Tight Bounds for Permutation Flow Shop Scheduling

Viswanath Nagarajan
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213.
email: viswa@cmu.edu

Maxim Sviridenko
IBM T.J. Watson Research center, Yorktown Heights, NY 10598.
email: sviri@us.ibm.com

In flow shop scheduling there are m machines and n jobs, such that every job has to be processed on the machines in the fixed order $1, \ldots, m$. In the *permutation flow shop* problem, it is also required that each machine processes the set of all jobs in the *same* order. Formally, given n jobs along with their processing times on each machine, the goal is to compute a single permutation of the jobs $\sigma : [n] \to [n]$, that minimizes the maximum job completion time (makespan) of the schedule resulting from σ. The previously best known approximation guarantee for this problem was $O(\sqrt{m \log m})$ [30]. In this paper, we obtain an improved $O(\min\{\sqrt{m}, \sqrt{n}\})$ approximation algorithm for the permutation flow shop scheduling problem, by finding a connection between the scheduling problem and the longest increasing subsequence problem. Our approximation ratio is relative to the lower bounds of maximum job length and maximum machine load, and is the best possible such result. This also resolves an open question from [21], by algorithmically matching the gap between permutation and non-permutation schedules.

We also consider the weighted completion time objective for the permutation flow shop scheduling problem. Using a natural linear programming relaxation, and our algorithm for the makespan objective, we obtain an $O(\min\{\sqrt{m}, \sqrt{n}\})$ approximation algorithm for minimizing the total weighted completion time, improving upon the previously best known guarantee of εm for any constant $\varepsilon > 0$ [31]. We give a matching lower bound on the integrality gap of our linear programming relaxation.

Key words: approximation algorithms; shop scheduling; increasing subsequence

MSC2000 Subject Classification: Primary: 90B35; Secondary: 68W25

OR/MS subject classification: Primary: Production, scheduling; Secondary: approximations, heuristic

---

1. Introduction In the *flow shop* problem, there are m machines located in a fixed order (say, 1 through m), and n jobs each of which consists of a sequence of operations on machines (in the order 1 through m). For any job $j \in \{1, \ldots, n\}$ and machine $i \in \{1, \ldots, m\}$ the length of the operation of job j on machine i is called its *processing time* p_{ij}. A schedule for the jobs is feasible if (i) each machine processes at most one job at any time; (ii) for each job, its operations on the machines are processed in the fixed order 1 through m; and (iii) each operation (of a job on a machine) is processed without interruption. The flow shop problem is a special case of *acyclic job shop* scheduling [5, 6], which in turn is a special case of the general *job shop* scheduling [3, 15].

We study the *permutation flow shop* problem, which is the flow shop problem with the additional constraint that each machine processes all the jobs in the *same* order. So any feasible schedule to the permutation flow shop problem corresponds to a permutation of the n jobs. It is well-known [4] that there exists an optimal schedule for the ordinary flow shop problem having the same processing order (of jobs) on the first two machines and the same processing order on the last two machines. So the permutation flow shop problem is equivalent to the ordinary flow shop problem for $m \leq 3$ machines. However, it is easy to construct an instance with $m = 4$ machines where no permutation schedule is optimal for the ordinary flow shop problem.

Two natural objective functions that are typically considered for scheduling problems are *makespan* and *weighted completion time*. The makespan of a schedule is the completion time of its last job, i.e. maximum completion time among all jobs. For the weighted completion time objective, each job j comes with a weight $w_j \geq 0$, and the weighted completion time of a schedule is the weighted sum of completion times over all jobs.

1.1 Related Work When the number of machines m is a fixed constant, a PTAS is known for the job-shop scheduling problem with the makespan objective due to Jansen et al. [12] and the total weighted completion time objective due to Fishkin et al. [7]. It seems quite likely, that similar techniques yield PTASes for the permutation flow shop scheduling problems with corresponding objective functions, although such results did not appear in the literature. For the ordinary flow shop problem with the

makespan objective, the best known approximation guarantee is $O(\log m(\log\log m)^{1+\epsilon})$, where $\epsilon > 0$ is any constant, due to Czumaj and Scheideler [5]; in fact this result holds for the more general class of acyclic-shop scheduling. Using the general result from [23] one can derive an approximation algorithm with the analogous performance guarantee for the flow shop scheduling problem with the total weighted completion time objective.

