Mathematical Programming
https://doi.org/10.1007/s10107-020-01571-5

FULL LENGTH PAPER

Series A

Performance guarantees of local search for minsum scheduling problems

José R. Correa[1] · **Felipe T. Muñoz**[2]

Received: 5 August 2019 / Accepted: 21 September 2020

Abstract
We study the worst-case performance guarantee of locally optimal solutions for the problem of minimizing the total weighted and unweighted completion time on parallel machine environments. Our method makes use of a mapping that maps a schedule into an inner product space so that the norm of the mapping is closely related to the cost of the schedule. We apply the method to study the most basic local search heuristics for scheduling, namely jump and swap, and establish their worst-case performance in the case of unrelated, restricted related and restricted identical machines.

Keywords Local search · Performance guarantee · Parallel machines · Total weighted completion time

Mathematics Subject Classification 90B35 · 90C59 · 68M20 · 68W40

1 Introduction

Machine scheduling problems have been widely studied in recent decades because of their practical relevance in production and communication systems and their theoretical interest. As these problems are typically NP-hard, designing and analyzing approximation algorithms has been a central object of study to the theoretical computer science community. However, these algorithms are often impractical so the real-world methods of choice boil down to heuristic approaches including integer programming, constraint programming and local search. In this paper, we focus on the latter method and study its worst-case performance guarantees. Local search methods are indeed

✉ José R. Correa
correa@uchile.cl

Felipe T. Muñoz
fmunoz@ubiobio.cl

[1] Department of Industrial Engineering, Universidad de Chile, Santiago, Chile

[2] Department of Industrial Engineering, Universidad del Bío-Bío, Concepción, Chile

Published online: 30 September 2020

widely used in practice and exhibit good empirical behavior, but little is known about their worst-case performance. We refer the reader to the surveys of Angel [3] and Michiels, Aarts and Korst [29] who present a review of performance guarantees and other theoretical aspects of local search for a wide variety of combinatorial problems, including scheduling problems.

Specifically, in this paper we consider the problem of minimizing the total weighted completion time on parallel machine environments, and study the worst-case performance guarantee of local optima solutions for jump and swap neighborhoods. We provide a systematic method to analyze the worst-case performance of these basic local search heuristics for scheduling with the weighted completion time objective. The method, which turns out to be quite simple, involves relating the local optimality conditions with an appropriate *schedule mapping* that maps a schedule into an inner product space, and allows to write the condition in terms of the inner product of the mappings of a global optimum and a local optimum [11].

1.1 Scheduling problems

In a typical parallel machine scheduling problem, we are given a set of n jobs to be scheduled on a set of m machines. Jobs must be scheduled without interruption on a single machine, and each machine can process one job at a time. Each job is associated with a weight and a processing time. The characteristics of the processing times of jobs define what is known as the machine environment. The simplest setting is that of identical parallel machines (P) in which the processing time of a job is equal in all machines. If, in addition, the machines have different processing speeds we have the so-called related parallel machine environment (Q), while in case the processing time of a job can arbitrarily depend on the machine which process it we have the unrelated machine environment (R). Furthermore, a common practical constraint refers to the fact that a given job can only be processed by a job-specific subset of machines ($\mathcal{M}_j$ for job j). These environments are known as restricted parallel machines and make sense in the case of identical machines or related machines. Of course, the most general machine environment is that of unrelated parallel machines.

For the problems considered in this paper, a schedule is an assignment of jobs to machines together with an ordering of the jobs within each machine. Here, we consider that the order within a machine is always taken to be the decreasing order of the weight to processing time ratio, also known as Smith ratio [35]. Therefore, for our purposes, a schedule is fully determined by the assignment of jobs to machines. Given a schedule, we denote by C_j the completion time of job j. With this, we can define the three most widely studied objective functions. First, in the *Total completion time* objective the goal is to find a schedule minimizing $\sum_j C_j$. Second, the *Total weighted completion time* objective aims at minimizing $\sum_j w_j C_j$, where $w_j \geq 0$ is the weight of job j. Finally, the *Makespan* objective looks for a schedule of minimum maximum completion time, i.e., minimizing $C_{\max} = \max_j \{C_j\}$.

Taking as reference the standard three field scheduling notation [19], we represent the problems by $\alpha|\beta|\gamma$, where α can be R, Q or P, β can be empty or $\mathcal{M}_j$ representing the machine eligibility restrictions (according to notation of the book of Pinedo [30])

and γ can be $\sum C_j$ or $\sum w_j C_j$. Specifically, the problems we study are: $R||\sum w_j C_j$, $R||\sum C_j$, $Q|\mathcal{M}_j|\sum w_j C_j$, $Q|\mathcal{M}_j|\sum C_j$, $P|\mathcal{M}_j|\sum w_j C_j$, and $P|\mathcal{M}_j|\sum C_j$.

Graham et al. [19] initiated the systematic study of algorithms and complexity for scheduling problems. For any parallel machine environment, minimizing the total completion time ($\sum C_j$) can be performed in polynomial time. More precisely, the problem $P||\sum C_j$ can be solved in $O(n \log n)$ using the Shortest processing time (SPT) rule [12]. For the problem $Q||\sum C_j$, complexity increases to $O(n \log nm)$ using a generalization of the SPT rule, where jobs are assigned sequentially to the machine that allows a shorter completion time, according to their workload and speed [12,23]. The problem $R||\sum C_j$ can be solved in polynomial time by bipartite matching techniques [22]. If the objective is to minimize the total weighted completion time ($\sum w_j C_j$) or the makespan (C_{max}), the problems are NP-hard [18], even for the case of two identical machines [8,26]. When m is part of the input, these problems are strongly NP-hard [17], independent of the parallel machine environment considered [23,25].

1.2 Local search and approximation

Since for the makespan and total weighted completion time objectives the machine scheduling problems become NP-hard, it is natural to look for approximate solutions. There is indeed a rich literature on approximation algorithms for scheduling problems and we mention the current best known factors for the problems considered in this paper in Table 1. Another approach, vastly used in practice and the main focus of this paper is to use local search techniques.

Typically, when designing a good local search heuristic we require that the size of the neighborhood (where new solutions will be sought) is small enough. Furthermore, a key aspect is that local optima are close, in objective function value, to a global optimum [2]. One way to evaluate the quality of local optima is by worst-case analysis, but this is often notoriously difficult, or simply impossible since there may exist very bad local optima. In this paper we provide a systematic study of two basic local search algorithms and prove that they have a good worst-case performance. Specifically we study the *Jump* neighborhood (also known as *move* or *insertion*), which is defined as moving a job from one machine to another. Also we study the *Swap* neighborhood (also known as *exchange* or *interchange*), in which a local move is defined as selecting two jobs assigned to different machines, and then interchanging their machine allocations. Both, jump and swap neighborhood are polynomial size neighborhoods.

Other neighborhoods that have been used for scheduling in parallel machines environments include *lexjump*, *push*, *multi-exchange* and *split*. Lexjump is a polynomial size neighborhood that extends the jump neighborhood. A push move can be seen as a sequence of jump moves while a *multi-exchange move* can be seen as a sequence of swap moves, so in a way these are generalization of the most basic local search algorithms. While push is of polynomial size, multi-exchange is exponential. Finally, split is an exponential size neighborhood introduced by Brueggemann et al. [7] for the problem $P||C_{max}$. In terms of their use, lexjump was first used for minimizing the makespan [7,31–33]. Push was introduced by Schuurman and Vredeveld

