RAIRO-Oper. Res. 56 (2022) 1079–1088
https://doi.org/10.1051/ro/2022045

RAIRO Operations Research
www.rairo-ro.org

PERFORMANCE GUARANTEE OF THE JUMP NEIGHBORHOOD FOR SCHEDULING JOBS ON UNIFORMLY RELATED MACHINES

Felipe T. Muñoz[1,*] and Alejandro A. Pinochet[2]

Abstract. We study the worst case performance guarantee of locally optimal solutions for the problem of scheduling jobs on uniformly related parallel machines with the objective of minimizing the total weighted completion time. The quality of locally optimal solutions under the jump neighborhood is analyzed, which consists of iteratively moving a single job from one machine to another, improving the total weighted completion time in each iteration and stopping once improvement is no longer possible. We propose an upper bound for the total weighted completion time for the solutions obtained by this local search, and upper and lower bounds for the performance guarantee of the obtained locally optimal solutions. Additionally, the case of identical parallel machines is analyzed.

Mathematics Subject Classification. 90B35, 90C59, 68M20, 68W40.

Received December 8, 2021. Accepted March 18, 2022.

1. Introduction

Local search methods are widely used to solve scheduling problems and exhibit good empirical behavior, but little is known about their theoretical worst-case performance. The reader is referred to [3, 24] for a survey of performance guarantees and other theoretical aspects of local search for a wide variety of combinatorial problems, including scheduling problems.

The problem of scheduling jobs on uniformly related parallel machines is considered with the objective of minimizing the total weighted completion time. This problem is denoted by $Q||\sum w_jC_j$ in the three-field scheduling notation [19]. More precisely, we are given a set of n jobs to be scheduled on a set of m machines. The jobs must be scheduled without interruption on a single machine, and each machine can process one job at a time. Each job has a non-negative weight and a non-negative processing requirement.

The solution called schedule, is an assignment of jobs to machines together with a sequence of the jobs within each machine. Here, it is considered that the sequence within a machine is always taken to be the decreasing order of weight to processing time requirement ratio, also known as the Smith ratio [28]. Therefore, 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 which the *total weighted completion time* can be computed by $\sum_j w_jC_j$, where w_j is the weight of job j. If jobs have unitary weight, the problem consists in minimizing the total completion time ($\sum_j C_j$).

Keywords. Parallel machines, total weighted completion time, local search, performance guarantee.

[1] Department of Industrial Engineering, Universidad del Bío-Bío, Concepción, Chile.
[2] Faculty of Forestry Sciences, Universidad de Concepción, Concepción, Chile.
*Corresponding author: fmunoz@ubiobio.cl

Two problems have particular interest for our purpose. The first being the particular case where all machines work at the same speed, denoted by $P||\sum w_jC_j$. The second one is the most general case where the jobs have arbitrary process times, denoted by $R||\sum w_jC_j$.

Minimizing the total completion time on parallel machine environments can be solved in polynomial time. The problem $R||\sum C_j$, can be solved by bipartite matching techniques [20]. The problem $Q||\sum C_j$ can be solved in $O(n \log nm)$ [11,21]. For the problem $P||\sum C_j$, complexity decreases to $O(n \log n)$ [11]. If the objective is to minimize the total weighted completion time, the problems are NP-hard [18], even for the case of two identical machines [8,22].

Since for the total weighted completion time objective the problem becomes NP-hard, it is natural to look for approximate solutions. Polynomial time approximation schemes (PTAS) for problems $P||\sum w_jC_j$ and $Q||\sum w_jC_j$ are studied in [14,27] respectively. While the best approximation reported for $R||\sum w_jC_j$ problem is $3/2 - c$, for $c > 1/6000$ [4,23]. Another approach, vastly implemented in practice and the main focus of this paper is to use local search techniques. There are two important aspects that determine the efficiency of a local search algorithm: neighborhood size and local optimum quality [2]. One way to evaluate the quality of a local optimum is by worst-case analysis. Specifically, we study the *Jump* neighborhood, also known as *move* or *insertion*, which is defined as moving a single job from one machine to another.

