Theoretical Computer Science 416 (2012) 87–94

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

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

A linear time approximation algorithm for permutation flow shop scheduling☆

Marcus Poggi *, David Sotelo

Departamento de Informática, PUC-Rio, Rua Marquês de São Vicente, 225 RDC, CEP 22451-900, RJ, Brazil

ARTICLE INFO

Article history:
Received 25 May 2009
Received in revised form 14 October 2011
Accepted 20 October 2011
Communicated by G. Ausiello

Keywords:
Permutation flow shop scheduling
Approximation algorithms
Erdös–Szekeres theorem

ABSTRACT

In the last 40 years, the permutation flow shop scheduling (PFS) problem with makespan minimization has been a central problem, known for its intractability, that has been well studied from both theoretical and practical aspects. The currently best performance ratio of a deterministic approximation algorithm for the PFS was recently presented by Nagarajan and Sviridenko, using a connection between the PFS and the longest increasing subsequence problem. In a different and independent way, this paper employs monotone subsequences in the approximation analysis techniques. To do this, an extension of the Erdös–Szekeres theorem to weighted monotone subsequences is presented. The result is a simple deterministic algorithm for the PFS with a similar approximation guarantee, but a much lower time complexity.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

1.1. Problem definition

Permutation flow shop scheduling is a production planning process consisting of a set $J = \{J_1, J_2, \ldots, J_n\}$ of n jobs to be executed in a set $M = \{M_1, M_2, \ldots, M_m\}$ of m machines. In this process, every job J_j is composed by m stages $O_{1,j}, O_{2,j}, \ldots, O_{m,j}$, named *operations*. Every operation $O_{i,j}$ has a non-negative processing time t_{ij} composing the matrix $T \in \Re^{+}_{M\times J}$. The job operation $O_{i,j}$ must be only executed on machine i. A machine cannot execute more than one operation per time. Operation $O_{i,j}$ can be executed only after operation $O_{i-1,j}$ have finished. Preemption is not allowed; i.e., once an operation is started, it must be completed without interruption. All jobs must be executed in the same order on every machine, defined by a permutation $\pi : \{1, \ldots, n\} \mapsto J$, with $\pi(i)$ indicating the i-th job to be executed. The completion time of an operation $O_{i,j}$, denoted by $C_{i,j}$, is defined by the recurrence:

$$C_{i,\pi(j)} = \begin{cases} t_{i\pi(j)} & \text{if } i = 1 \text{ and } j = 1 \\ C_{i,\pi(j-1)} + t_{i\pi(j)} & \text{if } i = 1 \text{ and } j > 1 \\ C_{i-1,\pi(j)} + t_{i\pi(j)} & \text{if } i > 1 \text{ and } j = 1 \\ \max(C_{i,\pi(j-1)}, C_{i-1,\pi(j)}) + t_{i\pi(j)} & \text{if } i > 1 \text{ and } j > 1. \end{cases}$$

The completion time of a job J_j is $C_{m,j}$. The makespan of a permutation is the maximum completion time of a job. The objective of the permutation flow shop scheduling problem (PFS) is to find a permutation π that minimizes the makespan.

☆ A short extended abstract appeared in MAPSP 2009.

* Corresponding author. Tel.: +55 21 35271500, +55 21 35274339 (Office); fax: +55 21 35271530.
E-mail addresses: poggi@inf.puc-rio.br (M. Poggi), dsilva@inf.puc-rio.br (D. Sotelo).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.10.013

1.2. Previous results

In the last 40 years, the PFS has been a central problem in scheduling and operations research communities; it is known for its intractability, and has been well studied from both theoretical and practical aspects. The PFS was proved strongly NP-hard by Garey et al. [4] for instances with three or more machines. In a seminal paper, Johnson [8] presented a polynomial time algorithm for instances with two machines. Gonzalez and Sahni [6] showed that every busy scheduling for the PFS has an approximation factor of m times the optimal solution.

Nowicki and Smutnicki [11–13] explored worst-case analysis on the approximation factor of several PFS algorithms, achieving a tight bound of $\lceil m/2 \rceil$. Potts et al. [14] proved the existence of some instances for which non-permutation-based solutions are $\Omega(\sqrt{m})$ less costly than permutation-based ones. Sevast'janov [18] developed geometric methods to analyze scheduling problems, including the PFS, introducing an algorithm that always produces a schedule with additive factor bounded by $O(m^2) \max\{t_{ij}\}$. Hall [7] presented a polynomial time approximation scheme (PTAS) for the PFS when $m = 3$. Sviridenko [20] presented a randomized algorithm based on Chernoff bounds arguments with a performance ratio of $O(\sqrt{m \log m})$ and additive factor limited to $O(m \log m) \max\{t_{ij}\}$. Approximation ratios for a large number of PFS heuristics were surveyed by Gupta et al. [3].

The best known approximation algorithm for the PFS to date, due to Nagarajan and Sviridenko [9], has a performance ratio of $O(\sqrt{\min\{n, m\}})$, and is based on a connection between the PFS and the longest increasing subsequence problem. This reduction allows proving that a random permutation achieves the claimed approximation guarantee. A corresponding deterministic algorithm is obtained further using the method of pessimistic estimators [15]. Finally, this algorithm resolves an open question from [14] matching the gap between optimal solutions for permutation and non-permutation schedules. Due to the relevance of the approximation factor obtained, the time complexity analysis of the deterministic approximation algorithm from [9] is not explicitly given in the paper. In the Appendix, a lower bound of $\Omega(n^4.m)$ is established.