Table 1 Performance guarantees

Problem	Best approximation	Jump and swap performance guarantee[a]	References
$R\|\|C_{\max}$	2 [27]	unbounded	[33]
$Q\|\mathcal{M}_j\|C_{\max}$	2	$[\sqrt{\frac{1}{4}+(m-1)s_{\max}}$, $\frac{1}{2}+\sqrt{\frac{1}{4}+(m-1)s_{\max}}]$	[31,32]
$P\|\mathcal{M}_j\|C_{\max}$	2	$\frac{1}{2}+\sqrt{m-\frac{3}{4}}$	[31,32]
$Q\|\|C_{\max}$	PTAS [14,21]	$\frac{1+\sqrt{4m-3}}{2}$	[10,33]
$P\|\|C_{\max}$	PTAS [20]	$2-\frac{2}{m+1}$	[15,33]
$R\|\|\sum w_jC_j$	$\frac{3}{2}-c$ [4,28][b]	2.618	[1], Theorems 1 and 5
$Q\|\mathcal{M}_j\|\sum w_jC_j$	$\frac{3}{2}-c$	2.618	Theorems 1 and 5
$P\|\mathcal{M}_j\|\sum w_jC_j$	$\frac{3}{2}-c$	[1.75, 1.809]	Theorems 3 and 7
$P\|\|\sum w_jC_j$	PTAS [34]	$\left[\frac{6}{5},\frac{3}{2}-\frac{1}{2m}\right]$[c]	[6]
$R\|\|\sum C_j$	Poly-time [22]	2	Theorems 2 and 6
$Q\|\mathcal{M}_j\|\sum C_j$	Poly-time	2	Theorems 2 and 6
$P\|\mathcal{M}_j\|\sum C_j$	Poly-time	[1.525, 1.618]	Theorems 4 and 8

[a] In the table, the numbers 1.618, 2.618 and 1.809 are actually numerical approximations of ϕ, $1+\phi$, and $1+\phi/2$, respectively, where ϕ is the golden ratio
[b] For $c > 1/6000$
[c] Only jump neighborhood

[33] for $P||C_{max}$ and $Q||C_{max}$, while Recalde et al. [31] extended its application to $P|\mathcal{M}_j|C_{max}$ and $Q|\mathcal{M}_j|C_{max}$. Multi-exchange was introduced by Frangioni et al. [16] for $R||C_{max}$, and then used for the problems P, Q, $R||C_{max}$ [33].

In Table 1 we present a summary of the performance guarantees that have been obtained for the Jump and Swap neighborhoods. In particular we remark that Brueggemann et al. [6] studied the performance guarantee of Jump solutions for $P||\sum w_jC_j$ and determined that for these solutions the performance guarantee is within the range $[6/5, (3m-1)/(2m)]$. Additionally, they determine that the performance guarantee for the special case where all jobs have the same Smith ratio is $(9-\sqrt{6})/6$. Finn and Horowitz [15], and Cho and Sahni [10] obtain upper bounds for the performance guarantee of Jump solutions for $P||C_{max}$ and $Q||C_{max}$. Later, Schuurman and Vredeveld [33] establish that these bounds are tight. Combined neighborhoods of jump with other neighborhoods for $P||C_{\max}$ were treated by Brueggemann et al. [7] while the performance guarantee of Jump for $R||C_{max}$, $Q|\mathcal{M}_j|C_{max}$ and $P|\mathcal{M}_j|C_{max}$ were studied by Recalde et al. [31], Rutten et al. [32] and Schuurman and Vredeveld [33]. The efficiency of local search methods under jump neighborhood for related problems have also received attention [5,33].

Decentralized and game theoretic versions of the problems $R||\sum w_jC_j$, $Q|\mathcal{M}_j|\sum w_jC_j$ and $P|\mathcal{M}_j|\sum w_jC_j$ have also been considered recently [1,9,11,13]. Here jobs are agents who selfishly decide the machine to be processed. Since this decision in made considering only the job's cost (rather than the overall objective) the implied

results for the price of anarchy [24] do not directly apply to our setting. Furthermore the focus of most of this line of research is to design machine processing policies that achieve good worst-case performance, whereas in our case this is just given by Smith rule. An exception is the result of Abed et al. [1] who consider the more general setting in which agents may own multiple jobs and as a consequence they establish that the performance guarantee of Jump optimal solutions for $R||\sum w_j C_j$ is at most 2.618.

1.3 Our results

In this paper we establish upper and lower bounds on the performance guarantee for the Jump and Swap neighborhoods in several scheduling settings, namely for the unrelated, restricted related and restricted parallel machines environments and the objective functions $\sum C_j$ and $\sum w_j C_j$. The specific bounds for each problem are summarized in Table 1. However, we believe that rather than the numeric improvements our main contribution is more conceptual. We devise a simple method for bounding the performance of local search heuristics borrowing ideas form the work of Cole et al. [11]. The rough idea is to first establish a simple necessary condition for local optimality that takes the form of a certain inequality. Then we relate the right-hand side of this inequality to the costs of an optimal schedule and a locally optimal schedule by using a scheduling mapping that assigns to each schedule an integrable function. Finally we apply some basic inequalities for real and/or integer numbers to obtain the final guarantee. We also present lower bound instances for all the considered problems.

Let us mention that the bound of 2.618 for $R||\sum w_j C_j$ is implied by the the work of Abed et al. [1]. However, our proof here is much simpler and does not go through the more complicated game theoretic setting.

Roadmap. The rest of the paper is structured as follows. We present preliminary background in § 2. The main tools of the proof are presented in § 3 while the upper bounds on the performance guarantee of jump are established in § 4. Then in § 5 we present tight or nearly tight lower bound instances for all considered problems.

2 Preliminaries

2.1 Basic definitions and problem statement

Consider $\mathcal{J}$ a set of n jobs to be scheduled on a set $\mathcal{M}$ of m machines. Let p_{ij} denote the non-negative processing time of job $j \in \mathcal{J}$ on machine $i \in \mathcal{M}$. Jobs must be scheduled without interruption on a single machine, and each machine can process one job at a time. In addition all jobs and machines are available from the beginning. Depending on the precise machine environment the processing times p_{ij} satisfy additional requirements. Specifically in this paper we consider the following:

- *Identical machines (P)* All machines are identical, meaning each job needs the same processing time on each machine: $p_{ij} = p_j, \forall i \in \mathcal{M}$.
- *Restricted identical machines ($P|\mathcal{M}_j|\cdot$)* This is a variant of identical machines according to which each job j can be processed only on some specified subset of

machines $\mathcal{M}_j \subseteq \mathcal{M}$. For this environment, processing times are: $p_{ij} = p_j, \forall i \in \mathcal{M}_j$; $p_{ij} = \infty, \forall i \notin \mathcal{M}_j$.
- *Related machines (Q)* The machines may have different speeds, and the processing time of a job is inversely proportional to the speed: $p_{ij} = p_j/s_i$, where s_i represents the speed of machine i.
- *Restricted related machines ($Q|\mathcal{M}_j|\cdot$)* This is a variant of related machines according to which each job j can be processed only on some specified subset of machines $\mathcal{M}_j \subseteq \mathcal{M}$. For this environment, processing times are: $p_{ij} = p_j/s_i, \forall i \in \mathcal{M}_j$; $p_{ij} = \infty, \forall i \notin \mathcal{M}_j$.
- *Unrelated machines (R)* The processing times are arbitrary. All environments of parallel machines are a particular case of unrelated parallel machines.

In terms of objective functions we consider two of the most prominent ones, namely the *total completion time* and the *total weighted completion time*. Denoting by w_j the weight of job j and by C_j its completion time under a particular schedule, in the former objective the goal is to minimize $\sum_{j\in\mathcal{J}} C_j$ while in the latter the goal is to minimize $\sum_{j\in\mathcal{J}} w_j C_j$.

More precisely, a schedule corresponds to an assignment of jobs to machines, represented by a vector $\boldsymbol{x}$, where x_j gives the machine to which job j is assigned, that is, $x_j = i$ if job j is assigned to machine i in schedule $\boldsymbol{x}$. Given such an assignment $\boldsymbol{x}$, it is standard to assume that jobs are processed according to the WSPT rule or Smith rule, which simply sequence jobs in decreasing order of w_j/p_{ij}, since this is optimal for a fixed assignment [35].[1] When there are ties in the ratio, these are broken arbitrarily and to avoid confusion we denote by $\prec_i$ the precedence relation of jobs in machine i. Therefore, letting $\mathcal{J}_i(\boldsymbol{x})$ the set of jobs assigned to machine i in schedule $\boldsymbol{x}$, we can write the completion time of job j in schedule $\boldsymbol{x}$ as

$$C_j(\boldsymbol{x}) = p_{x_j j} + \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\prec_{x_j} j}} p_{x_j k} = \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\preceq_{x_j} j}} p_{x_j k}.$$

Here, for convenience, we have introduced the notation $\preceq_i$ to include all predecessors of a job on machine i and the job itself.