Given a problem $\mathcal{P}$ and a neighborhood structure $\mathcal{N}$, the performance guarantee of a minimization criterion is defined as the maximum possible ratio of a local optimal solution with respect to the global optimum. More precisely, the performance guarantee can be formally defined by:

$$pg(\mathcal{P},\mathcal{N}) = \sup_{k\in\mathcal{I}} \sup_{\sigma\in\mathcal{L}_k} \left\{ \frac{cost(\sigma)}{opt(k)} \right\},$$

where $\mathcal{I}$ is the set of instances of the problem $\mathcal{P}$, $\mathcal{L}_k$ is the set of locally optimal solutions of instance k for neighborhood $\mathcal{N}$, $cost(\sigma)$ is the cost of solution σ (objective function), and $opt(k)$ is the optimal cost for the instance k. To determine the performance guarantee for a jump neighborhood, an upper bound is established and then an instance is proposed to establish a lower bound for the performance guarantee.

The study of performance guarantees for locally optimal solutions under a jump neighborhood on parallel machine environments is more extensive for the makespan objective than the total weighted completion time objective. The performance guarantees for the makespan objective has been studied for identical parallel machines [7,15,26], uniformly related parallel machines [9,26] and unrelated parallel machines [26]. The performance guarantees considering machine eligibility restrictions for identical and uniformly unrelated parallel machines was studied in [25].

For the total weighted completion time objective, the performance guarantees for the $P||\sum w_jC_j$ problem has been studied in [6], and it was determined that the performance guarantees are at least 1.2 and at most $(3m-1)/(2m)$. The $R||\sum w_jC_j$ problem was studied in [1,12] , establishing that the performance guarantee is at most 2.618. The performance guarantees considering machine eligibility restrictions for identical and uniformly unrelated parallel machines was studied in [12]. In [5,26] several related problems are studied.

Other neighborhoods that have been used for scheduling in parallel machine environments include *lexjump*, *push*, *multi-exchange* and *split*. Lexjump is a polynomial size neighborhood and is an extension of the jump neighborhood used for minimizing the makespan [7,25,26]. Push is also a polynomial size neighborhood, which was introduced in [26] for the minimization of makespan on identical and uniformly related parallel machine environments. Multi-exchange is an exponential size neighborhood introduced in [16] for unrelated parallel machine environments, and then used for other parallel machine environments in [26]. Split is another exponential size neighborhood introduced in [7] for the minimization of makespan on identical parallel machines. Combined Jump neighborhood with other neighborhoods was studied in [7].

In [1,6,12] the proof techniques used consist of establishing a local optimality condition and then using certain mathematical properties to establish the bound for the performance guarantee. In [1,12] a mapping proposed by Cole *et al.* [10] is used. With this mapping, a schedule can be mapped into an inner product space so that the norm of the mapping is closely related to the total weighted completion time of the schedule. On the other

hand, [6] uses an inequality proposed by Eastman *et al.* [13]. This inequality is presented in Section 2.3. Our proof technique is similar to the one used in [6].

Our results. It is proved that the performance guarantee for locally optimal solutions under jump neighborhood for $Q||\sum w_jC_j$ problem is at least 1.42285 and at most

$$\min\left\{2\left(1-\frac{2}{m+2}\right)\left(\frac{s_{\max}}{s_{\min}}+\frac{1}{2m}\right),2.618\right\}.$$

Additionally, it is proved that the performance guarantee for $P||\sum w_jC_j$ is at most $(3/2)-(1/2m)$. This bound was known from [6]. However, our proof treats the identical parallel machines case as a particular case of the uniformly related parallel machines. We also prove some inequalities for the total weighted completion time of optimal and locally optimal solutions.

The remainder of this paper is structured as follows. We present preliminary background in Section 2. All aspects related to the upper bound for the performance guarantee are included in Section 3. A lower bound for the performance guarantee is presented in Section 4. Finally, Section 5 provides the conclusions.

2. Preliminaries

2.1. Notation and problem statement

Throughout this paper, let $\mathcal{J}$ be the set of n jobs to be scheduled on a set $\mathcal{M}$ of $m \geq 2$ machines. Let p_j and w_j denote the non-negative processing requirement and weight of job $j \in \mathcal{J}$ respectively. Jobs must be scheduled on a single machine, and each machine can process one job at a time without preemptions. In addition all jobs and machines are available from the beginning.

