Operations Research Letters 38 (2010) 390–395

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Minimizing the sum of weighted completion times in a concurrent open shop

Monaldo Mastrolilli [a], Maurice Queyranne [b], Andreas S. Schulz [c], Ola Svensson [d], Nelson A. Uhan [e,*]

[a] IDSIA, Manno, Switzerland

[b] Sauder School of Business, University of British Columbia, Vancouver, BC, Canada

[c] Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, USA

[d] KTH Computer Science and Communication, Stockholm, Sweden

[e] School of Industrial Engineering, Purdue University, West Lafayette, IN, USA

ARTICLE INFO

Article history:
Received 15 November 2009
Accepted 24 April 2010
Available online 21 May 2010

Keywords:
Scheduling
Integrality gap
Approximation algorithm
Hardness of approximation

ABSTRACT

We study minimizing the sum of weighted completion times in a concurrent open shop. We give a primal–dual 2-approximation algorithm for this problem. We also show that several natural linear programming relaxations for this problem have an integrality gap of 2. Finally, we show that this problem is inapproximable within a factor strictly less than 6/5 if P $\neq$ NP, or strictly less than 4/3 if the Unique Games Conjecture also holds.

1. Introduction

Consider the following scheduling setting, sometimes known as the *concurrent open shop model*, or the *order scheduling model*. We have a set of machines $M = \{1, \ldots, m\}$, with each machine capable of processing one operation type. We have a set of jobs $N = \{1, \ldots, n\}$, with each job requiring specific quantities of processing for each of its m operation types. Each job $j \in N$ has a weight $w_j \in \mathbb{R}_{\geq 0}$, and the processing time of job j's operation on machine i is $p_{ij} \in \mathbb{R}_{\geq 0}$. Operations are independent of each other: in particular, operations from the same job can be processed in parallel. A job is completed when all its operations are completed. In this paper, we focus on minimizing the sum of weighted completion times in a concurrent open shop. Following the notation of Leung et al. [12], we denote this problem by PD $\parallel \sum w_j C_j$. Note that when $m = 1$, or when each job consists of operations all with equal processing time, PD $\parallel \sum w_j C_j$ reduces to the classic problem of minimizing the sum of weighted completion times on a single machine [18].

The concurrent open shop model can be considered as a variant of the classical open shop model in which operations belonging to the same job can be processed concurrently. This model has a variety of applications in manufacturing, including automobile and airplane maintenance and repair [23], and orders with multiple components in manufacturing environments [19]. This model also has applications in distributed computing [6].

The problem PD $\parallel \sum w_j C_j$ was first studied by Ahmadi and Bagchi [1]. A number of authors have since shown that various special cases of this problem are NP-hard [1,19,4,12]; it turns out this problem is strongly NP-hard, even when all jobs have unit weight, and the number m of machines is 2 [16]. Garg et al. [6] showed that PD $\parallel \sum w_j C_j$ is APX-hard, even when all jobs have unit weight and either zero or unit processing time.

Quite a bit of attention has been devoted to designing heuristics for this problem. For example, Sung and Yoon [19], Wang and Cheng [21], Ahmadi et al. [2] and Leung et al. [12] proposed various heuristics for this problem; all of the heuristics they studied were shown to either have a performance guarantee of m, or have an unbounded performance guarantee. Inspired by techniques in [7], Wang and Cheng [21] used an interval-indexed linear programming (LP) relaxation of this problem to obtain a 16/3-approximation algorithm. Finally, several groups of authors have independently observed that a linear programming relaxation of this problem in completion time variables with the parallel inequalities of Wolsey [22] and Queyranne [14], combined with a result of Schulz [17], yields a 2-approximation algorithm [4,6,13].

We begin in Section 2 by presenting some interesting properties of various natural linear programming relaxations for PD $\parallel \sum w_j C_j$; in particular, we show that all these LP relaxations have an integrality gap of 2. Then in Section 3, we present a combinatorial approximation algorithm that has a performance guarantee of 2. This algorithm can be seen either as a primal–dual algorithm, or as a greedy algorithm that starts at the end of the schedule.

* Corresponding author.
E-mail addresses: monaldo@idsia.ch (M. Mastrolilli), maurice.queyranne@ubc.ca (M. Queyranne), schulz@mit.edu (A.S. Schulz), osven@kth.se (O. Svensson), nuhan@purdue.edu (N.A. Uhan).

0167-6377/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2010.04.011

Although the approximation algorithm independently proposed by Chen and Hall [4], Garg et al. [6] and Leung et al. [13] achieves the same performance guarantee, their algorithm requires solving a linear program with an exponential number of constraints. Our algorithm, on the other hand, requires $O(n(m+n))$ elementary operations. Finally, in Section 4, we show that PD $\parallel \sum w_jC_j$ is inapproximable within a factor of $6/5-\varepsilon$ for any $\varepsilon > 0$, unless P $=$ NP; under the increasingly prevalent assumption that the Unique Games Conjecture holds, we can show that this scheduling problem is in fact inapproximable within a factor of $4/3-\varepsilon$ for any $\varepsilon > 0$, unless P $=$ NP. The construction used to show these hardness results, as well as the integrality gap result in Section 2, is an extension of the construction used by Garg et al. [6].