The following are two obvious lower bounds for the permutation flow shop scheduling problem with the makespan objective: maximum job length $l = \max_{j=1}^{n}\{\sum_{i=1}^{m} p_{i,j}\}$, and maximum machine load $L = \max_{i=1}^{m}\{\sum_{j=1}^{n} p_{i,j}\}$. Potts et al. [21] have shown a family of instances where the optimal makespan is $\Omega(\sqrt{\min\{m,n\}})$ times the trivial lower bound ($\max\{L,l\}$). It was an open question in [21] to determine whether this bound is tight. The previously best known approximation guarantee for the makespan problem is $O(\sqrt{m\log m})$ due to Sviridenko [30]; this guarantee is relative to the trivial lower bound. Prior to this, a number of algorithms [25, 19, 18] were shown to have a (tight) worst case guarantee of $\lceil\frac{m}{2}\rceil$. There are also some papers dealing with additive guarantees for this problem [26, 30]. Sevastianov [26] gave an algorithm that always produces a schedule of length at most $L+O(m^2)\max_{i,j} p_{i,j}$. Sviridenko [30] obtained a similar guarantee of $(1+\delta)L + K_\delta(m\log m)\max_{i,j} p_{i,j}$ for any $\delta > 0$ (here K_δ is a function depending on δ alone). The best multiplicative approximation guarantee obtainable from these results is the $O(\sqrt{m\log m})$ [30].

Smutnicki [31] gave a worst case analysis of several algorithms for the permutation flow shop problem with the weighted completion time objective. Assuming a ρ_k approximation guarantee for the problem on k machines, [31] gave an $\frac{m}{k}\rho_k$ approximation algorithm for the problem on m machines. Assuming that there exists a PTAS for the permutation flow shop scheduling problem with the weighted completion time objective and fixed number of machines, one could obtain an ϵm guarantee for minimizing weighted completion time (for any constant $\epsilon > 0$). Alternatively, we could use the $(2+\varepsilon)$-approximation algorithm from [28] that works basically for any shop scheduling problem with makespan criteria on constant number of machines. Otherwise, the best known guarantee is m. To the best of our knowledge this is the previously best known result for this problem.

The permutation flow shop scheduling problem has been very popular in the Operations Research community in the last 40 years and there is a significant body of work devoted to the design of heuristics for this problem. The survey paper [8] establishes a common framework for these heuristics, and describes main classes of algorithms and research directions.

1.2 Our Results and Paper Outline We give a simple randomized algorithm (Section 2) for minimizing makespan that achieves an approximation guarantee of $2\sqrt{\min\{m,n\}}$. This guarantee is relative to the trivial lower bounds. The analysis is based on the connection between the permutation flow shop scheduling problem and the longest increasing subsequence problem. This connection is new and might be interesting in its own right. It also allows us to apply non-trivial probabilistic results on the expected value and concentration of the longest increasing subsequence in a random sequence [16, 32, 9].

Hence we answer the open question in Potts et al. [21], by matching algorithmically the $\Omega(\sqrt{\min\{m,n\}})$ gap shown in [21]. We also show how this algorithm can be derandomized to obtain a deterministic approximation guarantee of $3\sqrt{\min\{m,n\}}$. This algorithm uses the derandomization technique of pessimistic estimators due to Raghavan [24] and certain ideas from the proof of concentration bound from [9]; the details are non-trivial and appear in Section 3. We note that among algorithms that are based on the trivial lower bounds, our algorithm is the best possible (up to a $2\sqrt{2}$ factor).

On the first sight our improvement of $O(\sqrt{\log m})$ upon the previously best known bound from [30] looks insignificant, but the proof in [30] is based on Chernoff Bounds and it is quite well-known that improving upon Chernoff Bounds based proofs requires substantially new insights on the problem structure. For example, for the famous combinatorial discrepancy problem the straightforward randomized algorithm yields the bound of $O(\sqrt{n\log n})$; but using more sophisticated techniques based on entropy and using a pigeon-hole principle, Spencer proved a better non-constructive bound of $O(\sqrt{n})$ [2]. It is one of the well-known open questions to obtain a constructive (algorithmic) proof of Spencer's result; it is also unknown if Spencer's guarantee holds in the case when the matrix entries are arbitrary real numbers in the interval $[0,1]$ while Chernoff bounds can be easily applied even in this more general case. The key to our result is a decomposition of the matrix of processing times into a sum of permutation matrices and noticing that the "intersection" of each such matrix with any critical path corresponds to an increasing

subsequence in a certain permutation.