1.3. This paper's contribution and organization

The purpose of this work is to present a simple and intuitive deterministic approximation algorithm for the PFS with performance ratio $2\sqrt{2n + m}$ and time complexity $\Theta(nm)$. These results were achieved independently [16] and in parallel with those of Nagarajan and Sviridenko, as mentioned in their published paper [10] (Section 1.2). When $n = \Theta(m)$, this is the best approximation algorithm already obtained for the PFS, achieving the same approximation factor as in [9] in linear time complexity. A novel technique for performance guarantee analysis of PFS solutions is also developed, exploring the correlation between weighted monotone subsequence problems and the PFS. We prove that a job permutation reaching the claimed approximation guarantee has the cells representing the longest operation of each job constituting a specific path on the processing time matrix. Furthermore, a permutation with this property can be obtained by a simple job sorting algorithm. Its approximation factor analysis is based on deterministic combinatorial arguments related to Erdös–Szekeres theorem extensions. The results presented on these extensions are further used to provide the lower bounds on the optimal value.

The main idea of this work involves the exploration of the PFS by a new perspective related to matrix games and monotone subsequences. The PFS is viewed as an equivalent, but more intuitive, matrix game problem between two players. In this game, the first player selects a permutation over the columns of an original matrix, creating a new matrix, and the second player tries to select a sequence of cells in such modified matrix with the maximum sum possible. The sequence of cells selected by player 2 must follow a specific property, composing what we call a path. The objective of player 1 is to find a permutation that makes the task of player 2 difficult. We argue that this game, with the objective of acting as player 1, is equivalent to that of the PFS.

Once the problem is defined, a natural strategy for player 1 is to choose a permutation that makes the choice of path of player 2 avoid as much as possible the cells of high weight. An approximation algorithm with such property, based on a simple ordering of jobs, is therefore presented. One of the main contributions in this work comes from the technique used to analyze the approximation guarantee of this algorithm. We prove that it is possible to obtain upper bounds on the approximation factor of the presented algorithm by a reduction of the matrix game problem into the *minimum double weighted sequence problem*. This new problem is a generalization of the classical problem considered by Erdös and Szekeres [2], in which every sequence element has two associated weights, one if it is considered in increasing subsequences and another if it is considered in decreasing ones. The objective is to define a sequence that minimizes the maximum weight of its increasing or decreasing subsequences. We provide extensions of the Erdös–Szekeres theorem to the case of weighted sequences, first considering the case in which every element has the same weight in increasing or decreasing subsequences and then generalizing it to the case of distinct weights.

This paper is organized as follows. In Section 2, we introduce the notion of weighted and double weighted sequences, proving an extension of the Erdös–Szekeres theorem applied to these concepts. Section 3 first presents the PFS by a matrix game perspective. This new approach is used on the development of a new technique to obtain upper bounds on the approximation guarantee of a PFS specific solution by its transformation into a *minimum double weighted sequence problem* corresponding solution. In Section 4, the greedy avoided path approximation algorithm is proposed and analyzed. Conclusions are drawn in Section 5. Finally, an Appendix gives a lower bound for the algorithm of Nagarajan and Sviridenko [9].

2. Weighted sequences

2.1. Weighted monotone sequences

Definition 1. Let $S = \langle s_1, s_2, \ldots, s_n\rangle$ be a sequence of distinct real elements. A monotone subsequence of S is a sequence $R = \langle s_{\varphi_1}, s_{\varphi_2}, \ldots, s_{\varphi_m}\rangle$ such that $1 \leq \varphi_1 < \varphi_2 < \cdots < \varphi_m \leq n$ and $s_{\varphi_1} \leq s_{\varphi_2} \leq \cdots \leq s_{\varphi_m}$ or $s_{\varphi_1} \geq s_{\varphi_2} \geq \cdots \geq s_{\varphi_m}$.

Definition 2. A set $R_1, R_2, \ldots, R_k$ of monotone subsequences of a sequence S is said to be an S-monotone partition of size k if $\bigcup_{i=1}^{k} R_i = S$ and $R_i \cap R_j = \emptyset, \forall i, j \in \{1, \ldots, k\}$.

The classical theorem of Erdös and Szekeres [2] states that from a sequence of $n^2 + 1$ distinct real elements it is always possible to extract a monotone subsequence of cardinality at least $n + 1$. In fact, the maximum cardinality monotone subsequence problem can be solved in polynomial time. However, finding a minimum size monotone partition of a given sequence is a NP-hard problem [21]. Bar-Yehuda and Fogel [1] presented an approximation algorithm for this latter problem based on the following lemma.

Lemma 1 (*[1]*). *Let $S = \langle s_1, s_2, \ldots, s_n\rangle$ be a sequence. There is an S-monotone partition of size at most $2\sqrt{n}$.*

The proof is obtained directly by the successive removal of maximum cardinality monotone subsequences. At this point, we define the notion of a weighted monotone subsequence and extend the Erdös–Szekeres theorem to this new concept.

Definition 3. Let $w : S \mapsto \Re^+$ be a weight function over a sequence S. The weight of a subsequence $R = \langle s_{\varphi_1}, s_{\varphi_2}, \ldots, s_{\varphi_l}\rangle$ of S, denoted by $w(R)$, is $\sum_{i=1}^{l} w\left(s_{\varphi_i}\right)$. Denote by $w(R_{\max})$ the maximum weight of a monotone subsequence of S.

Corollary 1. $w(R_{\max}) \geq \frac{w(S)}{2\sqrt{n}}$.