Recently, Bansal and Khot [3] and Kumar et al. [11] independently showed that if the Unique Games Conjecture holds, PD $\parallel \sum w_jC_j$ is in fact inapproximable within a factor of $2-\varepsilon$ for any $\varepsilon > 0$, unless P $=$ NP. In [3], this result is obtained by combining our inapproximability construction with a new and stronger inapproximability result for the minimum vertex cover problem on r-uniform hypergraphs. In [11], this result is obtained by combining our integrality gap construction with an integrality-gap-based inapproximability result for strict constraint satisfaction problems.

2. Linear programming relaxations

The existing mixed-integer programming formulations and linear programming relaxations for various machine scheduling problems (e.g. [15]) provide natural starting points for modeling the problem of minimizing the sum of weighted completion times in a concurrent open shop. We present two types of mathematical programming formulations for PD $\parallel \sum w_jC_j$, one based on completion time variables, and the other based on linear ordering variables.

2.1. Completion time variables

Chen and Hall [4] proposed the following linear programming relaxation of PD $\parallel \sum w_jC_j$:

$$\text{CT1}: \min \sum_{j\in N} w_jC_j \tag{1a}$$

$$\text{s.t.} \sum_{j\in S} p_{ij}C_{ij} \geq f_i(S) \quad \text{for all } i\in M,\ S\subseteq N, \tag{1b}$$

$$C_j \geq C_{ij} \quad \text{for all } i\in M,\ j\in N, \tag{1c}$$

where C_{ij} represents the completion time of job j's operation on machine i, C_j represents the completion time of job j, and

$$f_i(S) = \frac{1}{2}\sum_{j\in S} p_{ij}^2 + \frac{1}{2}\left(\sum_{j\in S} p_{ij}\right)^2 \quad \text{for all } i\in M,\ S\subseteq N.$$

The constraints (1b) are the so-called *parallel inequalities* [22,14] for each of the m machines. These inequalities are known to be valid for the completion time vectors of jobs on a single machine; in fact, they are sufficient to describe the convex hull of completion time vectors for jobs on a single machine. It immediately follows that CT1 is a valid relaxation for PD $\parallel \sum w_jC_j$.

By substituting the constraints (1c) into the constraints (1b), we obtain a further relaxation of PD $\parallel \sum w_jC_j$ in fewer completion time variables:

$$\text{CT2}: \min \sum_{j\in N} w_jC_j \tag{2a}$$

$$\text{s.t.} \sum_{j\in S} p_{ij}C_j \geq f_i(S) \quad \text{for all } i\in M,\ S\subseteq N. \tag{2b}$$

The relaxation CT2 will serve as the basis of our analysis of the algorithm presented in Section 3.

2.2. Linear ordering variables

In addition to explicitly modeling the completion times of each job on each machine, we can model the order in which the jobs are processed on each machine. For all $i \in M$ and $j,k \in N$ such that $j \neq k$, we define the decision variables δ_{jk}^i, where $\delta_{jk}^i = 1$ if the operation of job j precedes the operation of job k on machine i, and $\delta_{jk}^i = 0$ otherwise. These variables are known as *linear ordering variables*. Consider the following mixed-integer programming formulation for PD $\parallel \sum w_jC_j$:

$$\min \sum_{j\in N} w_jC_j \tag{3a}$$

$$\text{s.t.}\quad \delta_{jk}^i + \delta_{kj}^i = 1 \quad \text{for all } i\in M,\ j,k\in N, \tag{3b}$$

$$\delta_{jk}^i + \delta_{kl}^i + \delta_{lj}^i \leq 2 \quad \text{for all } i\in M,\ j,k,l\in N, \tag{3c}$$

$$\delta_{jk}^i \in \{0,1\} \quad \text{for all } i\in M,\ j,k\in N, \tag{3d}$$

$$C_{ij} \geq \sum_{\substack{k\in N:\\ k\neq j}} p_{ik}\delta_{kj}^i + p_{ij} \quad \text{for all } i\in M,\ j\in N, \tag{3e}$$

$$C_j \geq C_{ij} \quad \text{for all } i\in M,\ j\in N. \tag{3f}$$

For a given machine i, the set of vectors defined by the constraints (3b)–(3d) is known to define all permutations of N as described by these δ-variables (the convex hull of this set is known as the *linear ordering polytope*). It follows that the mixed-integer program (3a)–(3f) is a correct formulation of PD $\parallel \sum w_jC_j$.

A *permutation schedule* processes all jobs nonpreemptively, without unnecessary idle time and in the same order on each machine. Using concepts of Pareto minimality, Wagneur and Sriskandarajah [20] showed that one may restrict attention to permutation schedules without loss of optimality in problem PD $\parallel f(C)$ when the objective function $f(C)$ is nondecreasing in the job completion times $C = (C_j)_{j\in N}$ (i.e., when f is a *regular performance measure*). This result, which also implies that there is no advantage to preemption in problem PD$|$pmtn$|f(C)$, is in fact an easy consequence of the optimality of Jackson's [8] Earliest Due Date (EDD) rule for minimizing maximum lateness on a single machine, as we now show. (In a scheduling environment with a set of jobs N and due dates d_j for all $j \in N$, the lateness of a job j is defined as the difference between its completion time and its due date: $C_j - d_j$. Jackson's [8] EDD rule—schedule jobs in order of nondecreasing due dates—minimizes the maximum lateness on a single machine.)