Machines have different speeds, therefore, let $\boldsymbol{s}$ be the vector for the speeds, with $s_i > 0$ as the speed of machine i. Then, the processing time for job j on machine i is p_j/s_i. From vector $\boldsymbol{s}$, the maximum and minimum speeds are identified, $s_{\max}$ and $s_{\min}$ respectively. Without loss of generality, the speed and order of the machines is rescaled such that

$$1 = s_1 \leq s_2 \leq s_3 \leq \cdots \leq s_m = \frac{s_{\max}}{s_{\min}}. \tag{2.1}$$

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 $\boldsymbol{x}$. As a result of a schedule, each job $j \in \mathcal{J}$ will have a completion time with which the total weighted completion time is computed. Let $C_j(\boldsymbol{x},\boldsymbol{s})$ the completion time of job j in schedule $\boldsymbol{x}$ at speeds $\boldsymbol{s}$. The sequencing of jobs for any schedule is solved by the weighted shortest processing time (WSPT) rule ([28], Thm. 3) which simply sequences jobs in decreasing order of w_j/p_j ratio. When there are ties in the ratio, these are broken arbitrarily and to avoid confusion we denote by $\prec$ the precedence relation of jobs. 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

$$C_j(\boldsymbol{x},\boldsymbol{s}) = \frac{p_j}{s_{x_j}} + \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\prec j}} \frac{p_k}{s_{x_j}} = \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\preceq j}} \frac{p_k}{s_{x_j}}.$$

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

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

$$C(\boldsymbol{x},\boldsymbol{s}) = \sum_{j\in\mathcal{J}} w_jC_j(\boldsymbol{x},\boldsymbol{s}) = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}_i(\boldsymbol{x})} w_jC_j(\boldsymbol{x},\boldsymbol{s}), \tag{2.2}$$

$$\eta(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} \frac{w_j p_j}{s_{x_j}} = \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \frac{w_j p_j}{s_i}. \tag{2.3}$$

With the previous definitions, the following identities, (2.4) and (2.5), are immediate:

$$\begin{aligned} C(\boldsymbol{x},\boldsymbol{s}) &= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\preceq j}} \frac{w_j p_k}{s_i} = \eta(\boldsymbol{x}) + \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\prec j}} \frac{w_j p_k}{s_i} \\ &= \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\prec j}} \frac{w_j p_k}{s_{x_j}}, \end{aligned} \tag{2.4}$$

$$\begin{aligned} C(\boldsymbol{x},\boldsymbol{s}) &= \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\succeq j}} \frac{w_k p_j}{s_i} = \eta(\boldsymbol{x}) + \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}_i(\boldsymbol{x})} \sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{x})\\ k\succ j}} \frac{w_k p_j}{s_i} \\ &= \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\succ j}} \frac{w_k p_j}{s_{x_j}}. \end{aligned} \tag{2.5}$$

2.2. Jump neighborhood

We study the *Jump* neighborhood, also known as *move* or *insertion* neighborhood, which is a polynomial size 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 in this way, we have a local optimum and call this solution *Jump-Opt*. More precisely, let $\delta_j(\boldsymbol{x})$ be the amount by which $C(\boldsymbol{x},\boldsymbol{s})$ decreases if job j is removed from machine x_j, and $\delta'_j(\boldsymbol{x},h)$ the increment in $C(\boldsymbol{x},\boldsymbol{s})$ if job j is moved to machine h. Observe that job j has to be inserted on machine h at the appropriate position (defined by WSPT rule). Thus,

$$\begin{aligned} \delta_j(\boldsymbol{x}) &= w_j C_j(\boldsymbol{x},\boldsymbol{s}) + \sum_{\substack{k\in\mathcal{J}_{x_j}(\boldsymbol{x})\\ k\succ j}} \frac{w_k p_j}{s_{x_j}}, \\ \delta'_j(\boldsymbol{x},h) &= \frac{w_j p_j}{s_h} + \sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\prec j}} \frac{w_j p_k}{s_h} + \sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\succ j}} \frac{w_k p_j}{s_h}. \end{aligned}$$

Therefore, schedule $\boldsymbol{x}$ will be a Jump-Opt schedule for $Q||\sum w_j C_j$ problem, if and only if

$$\delta_j(\boldsymbol{x}) \le \delta'_j(\boldsymbol{x},h) \quad \forall j\in\mathcal{J}, \quad h\in\mathcal{M}\setminus\{x_j\}. \tag{2.6}$$