The cost of schedule $\boldsymbol{x}$ (total weighted completion time) and the weighted sum of processing times are defined as follows:

$$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x}) = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} w_j C_j(\boldsymbol{x}), \tag{1}$$

$$\eta(\boldsymbol{x}) = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} w_j p_{ij}. \tag{2}$$

[1] When there are no weights, these are taken to be 1, and therefore the rule is just the shortest processing time first rule.

With these definitions we have the following identities

$$C(\boldsymbol{x}) = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\preceq_i j}} w_j p_{ik} = \eta(\boldsymbol{x}) + \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\prec_i j}} w_j p_{ik}. \quad (3)$$

$$C(\boldsymbol{x}) = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\succeq_i j}} w_k p_{ij} = \eta(\boldsymbol{x}) + \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\succ_i j}} w_k p_{ij}. \quad (4)$$

2.2 Local search neighborhoods

For the general machine scheduling problems studied in this paper the goal is to minimize $C(\boldsymbol{x})$. In this paper we will consider only locally optimal solutions for the two most basic neighborhoods, namely *jump* and *swap*.

In the *jump* neighborhood, also known as *move* or *insertion* neighborhood, a *jump move* is defined as moving a single job from one machine to another. A successful jump reduces the objective function. Given a solution, if it is not possible to improve it in this way, we have a local optimum and call this solution *Jump-Opt*. More precisely, given a schedule $\boldsymbol{x}$ we look for a schedule $\boldsymbol{x}'$, such that $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ in exactly one coordinate and $C(\boldsymbol{x}') < C(\boldsymbol{x})$. Similarly, in a *swap* neighborhood, also known as *exchange* or *interchange* neighborhood, a *swap move* consists of selecting two jobs assigned to different machines, and then interchanging the machine allocations for these jobs. A successful swap reduces the objective function. Additionally, the *swap* neighborhood contains the *jump* neighborhood by allowing to take an *empty* job in the swap. Given a solution, if it is not possible to obtain improvements in this way, we have a local optimum and call it *Swap-Opt*. More precisely, given a schedule $\boldsymbol{x}$ we take two coordinates u, v and construct $\boldsymbol{x}'$ from $\boldsymbol{x}$ by exchanging the values in these coordinates and check whether $C(\boldsymbol{x}') < C(\boldsymbol{x})$. Both, jump and swap are polynomial size neighborhoods.

In our results for the worst-case performance of local search, we prove the upper bounds for the jump neighborhood and therefore the results also apply to the swap neighborhood. One the other hand, in all our lower bounds the Swap–Opt solutions are also Jump-Opt solutions and thus all our results apply to both neighborhoods.

2.3 Machine eligibility

Since all our results apply to machine environments in which the set of available machines for a given job can be a restricted set, we can consider a reduction to a situation in which each job j has only two available machines: the machine in which an optimal solution processes it and the machine in which a locally optimal solution processes it. More precisely, let $\mathcal{I}$ be an instance of the problem $R||\sum w_j C_j$, where the job j can be processed by a set $\mathcal{M}_j \subseteq \mathcal{M}$. That is, $p_{ij} = \infty$, for all $i \notin \mathcal{M}_j$. Let $\boldsymbol{x}^*$ be some optimal solution of $\mathcal{I}$, where the job j is processed by machine x^*_j, and $\boldsymbol{x}$ some Jump–Opt solution, where the job j is processed by the machine x_j. Consider a

new instance where $\mathcal{M}'_j = \{x_j, x^*_j\} \subseteq \mathcal{M}_j$ is the reduced set of machines which can process the job j, and call it $\mathcal{I}'$. This (machine eligibility) reduction satisfies:

i. $C(\boldsymbol{x})$ will be the same in both instances, $\mathcal{I}$ and $\mathcal{I}'$.
ii. $C(\boldsymbol{x}^*)$ will be the same in both instances.
iii. If $\boldsymbol{x}$ is a Jump-Opt solution of $\mathcal{I}$, then $\boldsymbol{x}$ is a Jump-Opt solution of $\mathcal{I}'$.
iv. If $\boldsymbol{x}^*$ is optimal for $\mathcal{I}$, then $\boldsymbol{x}^*$ is optimal for $\mathcal{I}'$.

Therefore, we can restrict ourselves to instances in which jobs can be assigned to at most two machines[2].

2.4 Basic inequalities

In order to determine upper bounds for the performance guarantees of Jump–Opt solutions, we need to establish some inequalities.

Lemma 1 *For non-negative real numbers a, b and β, it holds that*

$$ab \leq \frac{\beta}{2}a^2 + \frac{1}{2\beta}b^2.$$

Proof The statement follow from the fact that $(a\beta - b)^2 \geq 0$, for real numbers a, b and β. □

Lemma 2 *For every pair of non-negative integers a and b, it holds that*

$$ab \leq \frac{1}{2}\left(a^2 + b^2 - a + b\right).$$

Proof The statement follow from the fact that $(a-b)^2 \geq a - b$, for integer values of a and b. □

Lemma 3 *For every pair of non-negative integers a, b and $\phi = \frac{1+\sqrt{5}}{2}$, it holds that*

$$ab \leq \frac{1}{2\phi^2}a^2 + \frac{\phi^2}{2}b^2 + \frac{\phi}{2}a - \frac{\phi^2}{2}b.$$

Proof The inequality can be rewritten as $f\,(a,b) = \frac{1}{2\phi^2}a^2 + \frac{\phi^2}{2}b^2 + \frac{\phi}{2}a - \frac{\phi^2}{2}b - ab \geq 0$. Clearly, if either $a = 0$ or $b = 0$ the inequality follows directly. Thus it only remains to prove that $f\,(a,b) \geq 0$, for all $a \geq 1$ and $b \geq 1$. Then the problem can be formulated as $\min\{f(a,b) : a \geq 1, b \geq 1\}$. Note that $f(a,b) = (\frac{a}{\sqrt{2}\phi} - b\frac{\phi}{\sqrt{2}})^2 + \frac{\phi a}{2} - \frac{\phi^2 b}{2}$ and thus convex. Letting λ_a and λ_b be the multipliers of $a \geq 1$ and $b \geq 1$, respectively, the Karush-Kuhn-Tucker (KKT) conditions for the problem are:

$$\frac{a}{\phi^2} + \frac{\phi}{2} - b - \lambda_a = 0$$

[2] Note that it is possible that $|\mathcal{M}'_j| = 1$.

$$\phi^2 b - \frac{\phi^2}{2} - a - \lambda_b = 0$$
$$\lambda_a(a-1) = \lambda_b(b-1) = 0$$
$$a, b \geq 1$$
$$\lambda_a, \lambda_b \geq 0$$

As the solution for the KKT conditions is $a = b = 1$, $\lambda_a = \frac{1}{2\phi^2}$, and $\lambda_b = \frac{1}{2\phi}$, we conclude that the minimum value is $f(1,1) = 0$. Therefore $f(a,b) \geq 0$ for $a, b \geq 1$. □

3 Local optima and schedule mapping

3.1 Local optima condition

In this section we establish a local optima condition for Jump–Opt solutions for the general case of unrelated parallel machines.

