New approach of LP Rounding Algorithm for the Active Time Problem

Gruia Călinescu * [1] and Kai Wang† [2,3]

[1]Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA
[2]Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China
[3]Center for Advanced Studies in Management (CASiM), HHL Leipzig Graduate School of Management, Jahnallee 59, 04109 Leipzig, Germany

December 23, 2020

Abstract

In this paper, we work on the scheduling problem with active time model. We have a set of preemptive jobs with integral release times, deadlines and required processing lengths, while the preemption of jobs are only allowed at integral time points. We have a single machine that can process at most g distinct job units at any given time unit when the machine is switched on. The objective is to find a schedule that completes all jobs within their timing constraints and minimizes the time when the machine is on, i.e. the active time. This problem has been studied by Chang et al. where they proposed an LP rounding approach which gives a 2-approximation solution. In this paper, we also give a 2-approximation algorithm based on LP rounding approach with a different rounding technique and analysis. Finally, we give a new linear programming formulation for which we conjecture that the integrality gap is 5/3, which might bring new hope for beating the barrier of 2 for the approximation ratio.

1 Introduction

Scheduling with parallelism is a fundamental problem in computer science. Many studies have been done on parallel machine scheduling for decades. In this research, we focus on the parallelism over memory. In terms of energy saving, memory can be turned off when it is not accessed [1, 2]. At each time slot when the memory is on (active), the

*calinescu@iit.edu
†Corresponding author; kai.wang@my.cityu.edu.hk

scheduler can work on a group of at most g jobs, due to the capacity limit of the memory banks. Meanwhile, the memory bank consumes a fixed amount of energy during this active time slot, regardless of the number of jobs that access the memory. The objective is to schedule the jobs while minimizing the total energy consumption, which is equivalent to the total active time. This problem is referred to as *active time scheduling problem* and was introduced by Chang et al. [3]. They gave a general framework of this problem and studied the jobs with multiple available intervals and showed that the problem is NP-complete even for $g = 3$. Later, Chang et al. [4] worked on a more standard case where each job has only one available interval during which it can be scheduled. They considered preemptive jobs in the sense that the execution of a job could be interrupted and resumed later. They proposed a greedy algorithm with approximation ratio 3 which closes as many time slots as possible in an arbitrarily order as long as the new solution is feasible, and also an LP rounding approach for which they claim an approximation ratio of 2, while the complexity of the problem still remains open. More recently, Kumar and Khuller [6] proposed another greedy algorithm with approximation ratio of 2.

In this paper, we continue to work on the problem studied by Chang et al. [4]. We propose a new algorithm based on LP rounding approach and show that the solution is 2-approximation. This algorithm introduces a "chain" technique that might be useful for getting an approximation ratio smaller than 2, or for other problems. In Section 2 we formulate the problem and give the LP formulation referred from [4]. In Section 3 we introduce a rounding scheme which is the basis for the final rounding algorithm. In Section 4, we generalize the rounding scheme to make it work for the general case. We also introduce a potentially stronger LP relaxation for this problem.

2 Formulation

The input consists of a set J of jobs and a parallelism parameter g with $g \in \mathbb{N}_{\neq 0}$. Each job $j \in J$ is defined by its release time $r_j \in \mathbb{N}$, deadline $d_j \in \mathbb{N}_{\neq 0}$ and processing time $p_j \in \mathbb{N}_{\neq 0}$. We consider the schedule on a single machine in which the time horizon is discretized into many time slots by unit length where time slot t is defined to be the time interval $(t-1, t]$ with $t \in \mathbb{N}_{\neq 0}$ [1], and time slot t is called *active* if the machine is on during $(t-1, t]$. The parameter g indicates the parallelism ability of the machine, where for each time slot t, the machine can finish up to an amount of g units of workload of jobs while the workload of a single job assigned to time slot t is at most 1. We assume all the values in the input are integers. Jobs can be preempted, in other words, jobs can be interrupted during execution and resumed later, while the preemption is only allowed at integer time points, i.e. at the end of some time slot. In a feasible schedule, the machine is not necessarily switched on for all the time. Thus, the objective is to find a feasible schedule that uses the minimum number of active time slots to finish all the jobs. Without loss of generality, we assume the smallest job release time is 0 and the largest job deadline is T and denote $\mathcal{T} = \{1, 2, ..., T\}$ as the set of time slots. For each job $j \in J$, let $I_j = \{r_j + 1, ..., d_j\}$ be the set of available time slots for job j. Also, we denote time interval $(r_j, d_j]$ as job j's window. We use the phrase *open a time slot* as the meaning of turning on the machine during that time slot.

[1] It is necessary that the time interval of a time slot is defined to be a half-open interval, i.e. $(,\ ]$.

As this problem has been studied in [4], we cite their linear programming formulation as follows.

Linear Programming Formulation For $\forall t \in \mathcal{T}$, we create a binary variable x_t indicating whether time slot t is active. And $\forall t \in \mathcal{T}, \forall j \in J$, we create a binary variable y_{jt}, indicating the amount of workload of job j that is scheduled in time slot t.

$$
\begin{aligned}
&\min \sum_{t \in \mathcal{T}} x_t && \text{subject to} & \\
&\sum_{t \in I_j} y_{jt} = p_j && \forall j \in J & (1)\\
&\sum_{j \in J} y_{jt} \le g \cdot x_t && \forall t \in \mathcal{T} & (2)\\
&y_{jt} \le x_t && \forall j \in J, \forall t \in \mathcal{T} & (3)\\
&x_t \in \{0,1\} && \forall t \in \mathcal{T} & (4)\\
&y_{jt} \in \{0,1\} && \forall j \in J, \forall t \in \mathcal{T} & (5)
\end{aligned}
$$

It is easy to check that the integer linear program above corresponds to the active time problem. The linear program is obtained by replacing the constraints 4 and 5 by $x_t \ge 0 \ \forall t \in \mathcal{T}$ and $0 \le y_{jt} \le 1 \ \forall j \in J, \forall t \in \mathcal{T}$. The following fact is referred from [4], which is a corollary of the integrality flow theorem in Maximum Flow problem [5].

Fact 1. Given a set of integrally opened time slots (that is, $x_t \in \{0,1\} \ \forall t \in \mathcal{T}$), with feasible fractional assignment of jobs for the active time problem (that is, $0 \le y_{jt} \le 1 \ \forall j \in J, \forall t \in \mathcal{T}$), an integral assignment of jobs can be found at the end of the rounding procedure, via maximum flow computation.

In this paper, we use LP rounding technique to obtain a 2-approximation solution. Generally, we first solve the relaxed linear programming in which the variables x_t, y_{jt} are relaxed into floating numbers, i.e. $x_t \in [0,1], y_{jt} \in [0,1]$. Then in the rounding process, we open time slots (by making the values x_t integral; a time slot t being open if $x_t = 1$), and then construct a feasible assignment that assigns the fractional workload y_{jt} into the opened (i.e. active) time slots. Finally, a feasible schedule is computed according to the above fact.

3 LP Rounding

Given the values x_t, y_{jt} in the optimal LP solution, we aim to find an integral solution of 2-approximation. Before we present our rounding algorithm, we introduce several definitions over the LP solution. Here is an overview of the algorithm: first, we label the times at which the accumulated fractional active time from the LP solution reaches the multiples of 0.5, which is referred to as *time mark* (formally defined in the two paragraphs below). Then, for each time slot which contains one time mark, the rounding algorithm will open it, and for each time slot which contains two time marks, the rounding algorithm will open it and possibly also open one of its neighboring time slots as well, depending on the LP solution. Meanwhile, we create a feasible assignment that assigns the fractional workload y_{jt} into the opened time slots, which is the major focus of the rounding process.