2.3. Properties of the optimal schedules

In this section, two properties of the optimal schedules are presented that will be used to establish the upper bounds for the performance guarantees of Jump-Opt solutions. The first one is the inequality presented in Theorem 1 of [13], this is

$$Z \le mC(\tilde{\boldsymbol{x}},\mathbf{1}) - \frac{(m-1)}{2}\sum_{j\in\mathcal{J}} w_j p_j, \tag{2.7}$$

where $\tilde{\boldsymbol{x}}$ is the optimal solution if all machines have unitary speed, $C(\tilde{\boldsymbol{x}},\mathbf{1})$ is the total weighted completion time of $\tilde{\boldsymbol{x}}$ at unitary speed machines, and

$$Z = \sum_{j\in\mathcal{J}} \sum_{\substack{k\in\mathcal{J}\\ k\preceq j}} w_j p_k = \sum_{j\in\mathcal{J}} \sum_{\substack{k\in\mathcal{J}\\ k\succeq j}} w_k p_j, \tag{2.8}$$

is the total weighted completion time, if all jobs are assigned to a single machine that works at unitary speed.

The second inequality is presented in the following lemma.

Lemma 2.1. *Given a set of jobs, the optimal total weighted completion time on identical and uniformly related parallel machines at speeds $\boldsymbol{s}$, satisfy*

$$C(\tilde{\boldsymbol{x}}, \mathbf{1}) \leq \left(\frac{s_{\max}}{s_{\min}}\right) C(\boldsymbol{x}^*, \boldsymbol{s}).$$

Proof. First, note that if $\tilde{\boldsymbol{x}}$ is the optimal solution when all machines have unitary speed, and $\boldsymbol{x}^*$ is the optimal solution if machines work at speeds $\boldsymbol{s}$, then

$$C(\tilde{\boldsymbol{x}}, \mathbf{1}) \leq C(\boldsymbol{x}^*, \mathbf{1}). \tag{2.9}$$

Moreover, from Assumption (2.1) and (2.4), we have

$$C(\boldsymbol{x}^*, \boldsymbol{s}) = \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}_i(\boldsymbol{x}^*)} \sum_{\substack{k \in \mathcal{J}_i(\boldsymbol{x}^*) \\ k \preceq j}} \frac{w_j p_k}{s_i} \geq \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}_i(\boldsymbol{x}^*)} \sum_{\substack{k \in \mathcal{J}_i(\boldsymbol{x}^*) \\ k \preceq j}} \frac{w_j p_k}{s_m} = \frac{C(\boldsymbol{x}^*, \mathbf{1})}{s_m}.$$

Therefore,

$$C(\boldsymbol{x}^*, \mathbf{1}) \leq s_m C(\boldsymbol{x}^*, \boldsymbol{s}) = \left(\frac{s_{\max}}{s_{\min}}\right) C(\boldsymbol{x}^*, \boldsymbol{s}).$$

Finally, using (2.9) we conclude the proof. □

3. Performance guarantee

3.1. Uniformly related machines

In this section a parametric upper bound is established for the total weighted completion time of Jump-Opt solutions for $Q||\sum w_j C_j$ problem, and an upper bound is established for performance guarantee.

Lemma 3.1. *For any Jump-Opt schedule $\boldsymbol{x}$ of $Q||\sum w_j C_j$, it holds that*

$$C(\boldsymbol{x}, \boldsymbol{s}) \leq \left(\frac{s_{\max}}{s_{\min}}\right) C(\boldsymbol{x}^*, \boldsymbol{s}) + \frac{(m-2)}{2m}\eta(\boldsymbol{x}) + \frac{1}{2m}\eta(\boldsymbol{x}^*).$$

Proof. From local optimality condition (2.6), we have

$$w_j C_j(\boldsymbol{x}, \boldsymbol{s}) + \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{w_k p_j}{s_{x_j}} \leq \frac{w_j p_j}{s_h} + \sum_{\substack{k \in \mathcal{J}_h(\boldsymbol{x}) \\ k \prec j}} \frac{w_j p_k}{s_h} + \sum_{\substack{k \in \mathcal{J}_h(\boldsymbol{x}) \\ k \succ j}} \frac{w_k p_j}{s_h}, \quad \forall j \in \mathcal{J}, \quad h \in \mathcal{M} \setminus \{x_j\}.$$

