2
A Linear Time Approximation Algorithm for PFS

2.1 Introduction

In the last fifty years, PFS has been a central and well-studied problem in scheduling community, known by its intractability, from theoretical and practical aspects. Since PFS was proved to be *Strongly NP-Hard* by Garey, Johnson and Sethi [20], a considerable amount of work has been employed in finding good approximation algorithms for this problem.

The purpose of this chapter is to introduce a new approximation algorithm for the PFS problem. This algorithm achieves an approximation guarantee of $2\sqrt{2n+m}$ and runs in linear time over the instance size. This is the best performance ratio already obtained for the PFS problem in the case of $n = \Theta(m)$. Furthermore, a novel connection between PFS and monotone subsequence problems is established, resulting on an extension of the Erdös-Szekeres theorem to weighted monotone subsequences.

(a) Previous Results

Gonzalez and Sahni [23] showed that every *busy scheduling* for PFS has an approximation factor of m times the optimal solution. Nowicki and Smutnicki [43, 44, 45] explored worst-case analysis on the approximation factor of several PFS algorithms, achieving a tight bound of $\lceil m/2 \rceil$ for all them. Potts, Shmoys and Williamson [50] proved the existence of some instances for which non-permutation based solutions are $\Omega(\sqrt{m})$ less costly than permutation ones. Sevastjanov [58] developed geometric methods to analyze scheduling problems, including PFS, introducing an algorithm that always produces a permutation schedule with additive factor bounded by $O(m^2)\max\{t_{ij}\}$. Hall [30] presented a PTAS for PFS when $m = 3$. Sviridenko [62] introduced a randomized

algorithm for PFS based on Chernoff Bounds arguments with a performance ratio of $O(\sqrt{m \log m})$ and an additive factor limited to $O(m \log m) \max\{t_{ij}\}$. Approximation ratios for a large number of PFS heuristics were surveyed by Gupta, Koulamas and Kyparisis[27].

The best known up to date approximation algorithm for PFS, due to Nagarajan and Sviridenko [40], has a performance ratio of $O(\sqrt{\min\{n, m\}})$ and is based on a connection between 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 [51]. Finally, this algorithm answers an open question from Potts, Shmoys and Williamson [50] matching the gap between permutation and non-permutation schedules optimal solutions. Due to the relevance of the approximation factor obtained, the time complexity analysis of the deterministic approximation algorithm from Nagarajan and Sviridenko [40] was not explicitly given in their paper. In the appendix 2.6 of this chapter, a lower bound of $\Omega(n^4.m)$ is established for it.

(b) Contribution and Organization

The objective of this work is to present a simple and intuitive deterministic approximation algorithm for PFS with performance ratio $2\sqrt{2n + m}$ and time complexity $\Theta(nm)$. These results were achieved independently [60] and in parallel with the ones of Nagarajan and Sviridenko, as mentioned in their published paper [41](section 1.2). In the case that $n = \Theta(m)$ this is the best approximation algorithm already obtained for PFS, achieving the same approximation factor of [40], but in linear time complexity. A novel technique for performance guarantee analysis of PFS solutions is also developed, exploring the relationship between weighted monotone subsequence problems and PFS. We prove that a job permutation reaching the claimed approximation guarantee has the cells representing the longest operation of each job comprising a specific path on the processing times 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 some extensions of the Erdös-Szekeres Theorem, from which the lower bounds on optimal solutions are obtained.

The main idea considered in work involves the analysis of PFS by a new perspective related to matrix games and monotone subsequences. First of all,

PFS is considered 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 two must follow a specific property, composing what we call a *path*. The objective of player one is to find a permutation that turns the task of player two difficult. We argue that this game, with the objective of acting as player one, is equivalent to PFS.

Once the problem is defined, a natural strategy to player would be to choose a permutation that avoids cells with high weights on a path for player two. 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 to the Minimum Double Weighted Sequence Problem. This last problem is a generalization of the classical problem considered by Erdös and Szekeres [12] in which every sequence element has two associated weights, one if it is considered in increasing subsequences and other if considered in decreasing ones, and the objective is to define a sequence that minimizes the maximum weight of its increasing or decreasing subsequences. We provide extensions of 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 work is organized as follows. In section 2.2 we introduce the notion of weighted and double weighted sequences, proving an extension of Erdös-Szekeres Theorem applied to these concepts. Section 2.3 first presents 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 2.4, the approximation algorithm Greedy Avoided Path is proposed and analyzed. Final conclusions are drawn in section 2.5.

2.2 Weighted Sequences

(a) 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 $s_{\varphi_1} > s_{\varphi_2} > \ldots > s_{\varphi_m}$.*

The classical theorem of Erdös and Szekeres [12] states that from a sequence of n^2+1 distinct real elements is always possible extract a monotone subsequence of cardinality at least $n+1$.