In the following, we use t as the notation of a time slot $t \in \mathcal{T}$ and s a particular time $s \in [0, T]$. For each time slot t, we assume that the fractional active time x_t is uniformly distributed in time slot t. In other words, the total active time during time interval $(t-\eta, t]$ equals $\eta \cdot x_t$ for any $\eta \in [0, 1]$. Similarly, we assume each value y_{jt} is uniformly distributed in time slot t. Hence, given an interval $I \subseteq [0, T]$, we denote $x(I)$ as the total active time during time interval I, denote $y_j(I)$ as the total workload of job j that are assigned to interval I, and let $y(I) = \sum_{j \in \mathcal{J}} y_j(I)$ be the total workload of all jobs assigned in interval I. To abuse notation, for time slot t and interval $I \subseteq [0, T]$, we use $I \setminus t$ to denote $I \setminus (t-1, t]$, also we use $y_j(t)$ as the meaning of $y_j(I')$ where $I' = (t-1, t]$ for some time slot t [2]. Moreover, we define $\tau(a) = \min\{s \mid x([0, s]) = a, s \in [0, T]\}$ as the earliest time s when the amount of accumulated active time reaches a where $a \in \mathbb{R}_{\geq 0}$ and $a \leq \sum_{t=1}^{T} x_t$.

Let $\delta = 0.5$ be the rounding threshold. Especially, we say time s is a *time mark* if $s = \tau(k\delta)$ for some integer $k \in \mathbb{N}_{\neq 0}$. For each time slot t, if $(t-1, t]$ contains two time marks (resp. one time mark), we call time slot t *doubleton* (resp. *singleton*). Note that time slot t never contains three time marks because $\delta = 0.5$ and a time slot is a half-open interval by definition. Let Γ be the set of all singletons and doubletons. An interval $(s_1, s_2]$ is called *block* if it contains exactly one time slot t from Γ, i.e. $(t-1, t] \subseteq (s_1, s_2]$, and that the interval $(s_1, s_2]$ is maximal in the sense that expanding the interval into $(s_1, s_2 + \epsilon]$ or $(s_1 - \epsilon, s_2]$ for any $\epsilon > 0$ will make it intersect with another time slot from Γ. Note that two blocks can have non-empty intersection. By definition, a block is an extension of a singleton or doubleton and the endpoints of a block are integers. Note that a singleton or doubleton can be a block by itself, if the neighboring time slots are also in Γ. We use B_t to denote the block of time slot t. Two blocks $B_t, B_{t'}$ with $t < t'$ are referred to as *adjacent* if $\Gamma \cap \{t+1, t+2, ..., t'-1\} = \emptyset$.

Rounding Scheme. The rounding scheme works in many phases where in each phase, a time slot $t \in \Gamma$ is considered. First, we open time slot t. Moreover, if t is a doubleton, we may open an additional time slot $t-1$ or $t+1$. The rounding scheme is shown in Algorithm 1, which relies on definitions we introduce next. We aim to assign the fractional workload $y(B_t)$ into the slots we opened (i.e. subset of $\{t-1, t, t+1\}$) and show that the assignment is feasible for the jobs. The result, provided it is feasible, is a 2-approximation solution due to the fact that we open at most as many time slots as the total number of time marks.

If both time slots $t-1$ and $t+1$ are contained in Γ, block B_t equals time slot t and we are done, without opening any additional time slot. Thus we guarantee that any additional time slot opened in the algorithm is not included in Γ. One may note that the additional time slot $t-1$ or $t+1$ might be "opened twice" in the process by two adjacent blocks, possibly resulting in the infeasibility of the assignment of fractional workload. Therefore, Algorithm 1 is not the final algorithm and in Section 4 we propose a generalized rounding scheme for the final rounding algorithm and show that the assignment is feasible.

In order to assign the fractional workload $y(B_t)$ into the time slots t or its neighboring

[2] The endpoints of interval I are not restricted to be integers.

time slot $t-1$ or $t+1$, we need to guarantee that the assignment fulfills the LP constraints (2) (3), where constraints (2) make sure the total workload assigned into an active time slot is no more than g, and constraints (3) make sure the total workload of a single job assigned into an active time slot is no more than 1. This will be hard to achieve, but we can get help from this idea: note that two adjacent blocks might have an intersection, and the fractional workload during this intersection interval may be assigned twice, while in the algorithm we need to guarantee that this fractional workload is assigned at least once (to satisfy LP Constraint 1). Moreover, we can ignore any job j such that $y_j(B_t) > 0, t \notin I_j$ since the interval $B_t \cap (r_j, d_j]$ must be covered by a block which is adjacent to block B_t.

To begin with, we show that if t is a singleton, no additional time slot is needed and we simply assign $y_j(B_t)$ into time slot t for any job $j \in J$ with $t \in I_j$.

Lemma 1. If t is a singleton, the assignment that assigns $y_j(B_t)$ into time slot t for any job $j \in J$ with $t \in I_j$ fulfills the LP constraints (2) (3) for time slot t.

Proof. If t is a singleton, interval B_t only contains one time mark (regardless of endpoints of B_t), then by the definition of time marks, we have that $x(B_t) \leq 2\delta = 1$. Hence, from the LP solution, we have $y(B_t) \leq g \cdot x(B_t) \leq g$ and for each job $j \in J$, $y_j(B_t) \leq x(B_t) \leq 1$. Therefore, assigning the fractional workload $y(B_t)$ into time slot t fulfills the LP constraints (2) (3) for time slot t. $\square$

As a result, in the following, we only focus on the case where t is a doubleton. Let s_1, s_2 be the endpoints of block B_t, i.e. $B_t = (s_1, s_2]$. We consider the jobs that can be scheduled at time slot t and partition them into six disjoint classes, which are defined in the following and depicted in Figure 1.

Definition 2. For an interval I, let $\tilde{y}(I)$ be the workload of the LP solution in the interval I from the jobs j whose $(r_j, d_j]$ interval intersects $\Gamma \cap I$. In other words, we only consider the jobs that can be possibly scheduled in the singleton and doubletons that intersect I. Recall that $y(I)$ is the workload of the LP solution in the interval I of all the jobs.

Definition 3. We define job sets $\ddot{u}, \ddot{v}, \ddot{a}, \ddot{b}, \ddot{e}, \ddot{c}$ where jobs from sets $\ddot{a}, \ddot{u}$ have common deadline t with release time from $[0, s_1), [s_1, t-1)$ respectively, jobs from sets $\ddot{v}, \ddot{b}$ have common release time $t-1$ with deadline from $(t, s_2], (s_2, T]$ respectively, and jobs from set $\ddot{e}$ are tight jobs with release time $t-1$ deadline t, and jobs from set $\ddot{c}$ have release time strictly smaller than $t-1$ and deadline strictly larger than t. Let $\mathcal{Z} = \{\ddot{u}, \ddot{v}, \ddot{a}, \ddot{b}, \ddot{e}, \ddot{c}\}$. For each set of jobs $\ddot{z} \in \mathcal{Z}$, we denote z (resp. $\overline{z}$) as the summation of their fractional workload in time slot t (resp. during interval $B_t \setminus t$), i.e. $z = \sum_{j \in \ddot{z}} y_{jt}$ and $\overline{z} = \sum_{j \in \ddot{z}} (y_j(B_t) - y_{jt})$. In other words, in the following, we use the notations u, v, a, b, e, c and $\overline{u}, \overline{v}, \overline{a}, \overline{b}, \overline{e}, \overline{c}$ as the meaning of corresponding workload of jobs. We denote $\beta = \sum_{j \in J} y_{jt}$ as the total workload assigned to slot t in the LP solution.