Lemma 2.1 (*[20]*). *Given an instance of PD $\parallel f(C)$, let $C = (C_j)_{j\in N}$ be the completion times of a feasible (possibly preemptive) schedule. Then, there exists a permutation schedule with completion times $C^* = (C_j^*)_{j\in N}$ such that $C_j^* \leq C_j$ for all $j \in N$.*

Alternative Proof. Let $\sigma : \{1,\dots,n\} \to N$ be a permutation of N such that $C_{\sigma(1)} \leq \cdots \leq C_{\sigma(n)}$, and let $(C_{ij}^*)_{j\in N}$ be the completion times of the jobs on machine $i \in M$ scheduled according to the permutation σ. In addition, for each machine $i \in M$, define the due dates $d_j^i = C_j$ for all $j \in N$. In the schedule corresponding to the completion time vector C, for each machine $i \in M$, the maximum lateness over all jobs is nonpositive, by construction. Since Jackson's EDD rule is optimal, scheduling the jobs according to σ produces a permutation schedule in which the maximum lateness over all jobs for each machine $i \in M$ is nonpositive; that is, $C_{ij}^* \leq d_j^i = C_j$ for all $i \in M$ and $j \in N$. □

Lemma 2.1 implies that we only need to find one common ordering of the jobs to determine an optimal solution. Accordingly, for all $j,k \in N$ such that $j \neq k$, we define the decision variables δ_{jk}, where $\delta_{jk} = 1$ if job j precedes job k, and $\delta_{jk} = 0$ otherwise.

Consider the following mixed-integer programming formulation for PD $\parallel \sum w_j C_j$, now with only one set of linear ordering constraints:

$$\min \sum_{j \in N} w_j C_j \tag{4a}$$

$$\text{s.t.} \quad \delta_{jk} + \delta_{kj} = 1 \quad \text{for all } j, k \in N, \tag{4b}$$

$$\delta_{jk} + \delta_{kl} + \delta_{lj} \le 2 \quad \text{for all } j, k, l \in N, \tag{4c}$$

$$\delta_{jk} \in \{0, 1\} \quad \text{for all } j, k \in N, \tag{4d}$$

$$C_j \ge \sum_{\substack{k \in N: \\ k \ne j}} p_{ik}\delta_{kj} + p_{ij} \quad \text{for all } i \in M, j \in N. \tag{4e}$$

By Lemma 2.1, it follows that the above mixed-integer programming formulation is also valid for PD $\parallel \sum w_j C_j$.

Let LO3 be the linear programming relaxation of the mixed-integer program (3a)–(3f) obtained by replacing the binary constraints (3d) with nonnegativity constraints. Similarly, let LO4 be the linear programming relaxation of (4a)–(4e) obtained by replacing the binary constraints (4d) with nonnegativity constraints.

2.3. Relative strength of LP relaxations

For any linear programming relaxation X of PD $\parallel \sum w_j C_j$, let OPT_X be the optimal value of X. We show the following statement on the relative strength of the four linear programming relaxations presented above.

Lemma 2.2. *For any given instance of PD* $\parallel \sum w_j C_j$, *we have that* $\text{OPT}_{\text{CT1}} = \text{OPT}_{\text{CT2}} = \text{OPT}_{\text{LO3}} \le \text{OPT}_{\text{LO4}}$.

Proof. To simplify notation in this proof, when referring to a vector of completion time or linear ordering variables, we omit the associated set of indices; these sets should be clear from the context.

Fix an instance of PD $\parallel \sum w_j C_j$. Let (C_{ij}^1, C_j^1), (C_j^2), $(\delta_{jk}^{i3}, C_{ij}^3, C_j^3)$, and (δ_{jk}^4, C_j^4) be optimal solutions to CT1, CT2, LO3, and LO4, respectively.

Clearly, (C_j^1) is feasible in CT2, and so $\text{OPT}_{\text{CT2}} \le \text{OPT}_{\text{CT1}}$. Now define $C_{ij}^2 = C_j^2$ for all $i \in M$ and $j \in N$. Clearly, (C_{ij}^2, C_j^2) is feasible in CT1, and so $\text{OPT}_{\text{CT1}} \le \text{OPT}_{\text{CT2}}$. Therefore, $\text{OPT}_{\text{CT1}} = \text{OPT}_{\text{CT2}}$.