Proof. From Lemma 1, there is an S-monotone partition in at most $2\sqrt{n}$ monotone subsequences. Let $R_1, R_2, \ldots, R_k$ be such subsequences and $R^\star$ that of maximum weight. By the concept of monotone partition of a sequence, $\sum_{i=1}^{k} w(R_i) = w(S)$. So, $w(R^\star) \geq \frac{w(S)}{k}$. Since $k \leq 2\sqrt{n}$, it follows that $w(R^\star) \geq \frac{w(S)}{2\sqrt{n}}$. Since $w(R^\star) \leq w(R_{\max})$, the result is proved. □

The result of the previous corollary states a weighted version of the maximum cardinality monotone subsequence problem, studied by Erdös and Szekeres. Basically, it was proved that there will always be a monotone subsequence whose weights sum is at least the sum of all elements weights divided by the square root of the number of elements.

A survey of the Erdös–Szekeres theorem and its variations was presented by Steele [19]. As far as we know, the weighted monotone subsequence concept has not yet been explored.

2.2. Double weighted sequences

Definition 4. A double weighted set, denoted by (Γ, α, β), is composed by a set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\} \subset \Re$ of distinct elements and two weight functions $\alpha : \Gamma \mapsto \Re^+$ and $\beta : \Gamma \mapsto \Re^+$.

Definition 5. Let (Γ, α, β) be a double weighted set. A permutation $\pi : \{1, 2, \ldots, n\} \mapsto \Gamma$ defines a sequence $S = \langle \gamma_{\pi(1)}, \gamma_{\pi(2)}, \ldots, \gamma_{\pi(n)}\rangle$, named a double weighted sequence of (Γ, α, β).

Definition 6. Given a double weighted set (Γ, α, β) and a double weighted sequence S defined over it, let S_α be a maximum weighted **increasing** subsequence of S considering α as the weight function, where $C(S_\alpha)$ denotes the weight of S_α. Similarly, let S_β be a maximum weighted **decreasing** subsequence of S considering β as the weight function, where $C\left(S_\beta\right)$ denotes the weight of S_β. The weight of the double weighted sequence S, denoted by $C(S)$, is defined as $\max\{C(S_\alpha), C\left(S_\beta\right)\}$.

In order to illustrate these definitions, consider the following example: let (Γ, α, β) be a double weighted set with $\Gamma = \{1, 2, 3\}$, $\alpha = (\alpha(1), \alpha(2), \alpha(3)) = (10, 7, 8)$, and $\beta = (\beta(1), \beta(2), \beta(3)) = (6, 11, 9)$. Consider that the permutation π is the identity function, defining the sequence $S = \langle 1, 2, 3\rangle$. Clearly, the maximum weighted increasing subsequence of S is $S_\alpha = S$, and $C(S_\alpha) = \alpha(1) + \alpha(2) + \alpha(3) = 25$. Furthermore, the maximum weighted decreasing subsequence of S is $S_\beta = \langle 2\rangle$, and $C(S_\beta) = \beta(2) = 11$. Therefore, the weight of the double weighted sequence S is $C(S) = \max\{C(S_\alpha), C\left(S_\beta\right)\} = 25$.

The minimum double weighted sequence problem (MDWS) can be defined as follows: given a double weighted set (Γ, α, β), construct a double weighted sequence $S^\star$ such that $C(S^\star)$ is minimum. In this case, $S^\star$ is said to be an optimal double weighted sequence for (Γ, α, β).

Example. Considering the instance (Γ, α, β) given in the previous example, the sequence $S^\star = \langle 2, 3, 1\rangle$ is the unique optimal solution for the MDWS problem. In particular, $S^\star_\alpha = \langle 2, 3\rangle$, $S^\star_\beta = \langle 2, 1\rangle$, $C(S^\star_\alpha) = 15$ and $C(S^\star_\beta) = C(S^\star) = 17$.

Definition 7. Let $D_1 = (\Gamma_1, \alpha_1, \beta_1)$ and $D_2 = (\Gamma_2, \alpha_2, \beta_2)$ be double weighted sets. Assume, without loss of generality, that elements $\Gamma_1 = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ are given in increasing order. Consider that D_2 was constructed from D_1 by removal of an

element $\gamma_i \in \Gamma_1$ and insertion of two new elements γ_j' and γ_{j+1}' in Γ_2 such that $\gamma_i = \gamma_j' < \gamma_{j+1}'$, $\gamma_{j+1}' < \gamma_{i+1}$ if γ_{i+1} exists, $\alpha_1(\gamma_i) = \alpha_2(\gamma_j') + \alpha_2(\gamma_{j+1}')$, and $\beta_1(\gamma_i) = \beta_2(\gamma_j') = \beta_2(\gamma_{j+1}')$. It is said that D_2 is a split of D_1 and that element γ_i was split into γ_j' and γ_{j+1}'.

To illustrate the splitting process, consider the following example. $\Gamma_1 = \{2, 4, 7, 12\}, \alpha_1 = (\alpha_1(2), \alpha_1(4), \alpha_1(7), \alpha_1(12)) = (12, 7, 4, 8)$, $\beta_1 = (\beta_1(2), \beta_1(4), \beta_1(7), \beta_1(12)) = (7, 10, 9, 11)$. Element $\gamma_3 = 7$ can be split into two elements, $\gamma_3' = 7$ and $\gamma_4' = 10$, with weights $\alpha_2(\gamma_3') = 3$, $\alpha_2(\gamma_4') = 1$, and $\beta_1(\gamma_3) = \beta_2(\gamma_3') = \beta_2(\gamma_4') = 9$, creating a double weighted set ($\Gamma_2 = \{2, 4, 7, 10, 12\}, \alpha_2 = (12, 7, 3, 1, 8), \beta_2 = (7, 10, 9, 9, 11)$).