Lemma 4 *For any optimal schedule $\boldsymbol{x}^*$ and any Jump-Opt solution $\boldsymbol{x}$ of $R||\sum w_j C_j$, it holds that*

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}_i(\boldsymbol{x})} \sum_{k \in \mathcal{J}_i(\boldsymbol{x}^*)} w_j w_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}.$$

Proof From the machine eligibility reduction of § 2.3, we can assume that each job $j \in \mathcal{J}$ can only be processed by machines x_j or x_j^*, without changing the performance guarantee. Let δ_j be the amount by which $C(\boldsymbol{x})$ decreases if job j is removed from machine x_j and let δ'_j be the increment in $C(\boldsymbol{x})$ if job j is moved to machine x_j^*.[3] Thus,

$$\delta_j = w_j C_j(\boldsymbol{x}) + p_{x_j j} \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succ_{x_j} j}} w_k.$$

$$\delta'_j = w_j p_{x_j^* j} + w_j \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^* k} + p_{x_j^* j} \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k.$$

Note that it may happen that $x_j = x_j^*$ but in this case $\delta_j = \delta'_j$. Therefore, schedule $\boldsymbol{x}$ will be a Jump-Opt solution if and only if $\delta_j \leq \delta'_j$ for all $j \in \mathcal{J}$, which happens if and only if

[3] Observe that job j has to be inserted on machine x_j^* at the appropriate position (defined by WSPT rule).

$$w_j C_j(\boldsymbol{x}) + p_{x_j j} \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succ_{x_j} j}} w_k \leq w_j p_{x_j^* j} + w_j \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^* k} + p_{x_j^* j} \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k.$$

Adding $w_j p_{x_j j}$ to both sides we get that the latter is equivalent to

$$w_j C_j(\boldsymbol{x}) + p_{x_j j} \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succeq_{x_j} j}} w_k \leq w_j p_{x_j j} + w_j p_{x_j^* j} + w_j \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^* k} + p_{x_j^* j} \sum_{\substack{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k.$$

Finally, recalling that if $k \prec_{x_j^*} j$, then $\frac{p_{x_j^* k}}{w_k} \leq \frac{p_{x_j^* j}}{w_j}$, and otherwise $\frac{p_{x_j^* j}}{w_j} \leq \frac{p_{x_j^* k}}{w_k}$. Therefore, in a Jump-Opt solution we have that:

$$w_j C_j(\boldsymbol{x}) + p_{x_j j} \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succeq_{x_j} j}} w_k \leq w_j (p_{x_j j} + p_{x_j^* j}) + \sum_{k \in \mathcal{J}_{x_j^*}(\boldsymbol{x})} w_j w_k \min\left\{ \frac{p_{x_j^* k}}{w_k}, \frac{p_{x_j^* j}}{w_j} \right\}.$$

We sum over all $j \in \mathcal{J}$ and use Equations (2) and (4) to conclude that

$$\sum_{j \in \mathcal{J}} w_j C_j(\boldsymbol{x}) + C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}_i(\boldsymbol{x})} \sum_{k \in \mathcal{J}_i(\boldsymbol{x}^*)} w_j w_k \min\left\{ \frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j} \right\}.$$

Concluding the proof of the result. □

Clearly, we can apply this result to the more specific objectives considered in this paper by specializing the last lemma to more restricted settings.

3.2 Schedule mapping

Cole et al. [11] study a scheduling game where the jobs are allocated to unrelated parallel machines and propose decentralized mechanism for the problem, considering the minimization of the weighted completion time. To determine the price of anarchy Cole et al. [11] propose a mapping from the set of schedules to a certain inner product space such that the norm of the mapping will closely correspond to the cost of the schedule. We make use of this *schedule mapping* here as well. Consider the function $\varphi : \mathcal{M}^{\mathcal{J}} \to L_2([0, \infty))^{\mathcal{M}}$, which maps every schedule $\boldsymbol{s}$ to a vector of functions (one for each machine) as follows. If $\boldsymbol{f} = \varphi(\boldsymbol{s})$, then for each machine $i \in \mathcal{M}$

$$f_i(y) = \sum_{j \in \mathcal{J}_i(\boldsymbol{s}): \frac{p_{ij}}{w_j} \geq y} w_j. \tag{5}$$

We will use some basic properties of this mapping φ which are contained in the work of Cole et al. [11]. We provide a proof for the sake of completeness.

Lemma 5 *Let $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ be two schedules of an instance of $R||\sum w_j C_j$, with $\boldsymbol{f} = \varphi(\boldsymbol{s}_1)$ and $\boldsymbol{g} = \varphi(\boldsymbol{s}_2)$. Then*

$$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)dy = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} p_{ij}, \tag{6}$$

$$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)^2 dy = 2C(\boldsymbol{s}_1) - \eta(\boldsymbol{s}_1), \tag{7}$$

$$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)g_i(y)dy = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_2)} w_j w_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}. \tag{8}$$

Proof For (6),

$$\begin{aligned}
\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)dy &= \sum_{i\in\mathcal{M}} \int_0^\infty \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1):\ y\le\frac{p_{ij}}{w_j}} w_j dy \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} w_j \int_0^\infty \mathbb{1}_{y\le\frac{p_{ij}}{w_j}} dy \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} w_j \frac{p_{ij}}{w_j} \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} p_{ij}.
\end{aligned}$$

For (8),

$$\begin{aligned}
\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)g_i(y)dy &= \sum_{i\in\mathcal{M}} \int_0^\infty \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1):\ y\le\frac{p_{ij}}{w_j}} w_j \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_2):\ y\le\frac{p_{ik}}{w_k}} w_k dy \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_2)} w_j w_k \int_0^\infty \mathbb{1}_{y\le\frac{p_{ij}}{w_j}} \mathbb{1}_{y\le\frac{p_{ik}}{w_k}} dy \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_2)} w_j w_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}.
\end{aligned}$$

To prove (7), we start by using (8) and then apply Equations (3) and (4).

$$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)^2 dy = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_1)} w_j w_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}$$

$$
\begin{aligned}
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{k\in\mathcal{J}_i(\boldsymbol{s}_1)} \left(w_j p_{ik} \mathbb{1}_{\frac{p_{ik}}{w_k} \le \frac{p_{ij}}{w_j}} + w_k p_{ij} \mathbb{1}_{\frac{p_{ik}}{w_k} > \frac{p_{ij}}{w_j}} \right) \\
&= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{s}_1) \\ k\preceq_i j}} w_j p_{ik} + \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{s}_1)} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{s}_1) \\ k\succ_i j}} w_k p_{ij} \\
&= 2C(\boldsymbol{s}_1) - \eta(\boldsymbol{s}_1).
\end{aligned}
$$

□

4 Quality of local optima

In this section we combine the local optimality condition of Lemma 4, the properties of the scheduling mapping of Lemma 5, together with the basic inequalities shown in § 2.4 to obtain tight or nearly tight upper bounds on the performance of Jump–Opt solutions. The specific environments we consider include unrelated parallel machines with weighted and unweighted completion time objectives, and restricted identical parallel machines again with weighted and unweighted completion time objectives.

Theorem 1 *The performance guarantee of Jump–Opt solutions for* $R||\sum w_j C_j$ *is at most* $1+\phi \approx 2.618$.

Proof By Lemma 4 and (8), we have

$$
2C(\boldsymbol{x}) \le \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i\in\mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy,
$$

where $f_i(y)$ and $f_i^*(y)$ are the mappings of $\boldsymbol{x}$ and $\boldsymbol{x}^*$, respectively. Since both $f_i^*(y)$ and $f_i(y)$ are real numbers, we can use inequality from Lemma 1. Thus,

$$
2C(\boldsymbol{x}) \le \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i\in\mathcal{M}} \int_0^\infty \left(\frac{\beta}{2} f_i^{*2}(y) + \frac{1}{2\beta} f_i^2(y) \right) dy.
$$

By (7), we have

$$
\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \le \beta C(\boldsymbol{x}^*) + \left(1 - \frac{1}{2\beta}\right) \eta(\boldsymbol{x}) + \left(1 - \frac{\beta}{2}\right) \eta(\boldsymbol{x}^*).
$$

Also from (3), we have that $\eta(\boldsymbol{x}^*) \le C(\boldsymbol{x}^*)$ and $\eta(\boldsymbol{x}) \le C(\boldsymbol{x})$. So assuming that, $(1 - 1/(2\beta)) \ge 0$ and $(1 - \beta/2) \ge 0$ (i.e., $1/2 \le \beta \le 2$), we have that

$$
\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \le \left(1 - \frac{1}{2\beta}\right) C(\boldsymbol{x}) + \left(1 + \frac{\beta}{2}\right) C(\boldsymbol{x}^*).
$$

Thus for $1/2 \leq \beta \leq 2$ we have that

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} \leq \frac{(2+\beta)\beta}{2\beta - 1}.$$

Minimizing the right-hand side over β we get that the optimal choice is $\beta = \phi = \frac{1+\sqrt{5}}{2} \approx 1.618$, yielding the upper bound on the performance guarantee $C(\boldsymbol{x}) \leq (1+\phi)C(\boldsymbol{x}^*)$. □

Theorem 2 *The performance guarantee of Jump–Opt solutions for* $R||\sum C_j$ *is at most* 2.

