PUC

ISSN 0103-9741

Monografias em Ciência da Computação

n° 28/07

An Approximation Algorithm for Permutation Flowshop Scheduling Problem via Erdös-Szekeres Theorem Extensions

David Sotelo Pinheiro da Silva

Marcus Vinicius Soledade Poggi de Aragão

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 28/07 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Dezembro, 2007

An Approximation Algorithm for Permutation Flowshop Scheduling Problem via Erdös-Szekeres Theorem Extensions

David Sotelo P. da Silva and Marcus V. S. Poggi de Aragão

dsilva@inf.puc-rio.br, poggi@inf.puc-rio.br

Abstract. This work presents a deterministic approximation algorithm for Permutation Flowshop Scheduling Problem (PFS) with performance ratio $2\sqrt{2n+m}$ and time complexity $\Theta(nm + n\log n)$, where n is the number of jobs and m is the number of machines of an instance. This result consists in the first known deterministic approximation algorithm for PFS in which performance ratio is not a linear factor of n or m. In the case that $n = O(m)$ this is the best approximation algorithm already obtained for PFS. A novel technique for performance guarantee analysis of PFS solutions is developed, exploring its correlation with Weighted Monotone Subsequence Problems.

Keywords: Approximation Algorithms, Permutation Flowshop Scheduling, Monotone Subsequences, Erdös-Szekeres Theorem.

Resumo. Este trabalho apresenta um algoritmo aproximativo para o problema Permutation Flowshop Scheduling (PFS) com razão de performance $2\sqrt{2n+m}$ e complexidade de tempo $\Theta(nm + n\log n)$, onde n é o número de tarefas e m o número de máquinas de uma instância. Este resultado consiste no primeiro algoritmo aproximativo determinístico para o PFS cuja razão de performance não é um fator linear de n ou m. No caso em que $n = O(m)$ este é o melhor algoritmo aproximativo já obtido para o PFS. Uma nova técnica para análises de garantia de performance de soluções para o PFS é desenvolvida, explorando sua relação com problemas de Subseqüências Monótonas Ponderadas.

Palavras-chave: Algoritmos Aproximativos, Permutation Flowshop Scheduling, Subseqüências Monótonas, Teorema de Erdös-Szekeres.

1 Introduction

A *Permutation Flowshop Scheduling* is a production planing process consisting of a set $J = \{J_1, J_2, ..., J_n\}$ of n jobs to be executed in a set $M = \{M_1, M_2, ..., M_m\}$ of m machines. In this process every job J_j is composed by m stages $O_{1,j}, O_{2,j}, ..., O_{m,j}$ 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 by 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 Permutation Flowshop Scheduling Problem (PFS) is to find a permutation π that minimizes the makespan.

PFS was proved Strongly NP-Hard by Garey, Johnson e Sehti [4] for instances with three or more machines. In a seminal paper, Johnson [7] presented a polynomial time algorithm for instances with two machines. Gonzalez e Sahni [5] showed that every busy scheduling for PFS has an approximation factor of m times the optimal solution. Potts, Shmoys e Williamson [11] proved the existence of some instances for which non-permutation based solutions are $\Omega(\sqrt{m})$ less costly than permutation based. Williamson et al [15] proved an inapproximability result of 5/4 to PFS. Hall [6] presented a PTAS for PFS when the number of machines m is fixed.

To the best of our knowledge, the best known up to date deterministic approximation algorithm for PFS has performance guarantee of $\lceil m/2 \rceil$ [12, 8, 9, 10]. Approximation ratios for a large number of PFS heuristics were surveyed by Gupta, Koulamas and Kyparisis[3]. Sviridenko [14] proposed a randomized algorithm for PFS with $O(\sqrt{m \log m})$ expected approximation factor.

The purpose of this work is to present a deterministic approximation algorithm for PFS with performance ratio $2\sqrt{n+m}$ and time complexity $\Theta(n\dot{m} + n \log n)$. This result consists in the first known deterministic approximation algorithm for PFS in which performance ratio is not a linear factor of n or m. In the case that $n = O(m)$ this is the best approximation algorithm already obtained for PFS. A novel technique for performance guarantee analysis of PFS solutions is developed, exploring the correlation between Weighted Monotone Subsequence Problems and PFS.

This works is organized as follows. In section 2, extensions of Erdös-Szekeres Theorem [2] are obtained for weighted sequences. Section 3 presents a technique to obtain upper bounds on approximation guarantee of a PFS solution via its reduction to `Minimum Double Weighted Sequence Problem`. In section 4, the approximation algorithm `Greedy Avoided Path` is presented and analyzed. Final conclusions are drawn in section 5.