We propose a rounding scheme in Algorithm 1, which makes an assignment of fractional workload except $\overline{a}$ or $\overline{b}$. In other words, for jobs from $\ddot{a}$ (or $\ddot{b}$), their fractional workload during $B_t \setminus t$ will not be assigned in this phase. Moreover, we make the decision of opening time slot $t-1$ or $t+1$ according to the jobs from $\ddot{u}$ and $\ddot{v}$, as described later. In terms of feasibility, if we open time slot $t+1$ (i.e. time slot $t-1$ is closed), the fractional workload $\overline{u}$ has to be assigned into time slot t because jobs from set $\ddot{u}$

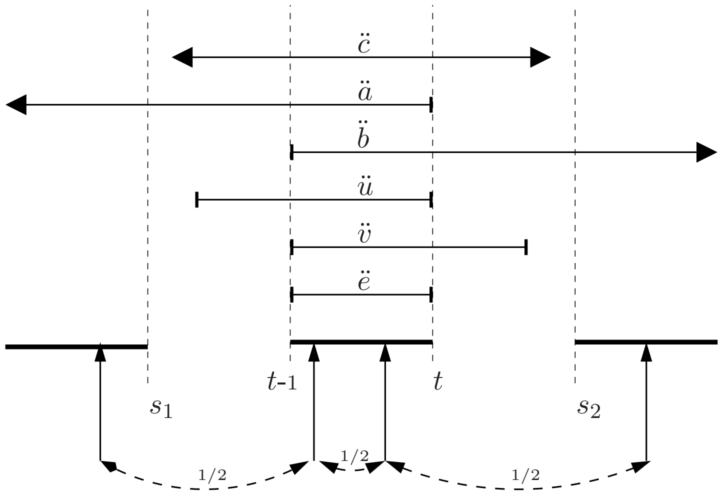

Figure 1: Illustration of the job partition as it pertains to one doubleton at time slot t, represented here by the middle solid bar. There are two other solid bars, one before and one after the doubleton, which are also in Γ. Time marks are represented by arrows pointing up. Job classes are represented by horizontal thinner lines, where an arrow represent that jobs in this class may or may not be longer than depicted in the direction of the arrow.

cannot be assigned into time slot $t+1$ due to their deadlines. Hence, in order to achieve a feasible assignment in which the total workload assigned to time slot t is no more than g, we adopt the strategy of shifting the fractional workload y_{jt} for any job $j \in \ddot{v}$ (i.e. v) into time slot $t+1$, in short, jobs $\ddot{u}$ in and jobs $\ddot{v}$ out. Symmetrically, if we open time slot $t-1$, we adopt a similar strategy of shifting the fractional workload y_{jt} for any job $j \in \ddot{u}$ (i.e. u) into time slot $t-1$. As a result, the rounding scheme checks which one of the two options is feasible, and we show that at least one of the two options is feasible if the following two properties are satisfied.

Property.

P(i) $p_j \leq 1, \forall j \in \ddot{u} \cup \ddot{v}$

P(ii) $y_{jt} \geq p_j/2, \forall j \in \ddot{u} \cup \ddot{v}$

We claim that the properties hold directly from the LP solution. Note the fact that no time mark occurs in interval $(s_1, t-1)$ and (t, s_2), hence $x([s_1, t-1]) \leq 0.5$, $x([t, s_2]) \leq 0.5$ and $x(B_t \setminus t) \leq 1$. As time slot t is a doubleton, at most two time marks occurs in B_t (regardless of interval endpoints), hence $x(B_t) \leq 1.5$. Since job $j \in \ddot{u} \cup \ddot{v}$ is fully contained in interval B_t (i.e. $(r_j, d_j] \subseteq B_t$), we have $p_j \leq x(B_t)$, i.e. $p_j = 1$. Moreover, due to the fact that jobs $\ddot{u}, \ddot{v}$ either have release time $t-1$ or deadline t, interval $(r_j, d_j) \setminus t$ is contained either in interval $(s_1, t-1)$ or (t, s_2), hence, $y_j(B_t \setminus t) \leq 0.5$, which implies $y_{jt} = p_j - y_j(B_t \setminus t) \geq 0.5$. Consequently, both properties hold directly from LP solution. Later on in Section 4, we will split some jobs so that the split jobs also conform to the above two properties.

Let $\psi = (u + \overline{u}) - (v + \overline{v})$ and $\lambda_1 = \overline{v} + \beta - u, \lambda_2 = \overline{u} + \beta - v$.

We claim that in the rounding scheme in Algorithm 1, the additional time slot t' is not included in Γ. If $\psi > 0$ we have $(u + \overline{u}) \geq \psi > 0$, hence $\ddot{u} \neq \emptyset$, i.e. $s_1 < t-1$. Therefore,

Algorithm 1 Rounding Scheme

```
for each: $t \in \Gamma$
 1: Open time slot $t$
 2: if $t$ is a doubleton then
 3:     Identify jobs $\ddot{u}, \ddot{v}, \ddot{a}, \ddot{b}, \ddot{c}, \ddot{e}$ and compute $\psi$ as described earlier.
 4:     Open slot $t' = t - 1$ if $\psi > 0$.
 5:     Open slot $t' = t + 1$ if $\psi < 0$.
 6:     Open a random slot $t'$ from $\{t-1, t+1\} \setminus \Gamma$ if $\psi = 0$.
 7: end if
 8: Call procedure JobAssignment
 9: procedure JobAssignment
10:     if $t' = t - 1$ then
11:         Assign fractional workload $\lambda_1 \to t, \overline{a} + \overline{c} \to t'$.
12:         Assign $y_j(B_t)$ into slots $t, t'$ whenever it is feasible, for each job $j \in \ddot{u}$.
13:     else if $t' = t + 1$ then
14:         Assign fractional workload $\lambda_2 \to t, \overline{b} + \overline{c} \to t'$.
15:         Assign $y_j(B_t)$ into slots $t, t'$ whenever it is feasible, for each job $j \in \ddot{v}$.
16:     else
17:         Assign fractional workload $\tilde{y}(B_t)$ into slot $t$.
18:     end if
19: end procedure
```

$t-1 \notin \Gamma$. Similarly, we have $t+1 \notin \Gamma$ if $\psi < 0$ due to $(v + \overline{v}) \geq -\psi > 0$. Moreover, if $\{t-1, t+1\} \setminus \Gamma = \emptyset$, then $u = \overline{u} = v = \overline{v} = 0$, and in the rounding scheme we do not open any additional time slot in this scenario; also in this case, $y(B_t)$ can definitely be assigned into slot t because in this case we have $s_1 = t-1, s_2 = t$. Therefore, the additional time slot t' that is opened by the algorithm is not included in Γ.

Lemma 2. In Algorithm 1 when doubleton $t \in \Gamma$ is considered, if $t' = t-1$, the fractional workload $\tilde{y}(B_t) - \overline{b}$ is feasibly assigned into the slots t and t'. If $t' = t+1$, the fractional workload $\tilde{y}(B_t) - \overline{a}$ is feasibly assigned into the slots t and t'.

We will show that Lemma 2 holds if the above two properties (P(i) and P(ii)) hold. Since in the rounding scheme we shift the fractional workload of jobs $\ddot{u}$ (or $\ddot{v}$) from time slot t to t', we have to guarantee that the workload assigned to t' for each of these jobs is no more than 1, hence the first property is necessary. The second property will guarantee that one of the two shifting strategies is feasible for time slot t.

Now, we are ready to prove Lemma 2.

Proof of Lemma 2. In the assignment of the rounding scheme, we never assign a job outside its window, i.e. jobs $\ddot{a}, \ddot{c}$ can be scheduled in time slot $t-1$ and jobs $\ddot{b}, \ddot{c}$ can be scheduled in time slot $t+1$. From the LP solution, we have

$$\overline{a} + \overline{b} + \overline{c} \leq g \cdot x(B_t \setminus t) \qquad \leq g \tag{6}$$

$$\beta \leq g \cdot x_t \qquad \leq g \tag{7}$$

Next, we show that the total workload that is assigned to each of the time slots t and t' is at most g. Using property P(ii), we have

$$\lambda_1 + \lambda_2 = 2\beta + (\overline{u} + \overline{v}) - (u + v) \leq 2\beta \leq 2g$$

Hence, $\min\{\lambda_1, \lambda_2\} \leq g$. Also, we have that $\lambda_2 - \lambda_1 = \psi$. If $\psi \geq 0$, we have $\lambda_1 = \min\{\lambda_1, \lambda_2\} \leq \beta \leq g$ and if $\psi \leq 0$ we have $\lambda_2 = \min\{\lambda_1, \lambda_2\} \leq \beta \leq g$. Therefore, the total fractional workload that is assigned to slot t is at most g. Moreover, the total fractional workload that is assigned to slot t' is at most g by inequality (6). We prove the lemma for the case $t' = t - 1$, as a symmetric argument can be constructed for the other case of $t' = t + 1$. We claim that the assignment in both time slots t, t' is feasible, without considering jobs $\ddot{u}$. Combining with the fact that $y_j(B_t \setminus t) \leq x(B_t \setminus t) \leq 1, \forall j \in J$, the assignment in slot t' ($\overline{a} + \overline{c} \to t'$) is feasible. Also, by property P(i), jobs $\ddot{u}, \ddot{v}$ have processing time at most 1, therefore, the assignment in slot t ($\lambda_1 \to t$) is feasible. The claim holds. Finally, for jobs $\ddot{u}$, in the rounding scheme we assign them into time slots t, t' whenever feasible. Because $y(B_t) \leq g \cdot x(B_t) \leq 1.5g$ and we opened two time slots t, t' in interval B_t, the assignment is feasible for jobs $\ddot{u}$. □

One may wonder why the workload $\overline{a}$ is not assigned in the case $t' = t + 1$, as we have just proved that there is enough capacity in slots t and $t + 1$. The issue is that for a job j in $\ddot{a}$, $y_j(B_t) > 1$ is possible, and only time slot t is available for such a job. A similar issue happens to $\overline{b}$ in the case $t' = t - 1$.