Proof By Lemma 4 and (8), we have

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy,$$

where $f_i(y)$ and $f_i^*(y)$ are the mappings of $\boldsymbol{x}$ and $\boldsymbol{x}^*$, respectively. Since $w_j = 1$ for all $j \in \mathcal{J}$, both $f_i(y)$ and $f_i^*(y)$ are non-negative integers, so we can use Lemma 2. Thus,

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \frac{1}{2} \sum_{i \in \mathcal{M}} \int_0^\infty \big(f_i^2(y) + f_i^{*2}(y) - f_i(y) + f_i^*(y)\big) dy.$$

Again using that $w_j = 1$ by (6) we can write $\eta(\boldsymbol{x}) = \sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) dy$ and $\eta(\boldsymbol{x}^*) = \sum_{i \in \mathcal{M}} \int_0^\infty f_i^*(y) dy$. Finally from (7) in Lemma 5, we have

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \frac{1}{2}\big(2C(\boldsymbol{x}) - \eta(\boldsymbol{x}) + 2C(\boldsymbol{x}^*) - \eta(\boldsymbol{x}^*) - \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*)\big),$$

thus, $C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}^*) + C(\boldsymbol{x}^*)$ and since by (3) we have $\eta(\boldsymbol{x}^*) \leq C(\boldsymbol{x}^*)$, we can conclude the upper bound on the performance guarantee. □

In the last two results we will consider restricted identical parallel machines and therefore we have that $p_{ij} = p_j$, for all $i \in \mathcal{M}$, $j \in \mathcal{J}$. Then, $\eta(\boldsymbol{x}) = \eta(\boldsymbol{x}^*)$. So, we can let $\eta = \eta(\boldsymbol{x}) = \eta(\boldsymbol{x}^*) = \sum_{j \in \mathcal{J}} w_j p_j$.

Theorem 3 *The performance guarantee of Jump–Opt solutions for* $P|\mathcal{M}_j|\sum w_j C_j$ *is at most* $1 + \frac{\phi}{2} \approx 1.809$.

Proof By Lemma 4 and (8) we have

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy,$$

where $f_i(y)$ and $f_i^*(y)$ are the mappings of $\boldsymbol{x}$ and $\boldsymbol{x}^*$, respectively. Since now $f_i^*(y)$ and $f_i(y)$ are real numbers, we use Lemma 1. Thus,

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \int_0^\infty \left(\frac{\beta}{2} f_i^{*2}(y) + \frac{1}{2\beta} f_i^2(y)\right) dy.$$

From (7), we conclude $2C(\boldsymbol{x}) \leq 2\eta + \frac{\beta}{2}(2C(\boldsymbol{x}^*) - \eta) + \frac{1}{2\beta}(2C(\boldsymbol{x}) - \eta)$ or equivalently

$$\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \leq \beta C(\boldsymbol{x}^*) + \left(2 - \frac{\beta}{2} - \frac{1}{2\beta}\right) \eta.$$

Finally, assuming that $2 - \beta/2 - 1/(2\beta) \geq 0$, and using $\eta \leq C(\boldsymbol{x}^*)$, we have that

$$\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \leq \left(2 + \frac{\beta}{2} - \frac{1}{2\beta}\right) C(\boldsymbol{x}^*).$$

Thus,

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} \leq \frac{\beta^2 + 4\beta - 1}{2(2\beta - 1)}.$$

Minimizing the right-hand side we obtain that the optimal choice is $\beta = \phi = \frac{1+\sqrt{5}}{2} \approx 1.618$ which moreover satisfies the assumption. Using $1+\phi = \phi^2$ amounts to conclude that $C(\boldsymbol{x}) \leq (1 + \phi/2) C(\boldsymbol{x}^*)$. □

Theorem 4 *The performance guarantee of Jump–Opt solutions for $P|\mathcal{M}_j|\sum C_j$ is at most $\phi \approx 1.618$.*

Proof By Lemma 4 and (8) we have

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy,$$

where $f_i(y)$ and $f_i^*(y)$ are the mappings of $\boldsymbol{x}$ and $\boldsymbol{x}^*$, respectively. Since $w_j = 1$ for all $j \in \mathcal{J}$, $f_i(y)$ and $f_i^*(y)$ are non-negative integers so we can use Lemma 3 to conclude that

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \int_0^\infty \left(\frac{1}{2\phi^2} f_i^2(y) + \frac{\phi^2}{2} f_i^{*2}(y) + \frac{\phi}{2} f_i(y) - \frac{\phi^2}{2} f_i^*(y)\right) dy.$$

Again by the unit weight assumption and (6), we have

$$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y) dy = \sum_{i\in\mathcal{M}} \int_0^\infty f_i^*(y) dy = \sum_{j\in\mathcal{J}} p_j = \eta.$$

Combining this with (7) we get that,

$$\left(2-\frac{1}{\phi^2}\right)C(\boldsymbol{x}) \leq \left(2-\frac{1}{2\phi^2}-\phi^2+\frac{\phi}{2}\right)\eta+\phi^2 C(\boldsymbol{x}^*).$$

Finally, since $\eta \leq C(\boldsymbol{x}^*)$ and $1+\phi=\phi^2$, we have $C(\boldsymbol{x}) \leq \phi C(\boldsymbol{x}^*)$. □

5 Lower bounds

In order to determine lower bounds for the performance guarantee of Jump–Opt solutions and Swap–Opt solutions, we present hard instances for both restricted related and restricted identical parallel machines environments for weighted and unweighted cases. These instances amount to conclude that the bounds in Theorems 1 and 2 are best possible. For Theorems 3 and 4 our lower bounds are close to the upper bounds but not tight. Our first two bounds apply even to the case of restricted related parallel machines.

5.1 Restricted related parallel machines, weighted case

Theorem 5 *The performance guarantee of Jump–Opt and Swap–Opt solutions for* $Q|\mathcal{M}_j|\sum w_jC_j$ *is at least* $\phi \approx 2.618$.

Proof We consider an instance $\mathcal{I}$ with n jobs and $m=n+1$ machines. Jobs have identical weight and processing time, $p_j=w_j=\phi^{2-j}$ for $j=1,\ldots,n$. Machine speed is $s_i=\phi^{4-2i}$ for machine $i=2,\ldots,n+1$, and $s_1=1$ for machine 1, where $\phi=\frac{1+\sqrt{5}}{2}$ is the golden ratio.

In optimal solution $\boldsymbol{x}^*$, job j is allocated on machine j, whereas in local optimum $\boldsymbol{x}$, it is allocated on machine $j+1$. Figure 1 shows a schematic representation of instance $\mathcal{I}$, where each machine is represented by a node, and each job is represented by a directed edge. The origin of edge k indicates the machine on which job k is assigned in the optimal solution, while the endpoint indicates the machine on which job k is assigned in the local optimum.

First, we note that any job $k \in \mathcal{J}$ can only be moved to machine k, and is not possible to perform any swap move. Then, a Jump-Opt solution is also a Swap-Opt solution for instance $\mathcal{I}$. Let δ_j be the reduction in $C(\boldsymbol{x})$ if job j is removed from machine $j+1$ and δ'_j be the increment in $C(\boldsymbol{x})$ if job j is moved to machine j. Since $w_j=p_j$, all jobs have Smith's ratio 1. Therefore, if more than one job is assigned to any machine, the sequencing is arbitrarily. Here, we assume that the moved job is placed last. Then,

$$\delta_j = w_j\frac{p_j}{s_{j+1}}, \text{ for all } j=1,\ldots,n$$

$$\delta'_j = \begin{cases} w_1p_1 & , \text{ for } j=1 \\ w_j\frac{(p_{j-1}+p_j)}{s_j} & , \text{ for all } j=2,\ldots,n \end{cases}$$