Using techniques from [17], it is straightforward to show that (C_j^3) is feasible in CT2, and so $\text{OPT}_{\text{CT2}} \le \text{OPT}_{\text{LO3}}$. To show the reverse inequality, for each machine $i \in M$ we define $P^i = \{(C_j) : \sum_{j \in S} p_{ij} C_j \ge f_i(S) \text{ for all } S \subseteq N\}$ and $B^i = \{(C_j) : \sum_{j \in N} p_{ij} C_j = f_i(N), \sum_{j \in S} p_{ij} C_j \ge f_i(S) \text{ for all } S \subset N\}$. As mentioned earlier, for each $i \in M$, the polyhedron P^i is the convex hull of completion time vectors for jobs on machine i. In addition, for each $i \in M$, the polytope B^i is the convex hull of completion time vectors corresponding to permutation schedules on machine i [22,14]. It follows that P^i is the dominant of B^i (see [15]). Therefore, for every machine $i \in M$, there exists a vector $(C_{ij}^2) \in B^i$ such that $C_{ij}^2 \le C_j^2$ for all $j \in N$. Also, for every machine $i \in M$, since $(C_{ij}^2) \in B^i$ represents a convex combination of permutation schedules on machine i, and each of these permutation schedules can be represented by a vector of linear ordering variables and completion time variables that satisfies (3b), (3c), (3e), and the nonnegativity constraints restricted to i, it follows by convexity that there exists a vector (δ_{jk}^{i2}) of linear ordering variables such that $(\delta_{jk}^{i2}, C_{ij}^2)$ satisfies the constraints (3b), (3c), (3e), and the nonnegativity constraints restricted to i. Therefore, $(\delta_{jk}^{i2}, C_{ij}^2, C_j^2)$ is a feasible solution to LO3, and so $\text{OPT}_{\text{LO3}} \le \text{OPT}_{\text{CT2}}$. So $\text{OPT}_{\text{CT2}} = \text{OPT}_{\text{LO3}}$.

Finally, define $\delta_{jk}^{i4} = \delta_{jk}^4$ for all $i \in M$ and $j, k \in N$ such that $j \ne k$. Also, define $C_{ij}^4 = C_j^4$ for all $i \in M$ and $j \in N$. Clearly, $(\delta_{jk}^{i4}, C_{ij}^4, C_j^4)$ is a feasible solution to LO3, and so $\text{OPT}_{\text{LO3}} \le \text{OPT}_{\text{LO4}}$. □

The inequality in Lemma 2.2 can be strict. Consider the instance with $m = 2, n = 2, w_1 = w_2 = 1, p_{11} = 2, p_{12} = 1, p_{21} = 1$, and $p_{22} = 2$. The optimal objective value of LO3 is 14/3, and the optimal objective value of LO4 is 5.

2.4. Integrality gaps for LP relaxations

Chen and Hall [4], Leung et al. [13], and Garg et al. [6] independently observed that scheduling jobs in order of nondecreasing optimal C_j to the linear program CT1 is a 2-approximation algorithm for the problem PD $\parallel \sum w_j C_j$. They showed this using a proof technique introduced in [17], which also implies that the integrality gap of CT1 is at most 2. (In this subsection, we slightly abuse terminology: for *any* relaxation X of the problem PD $\parallel \sum w_j C_j$, we say that the *integrality gap* of X is $\sup\{\text{OPT}(I)/\text{OPT}_\text{X}(I) : I$ is an instance of PD $\parallel \sum w_j C_j\}$, where $\text{OPT}(I)$ denotes the optimal value of PD $\parallel \sum w_j C_j$ under instance I, and $\text{OPT}_\text{X}(I)$ denotes the optimal value of the relaxation X under instance I.) Similarly, one can show that scheduling jobs in order of nondecreasing optimal C_j to the linear programs CT2, LO3, and LO4 are also 2-approximation algorithms, and that for all these linear programs, the integrality gap is at most 2. We show that the analyses of these LP relaxations are tight: the integrality gap is 2 for CT1, CT2, LO3, and LO4.

Theorem 2.3. *The integrality gap is 2 for the following linear programming relaxations:* CT1, CT2, LO3, *and* LO4.

Proof. As mentioned above, it follows from [4,13,6] that the integrality gap of CT1 is at most 2. We next show that the integrality gap of LO4 is at least 2.

Let (N, E) be a complete r-uniform hypergraph. (An *r-uniform hypergraph* is a pair (N, E) where N is a finite set, and E is a family of r-element subsets of N. The elements of N are called nodes, and the elements of E are called hyperedges. An r-uniform hypergraph (N, E) is *complete* if E is the family of all $\binom{n}{r}$ r-element subsets of N.) We construct an instance of PD $\parallel \sum w_j C_j$ as follows. Each node $j \in N$ corresponds to a job. Each hyperedge $i \in E$ corresponds to a machine, so $m = \binom{n}{r}$. For each $i \in M$ and $j \in N$, the processing time p_{ij} is 1 if j is in hyperedge i, and 0 otherwise. All jobs have unit weight. Note that in any feasible schedule without unnecessary idle time, every machine processes jobs only during the first r time units.

We first show that in any feasible schedule without idle time, there are at least $n-r+1$ jobs that complete at time r. We consider two cases.