Summing over all $h \in \mathcal{M} \setminus \{x_j\}$, gives

$$(m-1) w_j C_j(\boldsymbol{x}, \boldsymbol{s}) + (m-1) \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{w_k p_j}{s_{x_j}} \leq \sum_{h \in \mathcal{M}} \frac{w_j p_j}{s_h} - \frac{w_j p_j}{s_{x_j}} + \sum_{h \in \mathcal{M}} \sum_{\substack{k \in \mathcal{J}_h(\boldsymbol{x}) \\ k \prec j}} \frac{w_j p_k}{s_h} - \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \prec j}} \frac{w_j p_k}{s_{x_j}} + \sum_{h \in \mathcal{M}} \sum_{\substack{k \in \mathcal{J}_h(\boldsymbol{x}) \\ k \succ j}} \frac{w_k p_j}{s_h} - \sum_{\substack{k \in \mathcal{J}_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{w_k p_j}{s_{x_j}}.$$

Summing over all $j \in \mathcal{J}$, using (2.4) and (2.5), and grouping some terms, gives us

$$2mC(\boldsymbol{x},\boldsymbol{s}) \leq m\eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\frac{w_jp_j}{s_h} + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\prec j}}\frac{w_jp_k}{s_h} + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\succ j}}\frac{w_kp_j}{s_h}. \tag{3.1}$$

On the other hand, by using (2.1) and (2.8) we have the following three results:

$$\sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\frac{w_jp_j}{s_h} = \sum_{j\in\mathcal{J}}\frac{w_jp_j}{s_{x_j^*}} + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}\setminus\{x_j^*\}}\frac{w_jp_j}{s_h} = \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}\setminus\{x_j^*\}}\frac{w_jp_j}{s_h}$$
$$\leq \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}\setminus\{x_j\}} w_jp_j = \eta(\boldsymbol{x}^*) + (m-1)\sum_{j\in\mathcal{J}} w_jp_j, \tag{3.2}$$

$$\sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\prec j}}\frac{w_jp_k}{s_h} = \sum_{j\in\mathcal{J}}\sum_{\substack{k\in\mathcal{J}\\ k\prec j}}\frac{w_jp_k}{s_{x_k}} \leq \sum_{j\in\mathcal{J}}\sum_{\substack{k\in\mathcal{J}\\ k\prec j}} w_jp_k = Z - \sum_{j\in\mathcal{J}} w_jp_j, \tag{3.3}$$

$$\sum_{j\in\mathcal{J}}\sum_{h\in\mathcal{M}}\sum_{\substack{k\in\mathcal{J}_h(\boldsymbol{x})\\ k\succ j}}\frac{w_kp_j}{s_h} = \sum_{j\in\mathcal{J}}\sum_{\substack{k\in\mathcal{J}\\ k\succ j}}\frac{w_kp_j}{s_{x_k}} \leq \sum_{j\in\mathcal{J}}\sum_{\substack{k\in\mathcal{J}\\ k\succ j}} w_kp_j = Z - \sum_{j\in\mathcal{J}} w_jp_j. \tag{3.4}$$

Combining (3.2), (3.3) and (3.4) in (3.1), gives us

$$2mC(\boldsymbol{x},\boldsymbol{s}) \leq m\eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + (m-3)\sum_{j\in\mathcal{J}} w_jp_j + 2Z.$$

By (2.7), we have

$$2mC(\boldsymbol{x},\boldsymbol{s}) \leq m\eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) - 2\sum_{j\in\mathcal{J}} w_jp_j + 2mC(\tilde{\boldsymbol{x}},\boldsymbol{1}).$$

From (2.1), we have $s_i \geq 1$ for all $i \in \mathcal{M}$, then $\sum_{j\in\mathcal{J}} w_jp_j \geq \sum_{j\in\mathcal{J}} \frac{w_jp_j}{s_{x_j}} = \eta(\boldsymbol{x})$. Thus,

$$2mC(\boldsymbol{x},\boldsymbol{s}) \leq (m-2)\eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + 2mC(\tilde{\boldsymbol{x}},\boldsymbol{1}).$$

Note that in any parallel machine environment it is true that $m > 1$. Finally, using Lemma 2.1 we can conclude the proof. □

With the upper bound presented in Lemma 3.1, we can establish the following upper bound for the performance guarantee of Jump-Opt solutions for the $Q||\sum w_jC_j$ problem.