2 Weighted Subsequences

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 $T = \langle s_{\varphi_1}, s_{\varphi_2}, \ldots, s_{\varphi_m} \rangle$ such that $1 \leq \varphi_1 < \varphi_2 < \ldots < \varphi_m \leq n$ and $s_{\varphi_1} < s_{\varphi_2} < \ldots < s_{\varphi_m}$ or $s_{\varphi_1} > s_{\varphi_2} > \ldots > s_{\varphi_m}$.*

Definition 2 *A set $T_1, T_2, \ldots, T_k$ of monotone subsequences of a sequence S is said a S-monotone partition of size k if $\bigcup\limits_{i=1}^{k} T_i = S$ and $\bigcap\limits_{i=1}^{k} T_i = \emptyset$.*

The classic theorem of Erdös and Szekeres [2] enunciates that from a sequence of n^2+1 distinct real elements is always possible extract a monotone subsequence of cardinality at least $n+1$. The maximum cardinality monotone subsequence problem can be solved in polynomial time. However, finding a minimum size monotone partition is a NP-Hard problem [16]. Bar-Yehuda and Fogel [1] presented an approximation algorithm for minimum size monotone partition based on the following Lemma:

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

Lemma 1 can be proved directly by successive remotion of maximum cardinality monotone subsequences of S. Consider now a generalization of monotone subsequence concept. Let us define a weight function $w : S \mapsto \Re^+$ over a sequence S. The weight of a monotone subsequence $T = \langle s_{\varphi_1}, s_{\varphi_2}, \ldots, s_{\varphi_l} \rangle$ of S is $\sum\limits_{i=1}^{l} w(s_{\varphi_i})$. Let T_{max} be the maximum weight on a monotone subsequence of S. The following result is valid:

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

Proof: From Lemma 1, there is a S-monotone partition in at most $2\sqrt{n}$ monotone subsequences. Let $T_1, T_2, \ldots, T_k$ be such subsequences and $T^\star$ that of maximum weight. By the concept of monotone partition of a sequence, $\sum\limits_{i=1}^{k} w(T_i) = w(S)$. So, $w(T^\star) \geq \frac{w(S)}{k}$. Once $k \leq 2\sqrt{n}$ follows that $w(T^\star) \geq \frac{w(S)}{2\sqrt{n}}$. Once $T^\star \leq T_{max}$ the result is proved. ■

A survey on Erdös-Szekeres theorem and its variations was presented by Steele [13]. From the best we know, weighted monotone subsequence concept hasn't been explored so far.

2.2 Double Weighted Sequences

Definition 3 *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 4 *Let (Γ, α, β) a double weighted set. A permutation $\pi : \{1, 2, \ldots, n\} \mapsto \Gamma$ defines a sequence $S = \left\langle \gamma_{\pi(1)}, \gamma_{\pi(2)}, \ldots, \gamma_{\pi(n)} \right\rangle$ named a double weighted sequence of (Γ, α, β).*

The Minimum Double Weighted Sequence Problem: Let S_α be a maximum weighted increasing subsequence of S considering α as weight function and $C(S_\alpha)$ its cost. Similarly, let S_β be a maximum weighted decreasing subsequence of S considering β as weight function and $C(S_\beta)$ its cost. The cost of S, denoted by $C(S)$, is defined as $\mathit{max}\{C(S_\alpha), C(S_\beta)\}$. The `Minimum Double Weighted Sequence Problem` (MDWS) consists of, given a double weighted set (Γ, α, β), construct a double weighted sequence $S^\star$ such that $C(S^\star)$ is minimum.

Definition 5 *Let $D_1 = (\Gamma_1, \alpha_1, \beta_1)$ and $D_2 = (\Gamma_2, \alpha_2, \beta_2)$ be double weighted sets. Assume, w.l.o.g , that $\Gamma_1 = \{\gamma_1, \gamma_2, ..., \gamma_n\}$ elements are given in increasing order. Consider that D_2 was constructed from D_1 by remotion of an element $\gamma_i \in \Gamma_1$ and insertion of two new elements γ'_j and γ'_{j+1} in S_2 such that: $\gamma_i = \gamma'_j < \gamma'_{j+1}$, $\gamma'_{j+1} < \gamma_{i+1}$ if γ_{i+1} exists, $\alpha_1(\gamma_i) = \alpha_2\left(\gamma'_j\right) + \alpha_2\left(\gamma'_{j+1}\right)$ and $\beta_1(\gamma_i) = \beta_2\left(\gamma'_j\right) = \beta_2\left(\gamma'_{j+1}\right)$. It's 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 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 $\beta_1(\gamma_3) = \beta_2(\gamma'_3) = \beta_2(\gamma'_4) = 9$, giving origin to double weighted set $(\Gamma_2 = \{2, 4, 7, 10, 12\}, \alpha_2 = (12, 7, 3, 1, 8), \beta_2 = (7, 10, 9, 9, 11))$.