The above rounding scheme is the basis for the final rounding algorithm and the process in one phase is independent of another (recall that in each phase, just one time slot $t \in \Gamma$ is considered), however, it does not work directly. This is because the remaining fractional workload $\overline{a}$ or $\overline{b}$ may not be assigned in any phase, and also it could happen that the additional time slot t' is opened twice in two different phases. For the former case, we identify such situations (so-called "unlucky" jobs) and propose a job splitting method to adjust the rounding scheme and guarantee a feasible assignment. For the latter case, we propose a generalized rounding approach, which particularly identifies the doubletons that might cause reopening; this happens because of the so-called "bad" jobs. These doubletons are partitioned into so-called "doubleton chains" and handled in the next section.

4 Generalized Rounding Scheme

In this section, we propose the final rounding scheme in which all jobs are feasibly assigned. The scheme appears later in Algorithm 2, which relies on definitions we introduce next. We introduce the concepts of *unlucky* jobs and *doubleton chain* where in the previous rounding scheme, the former might have fractional workload that is not assigned and the latter could result in reopening time slots. In the algorithm, we propose a *job splitting* approach to obtain a feasible assignment for unlucky jobs. Moreover, we propose a new approach to open additional time slots for the doubleton chains.

Definition 4 (Unlucky Job)**.** Job $j \in J$ is called *unlucky* if $d_j - r_j \geq 3$ and both time slots $r_j + 1$ and d_j are doubletons such that $\Gamma \cap I_j = \{r_j + 1, d_j\}$.

Note that the definition of unlucky job is based on LP solution.

Definition 5 (Job Splitting)**.** A job *splitting process* of an unlucky job j is a process that replaces job j by two jobs j_1, j_2 such that jobs j_1, j_2 have job windows $(r_j, d_j - 1], (r_j + 1, d_j]$ respectively and $p_{j_1} + p_{j_2} = p_j$ with $p_{j_1} = \frac{\alpha_1}{\alpha_1+\alpha_2}$ if $p_j = 1$ and otherwise $p_{j_1} = 1$, where $\alpha_1 = y_j([r_j, r_j + 1])$ and $\alpha_2 = y_j([d_j - 1, d_j])$. Moreover, the fractional workload of job j during time interval $(r_j, t']$ (resp. $(t', d_j]$) in the LP solution belongs to job j_1 (resp. j_2), where t' is the earliest time such that $y_j([r_j, t']) = p_{j_1}$. (Note that $r_j + 1 \le t' \le d_j - 1$, and thus this fractional workload is inside a job's window).

In the following, we use *split jobs* to indicate the job set in which all the unlucky jobs are split (and other jobs remain unchanged). By applying splitting process for an unlucky job j, the two split jobs j_1, j_2 are categorized into job set $\ddot{v}$ (resp. $\ddot{u}$) when the rounding scheme processes the doubleton $r_j + 1$ (resp. d_j). The reason we use splitting is that, intuitively, Lemma 2 handles the jobs in $\ddot{u}$ and $\ddot{v}$ better than the jobs in $\ddot{a}$ and $\ddot{b}$. The generalized rounding scheme only considers the split jobs. This means that $\tilde{y}()$ in Definition 2 is adjusted to use the split jobs in the definition instead of the original (before split) jobs.

Lemma 3. For each unlucky job j, we have $p_j \le 2$ and properties P(i) and P(ii) hold for both jobs j_1, j_2 after splitting.

Proof. Since $\Gamma \cap I_j = \{r_j + 1, d_j\}$ by definition, no time mark occurs in interval $(r_j + 1, d_j - 1)$, hence we have $x([r_j + 1, d_j - 1]) \le 0.5$, which implies that $x([r_j, d_j]) \le 2.5$. As a result, $p_j \le 2$. If $p_j = 2$, we have $\alpha_1 + \alpha_2 \ge p_j - x([r_j + 1, d_j - 1]) \ge 1.5$. Also note that $p_{j_1} = p_{j_2} = 1$ by definition of splitting process. Because $\alpha_2 \le 1$, we have $\alpha_1 \ge 0.5$, and symmetrically $\alpha_2 \ge 0.5$. Otherwise $p_j = 1$. Since $\alpha_1 + \alpha_2 \ge p_j - x([r_j + 1, d_j - 1]) \ge 0.5$, we have $p_{j_1} = \frac{\alpha_1}{\alpha_1+\alpha_2} \le 2\alpha_1$ and $p_{j_2} = 1 - \frac{\alpha_1}{\alpha_1+\alpha_2} = \frac{\alpha_2}{\alpha_1+\alpha_2} \le 2\alpha_2$. In either case, properties P(i) and P(ii) hold for jobs j_1, j_2. □

Definition 6 (Doubleton Chain)**.** A set of doubletons $C = \{t_0, t_0 + 2, ..., t_0 + 2l\}$ with $t_0 \in \Gamma, l \in \mathbb{N}$ is called *doubleton chain* if $\{t_0, t_0 + 1, ..., t_0 + 2l\} \cap \Gamma = C$. Moreover, C is called *critical* if there exists no doubleton chain C' such that $C \subset C'$.

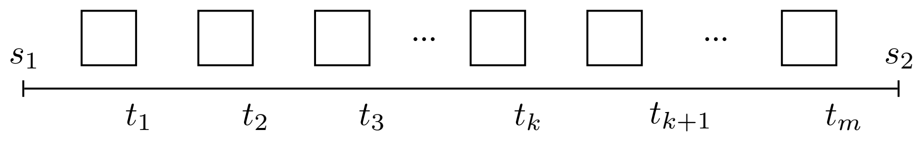

Figure 2: Doubleton chain

See Figure 2 for an illustration of doubleton chain. We first identify the critical doubleton chains from the LP solution, and for each critical doubleton chain we apply a recursive rounding scheme (describe later) to open the time slots and obtain assignment of the split jobs. Meanwhile, for the singletons, we stick to the same rounding scheme as Algorithm 1. Note that a doubleton chain could possibly contain only one doubleton. We will guarantee that in the generalized rounding scheme, no time slot is opened twice and the obtained assignment is feasible for the split jobs, as well as the original jobs.

Now, we introduce the generalized rounding scheme for a doubleton chain. Let $C = \{t_1, t_2, ..., t_m\}$ be a doubleton chain of m ($m \ge 1$) doubletons (so, for $1 \le i \le m$, we have

$t_i = t_1 + 2(i-1)$), and $\mathcal{I} = (s_1, s_2]$ be the union of blocks of these m doubletons (see also Figure 2). The goal is to open an additional time slot for each doubleton from C so that each additional time slot is opened only once and the fractional workload $\tilde{y}(\mathcal{I})$ can be feasibly assigned into the opened time slots. Similarly, we ignore any job j such that $C \cap I_j = \emptyset$ and $y_j(\mathcal{I}) > 0$, since for this job either $r_j \geq t_m$ or $d_j \leq t_1 - 1$ and the interval $\mathcal{I} \cap (r_j, d_j]$ must be covered by another block.