Lemma 3.2. *An upper bound for the performance guarantee of Jump-Opt solutions for the $Q||\sum w_jC_j$ problem is*

$$2\left(1 - \frac{2}{m+2}\right)\left(\frac{s_{\max}}{s_{\min}} + \frac{1}{2m}\right).$$

Proof. Note that $m \geq 2$ in any parallel machine environments. From (2.4), we have $\eta(\boldsymbol{x}) \leq C(\boldsymbol{x},\boldsymbol{s})$ and $\eta(\boldsymbol{x}^*) \leq C(\boldsymbol{x}^*,\boldsymbol{s})$. Then, using Lemma 3.1 we have

$$\left(1 - \frac{(m-2)}{2m}\right)C(\boldsymbol{x},\boldsymbol{s}) \leq \left(\frac{s_{\max}}{s_{\min}} + \frac{1}{2m}\right)C(\boldsymbol{x}^*,\boldsymbol{s}).$$

From here, the proof is immediate. □

Here, our main result is presented, which is based on the performance guarantee of Jump-Opt solutions of $R||\sum w_jC_j$ problem determined in [12] and the Lemma 3.2. Note that the performance guarantee of Jump-Opt solutions for $R||\sum w_jC_j$ can be considered as an upper bound for the performance guarantee of Jump-Opt solutions for $Q||\sum w_jC_j$. From Theorems 1 and 5 of [12] we have that the performance guarantee for $R||\sum w_jC_j$ is $1+\phi \approx 2.618$, where ϕ is the golden ratio.

Theorem 3.3. *The performance guarantee of Jump-Opt solutions for $Q||\sum w_jC_j$ problem is at most*

$$\min\left\{2\left(1-\frac{2}{m+2}\right)\left(\frac{s_{\max}}{s_{\min}}+\frac{1}{2m}\right), 2.618\right\}.$$

From the previous theorem, we have that the upper bound proposed in the Lemma 3.2 will be useful if and only if

$$\frac{s_{\max}}{s_{\min}} \leq \frac{\phi^2}{2}+\frac{1}{m}\left(\phi+\frac{1}{2}\right).$$

For two machines this bound is approximately 2.368, and when m grows this bound decreases monotonically and converges to 1.309.

3.2. Identical machines

In this section, an upper bound is presented for the performance guarantee of Jump-Opt solutions for the problem of scheduling jobs on identical parallel machines with a total weighted completion time objective. This problem is denoted by $P||\sum w_jC_j$ in the three-field scheduling notation [19].

For this case, we have that the machines have the same speed. Without loss of generality, it is assumed that this speed is unitary. Here, the following notations are redefined:

$$\begin{aligned} C(\boldsymbol{x}^*) &= C(\boldsymbol{x}^*, \mathbf{1}),\\ C(\boldsymbol{x}) &= C(\boldsymbol{x}, \mathbf{1}),\\ \eta = \eta(\boldsymbol{x}) &= \eta(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_jp_j. \end{aligned}$$

Then, from (2.4) we can determine the cost of any schedule $\boldsymbol{z}$ as

$$C(\boldsymbol{z}) = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}_i(\boldsymbol{z})}\sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{z})\\ k\preceq j}} w_jp_k = \eta + \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}_i(\boldsymbol{z})}\sum_{\substack{k\in\mathcal{J}_i(\boldsymbol{z})\\ k\prec j}} w_jp_k. \tag{3.5}$$

Theorem 3.4. *The performance guarantee of Jump-Opt solutions for $P||\sum w_jC_j$ is at most $\frac{3}{2}-\frac{1}{2m}$.*

Proof. For this problem we can rewrite Lemma 3.1 as,

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

From (3.5), we have $\eta \leq C(\boldsymbol{x}^*)$. Then, $\dfrac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} \leq \left(\dfrac{3m-1}{2m}\right)$. □

This upper bound for the performance guarantee was previously determined by [6], who also show a lower bound of 1.2.