1. *There are at most $r - 2$ jobs that complete at or before time $r - 1$.* Therefore, at least $n - r + 2$ jobs complete at time r, which directly implies the claim.
2. *There are at least $r - 1$ jobs that complete at or before time $r - 1$.* Let A be a set of $r - 1$ jobs that complete at or before time $r - 1$. Since (N, E) is a complete r-uniform hypergraph, for any job $j \in N \setminus A$, we have that $A \cup \{j\}$ is a hyperedge in (N, E). Since there are $r - 1$ jobs in A, this implies that every job $j \in N \setminus A$ cannot complete until at least time r on the machine corresponding to the hyperedge $A \cup \{j\}$. Since $|N \setminus A| = n - r + 1$, there are at least $n - r + 1$ jobs that complete at time r.

Let OPT denote the optimal value of this instance. It follows from the above observation that $\text{OPT} \ge r(n - r + 1)$. Now consider the following solution to LO4:

$$\delta_{jk} = 1/2 \quad \text{for all } j, k \in N : j \ne k,$$

$$C_j = \max_{i \in M} \left\{ \sum_{k \in N : k \ne j} p_{ik}\delta_{kj} + p_{ij} \right\} \quad \text{for all } j \in N.$$

It is straightforward to show that this solution is feasible. Also, note that $C_j = (r-1)/2 + 1$, and so $\mathrm{OPT}_{\mathrm{LO4}} \leq n(r+1)/2$. Letting $r = n^{3/4}$, we have that

$$\frac{\mathrm{OPT}}{\mathrm{OPT}_{\mathrm{LO4}}} \geq \frac{2n^{3/4}(n - n^{3/4} + 1)}{n(n^{3/4} + 1)},$$

which approaches 2 as n goes to infinity.

The result now follows from Lemma 2.2. □

3. A combinatorial 2-approximation algorithm

In this section, we present a simple combinatorial 2-approximation algorithm for $PD \parallel \sum w_j C_j$. Our algorithm can be seen as a primal–dual algorithm, or as a greedy algorithm starting from the end of the schedule. Unlike the LP-based approximation algorithms mentioned in Section 2.4, our algorithm does not require the solution of a linear program; in fact, our algorithm requires $O(n(m+n))$ elementary operations. Although it does not require solving the linear program CT2, we use this linear program and its dual in the analysis of our algorithm. Note that the dual of CT2 is

$$\max \sum_{i \in M} \sum_{S \subseteq N} f_i(S) y_{i,S} \tag{5a}$$

$$\text{s.t.} \quad \sum_{i \in M} p_{ij} \sum_{S \subseteq N : j \in S} y_{i,S} = w_j \quad \text{for all } j \in N, \tag{5b}$$

$$y_{i,S} \geq 0 \quad \text{for all } i \in M,\ S \subseteq N. \tag{5c}$$

Our algorithm works as follows. We find a permutation schedule by starting at the end of the schedule. We determine the *last* job to be scheduled by observing that its completion time is achieved on the machine with the maximum load when all jobs are scheduled; we choose the job with the minimum weight-to-processing time ratio on that machine. We adjust the weights of the other jobs to ensure dual feasibility, and proceed in determining the next-to-last job in a similar manner. A full description of the algorithm is below. We assume that all jobs require positive processing time on at least one machine; in other words,

$$\text{for all } j \in N, \quad p_{ij} > 0 \ \text{ for at least one } i \in M. \tag{6}$$

Note that this assumption is made without loss of generality: we can set aside the jobs that require zero processing time on all machines in a preprocessing step, and then schedule these jobs at the beginning of the permutation schedule for the remaining jobs constructed by the algorithm below.

Algorithm 3.1. Input: instance of $PD \parallel \sum w_j C_j$: number of jobs n; number of machines m; processing times $p_{ij} \in \mathbb{R}_{\geq 0}$ for all $i \in M$ and $j \in N$; weights $w_j \in \mathbb{R}_{\geq 0}$ for all $j \in N$.
Output: permutation schedule of jobs $\sigma : \{1, \ldots, n\} \to N$.

1. Initialize:
   a. $J \leftarrow N$ (unscheduled jobs)
   b. $L_i \leftarrow \sum_{j \in N} p_{ij}$ for all $i \in M$ (load of machine i)
   c. $\bar{w}_j \leftarrow w_j$ for all $j \in N$ (adjusted weights)
2. For $k = n, n-1, \ldots, 2, 1$:
   a. $\mu \leftarrow \arg\max_{i \in M} L_i$ (determine machine on which job $\sigma(k)$ completes)
   b. $\sigma(k) \leftarrow \arg\min_{j \in J}\{\bar{w}_j / p_{\mu,j}\}$ (determine job $\sigma(k)$)
   c. $\theta \leftarrow \bar{w}_{\sigma(k)} / p_{\mu,\sigma(k)}$
      $\bar{w}_j \leftarrow \bar{w}_j - \theta \cdot p_{\mu,j}$ for all $j \in J$ (adjust weights)
   d. $L_i \leftarrow L_i - p_{i,\sigma(k)}$ for all $i \in M$ (update machine loads)
   e. $J \leftarrow J \setminus \{\sigma(k)\}$ (update unscheduled jobs)
   When computing μ and $\sigma(k)$, break ties arbitrarily.

To show the performance guarantee of Algorithm 3.1, we need the following useful property of the set function f_i, first proved by Schulz [17] in the context of completion-time-variable LP relaxations for other scheduling problems.