Theorem 2 *Every sequence S of n^2+1 distinct elements has a monotone subsequence of at least $n+1$ elements.*

Proof: Let a_i be the i-th element from S and $f(a_i) : S \mapsto |S|$ a function that returns the number of elements of the longest increasing subsequence starting from a_i, where $i \in \{1, 2, \ldots, n^2+1\}$. If $f(a_i) \geq n+1$ for some i, then the theorem is proved. Otherwise, $1 \leq f(a_i) \leq n$, for all i. In this case we have, by the pigeonhole principle, at least $\lceil \frac{n^2+1}{n} \rceil$ elements a_i with the same value for $f(a_i)$. Let $S' = \langle a_{j_1}, a_{j_2}, \ldots, a_{j_{n+1}} \rangle$ be a subsequence of S composed by $n+1$ of these elements. We shall prove now that S' is a decreasing subsequence. Suppose this is a fallacy. Then, there are two elements a_i and a_j belonging to S', $i < j$, for which $f(a_i) = f(a_j)$ and $a_i < a_j$. Therefore, it is possible to obtain a subsequence of S with $f(a_j)+1$ elements starting from a_i and being followed by the $f(a_j)$ elements of a longest increasing subsequence starting from a_j. Hence, $f(a_i) \geq f(a_j)+1$, what is a contradiction to the hypothesis that $f(a_i) = f(a_j)$. ■

Example: In order to illustrate an application of the previous theorem let us consider the sequence $S = \langle 5, 9, 3, 7, 4, 10, 1, 8, 2, 6 \rangle$. By the theorem of Erdös and Szekeres, S should have a monotone subsequence of at least 4 elements. Following the definition of the function $f(a_i) : S \mapsto |S|$ given on the proof of theorem 2: $f(5) = 3, f(9) = 2, f(3) = 3, f(7) = 2, f(4) = 2, f(10) = 1, f(1) = 3, f(8) = 1, f(2) = 2, f(6) = 1$. Therefore, the longest increasing subsequence of S has length 3. However, the subsequence $S' = \langle 9, 7, 4, 2 \rangle$, formed by the elements a_i from S with $f(a_i) = 2$, is a decreasing subsequence of 4 elements.

Definition 3 *A set $T_1, T_2, \ldots, T_k$ of monotone subsequences of a sequence S is said to be a S-monotone partition of size k if $\bigcup_{i=1}^{k} T_i = S$ and $\bigcap_{i=1}^{k} T_i = \emptyset$.*

The maximum cardinality monotone subsequence problem can be solved in polynomial time. However, finding a minimum size monotone partition is a NP-Hard problem [65]. Bar-Yehuda and Fogel [3] presented an approximation algorithm for minimum size monotone partition based on the following Lemma, which can be proved directly by successive removal of maximum cardinality monotone subsequences of S:

Lemma 4 *[3] Let $S = \langle s_1, s_2, \ldots, s_n \rangle$ be a sequence. There is a S-monotone partition of size at most $2\sqrt{n}$.*

At this point, we define the notion of weighted monotone subsequence and extend the Erdös-Szekeres Theorem applied to such new concept.

Definition 5 *Let $w : S \mapsto \Re^+$ be a weight function over a sequence S. The weight of a subsequence $T = \langle s_{\varphi_1}, s_{\varphi_2}, \ldots, s_{\varphi_l} \rangle$ of S, denoted by $w(T)$, is $\sum_{i=1}^{l} w(s_{\varphi_i})$. Denote by $w(T_{max})$ the maximum weight on a monotone subsequence of S.*

Corollary 6 $w(T_{max}) \geq \frac{w(S)}{2\sqrt{n}}$.

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, $\sum_{i=1}^{k} w(T_i) = w(S)$. 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 $w(T^\star) \leq w(T_{max})$ the result is proved. ■

A survey on Erdös-Szekeres theorem and its variations was presented by Steele [61]. From the best we know, weighted monotone subsequence concept has not been explored so far.

(b) Double Weighted Sequences

Definition 7 *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 8 *Let* (Γ, α, β) *be 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).

Definition 9 *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* $C(S_\beta)$ *denotes the weight of* S_β. *The weight of the double weighted sequence* S, *denoted by* $C(S)$, *is defined as* $\max\{C(S_\alpha), C(S_\beta)\}$.

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 $C(S) = \max\{C(S_\alpha), C(S_\beta)\} = 25$.

Now we are ready to define the Minimum Double Weighted Sequence Problem:

The Minimum Double Weighted Sequence Problem (MDWS):
Given a double weighted set (Γ, α, β), construct a double weighted sequence $S^\star$ such that $C(S^\star)$ is minimum.

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 10 *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 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'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 $\beta_1(\gamma_3) = \beta_2(\gamma'_3) = \beta_2(\gamma'_4) = 9$, 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 11 *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 $\Phi_1 = \langle \gamma_{\phi_1(1)}, \gamma_{\phi_1(2)}, \ldots, \gamma_{\phi_1(n)} \rangle$. Construct a solution Ψ to D_2 from Φ_1 as follows: let $\gamma_i = \gamma_{\phi_1(k)}$ 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, $\alpha(\gamma_i) = \alpha(\gamma'_j) + \alpha(\gamma'_{j+1})$, every increasing subsequence of Ψ can be transformed into an increasing subsequence of Φ_1, with not smaller weight. 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, $\beta(\gamma_i) = \beta(\gamma'_j) = \beta(\gamma'_{j+1})$. Hence, $C(\Psi) \leq C(\Phi_1)$. Once $C(\Phi_2) \leq C(\Psi)$, the result follows. ■

The use of the split concept in conjunction with Corollary 6 permits obtaining a lower bound on optimal solutions of a `MDWS` instance.