Our second contribution is a partial explanation of great practical performance of the simple greedy algorithm for the permutation flow shop problem with the makespan objective. This algorithm was first suggested by Nawaz, Enscore and Ham [17] and is also known under the name of "insertion heuristic". This algorithm initially orders jobs in the decreasing order of the job lengths and inserts them one by one into the current schedule, always choosing the best position for a job in the current permutation. Although this algorithm is very simple and has superb practical performance [8], there is no analytical explanation of this phenomenon. Many practical algorithms either directly use the insertion heuristic or generalize it in some way. The natural way of analyzing such a heuristic would be to define a potential function which is improved on every step of the greedy procedure and prove some bound relating this function with the makespan. Although we were not able to apply this method to the insertion heuristic, our derandomization algorithm follows this framework. Moreover, our final derandomized algorithm resembles the greedy algorithm of Nawaz, Enscore and Ham [17]. The difference is that our algorithm greedily fixes the first few positions in the current permutation with respect to a certain potential function (derived from the concentration bound on length of the longest increasing subsequence [9]), while the greedy algorithm [17] just fixes a relative ordering of the first few jobs allowing unscheduled jobs to be scheduled in between later on.

We then consider the weighted completion time objective (Section 4) and use our algorithm for minimizing makespan to obtain an $O(\sqrt{\min\{m,n\}})$ approximation algorithm for this problem. This algorithm uses the linear relaxation of a natural integer programming formulation for the problem. Our rounding algorithm is similar to the approach used in Queranne and Sviridenko [23] (and many other papers on scheduling with the weighted completion time objective [11, 1, 10]); the difference is that we need to ensure that when we apply the approach of geometric partitioning of the time interval, the schedule for each such interval satisfies the permutation constraint. We also show a matching $\Omega(\sqrt{\min\{m,n\}})$ lower bound on the integrality gap of our LP relaxation.

Recently and independently of our work, Sotelo and Poggi de Aragao [29] designed a deterministic approximation algorithm for the permutation flow shop problem with makespan criteria. The performance guanrantee of their algorithm is $O(\sqrt{n})$ and is slighly worse than ours. Although the algorithm and analysis seem to be different from ours they also use the connection of the permutation flow shop problem and increasing subsequences in permutations.

1.3 Preliminaries An instance of the permutation flow shop problem with m machines and n jobs is given by an $m \times n$ matrix $P = \{p_{i,j} \mid i = 1, \cdots, m,\ j = 1, \cdots, n\}$ of processing times, where $p_{i,j}$ is the processing time of job j on machine i. We often denote the set $\{1, \cdots, n\}$ of all jobs by $[n]$, and the set $\{1, \cdots, m\}$ of all machines by $[m]$. Any feasible schedule for permutation flow shop corresponds to a permutation of the n jobs. Given a permutation $\pi : [n] \to [n]$ of jobs, the complete schedule of job-operations on machines can be obtained by running jobs on machines in the order π while maintaining the minimum possible wait-time between operations. It is convenient to think of π as a mapping from the set of n possible positions to the set of n jobs. Therefore, $\pi(p)$ denotes the job located at position p. For any permutation π of the n jobs and a job $j \in [n]$, we use C_j^π to denote the completion time of job j under the schedule π; we also denote the makespan of schedule π by $C_{max}^\pi = \max_{j=1}^n C_j^\pi$. Given non-negative weights $\{w_j\}_{j=1}^n$ for the jobs, the weighted completion time objective of the schedule corresponding to permutation π is $\sum_{j=1}^n w_j C_j^\pi$.

A *monotone path* (or *critical path*) in an $m \times n$ matrix is defined to be a sequence $\langle (x_1, y_1), \cdots, (x_t, y_t) \rangle$ of matrix positions such that $(x_1, y_1) = (1, 1)$, $(x_t, y_t) = (m, n)$, and for each $1 \leq i < t$ either $x_{i+1} = x_i + 1$ & $y_{i+1} = y_i$ or $x_{i+1} = x_i$ & $y_{i+1} = y_i + 1$. In particular, this definition implies that each monotone path in an $m \times n$ matrix consists of exactly $t = m + n - 1$ positions. We denote the set of all monotone paths in an $m \times n$ matrix by $\mathcal{M}_{m,n}$.