Lemma 3.2 ([17]). *For any $i \in M$, and $S \subseteq N$, we have that $\left(\sum_{j \in S} p_{ij}\right)^2 \leq \left(2 - \frac{2}{n+1}\right) f_i(S)$.*

Now we show the main result of this section.

Theorem 3.3. *Algorithm 3.1 is a $\left(2 - \frac{2}{n+1}\right)$-approximation algorithm for $PD \parallel \sum w_j C_j$.*

Proof. For ease of notation, let $\mu(k)$ denote the machine μ chosen in Step 2a at iteration k, let $\theta(k)$ denote the value θ computed in Step 2c at iteration k, and let $\bar{w}_j(k)$ denote the adjusted weights $\bar{w}_j$ computed in Step 2c at iteration k for all $j \in N$. In addition, let $J(k)$ denote the set of unscheduled jobs J *at the beginning of* iteration k; that is, $J(k) = \{\sigma(1), \ldots, \sigma(k)\}$.

Define the following dual solution: for all $i \in M$ and $S \subseteq N$,

$$y_{i,S} = \begin{cases} \theta(k) & \text{if } i = \mu(k),\ S = J(k) \text{ for some } k = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

We show that $y = (y_{i,S})_{i \in M, S \subseteq N}$ is a feasible solution to the dual linear program (5a)–(5c). Since $w_j \geq 0$ for all $j \in N$, Steps 1c, 2a and 2b, along with the assumption (6) imply that $\theta(n)$ is well-defined and that in fact, $\theta(n) \geq 0$. In addition, at any iteration $k = 2, \ldots, n$, the choice of $\sigma(k)$ in Step 2b implies that $\bar{w}_j(k) \geq 0$ for all $j \in J(k)$. It follows by Steps 2a and 2b and the assumption (6) that for $k = 1, \ldots, n-1$, $\theta(k)$ is well-defined and in fact, $\theta(k) \geq 0$. Therefore, y is well-defined and satisfies (5c). Next, observe that at every iteration $k = 1, \ldots, n$,

$$\bar{w}_j(k) = w_j - \sum_{l=k}^{n} p_{\mu(l),j}\theta(l) \quad \text{for all } j \in J(k).$$

It follows that y satisfies the constraints (5b), since for any job $\sigma(k)$ with $k = 1, \ldots, n$, we have

$$\sum_{i \in M} p_{i,\sigma(k)} \sum_{S \subseteq N : \sigma(k) \in S} y_{i,S} = \sum_{l=k}^{n} p_{\mu(l),\sigma(k)} y_{\mu(l),J(l)}$$
$$= \sum_{l=k}^{n} p_{\mu(l),\sigma(k)}\theta(l) = w_{\sigma(k)} - \bar{w}_{\sigma(k)}(k) \overset{\text{(i)}}{=} w_{\sigma(k)},$$

where (i) holds since Steps 2b and 2c imply that $\bar{w}_{\sigma(k)}(k) = 0$ for all $k = 1, \ldots, n$.

We now show that the schedule constructed by the algorithm is a $(2 - 2/(n+1))$-approximation. Note that the completion times $(C_j)_{j \in N}$ under the permutation schedule produced by the algorithm satisfy $C_{\sigma(1)} \leq C_{\sigma(2)} \leq \cdots \leq C_{\sigma(n)}$, and by Steps 2a and 2b, $C_{\sigma(k)} = \sum_{j \in J(k)} p_{\mu(k),j} = \sum_{j=1}^{k} p_{\mu(k),\sigma(j)}$ for all $k = 1, \ldots, n$. Let $(C_j^{\mathrm{LP}})_{j \in N}$ be an optimal solution to CT2, and let $(C_j^*)_{j \in N}$ be an optimal completion time vector. The objective value of the permutation schedule produced by the algorithm is

$$\sum_{j \in N} w_j C_j = \sum_{j \in N} \left(\sum_{i \in M} p_{ij} \sum_{S \subseteq N : j \in S} y_{i,S} \right) C_j$$
$$= \sum_{i \in M} \sum_{S \subseteq N} y_{i,S} \sum_{j \in S} p_{ij} C_j = \sum_{k=1}^{n} y_{\mu(k),J(k)} \sum_{j \in J(k)} p_{\mu(k),j} C_j$$
$$= \sum_{k=1}^{n} y_{\mu(k),J(k)} \sum_{j=1}^{k} p_{\mu(k),\sigma(j)} C_{\sigma(j)}$$
$$\overset{\text{(ii)}}{\leq} \sum_{k=1}^{n} y_{\mu(k),J(k)} \left(C_{\sigma(k)} \sum_{j=1}^{k} p_{\mu(k),\sigma(j)} \right)$$
$$\overset{\text{(iii)}}{=} \sum_{k=1}^{n} y_{\mu(k),J(k)} \left(\sum_{j=1}^{k} p_{\mu(k),\sigma(j)} \right)^2$$

$$\overset{\text{(iv)}}{\leq} \left(2 - \frac{2}{n+1}\right) \sum_{k=1}^{n} y_{\mu(k),J(k)} f_{\mu(k)}(J(k))$$
$$\overset{\text{(v)}}{\leq} \left(2 - \frac{2}{n+1}\right) \sum_{j\in N} w_j C_j^{\text{LP}} \leq \left(2 - \frac{2}{n+1}\right) \sum_{j\in N} w_j C_j^*,$$

where (ii) holds since $C_{\sigma(k)} \geq C_{\sigma(j)}$ for all $j = 1, \ldots, k$, (iii) holds since $C_{\sigma(k)} = \sum_{j=1}^{k} p_{\mu(k),\sigma(j)}$, (iv) holds by Lemma 3.2, and (v) holds since y is feasible in (5a)–(5c).