Lemma 2. *Let $D_1 = (\Gamma, \alpha, \beta)$ be a double weighted set, D_2 a split of D_1, and Φ_1 and Φ_2 optimal double weighted sequences for D_1 and D_2, respectively. Then, $C(\Phi_2) \leq C(\Phi_1)$.*

Proof. Consider that $\Phi_1 = \langle \gamma_{\phi_1(1)}, \gamma_{\phi_1(2)}, \ldots, \gamma_{\phi_1(n)} \rangle$. Construct a solution Ψ to D_2 from Φ_1 as follows: let $\gamma_i = \gamma_{\phi_1(k)}$ be the element from D_1 split into γ_j' and γ_{j+1}' in D_2. For all $k' < k$, do $\gamma_{\psi(k')} = \gamma_{\phi_1(k')}$. Let $\gamma_{\psi(k)} = \gamma_j'$ and $\gamma_{\psi(k+1)} = \gamma_{j+1}'$. For $k' = k + 2$ to $n + 1$, let $\gamma_{\psi(k')} = \gamma_{\phi_1(k'-1)}$. Since, by split definition, $\alpha(\gamma_i) = \alpha(\gamma_j') + \alpha(\gamma_{j+1}')$, every increasing subsequence of Ψ can be transformed into an increasing subsequence of Φ_1, with a not smaller weight. When elements γ_j' and γ_{j+1}' belong to such an increasing sequence, they can be both substituted by γ_i. All other elements are identical. An equivalent transformation is valid for decreasing subsequences of Ψ, in which only one of γ_j' or γ_{j+1}' can be present, and, by split definition, $\beta(\gamma_i) = \beta(\gamma_j') = \beta(\gamma_{j+1}')$. Hence, $C(\Psi) \leq C(\Phi_1)$. Since $C(\Phi_2) \leq C(\Psi)$, the result follows. □

The use of the split concept in conjunction with Corollary 1 permits obtaining a lower bound on optimal solutions of an MDWS instance.

Theorem 1. *Let Φ_1 be an optimal solution of an MDWS instance $D_1 = (\Gamma_1, \alpha_1, \beta_1)$, $|\Gamma_1| = n$. Then,*

$$C(\Phi_1) \geq \frac{\sum_{i=1}^{n} \alpha_1(i)}{\sqrt{4\left(n + \sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil\right)}}.$$

Proof. Consider a succession of splits that converts D_1 into a double weighted set $D_2 = (\Gamma_2, \alpha_2, \beta_2)$ such that $\alpha_2(i) \leq \beta_2(i)$, for all $i \in \{1, \ldots, |\Gamma_2|\}$. Clearly, $\sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$ splits are sufficient, which implies that $|\Gamma_2| \leq n + \sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$. Let Φ_2 be an optimal sequence for D_2. By Lemma 2, $C(\Phi_1) \geq C(\Phi_2)$. Consider now a double weighted set $D_3 = (\Gamma_3, \alpha_3, \beta_3)$ such that $\Gamma_3 = \Gamma_2$ and $\alpha_3 = \beta_3 = \alpha_2$. Let Φ_3 be an optimal sequence for D_3. Since $\Gamma_2 = \Gamma_3$, $\alpha_3(i) \leq \alpha_2(i)$, and $\beta_3(i) \leq \beta_2(i)$ for all $i \in \{1, \ldots, |\Gamma_2|\}$, it is true that $C(\Phi_2) \geq C(\Phi_3)$. Furthermore, since $\alpha_3 = \beta_3$, α_3 and β_3 can be viewed as an unique weight function. Then, by Corollary 1,

$$C(\Phi_3) \geq \sum_{i=1}^{|\Gamma_3|} \alpha_3(i)/(2\sqrt{|\Gamma_3|}).$$

Since $\alpha_3 = \alpha_2$ and D_2 is a split of D_1, it follows that

$$\sum_{i=1}^{|\Gamma_3|} \alpha_3(i) = \sum_{i=1}^{|\Gamma_2|} \alpha_2(i) = \sum_{i=1}^{n} \alpha_1(i).$$

Finally, since $|\Gamma_3| = |\Gamma_2| \leq n + \sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$, it is true that

$$C(\Phi_3) \geq \sum_{i=1}^{|\Gamma_3|} \alpha_3(i)/(2\sqrt{|\Gamma_3|}) \geq \sum_{i=1}^{n} \alpha_1(i) \Big/ \sqrt{4\left(n + \sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil\right)}.$$

Hence, $C(\Phi_1) \geq C(\Phi_2) \geq C(\Phi_3)$, and the result follows. □

3. Lower bounds for a matrix game

This section introduces the *matrix minimum maximum path game*, which is exactly the PFS viewed from a game perspective. A technique to construct lower bounds on *matrix minimum maximum path game* optimal solutions based on its transformation into the *minimum double weighted sequence problem* is presented.

3.1. Paths and anti-paths

Let $T \in \Re_{m \times n}^{+}$ be a matrix and $T_1, T_2, \ldots, T_n$ its columns. A permutation $\pi : \{1, 2, \ldots, n\} \mapsto \{T_1, T_2, \ldots, T_n\}$ over T defines a new matrix T^{π}, named the *permutated matrix*.

Table 3.1
A *path* and an *anti-path* on a permutated matrix.

	1	2	3	4	5
1	**5**	**2**	**4**	6	9
2	1	3	**7**	**8**	2
3	6	5	2	**4**	**8**
4	3	9	1	7	**5**

Path

	1	2	3	4	5
1	5	2	4	**6**	**9**
2	1	3	**7**	**8**	2
3	6	**5**	**2**	4	8
4	**3**	**9**	1	7	5

Anti-Path

Table 3.2
Maximum weight *path* shown over the completion times.