Theorem 12 *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, $\sum_{i=1}^{n} \lceil \alpha_1(i)/\beta_1(i) \rceil$ splits are sufficient, what 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 10, $C(\Phi_1) \geq C(\Phi_2)$. 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 6:

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

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

2.3 Lower Bounds for a Matrix Game

This section introduces the `Matrix Min-Max Path Problem`, which is exactly `PFS` viewed from a game perspective. A technique to construct lower bounds on `Matrix Min-Max Path Problem` optimal solutions based on its transformation into the `Minimum Double Weighted Sequence Problem` is presented.

(a) 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 13 *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 $\sum_{i=1}^{n+m-1} p_i$. P is said a maximum weight path if $W(P) \geq W(P')$ for every path P' over T^{π}.

Definition 14 *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* $\sum_{i=1}^{n+m-1} a_i$.

(b) 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 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 Matrix Min-Max 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 an $O(nm)$ dynamic-programming algorithm based on the recursive makespan definition presented at section 1.2 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 to act as player 1, with the objective pay the minimum possible value to player 2. Furthermore, due to the polynomial time algorithm that calculates the maximum path (makespan) on a permutated matrix, it will be considered that player 2 always select this maximum path.

(c) Approximation guarantees of PFS solutions

A technique to obtain upper bounds on approximation guarantees of PFS solutions using double weighted sequences is presented at this point. Consider that player 1 chose a permutation π over original matrix T, creating the 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. Furthermore, let $S^\star$ represent an optimal solution of `Minimum Double Weighted Sequence Problem` for (Γ, α, β).

Lemma 15 $C(S^\pi) = W(T^\pi)$

Proof: Once $S^\pi = \langle 1, 2, \ldots, n \rangle$, the maximum weight 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 16 *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. Similarly, 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. Therefore, $W(T^\sigma) \geq C(S^\sigma)$. ■

Corollary 17 $C(S^\star) \leq W(T^{OPT})$

Proof: By Theorem 16, $C(S^{OPT}) \leq W(T^{OPT})$. By optimal solution definition, $C(S^\star) \leq C(S^{OPT})$. Consequently, $C(S^\star) \leq W(T^{OPT})$. ■

From the previous results we have,

Theorem 18 $\frac{W(T^\pi)}{W(T^{OPT})} \leq \frac{C(S^\pi)}{C(S^\star)}$

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 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 (Γ, α, β), as described in this section, and analyzing the approximation factor of any permutation π applied as solution to such instance.

2.4 The Greedy Avoided Path Algorithm

This section presents a polynomial time deterministic algorithm which constructs a solution for PFS based on weighted monotone subsequences properties previously explored. Time complexity and approximation guarantee of 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 times 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, introduced in last sections.

Algorithm 1: Greedy Avoided Path

begin

for each job $j \in \mathbb{J}$: **set** $max\_machine(j)$ as the index of the machine with the longest operation of job j ;

construct a permutation π by sorting the jobs in non-increasing order of $max\_machine(j)$ variables ;

return permutation π. ;

end

Theorem 19 *Greedy Avoided Path is an* $2\sqrt{2n+m}$*-approximation algorithm for* PFS *and can be executed in* $\Theta(nm)$ *time.*

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 at section 2.3(c) it 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 12: $C(S^\star) \geq \sum_{i=1}^{n} \alpha(i) / \sqrt{4\left(n + \sum_{i=1}^{n} \lceil \alpha(i)/\beta(i) \rceil\right)}$.

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 18, $C(S^{\pi})/C(S^{\star}) \geq W(T^{\pi})/W(T^{OPT})$.

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

Time complexity analysis of `Greedy Avoided Path` algorithm is straightforward once observing 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, `Greedy Avoided Path` is a polynomial time $\Theta(nm)$ algorithm. ■

2.5 Conclusion

This work presents a deterministic approximation algorithm for `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 `PFS` achieving the same approximation factor found by Nagarajan and Sviridenko [40] 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 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 `PFS`.

2.6 Appendix

The algorithm from Nagarajan and Sviridenko [40] 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. On the selection of such edge-coloring algorithm it is important to observe that the number of edges on the bipartite multigraph representing the original processing time matrix would be *pseudopolynomial* in the input size. From the best we know, the most efficient algorithms to solve this problem are *weakly polynomial time algorithms* depending on a *log* factor of the maximum degree of a vertex [57]. In particular, the well-performing algorithm from Gabow

and Kariv [17] 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 can not 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 takes at least $\Omega(n)$. Hence, a 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 [40] is lower bounded by $\Omega(n^4.m)$.