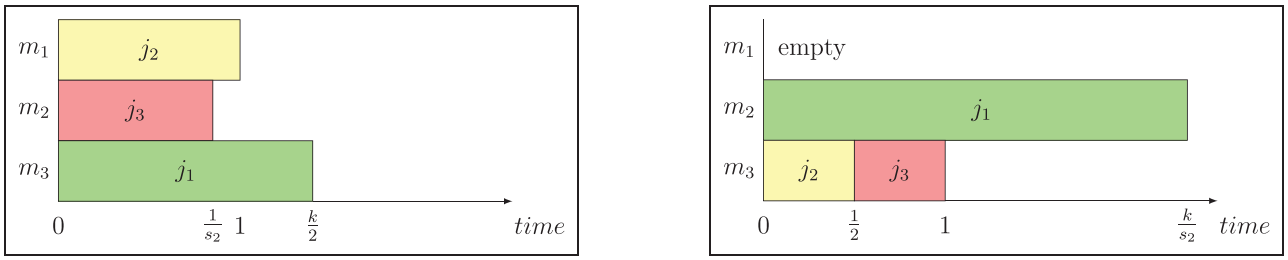

FIGURE 1. Gantt chart for optimal and Jump-Opt solutions. (A) Optimal solution. (B) Jump-Opt solution.

4. Lower bound

In order to establish a lower bound for the performance guarantee of Jump-Opt solutions, we propose the following instance.

Lemma 4.1. *The performance guarantee of Jump-Opt solutions for* $Q||\sum w_jC_j$ *is at least* 1.423.

Proof. An instance $\mathcal{I}$ is considered with $n = 3$ jobs and $m = 3$ machines. Jobs have identical weight and processing requirements, $p_1 = w_1 = k$ and $p_j = w_j = 1$ for $j = 2, 3$. The processing speed of the machines are $s_1 = 1$, $s_2 = \frac{2k}{k+2}$ and $s_3 = 2$.

The optimal and Jump-Opt solutions are $\boldsymbol{x}^* = (3, 1, 2)$ and $\boldsymbol{x} = (2, 3, 3)$ respectively (Fig. 1 shows a Gantt chart for $\boldsymbol{x}^*$ and $\boldsymbol{x}$). For $\boldsymbol{x}$, the local optimality conditions from (2.6) are reduced to $k \geq 2$. Then for $k \geq 2$ it is not possible to make jump moves without increasing the total weighted completion time. The total weighted completion time for $\boldsymbol{x}^*$ and $\boldsymbol{x}$ are

$$C(\boldsymbol{x}^*, \boldsymbol{s}) = 1 + \frac{1}{s_2} + \frac{k^2}{2} = \frac{k^3 + 3k + 2}{2k}, \quad \text{and}$$
$$C(\boldsymbol{x}, \boldsymbol{s}) = \frac{k^2}{s_2} + \frac{1}{2} + \frac{2}{2} = \frac{k^3 + 2k^2 + 3k}{2k}.$$

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

$$\frac{C(\boldsymbol{x}, \boldsymbol{s})}{C(\boldsymbol{x}^*, \boldsymbol{s})} = \frac{k^3 + 2k^2 + 3k}{k^3 + 3k + 2}.$$

Therefore, to determine the bound it is necessary to solve the problem

$$\operatorname*{argmax}_{k \geq 2} \left\{ \frac{k^3 + 2k^2 + 3k}{k^3 + 3k + 2} \right\}.$$

Finally, $k^* = 2 + \frac{9\sqrt{2}}{16}$ is an approximation to the solution of the problem, where $\frac{C(\boldsymbol{x}, \boldsymbol{s})}{C(\boldsymbol{x}^*, \boldsymbol{s})} \approx 1.423$. □

5. Conclusion

In this paper we establish an upper bound on the performance guarantee for the Jump neighborhood in uniformly related parallel machines environment with the objective of minimizing the total weighted completion

time. The rough idea is to first establish a necessary condition for local optimality that takes the form of a certain inequality. Then we apply some inequalities to obtain the final guarantee. We also propose an instance to establish a lower bound for the performance guarantee.

Acknowledgements. The authors gratefully acknowledge to Francisco Ramis and Centro Avanzado de Simulación de Procesos (CASP) of University of Bío-Bío. They also thank the editor-in-chief, the area editor, and the anonymous reviewers for their valuable comments on the earlier version of this manuscript.