	1	2	3	4	5
1	**5**	**5+2=7**	**7+4=11**	11+6=17	17+9=26
2	5+1=6	7+3=10	**11+7=18**	**18+8=26**	26+2=28
3	6+6=12	12+5=17	18+2=20	**26+4=30**	**30+8=38**
4	12+3=15	17+9=26	26+1=27	30+7=37	**38+5=43**

Completion Times

Definition 8. A **path**, defined over a permutated matrix T^π, is a sequence $P = \langle p_1, p_2, \ldots, p_{n+m-1}\rangle$ of distinct cells in T^π such that $p_1 = t^\pi_{1,1}$, $p_{n+m-1} = t^\pi_{m,n}$, and $p_k = t^\pi_{i_k,j_k}$ is the successor of $p_{k-1} = t^\pi_{i_{k-1},j_{k-1}}$ on P if and only if one of the two relations below is valid:

(1) $i_k = i_{k-1}$ and $j_k = j_{k-1} + 1$, or
(2) $i_k = i_{k-1} + 1$ and $j_k = j_{k-1}$.

The weight of P, $W(P)$, is defined as $\sum_{i=1}^{n+m-1} p_i$. P is said to be a maximum weight path if $W(P) \geq W(P')$ for every path P' over T^π.

Definition 9. An **anti-path**, defined over a permutated matrix T^π, is a sequence $A = \langle a_1, a_2, \ldots, a_{n+m-1}\rangle$ of distinct cells in T^π such that $a_1 = t^\pi_{m,1}$, $a_{n+m-1} = t^\pi_{1,n}$, and $a_k = t^\pi_{i_k,j_k}$ is the successor of $a_{k-1} = t^\pi_{i_{k-1},j_{k-1}}$ on A if and only if one of the two relations below is valid:

(1) $i_k = i_{k-1}$ and $j_k = j_{k-1} + 1$, or
(2) $i_k = i_{k-1} - 1$ and $j_k = j_{k-1}$.

The weight of A, $W(A)$, is defined as $\sum_{i=1}^{n+m-1} a_i$.

Example. In order to illustrate the concepts of path and anti-path, consider the following example. Let $\pi = \{1, 2, 3, 4, 5\}$ be a permutation and T^π the corresponding permutated matrix, as represented by Table 3.1. Here, the sequence $P = \langle(1, 1), (1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5), (4, 5)\rangle$ constitutes a *path* on T^π of weight 43, while the sequence $A = \langle(4, 1), (4, 2), (3, 2), (3, 3), (2, 3), (2, 4), (1, 4), (1, 5)\rangle$ represents an *anti-path* on T^π of weight 49. In particular, P is a maximum weight path on T^π.

Observe that the computation of the completion time $C_{m,\pi(n)}$ through the recurrence in Section 1.1 has the cells that give the maximum value at each step of the calculation defining a *path*. In particular, this path is a maximum weight path, and the sum of its cells give exactly the makespan associated to permutation π. This is presented for the example above in Table 3.2. So, when P is a maximum **path**, $W(P)$ is the makespan associated to the permutation of the jobs associated to matrix T^π.

3.2. The PFS and matrix games

The PFS can be viewed as a two-person matrix game. Given a matrix $T \in \Re_{m\times n}$ with positive elements, player 1 acts first, selecting a permutation π over the columns of T that creates a new matrix T^π. Then, player 2 selects a path P on matrix T^π, that is, a sequence of cells on T^π, starting from $t^\pi_{1,1}$ such that the cell after $t^\pi_{i,j}$ on P can only be $t^\pi_{i+1,j}$ or $t^\pi_{i,j+1}$ respecting the matrix limits, i.e., $i + 1 \leq n$ and $j + 1 \leq m$. At the end of game, player 1 pays to player 2 the sum of cells on P. Let us name this game the *matrix minimum maximum path game*, denoting it by MMP. The equivalence between PFS and MMP is clear. A schedule on PFS corresponds to a permutation on MMP and the makespan of such a schedule is exactly the cost of a maximum path chosen by player 2 given player 1's permutation. Therefore, player 2's objective of maximizing such a sum can be accomplished by an $O(nm)$ dynamic-programming algorithm based on the recursive makespan definition presented in Section 1. It computes a maximum path over T^π, i.e., the makespan of a selected schedule. The cost of a solution π for MMP is denoted by $W(T^\pi)$. From this point, the PFS is analyzed as the MMP, by considering that our objective is to act as player 1, paying the minimum possible value to player 2. Furthermore, due to the polynomial time algorithm that calculates the maximum path on a permutated matrix, player 1 always knows, after selecting a permutation, the maximum path that will be chosen by player 2.

3.3. Approximation guarantees of PFS solutions

Let $T \in \Re_{m\times n}$ be a processing time matrix from a PFS instance composed by n jobs and m machines and T^π be a permutated matrix corresponding to a schedule π, as defined in Section 3.2.

In this section, we are interested in techniques to obtain an upper bound on the approximation factor of the permutation schedule resulting from π. In particular, it will be proved that the approximation factor obtained by π, when applied as solution to an equivalent instance of the *minimum double weighted sequence problem*, is an upper bound on the approximation factor of schedule π when applied to the processing time matrix T.

We proceed now by showing how to construct an equivalent instance of the *minimum double weighted sequence problem*. Assume, without loss of generality, and for the remainder of this section, that π does not change the matrix T; i.e., $\pi = \{1, 2, \ldots, n\}$ and $T^\pi = \langle T_1, T_2, \ldots, T_n\rangle$ (this is true since it is always possible to rearrange the columns of the original processing time matrix T in such a way that π follows this definition). Furthermore, let P^π denote a maximum weight path on T^π and let A^π represent an arbitrary anti-path on T^π.