Theorem 1 *Let $D_1 = (\Gamma, \alpha, \beta)$ be a double weighted set, D_2 a split of D_1, 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 $\Phi_1 = \left\langle \gamma_{\phi_1(1)}, \gamma_{\phi_1(2)}, \ldots, \gamma_{\phi_1(n)} \right\rangle$. Construct a solution Ψ to D_2 from Φ_1 as follows: let $\gamma_i = \gamma_{\phi_1(k)}$ 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 $n + 1$ do $\gamma_{\psi(k')} = \gamma_{\phi_1(k'-1)}$. Once, by split definition, $\alpha(\gamma_i) = \alpha\left(\gamma'_j\right) + \alpha\left(\gamma'_{j+1}\right)$, every increasing subsequence of Ψ can be transformed into an increasing subsequence of Φ_1, with not smaller cost. When elements γ'_j and γ'_{j+1} belong to such 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, $\beta(\gamma_i) = \beta\left(\gamma'_j\right) = \beta\left(\gamma'_{j+1}\right)$. Hence, $C(\Psi) \leq C(\Phi_1)$. Once $C(\Phi_2) \leq C(\Psi)$, the result follows. ■

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

Theorem 2 *Let Φ_1 be an optimal solution of a MDWS instance
$D_1 = (\Gamma_1, \alpha_1, \beta_1)$, $|\Gamma_1| = n$.*

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

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, $\sum\limits_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$ splits are sufficient, what implies that $|\Gamma_2| \leq n + \sum\limits_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$. Let Φ_2 be an optimal sequence for D_2. By Theorem 1, $C(\Phi_1) \geq C(\Phi_2)$. 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. Once $\Gamma_2 = \Gamma_3$, $\alpha_3(i) \leq \alpha_2(i)$ and $\beta_3(i) \leq \beta_2(i)$ for all $i \in \{1, \ldots, |\Gamma_3|\}$, is true that $C(\Phi_2) \geq C(\Phi_3)$. Once $\alpha_3 = \beta_3$, α_3 and β_3 can be viewed as a unique weight function.
Then, by Corollary 1:
$C(\Phi_3) \geq \sum\limits_{i=1}^{|\Gamma_3|} \alpha_3(i)/(2\sqrt{|\Gamma_3|}) = \sum\limits_{i=1}^{n} \alpha_1(i) \Big/ \sqrt{4\left(n + \sum\limits_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil\right)}$.
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 Min-Max Path Problem`, which is exactly PFS viewed by a game perspective. A technique to construct lower bounds on `Matrix Min-Max Path Problem` optimal solutions based on its transformation into `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 *permutated matrix.*

Definition 6 *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 an only if one of the two relations below is valid:*

1. *$i_k = i_{k-1}$ and $j_k = j_{k-1} + 1$*
2. *$i_k = i_{k-1} + 1$ and $j_k = j_{k-1}$.*

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

Definition 7 *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 an only if one of the two relations below is valid:*

1. *$i_k = i_{k-1}$ and $j_k = j_{k-1} - 1$*
2. *$i_k = i_{k-1} - 1$ and $j_k = j_{k-1}$.*

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

3.2 PFS and Matrix Games

The PFS problem 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 switches the order between them, giving origin to 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 *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 schedule is exactly the cost of a maximum path chose by player 2 given player's 1 permutation. Therefore, player's 2 objective of maximize such sum can be accomplished by a well-known $O(nm)$ dynamic-programming algorithm based on the recursive makespan definition presented in Section 1 which 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, PFS problem is analyzed as MMP problem considering that our objective is act as player 1. 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 chose by player 2.