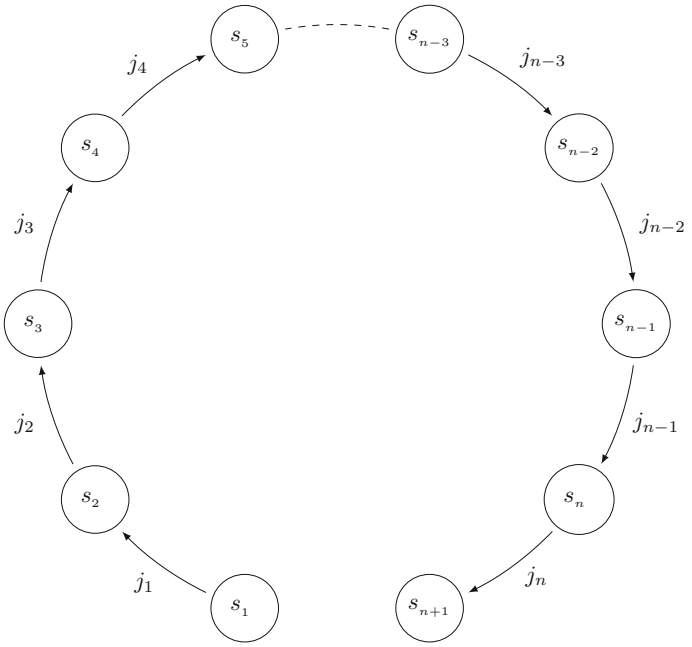

Fig. 1 Worst-case instance for restricted related parallel machines

Schedule $\boldsymbol{x}$ will be a Jump-Opt solution if and only if $\delta_j \leq \delta'_j$ for all $j = 1, \ldots, n$.

For instance $\mathcal{I}$, we have $p_{j-1} = \phi p_j$ for all $j \in \mathcal{J}$, and $\phi^2 s_{j+1} = s_j$ for $j \geq 2$ and $s_1 = s_2$. Thus, $\delta_j \leq \delta'_j$ is reduced to $\phi^2 \leq \phi + 1$ for all $j \geq 2$, and $1 \leq 1$ for $j = 1$. Therefore, $\boldsymbol{x}$ is a Jump-Opt solution for instance $\mathcal{I}$.

For $\boldsymbol{x}$ and $\boldsymbol{x}^*$, we have

$$C(\boldsymbol{x}^*) = \sum_{j=1}^{n} \frac{w_j p_j}{s_j} = \phi^2 + \sum_{j=2}^{n} \frac{w_j p_j}{s_j} = \phi^2 + n - 1 = n + \phi,$$

$$C(\boldsymbol{x}) = \sum_{j=1}^{n} \frac{w_j p_j}{s_{j+1}} = \sum_{j=1}^{n} \phi^2 = n\phi^2.$$

Then, the performance guarantee for $\boldsymbol{x}$ is

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{n\phi^2}{n+\phi} = \frac{\phi^2}{1+\frac{\phi}{n}}.$$

Finally, for any $\varepsilon > 0$ and for sufficiently large n, the performance guarantee is $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \phi^2 - \varepsilon$, which completes the proof. □

5.2 Restricted related parallel machines, unweighted case

Theorem 6 *The performance guarantee of Jump–Opt and Swap–Opt solutions for* $Q|\mathcal{M}_j|\sum C_j$ *is at least* 2.

Proof We consider an instance $\mathcal{I}$ similar to the instance presented in proof of Theorem 5. Instance $\mathcal{I}$ have n jobs and $m = n + 1$ machines. Each job have unit weight and processing time, so $w_j = p_j = 1$ for $j = 1, \ldots, n$. Machine speed is $s_1 = 1$ for machine 1, and $s_i = 2^{2-i}$ for machines $i = 2, \ldots, n + 1$.
In optimal solution $\boldsymbol{x}^*$, job j is allocated on machine j, whereas in local optimum $\boldsymbol{x}$, it is allocated on machine $j + 1$. Figure 1 shows a schematic representation of instance $\mathcal{I}$, where each machine is represented by a node, and each job is represented by a directed edge. The origin of edge k indicates the machine on which job k is assigned in the optimal solution, while the endpoint indicates the machine on which job k is assigned in the local optimum.

First, we note that any job $k \in \mathcal{J}$ can only be moved to machine k, and is not possible to perform any swap move. Then, a Jump-Opt solution is also a Swap-Opt solution for instance $\mathcal{I}$.

Let δ_j be the reduction in $C(\boldsymbol{x})$ if job j is removed from machine $j + 1$ and δ'_j be the increment in $C(\boldsymbol{x})$ if job j is moved to machine j. Since $w_j = p_j$, all jobs have Smith's ratio 1. Therefore, if more than one job is assigned to any machine, we assume that the moved job is the last. Then,

$$\delta_j = \frac{1}{s_{j+1}}, \text{ for all } j = 1, \ldots, n$$

$$\delta'_j = \begin{cases} 1 & , \text{ for } j = 1 \\ \frac{2}{s_j} & , \text{ for all } j = 2, \ldots, n \end{cases}$$

Schedule $\boldsymbol{x}$ will be a Jump-Opt solution if and only if $\delta_j \leq \delta'_j$ for all $j = 1, \ldots, n$.

For instance $\mathcal{I}$, we have $s_j = 2s_{j+1}$ for all $j \geq 2$, and $s_1 = s_2$. Thus, $\delta_j \leq \delta'_j$ is reduced to $1 \leq 1$ for all $j \geq 1$. Therefore, $\boldsymbol{x}$ is a Jump-Opt solution for instance $\mathcal{I}$.

For $\boldsymbol{x}$ and $\boldsymbol{x}^*$, we have

$$C(\boldsymbol{x}^*) = \sum_{j=1}^{n} \frac{1}{s_j} = 1 + \sum_{j=2}^{n} \frac{1}{s_j} = 1 + \sum_{j=2}^{n} 2^{j-2} = 1 + \sum_{j=0}^{n-2} 2^j = 2^{n-1},$$

$$C(\boldsymbol{x}) = \sum_{j=1}^{n} \frac{1}{s_{j+1}} = \sum_{j=1}^{n} 2^{j-1} = \sum_{j=0}^{n-1} 2^j = 2^n - 1.$$

Then, the performance guarantee for $\boldsymbol{x}$ is

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{2^n - 1}{2^{n-1}} = 2 - \frac{1}{2^{n-1}}.$$

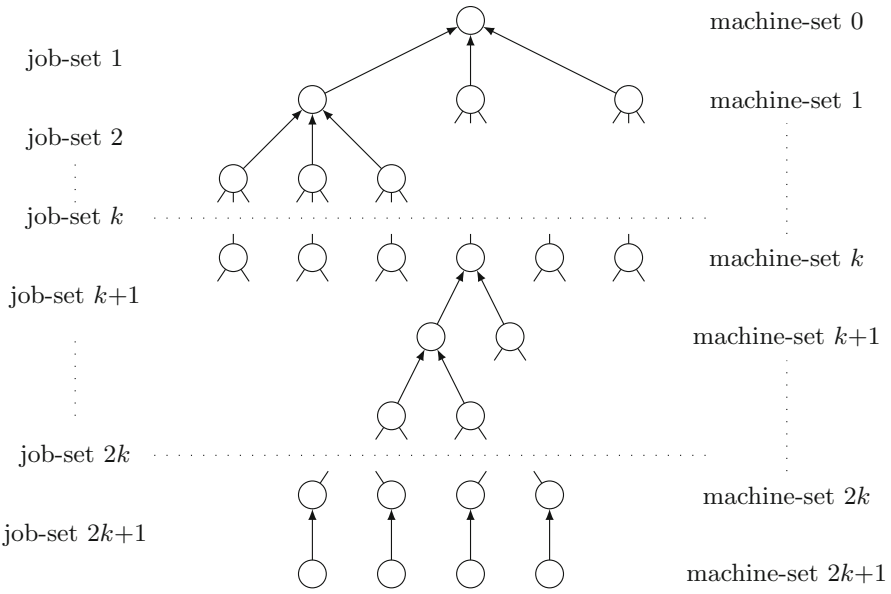

Fig. 2 Worst-case instance for $P|\mathcal{M}_j|\sum w_jC_j$

Finally, for any $\varepsilon > 0$ and for sufficiently large n, the performance guarantee is $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = 2 - \varepsilon$, which completes the proof. □

5.3 Restricted identical parallel machines, weighted case

To establish a lower bound instance for the problem $P|\mathcal{M}_j|\sum w_jC_j$, we use an instance inspired by the work of Caragiannis et al. [9] who obtained a lower bound for the price of anarchy of weighted load balancing in identical servers.