A map $\tau : [n] \to X \cup \{\phi\}$ where $X \subseteq [m]$ is called a *partial permutation* if there is a subset $Y \subseteq [n]$ with $|Y| = |X|$ such that (i) $\tau(Y) = X$ (τ is a one-to-one map from Y to X); and (ii) $\tau(z) = \phi$ for all $z \in [n] \setminus Y$. For such a partial permutation τ, we refer to the set X as its *image*, denoted $\mathsf{Image}(\tau)$. A 0-1 $m \times n$ matrix Π is called a *permutation matrix* if every row and column has at most one entry that is 1 (all other entries are 0s). Note that there is an obvious correspondence between partial permutations and permutation matrices. In the rest of the paper we will use partial permutations that map a subset

of jobs into a set of machines.

2. Randomized Algorithm for Minimizing Makespan In this section, we give a randomized $\Theta(\sqrt{\min\{m,n\}})$ approximation guarantee for minimizing makespan in the permutation flow shop problem. From the results of Potts et al. [21], it follows that this result is the best possible using the known lower bounds for this problem (namely, machine load & job length). Our algorithm is extremely simple: always output a permutation chosen uniformly at random. The rest of this section proves that this algorithm achieves a guarantee of $2\sqrt{\min\{m,n\}}$.

Given any instance of permutation flow shop, consider the $m \times n$ matrix P of processing times. We first show how P can be decomposed into a collection of smaller matrices having certain properties.

LEMMA 2.1 *Given any matrix $P \in \mathbb{N}_{m\times n}$, there exist $h = \max\{l, L\}$ permutation matrices $\{\Pi_k\}_{k=1}^{h}$ such that $P = \sum_{k=1}^{h} \Pi_k$, where $l = \max_{j=1}^{n}\{\sum_{i=1}^{m} p_{i,j}\}$ and $L = \max_{i=1}^{m}\{\sum_{j=1}^{n} p_{i,j}\}$.*

PROOF. Define a bipartite multi-graph graph G corresponding to P as follows. G has vertex bipartition $[m]$ and $[n]$. For every $i \in [m]$ & $j \in [n]$, G contains $p_{i,j}$ parallel edges between i & j. Note that the maximum degree of G is exactly $h = \max\{l, L\}$. By the König edge-coloring theorem for bipartite graphs there is a valid coloring of the edges of G (no two adjacent edges receive the same color) with h colors. Let $E_1, \cdots, E_h$ denote the edges in each color class of such a valid coloring. For each $1 \le k \le h$, let Π_k denote the $m \times n$ 0-1 matrix that contains 1s in the positions corresponding to edges of E_k, and 0s everywhere else. Since we have a valid coloring, each E_k is a matching in G; in other words, the matrices $\{\Pi_k\}_{k=1}^{h}$ are all permutation matrices. Further, since each edge of G is assigned some color, we have $P = \sum_{k=1}^{h} \Pi_k$. □

Recall that $\mathcal{M}_{m,n}$ denotes the set of all monotone paths in an $m \times n$ matrix. In this section, m and n are fixed; so we abbreviate $\mathcal{M} = \mathcal{M}_{m,n}$. For any permutation σ of the n jobs, monotone paths can be used to characterize the makespan of the schedule resulting from σ as follows:

$$C^{\sigma}_{max} = \max_{\alpha\in\mathcal{M}} \sum_{(i,q)\in\alpha} p_{i,\sigma(q)}.$$

This well-known characterization follows from the fact that the makespan C^{σ}_{max} is lower bounded by the total length of any monotone path since every two consecutive operations in such a path are either consecutive operations of some job or consecutive operations on the same machine. It is also easy to build a monotone path that attains the equality. Starting with the operation that finishes last (on the last machine) include all operations preceding to that operation on the same machine till the last idle time. Let J_j be the job whose m-th operation starts on the last machine after that idle time. Include the $m-1$st operation of job J_j into the monotone path. Include all operations preceding that operation on machine $m-1$ till the last idle time before that operation and so on.