From the definition of a doubleton chain, we have the following Fact.

Fact 7. $x([s_1, t_1 - 1]) \leq 0.5$, $x([t_m, s_2]) \leq 0.5$ and $x([t_i, t_i + 1]) \leq 0.5, \forall i \in [1, m]$.

After this paragraph, we assume $s_1 < t_1 - 1, s_2 > t_m$ as the other cases are easy, as shown below. If $s_1 = t_1 - 1$, we open the additional time slots $\{t_i + 1 \mid t_i + 1 \leq s_2, i \in [1, m]\}$. Note that the whole interval $(s_1, t_m]$ is completely active. For interval $(t_m, s_2]$, if $s_2 > t_m$ we assign all the corresponding fractional workload during interval $(t_m, s_2]$ to time slot $t_m + 1$; this assignment is feasible due to Fact 7. Symmetrically, if $s_2 = t_m$, another easy solution can be constructed.

We start from a critical double chain C, and reduce to sub-problem by reducing the length of the doubleton chain and making partial assignment in some blocks. In the beginning of each iteration, we maintain the invariant that time slots $t_1 - 1$ and $t_m + 1$ have not been opened and apply one of the following operations.

Definition 8. The two operations we use for opening additional time slots when processing a doubleton chain are:

- **Option 1:** open additional time slot $t_i + 1$ for each $i \in [1, m]$.
- **Option 2:** select $k \in [1, m]$, open additional time slot $t_i - 1$ for each $i \in [1, k]$ and reduce to the sub-problem with doubleton chain $\{t_{k+1}, t_{k+2}, ..., t_m\}$.

Before we describe the process of choosing from the above two options, we introduce necessary definitions related to certain unlucky jobs that we call "bad", as the choice highly depends on bad jobs.

Definition 9. A job j is called *bad* if it is an unlucky job with $d_j - r_j = 3, p_j = 2$. For doubleton $t_i, i \in [1, m]$, let $\ddot{h}_i$ be the set of bad jobs who have release time $t_i - 1$. Recall the definition of $\ddot{u}$ in Definition 3, we use a similar notation $\ddot{u}_i$ (resp. $\ddot{v}_i, \ddot{a}_i, \ddot{b}_i, \ddot{c}_i,$) with respect to the doubleton t_i for the split jobs. Moreover, when we split a bad job from $\ddot{h}_i$, let $\ddot{h}_i^+$ and $\ddot{h}_i^-$ be the sets containing the two split jobs respectively. Let h_i^+ (resp. h_i^-) be the total fractional workload of the split jobs $\ddot{h}_i^+$ in slot t_i (resp. t_{i+1}), in other words, $h_i^+ = \sum_{j_1 \in \ddot{h}_i^+} y_{j_1}(t_i)$ and $h_i^- = \sum_{j_2 \in \ddot{h}_i^-} y_{j_2}(t_{i+1})$. And let $\overline{h_i^+} = (\sum_{j_1 \in \ddot{h}_i^+} p_{j_1}) - h_i^+$ and $\overline{h_i^-} = (\sum_{j_2 \in \ddot{h}_i^-} p_{j_2}) - h_i^-$.

See Figure 3 for an illustration. Note that by definition of the splitting process, we have $\ddot{h}_i^+ \subseteq \ddot{v}_i, \ddot{h}_i^- \subseteq \ddot{u}_{i+1}$. Also note that for an unlucky job which is not a bad job, set $\ddot{u}_i$ already contains the corresponding split job. Moreover, since time slot t_m is the last time slot of a critical doubleton chain, we have $\ddot{h}_m = \emptyset$.

Fact 10. For each $i \in [1, m]$, we have $\max\{\overline{h_i^-}, \overline{h_i^+}\} \leq \min\{h_i^+, h_i^-\}$.

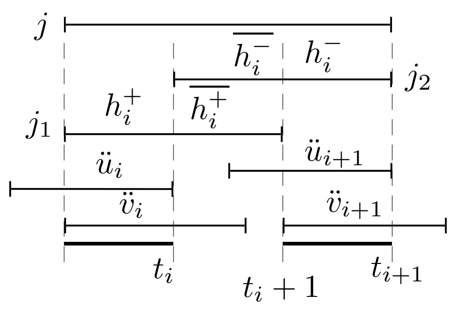

Figure 3: Illustration of Definition 9. As before, doubletons are depicted by solid bars, and jobs are represented by horizontal thinner lines. Bad job j covering two doubletons t_i and t_{i+1} is split into two jobs j_1 and j_2. The split job j_1 belongs to job set $\ddot{h}_i^+$, and jobs $\ddot{h}_i^+$ is part of $\ddot{v}_i$, i.e. $j_1 \in \ddot{h}_i^+ \subseteq \ddot{v}_i$. Similar for split job j_2 that $j_2 \in \ddot{h}_i^- \subseteq \ddot{u}_{i+1}$.

Proof. Given a bad job $j \in \ddot{h}_i$, let j_1, j_2 be the two jobs after splitting. Recall that $p_j = 2, p_{j_1} = p_{j_2} = 1$. By property P(ii), we have $y_{j_1}(t_i) \geq 0.5$ and $y_{j_2}(t_{i+1}) \geq 0.5$, which implies $y_{j_1}(t_i + 1) \leq 0.5$ and $y_{j_2}(t_i + 1) \leq 0.5$. Hence, by summing up this over the bad jobs of $\ddot{h}_i$, we get that the fact is true. □

Definition 11. Given a constant $\omega \in [0, 1]$, a *forward shifting process* of a bad job $j \in \ddot{h}_i$ is a process that moves an amount of $\omega \cdot y_{j_1}(t_i)$ workload from time slot t_i into time slot $t_i + 1$ and moves an amount of $\omega \cdot y_{j_2}(t_i + 1)$ workload from time slot $t_i + 1$ into time slot t_{i+1}, i.e. $\omega \cdot y_{j_1}(t_i) \rightarrow t_i + 1, \omega \cdot y_{j_2}(t_i + 1) \rightarrow t_{i+1}$. Similarly, a *backward shifting process* is defined as $t_i \leftarrow \omega \cdot y_{j_1}(t_i + 1), t_i + 1 \leftarrow \omega \cdot y_{j_2}(t_{i+1})$, which is the reverse of forward shifting process.

In the forward shifting process of a bad job j, the total amount of workload assigned to time slot $t_i + 1$ of job j is at most 1 for any $\omega \in [0, 1]$, that is

$$y_{j_1}(t_i + 1) + \omega \cdot y_{j_1}(t_i) + (1 - \omega) \cdot y_{j_2}(t_i + 1) \leq y_{j_1}(t_i + 1) + y_{j_1}(t_i) = 1 \tag{8}$$

The inequality holds due to the fact that, using Fact 7 and Property P(ii), $y_{j_2}(t_i + 1) \leq x([t_i, t_i + 1]) \leq 0.5 \leq y_{j_1}(t_i)$. Therefore, the new assignment of job j in time slot $t_i + 1$ is feasible. Also, shifting does not lead to the after-shifting $y_{j_2}(t_{i+1}) > 1$, since $y_{j_2}(t_i + 1) + y_{j_2}(t_{i+1}) = p_{j_2} = 1$. A similar argument can be constructed to show that after the backward shifting process of a bad job j, the new assignment is also feasible for job j. That is $(1 - \omega) y_{j_1}(t_i + 1) + y_{j_2}(t_i + 1) + \omega \cdot y_{j_2}(t_{i+1}) \leq 1$. (We will consider the LP constraints 2 later, in lemmas 5 and 6).