Theorem 7 *The performance guarantee of Jump–Opt and Swap–Opt solutions for* $P|\mathcal{M}_j|\sum w_jC_j$ *is at least* 1.75.

Proof We consider an instance $\mathcal{I}$ with $2k+1$ sets of jobs and $2k+2$ sets of machines. Inside each set, jobs and machines are identical. The instance is depicted in Fig. 2, where each machine is represented by a node, and each job is represented by a directed arc. The origin of the arc k indicates the machine where the job k is allocated in the optimal solution $\boldsymbol{x}^*$. The endpoint indicates the machine in which it is allocated in local optimum $\boldsymbol{x}$.

First, we note that for instance $\mathcal{I}$ is not possible to perform any swap move. Then, a Jump-Opt solution is also a Swap-Opt solution.

Let j_t be some job of job-set t, for $t = 1, \ldots, 2k+1$. These jobs can only be processed in some machine of machine-sets $t-1$ and t. In $\boldsymbol{x}$, some machine of machine-set $t-1$ processes job j_t, while in $\boldsymbol{x}^*$ it is processed by some machine of machine-set t.

For all jobs in job-set t, the processing time and weight is defined by:

$$w_t = p_t = \begin{cases} \left(\frac{2}{3}\right)^t, & \forall t = 1, \ldots, k \\ \left(\frac{2}{3}\right)^k \left(\frac{1}{2}\right)^{t-k-1}, & \forall t = k+1, \ldots, 2k \\ 2\left(\frac{1}{3}\right)^k, & t = 2k+1 \end{cases}$$

The cardinality of each machine-set is,

$$q_t = \begin{cases} 3^t, & \forall t = 0, \ldots, k \\ 3^k 2^{t-k}, & \forall t = k+1, \ldots, 2k \\ 6^k, & t = 2k+1 \end{cases}$$

For schedule $\boldsymbol{x}^*$, all machines process a single job, except for the machine in machine-set 0, which does not process any job. For schedule $\boldsymbol{x}$, machines in machine-sets $i = 0, \ldots, k-1$ will process three jobs, machines in machine-sets $i = k, \ldots, 2k-1$ will process two jobs, machines in machine-set $2k$ will process only one job, and machines in machine-set $2k+1$ do not process any job.

Let δ_t be the reduction in $C(\boldsymbol{x})$ if some job in job-set t is removed from the machine where it was assigned in the local optimum and δ'_t be the increment in $C(\boldsymbol{x})$ if that job is moved to some eligible machine in machine-set t. Then, schedule $\boldsymbol{x}$ will be a Jump-Opt solution if and only if $\delta_t \leq \delta'_t$ for all $t = 1, \ldots, 2k+1$.

For $t = 1, \ldots, k-1$, if some job in job-set t is moved from some machine in machine-set $t-1$ to some elegible machine in machine-set t, it would have $\delta_t = 3w_t p_t$ and $\delta'_t = w_t (3p_{t+1} + p_t)$. As $p_{t+1} = (2/3)p_t$, it holds that $\delta_t = \delta'_t$. Hence, no moves are made for jobs in job-sets $t = 1, \ldots, k-1$.

For $t = k$, if some job in job-set k is moved from some machine in machine-set $k-1$ to some elegible machine in machine-set k, it would have $\delta_k = 3w_k p_k$ and $\delta'_k = w_k (2p_{k+1} + p_k)$. As $p_{k+1} = p_k$, it holds that $\delta_k = \delta'_k$. Hence, no moves are made for jobs in job-set k.

For $t = k+1, \ldots, 2k-1$, if some job in job-set t is moved from some machine in machine-set $t-1$ to some elegible machine in machine-set t, it would have $\delta_t = 2w_t p_t$ and $\delta'_t = w_t (2p_{t+1} + p_t)$. As $p_{t+1} = (1/2)p_t$, it holds that $\delta_t = \delta'_t$. Hence, no moves are made for jobs in job-sets $j_t,\ t = k+1, \ldots, 2k-1$.

For $t = 2k$, if some job in job-set $2k$ is moved from some machine in machine-set $2k-1$ to some elegible machine in machine-set $2k$, it would have $\delta_{2k} = 2w_{2k} p_{2k}$ and $\delta'_{2k} = w_{2k} (p_{2k+1} + p_{2k})$. As $p_{2k+1} = p_{2k}$, it holds that $\delta_{2k} = \delta'_{2k}$. Hence, no moves are made for jobs in job-set $2k$.

Finally, if some job in job-set $2k+1$ is moved from some machine in machine-set $2k$ to some elegible machine in machine-set $2k+1$, no changes occur in $C(\boldsymbol{x})$, because machines in machine-set $2k+1$ don't have any jobs assigned in $\boldsymbol{x}$. Therefore, $\boldsymbol{x}$ is a Jump-Opt solution for instance $\mathcal{I}$.

The cost for Jump-Opt solution $\boldsymbol{x}$ is

$$\begin{aligned} C(\boldsymbol{x}) &= 6\sum_{t=1}^{k} q_{t-1}w_t p_t + 3\sum_{t=k+1}^{2k} q_{t-1}w_t p_t + q_{2k}w_{2k+1}p_{2k+1} \\ &= 6\sum_{t=1}^{k} 3^{t-1}\left(\tfrac{2}{3}\right)^{2t} + 3\sum_{t=k+1}^{2k} 3^k 2^{t-k-1}\left(\tfrac{2}{3}\right)^{2k}\left(\tfrac{1}{2}\right)^{2(t-k-1)} + 4\cdot 6^k\left(\tfrac{1}{3}\right)^{2k} \\ &= \left(\tfrac{8}{3}\right)\sum_{t=0}^{k-1}\left(\tfrac{4}{3}\right)^t + 3\left(\tfrac{4}{3}\right)^k\sum_{t=0}^{k-1}\left(\tfrac{1}{2}\right)^t + 4\left(\tfrac{2}{3}\right)^k = 14\left(\tfrac{4}{3}\right)^k - 8 - 2\left(\tfrac{2}{3}\right)^k. \end{aligned}$$

The cost for optimal solution $\boldsymbol{x}^*$ is

$$\begin{aligned} C(\boldsymbol{x}^*) &= \sum_{t=1}^{2k+1} q_t w_t p_t \\ &= \sum_{t=1}^{k} q_t w_t p_t + \sum_{t=k+1}^{2k} q_t w_t p_t + q_{2k+1}w_{2k+1}p_{2k+1} \\ &= sum_{t=1}^{k} 3^t\left(\tfrac{2}{3}\right)^{2t} + \sum_{t=k+1}^{2k} 3^k 2^{t-k}\left(\tfrac{2}{3}\right)^{2k}\left(\tfrac{1}{2}\right)^{2(t-k-1)} + 4\cdot 6^k\left(\tfrac{1}{3}\right)^{2k} \\ &= \sum_{t=1}^{k}\left(\tfrac{4}{3}\right)^t + 2\left(\tfrac{4}{3}\right)^k\sum_{t=0}^{k-1}\left(\tfrac{1}{2}\right)^t + 4\left(\tfrac{2}{3}\right)^k = 8\left(\tfrac{4}{3}\right)^k - 4. \end{aligned}$$

Then, the performance guarantee for $\boldsymbol{x}$ is

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{14\left(\frac{4}{3}\right)^k - 8 - 2\left(\frac{2}{3}\right)^k}{8\left(\frac{4}{3}\right)^k - 4} = \frac{7 - 4\left(\frac{3}{4}\right)^k - \left(\frac{1}{2}\right)^k}{4 - 2\left(\frac{3}{4}\right)^k}.$$

Finally, for any $\varepsilon > 0$ and for sufficiently large k, the performance guarantee is $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{7}{4} - \varepsilon$, which completes the proof. □

5.4 Restricted identical parallel machines, unweighted case

For the problem $P|\mathcal{M}_j|\sum C_j$ we consider a fixed finite instance depicted in Fig. 3, and use it to determine a lower bound for the performance guarantee of Jump–Opt and Swap–Opt solutions. The instance has 186 jobs and 168 machines. All jobs have unit weight and processing time. As usual, each machine is represented by a node, and each job is represented by a directed edge. The origin of edge k indicates the machine on which job k is assigned in the optimal solution $\boldsymbol{x}^*$, whereas the endpoint of edge k indicates the machine on which job k is assigned in the local optima solution