Consider any permutation matrix Π_k $(k = 1, \cdots, h)$ in the decomposition of Lemma 2.1. Let the 1-entries of Π_k be in positions $\{(x^k_1, y^k_1), \cdots, (x^k_r, y^k_r)\}$, where $1 \le x^k_1 < \cdots < x^k_r \le m$ and $y^k_1, \cdots, y^k_r \in [n]$ are distinct elements. Denote $X_k = \{x^k_1, \cdots, x^k_r\}$ and $Y_k = \{y^k_1, \cdots, y^k_r\}$; clearly, $|X_k| = |Y_k| = r \le \min\{m, n\}$. Define the map $\tau_k : [n] \to X_k \cup \{\phi\}$ where $\tau_k(y^k_g) = x^k_g$ (for all $1 \le g \le r$) and $\tau_k(z) = \phi$ for $z \notin Y_k$. Since each Π_k is a permutation matrix, it follows that the τ_k is a partial permutation for $k = 1, \ldots, h$.

Finally, for any sequence S of elements from $[m] \cup \phi$, define $I(S)$ to be the length of the longest increasing subsequence of numbers in S (ignoring all occurrences of the null element ϕ).

LEMMA 2.2 *For any permutation σ on jobs and any monotone path $\alpha \in \mathcal{M}$,*

$$\sum_{(i,q)\in\alpha} p_{i,\sigma(q)} \le \sum_{k=1}^{h} I(\tau_k \circ \sigma[n])$$

PROOF. Clearly we have:

$$\sum_{(i,q)\in\alpha} p_{i,\sigma(q)} = \sum_{(i,q)\in\alpha} [\sum_{k=1}^{h} \Pi_k(i, \sigma(q))] = \sum_{k=1}^{h} \sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q))$$

Now consider a particular permutation matrix Π_k (for $k = 1, \cdots, h$) and the sum $\sum_{(i,q)\in\alpha} \Pi_k(i, \sigma(q))$.

Let $S_k = \{(i,q) \in \alpha \mid \Pi_k(i,\sigma(q)) = 1\}$; then the sum $\sum_{(i,q)\in\alpha} \Pi_k(i,\sigma(q)) = |S_k|$. Since Π_k has at most one non-zero entry in each row and column (given by the partial permutation τ_k) and α is a monotone path, we obtain that $S_k = \{(i_1,q_1), \cdots, (i_t,q_t)\}$ (where $t = |S_k|$), with the following properties:

(i) $1 \le i_1 < \cdots < i_t \le m$ and $\{i_1, \cdots, i_t\} \subseteq X_k$.
(ii) $1 \le q_1 < \cdots < q_t \le n$.
(iii) $\tau_k(\sigma(q_g)) = i_g$ for all $1 \le g \le t$.

From the above, we have that $i_1 < i_2 < \cdots < i_t$ is an increasing subsequence of length t in the sequence $\tau_k \circ \sigma[n] = \langle \tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(n) \rangle$; namely given by the positions $q_1 < q_2 < \cdots < q_t$. Thus the longest increasing subsequence in $\tau_k \circ \sigma[n]$ has length at least $|S_k|$. In other words, $\sum_{(i,q)\in\alpha} \Pi_k(i,\sigma(q)) \le I(\tau_k \circ \sigma[n])$. Summing this expression over all permutation matrices Π_k for $k = 1, \cdots, h$, we obtain the statement of the Lemma. □

Note that the right hand side in the inequality in Lemma 2.2 does not depend on the monotone path α; hence we obtain that $C^\sigma_{max} = \max_{\alpha\in\mathcal{M}} \sum_{(i,q)\in\alpha} p_{i,\sigma(q)}$
$\le \max_{\alpha\in\mathcal{M}} \sum_{k=1}^{h} I(\tau_k \circ \sigma[n]) = \sum_{k=1}^{h} I(\tau_k \circ \sigma[n])$. We will also need the following:

Lemma 2.3 (Logan & Shepp [16]; Vershik & Kerov [32]) *The expected length of the longest increasing subsequence of a uniformly random permutation on r elements is $(2 + o(1))\sqrt{r}$.*

We are now ready for the main theorem of this section.

Theorem 2.1 *$E_\sigma[C^\sigma_{max}] \le (2 + o(1))h \cdot \sqrt{\min\{m,n\}}$. Hence there is a randomized polynomial time $(2\sqrt{\min\{m,n\}})$-approximation algorithm for the permutation flow shop problem.*

Proof. From the linearity of expectation, Lemma 2.2 and the comment following it, it suffices to bound $E_\sigma[I(\tau_k \circ \sigma[n])]$ for each $1 \le k \le h$. Fix a $1 \le k \le h$: since σ is chosen uniformly at random over all permutations, the jobs from Y_k are ordered uniformly at random. Thus $\tau_k \circ \sigma[n]$ is a uniformly random ordering of the elements X_k (ignoring occurrences of ϕ). Applying Lemma 2.3, we immediately obtain the following which proves the theorem.