Now, let us construct a double weighted set $(\Gamma, \alpha^\pi, \beta^\pi)$ from T^π, P^π and A^π. First, make $S^\pi = \{1, 2, \ldots, n\}$, following π. Second, let $\alpha^\pi(i)$ be the sum of all cells on the path P^π that belong to the column i of the matrix T^π. Finally, let $\beta^\pi(i)$ be the sum of all cells on the anti-path A^π that belong to the column i of the matrix T^π.

Example. Let P^π be a maximum path, A^π an arbitrary anti-path, and T^π a permutated matrix as represented in the example of Section 3.1. Since $n = 5$, $\Gamma = \{1, 2, 3, 4, 5\}$. Furthermore, the weight function α^π is defined as $\alpha^\pi(1) = (1, 1) = 5$, $\alpha^\pi(2) = (1, 2) = 2$, $\alpha^\pi(3) = (1, 3) + (2, 3) = 11$, $\alpha^\pi(4) = (2, 4) + (3, 4) = 12$, and $\alpha^\pi(5) = (3, 5) + (4, 5) = 13$. Finally, the weight function β^π is defined as $\beta^\pi(1) = (4, 1) = 3$, $\beta^\pi(2) = (4, 2) + (3, 2) = 14$, $\beta^\pi(3) = (3, 3) + (2, 3) = 9$, $\beta^\pi(4) = (2, 4) + (1, 4) = 14$, and $\beta^\pi(5) = (1, 5) = 9$.

Lemma 3. $C(S^\pi) = W(T^\pi)$.

Proof. Since $S^\pi = \langle 1, 2, \ldots, n\rangle$, and all the weights are non-negative, the maximum weight monotone subsequence of S^π is either the increasing subsequence S^π itself or a decreasing subsequence that can have at most one element. No coefficient $\beta(i)$ can have a value larger than the maximum **path**, since it would then be part of the maximum **path**. Hence, $C(S^\pi)$ is given by the increasing subsequence S^π itself, and we have $C(S^\pi) = \sum_{i=1}^{n} \alpha(i) = W(P^\pi) = W(T^\pi)$, as the makespan is given by the maximum **path** and each $\alpha(i)$ is the sum of the cells in the maximum path in each column. □

Now, we establish the relation between $C(S)$ and $W(T)$ for all sequences with respect to the **path** and the **anti-path** obtained for the initial permutation π.

Theorem 2. *Let σ be an arbitrary permutation over Γ and $S^\sigma = \{\sigma_1, \ldots, \sigma_n\}$, T^σ the permutated matrix obtained applying σ to T and the double weighted set $(\Gamma, \alpha^\pi, \beta^\pi)$. Then $C(S^\sigma) \leq W(T^\sigma)$.*

Proof. Every weighted increasing subsequence of S^σ, taking α^π as weight function, is equivalent to a subsequence of a path in T^σ whose cells belong only to P^π. Similarly, every weighted decreasing subsequence of S^σ, taking β^π as weight function, is equivalent to a subsequence of a path in T^σ whose cells belong only to A^π. Consequently, the maximum weight monotone subsequence of S^σ is equivalent to a subsequence of a path in T^σ whose cells belong exclusively to P^π or A^π. This can be seen observing that for any permutation an increasing subsequence will only have n elements when it is $\{1, 2, \ldots, n\}$; otherwise, it will have fewer elements and its value, $C(S^\sigma_\alpha)$, will be strictly less than $W(P^\pi)$. In a symmetric way, a decreasing subsequence will only have n elements when it is $\{n, n-1, \ldots, 1\}$; otherwise, it will have fewer elements and its value, $C(S^\sigma_\beta)$, will be strictly less than $W(A^\pi)$. Finally, observe that the maximum path on T^σ will be able to choose the cells that lead to the largest sum from A^π and P^π. In particular, if $S^\sigma = \{n, n-1, \ldots, 1\}$ it will be at least $W(A^\pi)$, since A^π will be a possible path. Therefore, $W(T^\sigma) \geq C(S^\sigma)$. □

Let $S^\star$ represent an optimal solution of the *minimum double weighted sequence problem* for (Γ, α, β).

Corollary 2. $C(S^\star) \leq W(T^{OPT})$.

Proof. By Theorem 2, $C(S^{OPT}) \leq W(T^{OPT})$. By optimal solution definition, $C(S^\star) \leq C(S^{OPT})$. Consequently, $C(S^\star) \leq W(T^{OPT})$. □

By previous results we have the following.

Theorem 3. $\frac{W(T^\pi)}{W(T^{OPT})} \leq \frac{C(S^\pi)}{C(S^\star)}$.

Proof. $\frac{W(T^\pi)}{W(T^{OPT})} \leq \frac{W(T^\pi)}{C(S^{OPT})} \leq \frac{W(T^\pi)}{C(S^*)} = \frac{C(S^\pi)}{C(S^*)}$. □

As consequence of the last theorem it is possible to obtain an upper bound on the approximation guarantee of a PFS specific solution π by constructing an equivalent MDWS instance $(\Gamma, \alpha^\pi, \beta^\pi)$, as described in this section, and analyzing the approximation factor of any permutation π applied as a solution to such instance.

4. The greedy avoided path algorithm

This section presents a polynomial time deterministic algorithm which constructs a solution for the PFS based on weighted monotone subsequences properties previously explored. The time complexity and approximation guarantee of the algorithm are also analyzed. We prove that a job permutation, in which cells representing the longest operation of each job constitute an avoided path on processing time matrix, reach the claimed approximation guarantee. Furthermore, a permutation with this property can be obtained by a simple job sorting algorithm. Its approximation factor analysis is based on deterministic combinatorial arguments based on Erdös–Szekeres theorem extensions.