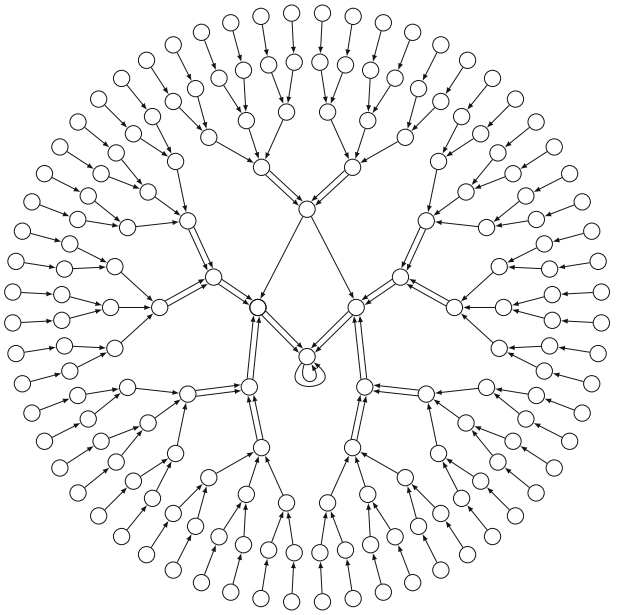

Fig. 3 Worst-case instance for restricted identical parallel machines

Table 2 Local optima for $P|\mathcal{M}_j|\sum C_j$ instance

Machines quantity	Jobs assigned	Unitary contribution	Partial contribution
1	6	21	21
2	5	15	30
5	4	10	50
10	3	6	60
30	2	3	90
60	1	1	60
60	0	0	0
			$C(\boldsymbol{x}) = 311$

$\boldsymbol{x}$. Checking that these solutions form a global and local optimum respectively is straightforward. For this instance is not possible to perform any swap move, so a Jump-Opt solution is also a Swap-Opt solution.

For $\boldsymbol{x}^*$, we have 18 machines that process two jobs and 150 machines that process only one job. Then, the cost for $\boldsymbol{x}^*$ is $C(\boldsymbol{x}^*) = 204$. While for the solution $\boldsymbol{x}$, we have $C(\boldsymbol{x}) = 311$ (the details are shown in Table 2). Then, the performance guarantee for $\boldsymbol{x}$ is $\frac{311}{204} \approx 1.5245$.

This amounts to conclude the following result.

Theorem 8 *The performance guarantee of Jump–Opt and Swap–Opt solutions for* $P|\mathcal{M}_j|\sum C_j$ *is at least* $311/204$.

We remark that by increasing the size of the instance and considering the local optimality conditions, we have been able to design instances with a performance guarantee of 1.533, however it is unclear how to further approach the upper bound of Theorem 4.

Acknowledgements We thank Fidaa Abed for providing the instance presented in § 5.4. We also thank two anonymous referees for many helpful suggestions that greatly improved the presentation of the paper. This work was partially supported by ANID Chile through grants BASAL AFB-180003 and BASAL AFB-170001.

References

1. Abed, F., Correa, J.R., Huang, C.: Optimal coordination mechanisms for multi-job scheduling games. In: ESA (2014)
2. Ahuja, R., Ergun, O., Orlin, J., Punnen, A.: A survey of very large-scale neighborhood search techniques. Discrete Appl. Math. **123**(1–3), 75–102 (2002)
3. Angel, E.: A survey of approximation results for local search algorithms. In: Bampis, E., Jansen, K., Kenyon, C. (eds.) Efficient Approximation and Online Algorithms: Recent Progress on Classical Combinatorial Optimization Problems and New Applications, vol. 3484, pp. 30–73. Springer, Berlin (2006)
4. Bansal, N., Srinivasan, A., Svensson, O.: Lift-and-round to improve weighted completion time on unrelated machines. In: STOC (2016)
5. Brucker, P., Hurink, J., Werner, F.: Improving local search heuristics for some scheduling problems. part II. Discrete Appl. Math. **72**(1), 47–69 (1997)
6. Brueggemann, T., Hurink, J.L., Kern, W.: Quality of move-optimal schedules for minimizing total weighted completion time. Op. Res. Lett. **34**(5), 583–590 (2006)
7. Brueggemann, T., Hurink, J.L., Vredeveld, T., Woeginger, G.J.: Exponential size neighborhoods for makespan minimization scheduling. Naval Res. Logist. **58**(8), 795–803 (2011)
8. Bruno, J., Coffman Jr., E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM **17**(7), 382–387 (1974)
9. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., Moscardelli, L.: Tight bounds for selfish and greedy load balancing. Algorithmica **61**(3), 606–637 (2011)
10. Cho, Y., Sahni, S.: Bounds for list schedules on uniform processors. SIAM J. Comput. **9**(1), 91–103 (1980)
11. Cole, R., Correa, J.R., Gkatzelis, V., Mirrokni, V., Olver, N.: Decentralized utilitarian mechanisms for scheduling games. Games Econ. Behav. **92**, 306–326 (2015)
12. Conway, R., Maxwell, W., Miller, L.: Theory of Scheduling. Addison-Wesley, Reading (1967)
13. Correa, J.R., Queyranne, M.: Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. Naval Res. Logist. **59**(5), 384–395 (2012)
14. Epstein, L., Sgall, J.: Approximation schemes for scheduling on uniformly related and identical parallel machines. Algorithmica **39**(1), 43–57 (2004)
15. Finn, G., Horowitz, E.: A linear time approximation algorithm for multiprocessor scheduling. BIT Numer. Math. **19**(3), 312–320 (1979)
16. Frangioni, A., Necciari, E., Scutellà, M.G.: A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. J. Comb. Optim. **8**(2), 195–220 (2004)
17. Garey, M.R., Johnson, D.S.: Strong NP-Completeness results: motivation, examples, and implications. J. ACM **25**(3), 499–508 (1978)
18. Garey, M.R., Johnson, D.S.: Computers and Intractability, a Guide to the Theory of NP-Completness. WH Freeman and Co, San Francisco (1979)
19. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)
20. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems theoretical and practical results. J. ACM **34**(1), 144–162 (1987)

21. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. SIAM J. Comput. **17**(3), 539–551 (1988)
22. Horn, W.A.: Technical note-minimizing average flow time with parallel machines. Op. Res. **21**(3), 846–847 (1973)
23. Horowitz, E., Sahni, S.: Exact and approximate algorithms for scheduling nonidentical processors. J. ACM **23**(2), 317–327 (1976)
24. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. Comput. Sci. Rev. **3**(2), 65–69 (2009)
25. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: algorithms and complexity. Handb. Op. Res. Manag. Sci. **4**, 445–522 (1993)
26. Lenstra, J., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. Ann. Discrete Math. **1**, 343–362 (1977)
27. Lenstra, J.K., Shmoys, D.B., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Math. Program. **46**(1–3), 259–271 (1990)
28. Li, S.: Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In: FOCS (2017)
29. Michiels, W., Aarts, E., Korst, J.: Theoretical Aspects of Local Search. Springer Science & Business Media, Berlin (2007)
30. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. Springer, Berlin (2016)
31. Recalde, D., Rutten, C., Schuurman, P., Vredeveld, T.: Local search performance guarantees for restricted related parallel machine scheduling. In: LATIN (2010)
32. Rutten, C., Recalde, D., Schuurman, P., Vredeveld, T.: Performance guarantees of jump neighborhoods on restricted related parallel machines. Op. Res. Lett. **40**(4), 287–291 (2012)
33. Schuurman, P., Vredeveld, T.: Performance guarantees of local search for multiprocessor scheduling. INFORMS J. Comput. **19**(1), 52–63 (2007)
34. Skutella, M., Woeginger, G.J.: A ptas for minimizing the total weighted completion time on identical parallel machines. Math. Op. Res. **25**(1), 63–75 (2000)
35. Smith, W.E.: Various optimizers for single-stage production. Naval Res. Logist. **3**(1–2), 59–66 (1956)