3.3 Approximation guarantees of PFS solutions

A technique to obtain upper bounds on approximation guarantees of PFS solutions using double weighted sequences is exposed at this point. Consider that player 1 chose a permutation π over original matrix T, giving origin to matrix T^{π}. Assume w.l.o.g that $\pi = \langle 1, 2, \ldots, n \rangle$. Let T^{OPT} be the optimal permuted matrix of T and OPT its corresponding optimal permutation, P^{π} and P^{OPT} maximum paths over T^{π} and T^{OPT}, respectively, and A^{π} an anti-path of T^{π}. Construct a double weighted set (Γ, α, β) as follows: make $\Gamma = \{\pi(1), \pi(2), \ldots, \pi(n)\} = \{1, 2, \ldots, n\}$. Let $\alpha(i)$ be the sum of all P^{π} cells over column T^{π}_{i} and $\beta(i)$ be the sum of all A^{π} cells over the same column. From here to the end of this section, consider that (Γ, α, β) was constructed from permutation π, chosen by player 1.

Lemma 2 $C(S^{\pi}) = W(T^{\pi})$

Proof: Once $S^{\pi} = \langle 1, 2, \ldots, n \rangle$, the maximum weigth monotone subsequence of S^{π} is exactly the increasing subsequence S^{π}. Hence, $C(S^{\pi}) = \sum_{i=1}^{n} \alpha(i) = W(T^{\pi})$. ■

Theorem 3 *Let σ be an arbitrary permutation, T^{σ} the permutated matrix obtained applying σ to T and S^{σ} the sequence constructed applying σ to (Γ, α, β). Then $C(S^{\sigma}) \leq W(T^{\sigma})$*

Proof: Every weighted increasing subsequence of S^{σ}, taking α as weight function, is equivalent to a subsequence of a path in T^{σ} whose cells belong only to P^{π}. Similary, every weighted decreasing subsequence of S^{σ}, taking β 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}. Consider now a maximum path P^{σ} in T^{σ}. P^{σ} can be composed by cells in P^{π}, A^{π} and $T^{\sigma} - (P^{\pi} \cup A^{\pi})$. Then, $W(P^{\sigma}) = W(T^{\sigma}) \geq C(S^{\sigma})$ and the result follows. ■

Let $S^{\star}$ represent an optimal solution of `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,

Theorem 4 $\frac{W(T^{\pi})}{W(T^{OPT})} \leq \frac{C(S^{\pi})}{C(S^{\star})}$

Proof: $\frac{W(T^{\pi})}{W(T^{OPT})} \leq \frac{W(T^{\pi})}{C(S^{OPT})} \leq \frac{W(T^{\pi})}{C(S^{\star})} = \frac{C(S^{\pi})}{C(S^{\star})}$. ■

As consequence of last theorem it is possible to obtain an upper bound on the approximation guarantee of a PFS specific solution π by construction of an equivalent MDWS instance (Γ, α, β), as described in this section, and analysis of the approximation factor of any permutation π applied as solution to such instance.

4 The Greedy Avoided Path Algorithm

This section presents a polynomial time deterministic algorithm that construct a solution for PFS based on weigthed monotone subsequences properties previously explored. Time complexity and approximation guarantee of algorithm are also analyzed.

Theorem 5 *Greedy Avoided Path is an $2\sqrt{2n+m}$-approximation algorithm for PFS.*

Proof: Let π be the solution returned by `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 $(MaxMachine_j, j)$ in T belong to A^{π}. Once π was obtained by application of `Greedy Avoided Path` algorithm the construction of such anti-path in T^{π} is possible. Let T^{OPT} be an optimal permutated matrix of T. The approximation factor of solution π is, by definition, $W(T^{\pi})/W(T^{OPT})$. By the technique presented in section 3.3 is possible, from T^{π}, P^{π} and A^{π}, to construct a solution S^{π} for an instance (Γ, α, β) of MDWS problem such that $C(S^{\pi}) = W(T^{\pi})$. Consider that (Γ, α, β) was constructed following such technique.
Let $C(S^{\star})$ be an optimal solution for (Γ, α, β).
By Theorem 2: $C(S^{\star}) \geq \sum_{i=1}^{n} \alpha(i) / \sqrt{4\left(n + \sum_{i=1}^{n} \lceil \alpha(i)/\beta(i) \rceil\right)}$.
By A^{π} property, $\sum_{i=1}^{n} \lceil \alpha(i)/\beta(i) \rceil \leq n + m - 1$.

Consequently, $C(S^\star) \geq \sum_{i=1}^{n} \alpha(i)/2\sqrt{2n+m)} = C(S^\pi)/2\sqrt{2n+m}$.
By Theorem 4, $C(S^\star)/C(S^\pi) \geq W(T^\pi)/W(T^{OPT})$.