Consider Option 1 from Definition 8, in order to make jobs $\ddot{u}_1$ feasible, we have to assign the fractional workload $\overline{u_1}$ into time slot t_1 and hence shift some fractional workload of jobs $\ddot{v}_1$ from time slot t_1 into time slot $t_1 + 1$. Specifically, for jobs $\ddot{v}_1$, we first shift the fractional workload of jobs $\ddot{v}_1 \setminus \ddot{h}_1^+$ and then jobs $\ddot{h}_1^+$. For each job of $\ddot{h}_1^+$, we apply the forward shifting process with gradually increasing parameter ω from 0 to 1 such that a total amount of $\overline{u_1} - (v_1 - h_1^+)$ workload of jobs of $\ddot{h}_1^+$ is shifted out of time slot t_1 (recall that $\ddot{h}_i^+ \subseteq \ddot{v}_i$ and therefore h_1^+ is "included" in v_1). Then for doubleton t_2, since some workload of jobs of $\ddot{h}_1^-$ is shifted into time slot t_2, we again need to shift some fractional workload of jobs of $\ddot{v}_2$ (which includes $\ddot{h}_2^+$) out of time slot t_2. We continue such shifting process until at some time slot t_k ($1 \leq k \leq m$), the fractional workload v_k

is not enough to compensate for the incoming workload that is shifted into time slot t_k, or all the workload $\overline{u_1}$ are shifted. If the former occurs, we then show that Option 2 from Definition 8 is feasible, by applying backward shifting from time slot t_k to t_1. If the latter occurs, we take Option 1 from Definition 8 and shift the workload from t_1 to t_k as above.

Definition 12. A *path* is a vector $\langle q_1, q_2, ..., q_k\rangle$ with $k \in [1, m]$ and $q_i \in \mathbb{R}_{\geq 0}, i \in [1, k]$ such that

$$\begin{array}{rll} & q_1 & \leq \overline{u_1} \\ \forall 1 \leq i < k, & q_{i+1} & = \begin{cases} (q_i - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+ & , \text{ if } (v_i - h_i^+) < q_i \\ 0 & , \text{ otherwise} \end{cases} \\ \forall 1 \leq i \leq k, & q_i & \leq v_i \end{array}$$

A path $\langle q_1, q_2, ..., q_k\rangle$ defines a way to transfer a portion of workload of $\overline{u_1}$ into time slot t_k via the forward shifting process and q_i indicates the amount of workload shifted into time slot t_i in the path. In Definition 12, the value q_{i+1} in a path is always properly defined as long as the last constraint holds, i.e. $q_i \leq v_i$. This is true because $q_i - (v_i - h_i^+) > 0$ implies $h_i^+ > v_i - q_i \geq 0$, hence the term $\overline{h_i^-}/h_i^+$ is properly defined. Part of the following lemma is used as a procedure in Algorithm 2.

Lemma 4. Let $P = \langle q_1, q_2, ..., q_k\rangle$ be a path with $q_1 = \overline{u_1}, k = m$, and let $Q = \langle q_1', q_2', ..., q_k'\rangle$ be a path with $1 \leq k \leq m$, $q_k' = v_k$, then path P exists or path Q exists. Moreover, path Q with length k (if exists) can be computed in $O(k)$ operations, and path P (if exists) can be computed in $O(m)$ operations.

Proof. Consider the process of constructing a path with $q_1 = \overline{u_1}$, let k be the largest integer with $1 \leq k \leq m$ such that $\forall 1 \leq i < k$, $q_i \in (v_i - h_i^+, v_i], q_{i+1} = (q_i - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+$. If $q_k \in (v_k - h_k^+, v_k]$, we have $k = m$ as otherwise k is not the largest integer as defined, therefore, we take path P to be $\langle q_1, q_2, ..., q_k\rangle$ and it is properly defined. If $q_k \leq v_k - h_k^+$, then path P exists as the next term q_{k+1} equals 0 according to Definition 12, and we could always extend the path by adding zero terms. Otherwise, $q_k > v_k$. Obviously, we have $1 \leq i < k, \overline{h_i^-} > 0, h_i^+ > 0$ as $q_k > 0$. Consider path $\langle q_1', q_2', \ldots, q_k'\rangle$ where $\forall 1 \leq i \leq k, q_i' = q_i - (q_k - v_k) \prod_{i'=i}^{k-1} h_{i'}^+/\overline{h_{i'}^-}$ (the product, as usually, is taken to be 1 when $i = k$). Note that $\forall 1 \leq i < k, q_{i+1}' - q_{i+1} = (q_i' - q_i) \cdot \overline{h_i^-}/h_i^+ < 0$, and also $q_{i+1}' = (q_i' - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+$. As a result, path $\langle q_1', q_2', ..., q_k'\rangle$ is a valid path and $q_k' = v_k$, hence we take it as path Q. As a consequence, path P can be computed in $O(m)$ operations, and path Q can be computed in $O(k)$ operations. □

In the generalized rounding scheme, if path P exists, we take Option 1 from Definition 8, otherwise path Q exists, we take Option 2 from Definition 8, with value k. We prove in lemmas 5 and 6 that for both options, a feasible assignment exists. As an aside, when $m = 1$, path P with $q_1 = \overline{u_1}$ exists if $\overline{u_1} \leq v_1$, and path Q exists otherwise. The options taken are not the same as those in Algorithm 1, but are closely related and one can reprove Lemma 2 with both of these options.

Fact 13. Given a path $\langle q_1, q_2, ..., q_k\rangle$, we have $q_i \geq q_{i+1}, \forall 1 \leq i < k$.

Algorithm 2 Generalized Rounding Scheme

```
1: Open all time slots in $\Gamma$
2: $\Gamma_s \leftarrow$ collection of singletons
3: $\mathcal{C} \leftarrow$ collection of critical doubleton chains
4: for $t \in \Gamma_s$ do
5:     Assign fractional workload $\tilde{y}(B_t)$ into slot $t$.
6: end for
7: for $C \in \mathcal{C}$ do                                   ▷ doubleton chain $C$
8:     while $C \neq \emptyset$ do
9:         $C \leftarrow$ AssignmentPath($C$)
10:    end while
11: end for
12: procedure AssignmentPath($C = \{t_1, t_2, ..., t_m\}$)
13:     if Path $P$ exists then                                  ▷ Lemma 4
14:         Open time slots $\{t_1 + 1, t_2 + 1, ..., t_m + 1\}$
15:         Do assignment($P$)                                   ▷ Definition 14
16:         Return $\emptyset$
17:     else
18:         Obtain Path $Q = \langle q_1, q_2, ..., q_k \rangle$.             ▷ Lemma 4
19:         Open time slots $\{t_1 - 1, t_2 - 1, ..., t_k - 1\}$
20:         Do assignment($Q$)                                   ▷ Definition 15
21:         Return $\{t_{k+1}, t_{k+2}, ..., t_m\}$
22:     end if
23: end procedure
```

Proof. $\forall 1 \le i < k$, if $h_i^+ = 0$, we have $q_i - (v_i - h_i^+) \le 0$, i.e. $q_{i+1} = 0$. Hence, if $q_{i+1} > 0$, there must be $q_i - (v_i - h_i^+) > 0$ and $h_i^+ > 0$. Therefore, $q_{i+1} = (q_i - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+ \le q_i - (v_i - h_i^+) \le q_i$ due to $\overline{h_i^-} \le h_i^+$ in Fact 10 (recall also that $\ddot{h}_i^+ \subseteq \ddot{v}_i$ and therefore $h_i^+ \le v_i$). □

The following definition is used as a procedure in Algorithm 2. Recall that $\mathcal{I} = (s_1, s_2]$ is the union of blocks of the m doubletons (see also Figure 2).

Definition 14. Given path $P = \langle q_1, q_2, ..., q_k \rangle$ with $q_1 = \overline{u_1}$ and $k = m$, we open time slots $\{t \mid t_1 \le t \le t_m + 1, t \in \mathbb{N}\}$ and let *assignment*(P) be the assignment of fractional workload $\tilde{y}(\mathcal{I})$ obtained by applying the following process on the LP solution.

i. For each t_i, $i \in [1, m]$, we move z_i amount of fractional workload of jobs $\ddot{v}_i \setminus \ddot{h}_i^+$ from time slot t_i into time slot $t_i + 1$ where $z_i = \min\{q_i, v_i - h_i^+\}$.

ii. For each t_i, $i \in [1, m)$, for each bad job $j \in \ddot{h}_i$, apply the forward shifting process with parameter ω_i where $\omega_i = \frac{q_i - (v_i - h_i^+)}{h_i^+}$ if $q_i - (v_i - h_i^+) > 0$, and otherwise $\omega_i = 0$. Note that $\omega_i \le 1$ since $q_i \le v_i$ in Definition 12.

iii. assign $\overline{u_1} \to t_1$, $y'([s_1, t_1 - 1]) - \overline{u_1} - \overline{a_1} \to t_1 + 1$, $y'([t_m, s_2]) \to t_m + 1$. Here $y'()$ excludes from $y()$ the fractional workload of $y(\mathcal{I})$ that does not count for $\tilde{y}(\mathcal{I})$.