We next present the GREEDY AVOIDED PATH ALGORITHM. The main concept is to construct a permutation π in which the maximum path avoids as much as possible the large coefficients in the T^{π}. We propose to do this in a greedy way. This amounts to, first, determining for each job the machine in which the corresponding operation processing time is maximum. Next, the jobs are ordered in such a way that those cells are as close as possible to an **anti-path**. This order is the one in which the first job has the largest machine index for its longest operation. The second job is the one with the second largest machine index for its longest operation, and so on. The algorithm follows.

GREEDY AVOIDED PATH ALGORITHM(J, M, T)

$\triangleright\ J = \{J_1, J_2, \ldots, J_n\}$, the set of jobs.
$\triangleright\ M = \{M_1, M_2, \ldots, M_m\}$, the set of machines.
$\triangleright\ T \in \Re^{+}_{M\times J}$, the processing time matrix.

1 **For each** $j \in J$: **Set** $MaxMachine_j$ as the index of the machine with the longest operation of job j.

2 **Construct** a permutation $\pi : \{1, \ldots, n\} \mapsto J$ sorting the jobs in non-increasing order of the values of $MaxMachine_j$.

3 **Return** permutation π.

Theorem 4. *The* GREEDY AVOIDED PATH ALGORITHM *returns a permutation that is a* $2\sqrt{2n+m}$*-approximation for the PFS.*

Proof. Let π be the solution returned by the GREEDY AVOIDED PATH ALGORITHM, T^{π} the permutated matrix of T, and P^{π} a maximum path in T^{π}. Consider that A^{π} is an anti-path in T^{π} with the following property: all cells on positions $(MaxMachine_j, j)$, for $j = 1, \ldots, n$, in T^{π} belong to A^{π}. Once π is obtained by application of the GREEDY AVOIDED PATH ALGORITHM, the construction of such an anti-path in T^{π} is possible, since the indices $MaxMachine_j$, for $j = 1, \ldots, n$, are in non-increasing order. Let T^{OPT} be an optimal permutated matrix of T. The approximation factor of solution π is, by definition, $W(T^{\pi})/W(T^{OPT})$.

Following the steps in Section 3.3, without loss of generality, we can have $\pi = \{1, 2, \ldots, n\}$. Then, let $S^{\pi} = \pi$ be a solution for the instance $(\Gamma, \alpha^{\pi}, \beta^{\pi})$ of the MDWS problem. Therefore, by Lemma 3, $C(S^{\pi}) = W(T^{\pi})$. Let now $C(S^{\star})$ be an optimal solution for $(\Gamma, \alpha^{\pi}, \beta^{\pi})$. By Theorem 1, $C(S^{\star}) \geq \sum_{i=1}^{n} \alpha(i)/\sqrt{4\left(n + \sum_{i=1}^{n} \lceil \alpha(i)/\beta(i) \rceil\right)}$. Since A^{π} was chosen in such a way that the longest operation of each job belongs to it, a ratio $\lceil \alpha(i)/\beta(i) \rceil$ can only be greater than one for jobs, columns of T^{π}, where the maximum path P^{π} chooses two or more cells. On all others this ratio must be one. As there are exactly $n + m - 1$ cells on P^{π} and A^{π}, we have $\sum_{i=1}^{n} \lceil \alpha(i)/\beta(i) \rceil \leq n + m - 1$. Observe that this upper bound is tight when one job has all its cells in the maximum path and all operations with the same processing time, or all jobs and all operations have the same processing time.

Consequently, $C(S^{\star}) \geq \sum_{i=1}^{n} \alpha(i)/2\sqrt{2n+m)}$.

Since $S^{\pi} = \pi$ and all $\alpha(i)$ and $\beta(i)$ are non-negative, a largest subsequence is the increasing subsequence $1, 2, \ldots, n$, and therefore $C(S^{\pi}) = \sum_{i=1}^{n} \alpha(i)$. So, we can state that $C(S^{\star}) \geq C(S^{\pi})/2\sqrt{2n+m}$.

Finally, by Theorem 3, $C(S^{\pi})/C(S^{\star}) \geq W(T^{\pi})/W(T^{OPT})$. Hence, $W(T^{\pi})/W(T^{OPT}) \leq 2\sqrt{2n+m}$. □

Example. Consider matrix T of Table 3.1. The *MaxMachine* vector is $(3, 4, 2, 2, 1)$. Therefore, the GREEDY AVOIDED PATH ALGORITHM would produce a permutation in which the second column is the first one, while the first appears second and all remaining ones keep their positions. The resulting T^{π} matrix is presented in Table 4 with the $(MaxMachine_j, j)$ cells for $j = 1, 2, \ldots, n$ in bold. An **anti-path** containing all these cells is represented by the underlined cells.

The time complexity analysis for the GREEDY AVOIDED PATH ALGORITHM is straightforward. First, observe that the sorting phase can be done in linear time using, for example, counting sort. Line 1 can be executed in $\Theta(nm)$ time. Permutation construction on line 2 can be achieved sorting jobs in $\Theta(n+m)$ time using *MaxMachine* variables as key. Line 3 costs $\Theta(n)$ steps. Hence, the GREEDY AVOIDED PATH ALGORITHM is a polynomial time $\Theta(nm)$ algorithm.

5. Conclusion

This work presents a deterministic approximation algorithm for the PFS with performance ratio $2\sqrt{2n+m}$ and time complexity $\Theta(nm)$. In the case that $n = \Theta(m)$ this is the best approximation algorithm already obtained for the PFS

Table 4
Permutation matrix generated by the Greedy Avoided Path Algorithm.