Hence, $W(T^\pi)/W(T^{OPT}) \leq 2\sqrt{2n+m}$. ■

Algorithm 1: Greedy Avoided Path

```
Input  : A set $J = \{J_1, J_2, ..., J_n\}$ of jobs, a set $M = \{M_1, M_2, ..., M_m\}$ of machines
         and a processing times matrix $P \in \Re^+_{J \times M}$. A cell $t_{ij} \in T$ represents the
         processing time of operation $O_{i,j}$.
Output: A permutation function $\pi : \{1, \ldots, n\} \mapsto J$.
1 for $j \leftarrow 1$ to $n$ do
2     $MaxTime_j \leftarrow t_{1j}$;
3     $MaxMachine_j \leftarrow 1$;
4     for $i \leftarrow 2$ to $m$ do
5         if $MaxTime_j < t_{ij}$ then
6             $MaxTime_j \leftarrow t_{ij}$ ;
7             $MaxMachine_j \leftarrow i$ ;
8 Construct a permutation $\pi : \{1, \ldots, n\} \mapsto J$ such that: $\forall a, b \in J$,
  $\pi^{-1}(a) \leq \pi^{-1}(b) \iff MaxMachine_a \geq MaxMachine_b$;
9 Return permutation $\pi$;
```

Time complexity analysis of `Greedy Avoided Path` algorithm can be done directly. Lines 1 to 8 can be executed in $\Theta(nm)$ time. Permutation construction on line 9 can be achieved sorting jobs in $\Theta(n \log n)$ time using *MaxMachine* variables as key. Line 9 costs $\Theta(n)$ steps. In resume, `Greedy Avoided Path` is a polynomial time $\Theta(nm + n \log n)$ algorithm.

5 Conclusion

This work presented a deterministic approximation algorithm for PFS with performance ratio $2\sqrt{2n+m}$ and time complexity $\Theta(nm + n \log n)$. In the case that $n = O(m)$ this is the best approximation algorithm already obtained for PFS. The Erdös-Szekeres Theorem was extended, considering a weighted version in which elements of monotone subsequences can have different weights. As consequence, a novel technique to obtain upper bounds on approximation guarantees of PFS solutions using double weighted sequences was introduced, exploring the correlation between Weighted Monotone Subsequence Problems and PFS.

References

[1] R. Bar-Yehuda, S. Fogel, *Partitioning a sequence into few monotone subsequences*, Acta Inform. 35 (1998), 421-440.

[2] P. Erdös and 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 Journal of Operational Research 169 (2006), 865-872.

[4] M. R. Garey, D. S. Johnson and R. Sethi, *The Complexity of Flowshop and Jobshop Scheduling*, Math. Oper. Res. 1 (1976), 117-129.

[5] T. Gonzalez, S. Sahni, *Flowshop and jobshop schedules: complexity and approximation*, Operations Research 26 (1978), 36-52.

[6] L. A. Hall, *Approximability of flow shop scheduling*, Math. Program. 82 (1998), 175-190.

[7] S. M. Johnson, *Optimal two- and three-stage production schedules with setup times included*, Naval Res. Logist. Quart. 1 (1954), 61-68.

[8] E. Nowicki and C. Smutnicki, *Worst-case analysis of an approximation algorithm for flow- shop scheduling*, Oper. Res. Lett. 8 (1989), 171-177.

[9] E. Nowicki and C. Smutnicki, *Worst-case analysis of Dannenbring's algorithm for flow-shop scheduling*, Oper. Res. Lett. 10 (1991), 473-480.

[10] E. Nowicki and C. Smutnicki, *New results in the worst-case analysis for flow-shop scheduling*, Discrete Appl. Math. 46 (1993), 21-41.

[11] C. Potts, D. Shmoys and D. Williamson, *Permutation vs. nonpermutation flow shop schedules*, Operations Research Letters 10 (1991), 281-284.

[12] H. Röck and G. Schmidt, *Machine aggregation heuristics in shop-scheduling*, Methods of Operations Research 45 (1983), 303-314.

[13] J. M. Steele, *Variations on the monotone subsequence theme of Erdös and Szekeres*, IMA Vol. Math. Appl. 72 (1995), 111-131.

[14] M. Sviridenko, *A Note on Permutation Flow Shop Problem*, Annals of Operations Research v. 129 (2004), 247-252.

[15] D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov, D. B. Shmoys, *Short shop schedules*, Oper. Res. 45 (1997), 288-294.

[16] K. Wagner, *Monotonic coverings of finite sets*, Elektron. Informationsverarb. Kybernet., 20 (1984), 633-639.