Finally, we analyze the running time of the algorithm. The algorithm runs through an initialization and n iterations. Each step in the initialization of the algorithm takes at most nm elementary operations. Each step in each iteration of the algorithm takes either at most m elementary operations or at most n elementary operations. Therefore, the algorithm requires $O(n(m + n))$ elementary operations. □

The above analysis of the performance guarantee of Algorithm 3.1 is tight. Consider the following instance with $m = n$, and

$$p_{ij} = \begin{cases} \frac{n}{i} & \text{if } j \leq i, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i = 1, \ldots, n \text{ and } j = 1, \ldots, n.$$

All jobs have unit weight. It is straightforward to show that the total completion time of the permutation schedule $(n, n-1, \ldots, 2, 1)$ is $n(n+1)/2$. On the other hand, suppose that Algorithm 3.1, when computing μ and $\sigma(k)$, breaks ties by always choosing the machine or job with the highest index. It turns out that when using this tiebreaking rule, Algorithm 3.1 outputs the permutation schedule $(1, \ldots, n)$ which has a total completion time of n^2. Therefore, using the objective value of the permutation schedule $(n, n-1, \ldots, 2, 1)$ as an upper bound on the optimal value, the performance guarantee of Algorithm 3.1 cannot be better than $2 - 2/(n+1)$.

4. Hardness of approximation

In this section, we give lower bounds on the approximability of the problem PD $\|$ $\sum C_j$ (all jobs have unit weight), both under the standard assumption P $\neq$ NP, as well as under the increasingly prevalent additional assumption that the Unique Games Conjecture [9] holds. In order to show these inapproximability results, we make use of the following theorems on the inapproximability of the maximum cardinality independent set problem on r-uniform hypergraphs. (An *independent set* of an r-uniform hypergraph (N, E) is a subset I of N such that $i \setminus I \neq \emptyset$ for every hyperedge $i \in E$.)

Theorem 4.1 ([5]). *For any $\gamma \in (0, 1)$ and $\delta \in (0, 1/2)$, the following problem is NP-hard: given an r-uniform hypergraph $G = (N, E)$ with $r \geq 3$, decide whether*

(i) *G contains an independent set of size greater than or equal to $(1 - \frac{1}{r-1} - \delta)|N|$, or*
(ii) *all independent sets of G have size strictly less than $\gamma|N|$.*

Theorem 4.2 ([10]). *Assuming the Unique Games Conjecture is true, for any $\gamma \in (0, 1)$ and $\delta \in (0, 1/2)$, the following problem is NP-hard: given an r-uniform hypergraph $G = (N, E)$ with $r \geq 2$, decide whether*

(i) *G contains an independent set of size greater than or equal to $(1 - \frac{1}{r} - \delta)|N|$, or*
(ii) *all independent sets of G have size strictly less than $\gamma|N|$.*

Using the above results, we can show the following.

Theorem 4.3. (a) *PD $\|$ $\sum C_j$ is hard to approximate within a factor of $6/5 - \varepsilon$ for any $\varepsilon > 0$, unless P = NP.*
(b) *Assuming the Unique Games Conjecture is true, PD $\|$ $\sum C_j$ is hard to approximate within a factor of $4/3 - \varepsilon$ for any $\varepsilon > 0$, unless P = NP.*

Proof. First, we show (a). Let $G = (N, E)$ be an r-uniform hypergraph. We construct an instance of PD $\|$ $\sum C_j$ as we did in the proof of Theorem 2.3: each node $j \in N$ corresponds to a job, and each hyperedge $i \in E$ corresponds to a machine. For each $i \in M$ and $j \in N$, the processing time p_{ij} is 1 if j is in hyperedge i, and 0 otherwise. As before, in any feasible schedule without unnecessary idle time, every machine processes jobs only during the first r time units. The key observation is as follows: $I \subseteq N$ is an independent set in G if and only if each job in I can be completed by time $r - 1$.