Lemma 5. If path P exists, in *assignment*(P) the fractional workload $\tilde{y}(\mathcal{I}) - \overline{a_1}$ is feasibly assigned into time slots $\{t \mid t_1 \le t \le t_m + 1, t \in \mathbb{N}\}$.

Proof. Recall that we ignore any job j such that $C \cap I_j = \emptyset, y_j(\mathcal{I}) > 0$, as such a job does not contribute to $\tilde{y}(\mathcal{I})$. Due to the fact that jobs in $\ddot{u}_1$ and $\ddot{a}_1$ have the same deadline t_1, any job j that contributes to the fractional workload $y'([s_1, t_1 - 1]) - \overline{u_1} - \overline{a_1}$ must have deadline larger than t_1, hence j can be assigned into time slot $t_1 + 1$. Also note that $q_1 = \overline{u_1} \le x([s_1, t_1 - 1]) \cdot g \le 0.5g$ by Fact 7.

We claim that the total workload assigned to each time slot t_i, $i \in [1, m]$ remains the same. According to the first two steps in Definition 14, the workload shifted out of time slot t_i equals $z_i + \omega_i \cdot h_i^+$ $\forall i \in [1, m]$, which equals q_i. Moreover, the workload shifted into time slot $t_{i+1}, \forall i \in [1, m)$ equals $\omega_i \cdot \overline{h_i^-}$, which equals q_{i+1} by the definition of path P. Combining with the fact that $q_1 = \overline{u_1}$ is the amount of workload shifted into time slot t_1, the total amount of workload that is shifted in and out of time slot t_i equals q_i, $\forall i \in [1, m]$, hence, the claim is true.

Next we claim that for each additional time slot $t_i + 1$, $i \in [1, m]$, the total workload assigned in $t_i + 1$ is no more than g (thus fulfilling the LP constraints 2). For time slot $t_1 + 1$, as argued earlier the total workload shifted from time slot t_1 into $t_1 + 1$ equals q_1, which equals $\overline{u_1}$. Moreover, in the last step in Definition 14, we assign at most $y'([s_1, t_1 - 1]) - \overline{u_1} - \overline{a_1}$ workload into time slot $t_1 + 1$. Hence in total the workload assigned into time slot $t_1 + 1$ is at most $y'([s_1, t_1 - 1]) - \overline{a_1} + y([t_1, t_1 + 1]) \le g \cdot x([s_1, t_1 - 1]) + g \cdot x([t_1, t_1 + 1]) \le g$ due to Fact 7. Moreover, note that we have $q_{i+1} \le q_i, i \in [1, m)$ (Fact 13). Therefore, the increase of workload during time slot $t_i + 1, \forall i \in (1, m]$ is no more than q_1, which is at most $0.5g$. Then we conclude that the total workload assigned in each additional time slot $t_i + 1, i \in [1, m]$ is at most g.

Finally, using the definition of the forward shifting process (precisely, Equation (8) and the discussion that follows it), and the fact that jobs in $\ddot{v}_i \setminus \ddot{h}_i^+$ have processing time at most 1, for each job $j \in J$, for each time slot $t, t_1 \leq t \leq t_m + 1$, the total workload of job j that is assigned into t is at most 1. With this, the lemma is proved. $\square$

Now we consider the opposite case where path Q as in Lemma 4 exists. The following definition is used as a procedure in Algorithm 2.

Definition 15. Given path $Q = \langle q'_1, q'_2, ..., q'_k \rangle$ with $1 \leq k \leq m$, $q'_k = v_k$, let vector $\hat{Q} = \langle \hat{q}_1, \hat{q}_2, ..., \hat{q}_k \rangle$ where $\hat{q}_i = 0, \forall 1 \leq i \leq k$ if $v_k = 0$, otherwise

$$\begin{aligned} \hat{q}_k &= \overline{v_k} \\ \hat{q}_i &= \hat{q}_{i+1} \cdot \overline{h_i^+}/h_i^-, \quad \forall 1 \leq i < k \end{aligned}$$

We open time slots $\{t \mid t_1 - 1 \leq t \leq t_k, t \in \mathbb{N}\}$ and let *assignment*(Q) be the assignment for fractional workload $\tilde{y}(\mathcal{I}')$ obtained by applying the following process on the LP solution where $\mathcal{I}' = (s_1, t_k + 1]$.

i. For time slot t_1, we move $\hat{q}_1$ amount of fractional workload of jobs $\ddot{u}_1$ from time slot t_1 into time slot $t_1 - 1$.

ii. For each t_i, $i \in [1, k)$, for each bad job $j \in \ddot{h}_i$, apply the backward shifting process with parameter ω_i where $\omega_i = \frac{\hat{q}_{i+1}}{h_i^-}$. We prove that $\omega_i \leq 1$ after the definition.

iii. assign $\overline{v_k} \to t_k$, $y(t_k+1) - \overline{v_k} - \overline{b_k} \to t_k - 1$, $y'([s_1, t_1 - 1]) \to t_1 - 1$. Here, as before, $y'()$ excludes from $y()$ the fractional load of $y(\mathcal{I})$ that does not count for $\tilde{y}(\mathcal{I})$.

Note that indeed $\omega_i \leq 1$, since as shown in the next lemma, $\hat{q}_i \leq q'_i, \forall 1 \leq i \leq k$, and if $q'_{i+1} > 0$, then (from Definition 12) $q'_{i+1} = (q'_i - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+$. Now, $q'_i \leq v_i$ (also from Definition 12) and therefore $(q'_i - (v_i - h_i^+)) \cdot \overline{h_i^-}/h_i^+ \leq \overline{h_i^-}$. Finally, $\overline{h_i^-} \leq h_i^-$ from Fact 10. We conclude that $\hat{q}_{i+1} \leq h_i^-$, and therefore $\omega_i \leq 1$.

Lemma 6. If path Q exists, in *assignment*(Q) the fractional workload $\tilde{y}(\mathcal{I}') - \overline{b_k}$ is feasibly assigned into time slots $\{t \mid t_1 - 1 \leq t \leq t_k, t \in \mathbb{N}\}$ where $\mathcal{I}' = (s_1, t_k + 1]$.

Proof. First, we claim that $\hat{q}_i \leq q'_i, \forall 1 \leq i \leq k$. Indeed, for path Q, it follows from Fact 13 that $q'_i \geq q'_{i+1}, \forall 1 \leq i < k$. If $v_k = 0$, then $\overline{v_k} = 0$ by Property P(ii) and therefore $\hat{q}_i = 0$ for $1 \leq i \leq k$, and the claim follows. Otherwise, $v_k > 0$ and therefore $q'_k > 0$. Since $q'_k > 0$, we have $\ddot{h}_i \neq \emptyset$, $\forall i \in [1, k)$ as otherwise $q'_{i+1} = 0$ by the definition of path Q. In path $\hat{Q}$, we have $\hat{q}_i \leq \hat{q}_{i+1}, \forall 1 \leq i < k$ due to $\overline{h_i^+} \leq h_i^-$ from Fact 10. Moreover, in time slot t_k, we have $\hat{q}_k = \overline{v_k} \leq v_k = q'_k$ due to $\overline{v_k} \leq v_k$. Therefore, the claim is true. As a result, we have $\hat{q}_1 \leq q'_1 \leq \overline{u_1} \leq u_1$.