References

[1] F. Abed, J.R. Correa and CC. Huang, Optimal coordination mechanisms for multi-job scheduling games. In: *ESA* (2014).

[2] R. Ahuja, O. Ergun, J. Orlin and A. Punnen, A survey of very large-scale neighborhood search techniques. *Disc. Appl. Math.* **123** (2002) 75–102.

[3] E. Angel, A survey of approximation results for local search algorithms. In: Efficient Approximation and Online Algorithms, edited by E. Bampis, K. Jansen and C. Kenyon. Springer (2006) 30–73.

[4] N. Bansal, A. Srinivasan and O. Svensson, Lift-and-round to improve weighted completion time on unrelated machines. In: *STOC* (2016).

[5] P. Brucker, J. Hurink and F. Werner, Improving local search heuristics for some scheduling problems. Part II. *Disc. Appl. Math.* **72** (1997) 47–69.

[6] T. Brueggemann, J.L. Hurink and W. Kern, Quality of move-optimal schedules for minimizing total weighted completion time. *Oper. Res. Lett.* **34** (2006) 583–590.

[7] T. Brueggemann, J.L. Hurink, T. Vredeveld and G.J. Woeginger, Exponential size neighborhoods for makespan minimization scheduling. *Naval Res. Logist.* **58** (2011) 795–803.

[8] J. Bruno, E.G. Coffman and R. Sethi, Scheduling independent tasks to reduce mean finishing time. *Commun. ACM* **17** (1974) 382–387.

[9] Y. Cho and S. Sahni, Bounds for list schedules on uniform processors. *SIAM J. Comput.* **9** (1980) 91–103.

[10] R. Cole, J.R. Correa, V. Gkatzelis, V. Mirrokni and N. Olver, Decentralized utilitarian mechanisms for scheduling games. *Games Econ. Behav.* **92** (2015) 306–326.

[11] R.W. Conway, W.L. Maxwell and L.W. Miller, Theory of Scheduling. Addison-Wesley (1967).

[12] J.R. Correa and F.T. Muñoz, Performance guarantees of local search for minsum scheduling problems. *Math. Program.* (2020). https://doi.org/10.1007/s10107-020-01571-5.

[13] W.L. Eastman, S. Even and I.M. Isaacs, Bounds for the optimal scheduling of n jobs on m processors. *Manage Sci.* **11** (1964) 268–279.

[14] L. Epstein and J. Sgall, Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica* **39** (2004) 43–57.

[15] G. Finn and E. Horowitz, A linear time approximation algorithm for multiprocessor scheduling. *BIT Numer. Math.* **19** (1979) 312–320.

[16] A. Frangioni, E. Necciari and M.G. Scutellà, A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. *J. Comb. Optim.* **8** (2004) 195–220.

[17] M. Garey and D. Johnson, Strong NP-completeness results: Motivation, examples, and implications. *J. ACM* **25** (1978) 499–508.

[18] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman and Co., San Francisco (1979).

[19] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Disc. Math.* **5** (1979) 287–326.

[20] W.A. Horn, Minimizing average flow time with parallel machines. *Oper. Res.* **21** (1973) 846–847.

[21] E. Horowitz and S. Sahni, Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM* **23** (1976) 317–327.

[22] J.K. Lenstra, A.H.G. Rinooy Kan and P. Brucker, Complexity of machine scheduling problems. *Ann. Disc. Math.* **1** (1977) 343–362.

[23] S. Li, Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations. In: *FOCS* (2017).

[24] W. Michiels, E. Aarts and J. Korst, Theoretical Aspects of Local Search. Springer Science & Business Media, Berlin (2007).

[25] C. Rutten, D. Recalde, P. Schuurman and T. Vredeveld, Performance guarantees of jump neighborhoods on restricted related parallel machines. *Oper. Res. Lett.* **40** (2012) 287–291.

[26] P. Schuurman and T. Vredeveld, Performance guarantees of local search for multiprocessor scheduling. *INFORMS J. Comput.* **19** (2007) 52–63.

[27] M. Skutella and G.J. Woeginger, A PTAS for minimizing the total weighted completion time on identical parallel machines. *Math. Oper. Res.* **25** (2000) 63–75.
[28] W.E. Smith, Various optimizers for single-stage production. *Naval Res. Logist.* **3** (1956) 59–66.