	1	**2**	**3**	**4**	**5**
1	2	5	4	6	**9**
2	3	1	**7**	**8**	2
3	5	**6**	2	4	8
4	**9**	3	1	7	5

achieving the same approximation factor found by Nagarajan and Sviridenko [9] in linear time, reducing its complexity from $\Omega(n^4.m)$. The Erdös–Szekeres theorem was extended, considering a weighted version in which elements of monotone subsequences can have different weights. As a consequence, a novel technique to obtain upper bounds on approximation guarantees of PFS solutions using double weighted sequences was introduced, exploring the connection between weighted monotone subsequence problems and the PFS.

Acknowledgements

We are grateful to an anonymous referee for several suggestions and comments. We thank the partial support of CNPq to MP and of Petrobras to DS.

Appendix

The algorithm from Nagarajan and Sviridenko [9] has two phases. The first phase comprises decomposing the original processing time matrix into $k \leq n.m$ permutation matrices. Such decomposition can be achieved by applying an algorithm for minimum edge-coloring on bipartite multigraphs. To the best of our knowledge, the current most efficient algorithm to solve this problem depends on a log factor of the maximum degree of a vertex [17]. In particular, the well-performing algorithm from Gabow and Kariv [5] with a time complexity of $O(|V|.|\tilde{E}|.\log\mu)$ could be applied here, where $|V| = n + m$, $|\tilde{E}| = n.m$ and $\mu = \max\{\max_i \sum_j t_{ij}, \max_j \sum_i t_{ij}\}$. Disregarding the μ factor, a lower bound on the time complexity of the first phase of the algorithm would be $\Omega(n^2.m)$. The second phase of the algorithm constructs the permutation schedule and is composed by n steps. Each step $1 \leq i \leq n$ selects a job to be inserted at position i of the schedule. Once a job is selected to be inserted on step i, its position cannot be changed. The selection of the job to be inserted at position i comprises testing all $n - i$ remaining jobs, calculating the insertion cost of each one (estimated by an upper bound function U_i), and selecting that of minimum insertion cost at position i. The function U_i is defined as the weighted sum of $k \leq n.m$ functions U_i^k. The calculus of each function U_i^k cannot take less than $\Omega(n)$. Hence, a (possibly far from tight) lower bound on the time complexity of the second phase of the algorithm would be $\Omega(n^4.m)$. Therefore, the time complexity of the deterministic algorithm from [9] is lower bounded by $\Omega(n^4.m)$.

References

[1] R. Bar-Yehuda, S. Fogel, Partitioning a sequence into few monotone subsequences, Acta Inform. 35 (1998) 421–440.
[2] P. Erdös, G. Szekeres, A combinatorial problem in geometry, Compositio Math. 2 (1935) 463–470.
[3] J. Gupta, C. Koulamas, G. Kyparisis, Performance guarantees for flowshop heuristics to minimize makespan, European J. Oper. Res. 169 (2006) 865–872.
[4] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, Math. Oper. Res. 1 (1976) 117–129.
[5] H.N. Gabow, O. Kariv, Algorithms for edge coloring bipartite graphs and multigraphs, SIAM J. Comput. 11 (1982) 117–129.
[6] T. Gonzalez, S. Sahni, Flowshop and jobshop schedules: complexity and approximation, Oper. Res. 26 (1978) 36–52.
[7] L.A. Hall, Approximability of flow shop scheduling, Math. Program. 82 (1998) 175–190.
[8] S.M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Res. Logist. Quart. 1 (1954) 61–68.
[9] V. Nagarajan, M. Sviridenko, Tight bounds for permutation flow shop scheduling, in: Proceedings of IPCO 2008, pp. 154–168.
[10] V. Nagarajan, M. Sviridenko, Tight bounds for permutation flow shop scheduling, Math. Oper. Res. 34 (2009) 417–427.
[11] E. Nowicki, C. Smutnicki, Worst-case analysis of an approximation algorithm for flow-shop scheduling, Oper. Res. Lett. 8 (1989) 171–177.
[12] E. Nowicki, C. Smutnicki, Worst-case analysis of Dannenbring's algorithm for flow-shop scheduling, Oper. Res. Lett. 10 (1991) 473–480.
[13] E. Nowicki, C. Smutnicki, New results in the worst-case analysis for flow-shop scheduling, Discrete Appl. Math. 46 (1993) 21–41.
[14] C. Potts, D. Shmoys, D. Williamson, Permutation vs. nonpermutation flow shop schedules, Oper. Res. Lett. 10 (1991) 281–284.
[15] P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs, J. Comput. System Sci. 37 (1988) 130–143.
[16] D. Sotelo, M. Poggi, An approximation algorithm for the permutation flow shop scheduling problem via Erdös–Szekeres theorem extensions, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 28 (2007).
[17] A. Schrijver, Combinatorial Optimization. Polyhedra and Efficiency. Algorithms and Combinatorics, Springer-Verlag, Berlin, 2003.
[18] S. Sevast'janov, On some geometric methods in scheduling theory: a survey, Discrete Appl. Math. 55 (1994) 59–82.
[19] J.M. Steele, Variations on the monotone subsequence theme of Erdös and Szekeres, IMA Vol. Math. Appl. 72 (1995) 111–131.
[20] M. Sviridenko, A note on permutation flow shop problem, Ann. Oper. Res. 129 (2004) 247–252.
[21] K. Wagner, Monotonic coverings of finite sets, Elektron. Informationsverarb. Kybernetes 20 (1984) 633–639.