Then we claim that the total workload assigned to each time slot t_i, $i \in [1, k]$ remains the same. According to the second step in Definition 15, for each $i \in [1, k)$, the workload shifted out of time slot t_{i+1} equals $\omega_i \cdot h_i^-$ during the backward shifting process, which equals $\hat{q}_{i+1}$. Moreover, the workload shifted into time slot t_i equals $\omega_i \cdot \overline{h_i^+}$, which equals $\hat{q}_i$ by definition of vector $\hat{Q}$. Therefore, the total amount of workload that is shifted in and out of time slot t_i equals $\hat{q}_i$, $\forall i \in [1, k]$. Hence, the claim is true.

Finally, using $\hat{q}_i \leq q'_i \leq q'_1 \leq \overline{u_1} \leq 0.5g$, a similar argument as in the proof of Lemma 5 could be constructed to show that the fractional workload $\tilde{y}(\mathcal{I}') - \overline{b_k}$ is feasibly assigned into time slots $\{t \mid t_1 - 1 \leq t \leq t_k, t \in \mathbb{N}\}$. □

As a consequence, the recursive rounding scheme either assigns the workload $\tilde{y}(\mathcal{I}) - \overline{b_m}$ or $\tilde{y}(\mathcal{I}) - \overline{a_1}$ into the opened time slots.

Lemma 7. For each job $j \in J$, all the fractional workload of j is assigned in the generalized rounding scheme (Algorithm 2).

Proof. Suppose for a contradiction that interval I' is the biggest interval such that there exists a job $j \in J$ and the positive workload $y_j(I')$ is not assigned at all. Then combining with Lemma 5 and 6, the interval I' must be the gap between two consecutive time slots from Γ, i.e. $I' = (t'_1, t'_2 - 1]$ where time slots $t'_1 \in \Gamma, t'_2 \in \Gamma$ and $t'_2 - t'_1 > 1$ (the cases $t'_1 = 0$ or $t'_2 - 1 = T$ are handled by the same arguments). We have $r_j < t'_2 - 1$ and $d_j > t'_1$. If $d_j > t'_2$, the workload $y_j(I')$ must be assigned when we process the time slot t'_2 since $r_j < t'_2 - 1 < t'_2 < d_j$, either in the process of a doubleton chain containing t'_2 or in the process of a singleton t'_2. Therefore, we have $d_j \leq t'_2$. A symmetric argument can be constructed to show that $r_j \geq t'_1 - 1$. Note the fact that $x(I') \leq 0.5$. Hence, if $r_j > t'_1 - 1$ we have $d_j = t'_2$ and if $d_j < t'_2$ we have $r_j = t'_1 - 1$, in either case $y_j(I')$ must be assigned, since in this scenario job j will not be categorized into job set $\ddot{a}$ (resp. $\ddot{b}$).

Then we discuss the final case where $r_j = t'_1 - 1, d_j = t'_2$. If either t'_1 or t'_2 is a singleton, $y_j(I')$ must be assigned in Step 4 of Algorithm 2. Otherwise if both t'_1 and t'_2 are doubletons, then job j is a unlucky job. We have shown in the generalized rounding scheme that the split jobs of an unlucky job are categorized into job set $\ddot{v}$ (resp. $\ddot{u}$) when the rounding scheme processes the doubleton t'_1 (resp. t'_2), and hence $y_j(I')$ is assigned, which is a contraction. □

Running Time Analysis We assume there are efficient algorithms to obtain the optimal LP solution, in this part we only discuss the running time of the rounding algorithm (Algorithm 2). We will only focus on the necessary operations to open the time slots, while the final job assignment can be obtained by applying max-flow algorithm over the active time slots. The job assignment in the algorithm is only for analysis of correctness and feasibility.

It takes $O(T)$ operations to identify all time marks, singletons, doubletons, blocks, critical doubleton chains. It takes $O(1)$ operations to identify whether a job is an unlucky job and split an unlucky job, and it takes $O(n)$ time to identify jobs $\ddot{u}, \ddot{v}$ for all doubletons since for all these jobs either their release times or deadlines are equal to the endpoints of a doubleton. Job sets $\ddot{a}, \ddot{b}, \ddot{c}, \ddot{e}$ can be ignored as they does not affect the decision of opening time slots. Moreover, for each set $\ddot{u}$, it takes $O(|\ddot{u}|)$ operations to compute the values u and $\overline{u}$ (analogous for job sets $\ddot{v}$), where for each job in $\ddot{u}$ it takes $O(1)$ operations correspondingly. For each critical doubleton chain with m time slots, it takes $O(k)$ operations to obtain a path Q with length $k < m$ (if exists) and $O(m)$ operations to obtain a path P (if exists), that is to say, it takes $O(m)$ operations in total to identify all paths Q. Therefore, each critical doubleton chain needs $O(m)$ operations.

In conclusion, the rounding algorithm takes $O(n + T)$ operations to obtain active time slots. As a comparison, the algorithm needs to invoke max-flow algorithm only once to get job assignment, while the previous algorithm [4] needs to invoke many times of max-flow algorithm, depending on the active time slots. Also, our rounding algorithm can be easily parallelized since the rounding scheme for each block of singleton or critical doubleton chain is independent of another, while in the previous method the decision at time slot t relies on the decision of earlier time slots.

The algorithm is indeed a 2-approximation algorithm due to the fact that we open at most as many time slots as the total number of time marks.

New Linear Programming Formulation In this part, we show a new linear programming formulation by adding extra necessary constraints in the original LP rounding formulation.

Definition 16. Given an interval $I = (t_1, t_2]$ with $t_1 \in \mathcal{T}$, $t_2 \in \mathcal{T}$, for each job $j \in J$, let $p_j(I) = \max\{0, p_j - |(r_j, d_j] \setminus I|\}$, where $|(r_j, d_j] \setminus I|$ indicates the length of $(r_j, d_j] \setminus I$ (if nonempty, the set $(r_j, d_j] \setminus I$ consists of one or two intervals).

The value $p_j(I)$ indicates the minimum amount of workload of job j that has to be assigned during interval I in a feasible schedule, providing that every time slot outside I is already active. Then we add the following constraints (valid for any integral solution) into the existing LP formulation and obtain a new LP formulation:

$$\sum\nolimits_{t=t_1+1}^{t_2} x_t \geq \left\lceil \frac{\sum_{j \in J} p_j(I)}{g} \right\rceil, I = (t_1, t_2], t_2 > t_1, \forall t_1 \in \mathcal{T}, \forall t_2 \in \mathcal{T}$$

We conjecture that the new LP formulation has integrality gap $\frac{5}{3}$, while the existing LP is known to have an integrality gap that converges to 2 for large g and $|J|$.

In the following, we give an example achieving integrality gap $\frac{5}{3+2/g}$. Consider the set of jobs where we have g jobs with release time 0, deadline 2 and processing time 1, another g jobs with release time 5, deadline 7 and processing time 1 and finally one job with release time 1, deadline 6 and processing time 3. One relaxed linear programming solution is $x_2 = x_4 = x_6 = 1$ and $x_1 = x_7 = 1/g$. One can verify that the integer linear programming solution has to open 5 time slots, therefore the lower bound on the integrality gap is $\frac{5}{3+2/g}$.

References

[1] Hrishikesh Amur, James Cipar, Varun Gupta, Gregory R Ganger, Michael A Kozuch, and Karsten Schwan. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 217–228. ACM, 2010.

[2] Gruia Calinescu, Chenchen Fu, Minming Li, Kai Wang, and Chun Jason Xue. Energy optimal task scheduling with normally-off local memory and sleep-aware shared memory with access conflict. *IEEE Transactions on Computers*, (1):1–1, 2018.

[3] Jessica Chang, Harold N. Gabow, and Samir Khuller. A model for minimizing active processor time. *Algorithmica*, 70(3):368–405, 2014.

[4] Jessica Chang, Samir Khuller, and Koyel Mukherjee. LP rounding and combinatorial algorithms for minimizing active and busy time. *J. Scheduling*, 20(6):657–680, 2017.

[5] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[6] Saurabh Kumar and Samir Khuller. Brief announcement: A greedy 2 approximation for the active time problem. 2018.