	6.521	9.490	10	10	7.125	4.375	0	0	0	0
x_6	0	0	0	0.434	2.909	1.125	0	0	0	0	0	0	0
x_7	9.663	11.333	12	12.868	0	0	0	0	0	0	0	0	0
x_8	10.995	5.1917	3.125	0	0	0	0	0	0	0	0	0	0

communication time can be ignored as if $c_i = s_i = 0$ for any $P_i \in \mathcal{P}$.

Another scenario is typical for a pipeline-like computing mode. Load scattering and processing are interleaved so that communications and computations are performed at different stages. The load is distributed in one interval (say interval i) and processed in the next interval $(i + 1)$. If there is a common communication time τ_{comm} for all processors and a common computation time τ_{comp}, with $\tau_{comm} \leq \tau_{comp}$, then the communications executed in interval i do not determine partitioning of the load for minimum computing time and the cost in interval $i + 1$. It can be shown that the general case of the pipeline mode, characterized by $s_i > 0, a_i > 0$, is **NP**-hard [see, e.g., DLS with processor release times in Drozdowski and Lawenda (2005)].

5 Conclusions

In this paper, we analyze the time/cost optimization for divisible load scheduling problems with arbitrary processor memory sizes, ready times, deadlines, communication and computation start-up costs. Three versions of the problem are studied: $\mathrm{DL}_{\mathrm{time}}(K)$—schedule length minimization for the given limited budget K, $\mathrm{DL}_{\mathrm{cost}}(T)$—cost minimization for the given schedule length limit T, and $\mathrm{DL}_{\mathrm{bicrit}}$—constructing the set of time–cost Pareto-optimal solutions. All three versions can be solved in polynomial time for fixed m.

The case with given upper bounds on the schedule length and cost appears to be **NP**-hard even if all fixed overheads are zero ($p_i = s_i = f_i = 0$ for all $P_i \in \mathcal{P}$). This result is rather unusual: all previous **NP**-hardness results in the divisible load theory assumed nonzero fixed overheads. Interestingly, a divisible load problem is linked to scheduling problems with preemption, for which **NP**-hardness results are rather atypical (see, e.g., Sitters 2001; Drozdowski et al. 2017).

We leave an open question regarding the time complexity of finding a shortest schedule with processor availability constraints, but with zero fixed communication and computation overheads, regardless of the computation cost ($K = \infty$). We believe that the latter problem is computationally hard, see Conjecture 1. Contrarily, the version with negligible communication times is solvable in $O(m \log m)$ time even in its bicriteria setting.

Our summary table provided in Introduction presents the state-of-the-art results in divisible load scheduling and can be used as a guideline for future research.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

Adler, I., & Monteiro, R. D. C. (1992). A geometric view on parametric linear programming. *Algorithmica*, *8*, 161–176.

Agrawal, R., & Jagadish, H. V. (1988). Partitioning techniques for large-grained parallelism. *IEEE Transactions on Computers*, *37*, 1627–1634.

Balas, E., & Zemel, E. (1980). An algorithm for large zero-one knapsack problems. *Operations Research*, *28*, 1130–1154.

Bharadwaj, V., Ghose, D., & Mani, V. (1994). Optimal sequencing and arrangement in distributed single-level tree networks with communication delays. *IEEE Transactions on Parallel and Distributed Systems*, *5*, 968–976.

Bharadwaj, V., Ghose, D., Mani, V., & Robertazzi, T. G. (1996). *Scheduling divisible loads in parallel and distributed systems*. Los Alamitos: IEEE Computer Society Press.

Blazewicz, J., & Drozdowski, M. (1997). Distributed processing of divisible jobs with communication startup costs. *Discrete Applied Mathematics*, *76*, 21–41.

Cheng, Y.-C., & Robertazzi, T. G. (1988). Distributed computation with communication delay. *IEEE Transactions on Aerospace and Electronic Systems*, *24*, 700–712.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). Cambridge: MIT Press and McGraw-Hill.

Drozdowski, M. (2009). *Scheduling for parallel processing*. London: Springer.

Drozdowski, M., Jaehn, F., & Paszkowski, R. (2017). Scheduling position-dependent maintenance operations. *Operations Research*, *65*, 1657–1677.

Drozdowski, M., & Lawenda, M. (2005). The combinatorics in divisible load scheduling. *Foundations of Computing and Decision Sciences*, *30*, 297–308.

Goldfarb, D., & Todd, M. J. (1989). Chapter II: Linear programming. In G. L. Nemhauser, A. H. G. Rinooy Kan, & M. J. Todd (Eds.), *Handbooks in operations research and management science*. Optimization (Vol. 1, pp. 73–170). Elsevier Science Publishers B.V. (North-Holland).

Kim, H. J. (2003). A novel optimal load distribution algorithm for divisible loads. *Cluster Computing*, *6*, 41–46.

Robertazzi, T. G. (2003). Ten reasons to use divisible load theory. *IEEE Computer*, *36*, 63–68.

Shakhlevich, N. V. (2013). *Scheduling divisible loads to optimize the computation time and cost* (Vol. 8193, pp. 138–148)., Lecture notes in computer science Cham: Springer.

Sitters, R. A. (2001). *Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines* (Vol. 2081, pp. 396–405)., Lecture Notes in Computer Science Berlin: Springer.

Yang, Y., Casanova, H., Drozdowski, M., Lawenda, M., & Legrand, A. (2007) *On the complexity of multi-round divisible load scheduling*. INRIA Rône-Alpes, Research Report No. 6096, 2007.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.