Let OPT denote the optimal value of this instance of PD $\|$ $\sum C_j$. Suppose that condition (i) from Theorem 4.1 holds. Let I be such an independent set. By the observation in the previous paragraph, we know that all jobs in I can be completed by time $r - 1$, and that all the remaining jobs $N \setminus I$ can be completed by time r. Therefore, in this case,

$$\text{OPT} \leq (r-1) \cdot \left(1 - \frac{1}{r-1} - \delta\right)|N| + r \cdot \left(\frac{1}{r-1} + \delta\right)|N|$$
$$= \left((r-1) + \frac{1}{r-1} + \delta\right)|N|.$$

Now suppose that condition (ii) from Theorem 4.1 holds. This implies that in any schedule, at least $(1-\gamma)|N|$ jobs are forced to be completed at time r. Therefore, in this case,

$$\text{OPT} \geq 1 \cdot \gamma|N| + r \cdot (1-\gamma)|N| = (r - (r-1)\gamma)|N|.$$

It follows that a $\left(\frac{r(r-1)}{(r-1)^2+1} - \varepsilon\right)$-approximation algorithm for PD $\|$ $\sum C_j$ can solve the decision problem in Theorem 4.1. When $r = 3$, we have that $\frac{r(r-1)}{(r-1)^2+1} = 6/5$.

Using the above ideas in conjunction with Theorem 4.2, and by setting $r = 2$, one can show (b). □

Acknowledgements

The authors thank Uriel Feige and Prasad Raghavendra for an inspiring discussion, and an anonymous referee for helpful suggestions. The second author acknowledges support from a Discovery Grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada. The work of the third author is supported by the National Science Foundation (NSF) under Grant No. 0700044.

References

[1] R.H. Ahmadi, U. Bagchi, Scheduling of multi-job customer orders in multi-machine environments, ORSA/TIMS, Philadelphia, 1990.
[2] R.H. Ahmadi, U. Bagchi, T. Roemer, Coordinated scheduling of customer orders for quick response, Naval Research Logistics 52 (2005) 493–512.
[3] N. Bansal, S. Khot, Inapproximability of hypergraph vertex cover and applications to scheduling problems, Working paper, 2009.
[4] Z.-L. Chen, N.G. Hall, Supply chain scheduling: assembly systems, working paper, Department of Systems Engineering, University of Pennsylvania, 2001.
[5] I. Dinur, V. Guruswami, S. Khot, O. Regev, A new multilayered PCP and the hardness of hypergraph vertex cover, SIAM Journal on Computing 34 (2005) 1129–1146.
[6] N. Garg, A. Kumar, V. Pandit, Order scheduling models: hardness and algorithms, in: V. Arvind, S. Prasad (Eds.), Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2007, in: Lecture Notes in Computer Science, vol. 4855, Springer, Berlin, 2007, pp. 96–107.
[7] L.A. Hall, A.S. Schulz, D.B. Shmoys, J. Wein, Scheduling to minimize average completion time: off-line and on-line approximation algorithms, Mathematics of Operations Research 22 (1997) 513–544.

[8] J.R. Jackson, Scheduling a production line to minimize maximum tardiness, Management Science Research Project Research Report 43, University of California, Los Angeles, 1955.
[9] S. Khot, On the power of unique 2-prover 1-round games, in: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, STOC 2002, 2002, pp. 767–775.
[10] S. Khot, O. Regev, Vertex cover might be hard to approximate within $2-\varepsilon$, Journal of Computer and System Sciences 74 (2008) 335–349.
[11] A. Kumar, R. Manokaran, M. Tulsiani, N.K. Vishnoi, On LP-based approximability for strict CSPs, Working paper, 2010.
[12] J.Y.-T. Leung, H. Li, M.L. Pinedo, Order scheduling in an environment with dedicated resources in parallel, Journal of Scheduling 8 (2005) 355–386.
[13] J.Y.-T. Leung, H. Li, M. Pinedo, Scheduling orders for multiple product types to minimize total weighted completion time, Discrete Applied Mathematics 155 (2007) 945–970.
[14] M. Queyranne, Structure of a simple scheduling polyhedron, Mathematical Programming 58 (1993) 263–285.
[15] M. Queyranne, A.S. Schulz, Polyhedral approaches to machine scheduling, Working paper, 2004.
[16] T.A. Roemer, A note on the complexity of the concurrent open shop problem, Journal of Scheduling 9 (2006) 389–396.
[17] A.S. Schulz, Scheduling to minimize total weighted completion time: performance guarantees of LP-based heuristics and lower bounds, in: W.H. Cunningham, S.T. McCormick, M. Queyranne (Eds.), Integer Programming and Combinatorial Optimization, IPCO 1996, in: Lecture Notes in Computer Science, vol. 1084, Springer, Berlin, 1996, pp. 301–315.
[18] W.E. Smith, Various optimizers for single-stage production, Naval Research Logistics Quarterly 3 (1956) 59–66.
[19] C.S. Sung, S.H. Yoon, Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines, International Journal of Production Economics 54 (1998) 247–255.
[20] E. Wagneur, C. Sriskandarajah, Open shops with jobs overlap, European Journal of Operational Research 71 (1993) 366–378.
[21] G. Wang, T.C.E. Cheng, Customer order scheduling to minimize total weighted completion time, in: Proceedings of the First Multidisciplinary Conference on Scheduling Theory and Applications, 2003, pp. 409–416.
[22] L.A. Wolsey, Mixed integer programming formulations for production planning and scheduling problems, in: Invited Talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, MA, 1985.
[23] J. Yang, Scheduling with batch objectives, Ph.D. Thesis, Industrial and Systems Engineering, The Ohio State University, 1998.