$$E_\sigma[I(\tau_k \circ \sigma[n])] \le (2 + o(1))\sqrt{|X_k|} \le (2 + o(1))\sqrt{\min\{m,n\}}.$$

□

Thus we have a very simple randomized $\Theta(\sqrt{\min\{m,n\}})$-approximation algorithm for the permutation flow shop problem, based on the trivial lower bound. Potts et al. [21] gave a family of examples where the optimal permutation schedule has length at least $\frac{1}{\sqrt{2}}\sqrt{\min\{m,n\}}$ times the lower bound. Hence our result is the best possible guarantee (within a factor of $2\sqrt{2}$) using these lower bound. We note that Theorem 2.1 also implies that for any instance of flow shop scheduling, there is a permutation schedule of length at most $2\sqrt{\min\{m,n\}}$ times the length of an optimal *non-permutation* schedule; hence this resolves positively the open question in Potts et al. [21] regarding the gap between permutation & non-permutation schedules.

Tight Example. The following simple example shows that the performance guarantee of this randomized algorithm is tight. There are n jobs and $m = 2n$ machines. Each job j (for $1 \le j \le n$) has processing time 1 on machines j and $n + j$, and 0 elsewhere. The optimal permutation of jobs is $n, n-1, \cdots, 1$ which results in a makespan of 2. However, it follows from Lemma 2.3 that a random permutation has expected makespan at least $2\sqrt{n}$.

3. A Deterministic Algorithm We apply the technique of *pessimistic estimators* due to Raghavan [24] to derandomize the algorithm of the previous section, and obtain a deterministic $\Theta(\sqrt{\min\{m,n\}})$-approximation guarantee. We first apply the decomposition of Lemma 2.1 to obtain h permutation-matrices $\Pi_1, \cdots, \Pi_h$ corresponding to P. By assigning weights $w_1, \cdots, w_h \in \mathbb{N}$ to each of these permutations, we can ensure that $P = \sum_{k=1}^{h} w_k \cdot \Pi_k$ and $h \le mn$; here $\sum_{k=1}^{h} w_k$ is the trivial lower-bound

for the flowhop instance. This computation can be done easily in polynomial time by iteratively using any bipartite matching algorithm. There are many more efficient algorithms for computing an edge-coloring in bipartite multigraphs (See the table in Section 20.9b [27] for running times and references for various edge-coloring algorithms). Further Lemma 2.2 implies that for any permutation $\sigma : [n] \to [n]$ of the jobs, the resulting makespan $C^{\sigma}_{max} \leq C^*(\sigma) \doteq \sum_{k=1}^{h} w_k \cdot I(\tau_k \circ \sigma)$, where τ_ks are the partial permutations corresponding to the permutation-matrices Π_ks. From the previous section, we have that $E_\sigma[C^*(\sigma)] \leq 2\sqrt{\min\{m,n\}} \cdot \sum_{k=1}^{h} w_k$. In this section, we give a deterministic algorithm that obtains a permutation σ satisfying $C^*(\sigma) \leq 3\sqrt{\min\{m,n\}} \cdot \sum_{k=1}^{h} w_k$.

In particular, we show that given any collection of h partial permutations $\tau_1, \cdots, \tau_h : [n] \to [m] \cup \{\phi\}$, each having a non-empty value on at most r elements, and associated weights $\{w_k\}_{k=1}^{h}$, there is a polynomial time deterministic algorithm that computes a single permutation $\sigma : [n] \to [n]$ satisfying $C^*(\sigma) = \sum_{k=1}^{h} w_k \cdot I(\tau_k \circ \sigma[n]) \leq 3\sqrt{r} \cdot \sum_{k=1}^{h} w_k$. This immediately implies the desired deterministic approximation guarantee for the permutation flow shop problem since each partial permutation has an image of size at most $r \leq \min\{m,n\}$. In the following, we refer to a permutation that is chosen uniformly at random as u.a.r. permutation.

The algorithm first computes the partial permutations τ_k and weights w_k for $k = 1, \ldots, h$, and then builds the solution σ incrementally. In each step $i = 1, \ldots, n$ we suitably fix the value of $\sigma(i)$ that results in a prefix of the permutation $\langle \sigma(1), \ldots, \sigma(i) \rangle$, i.e. we fix jobs located in the first i positions. The choices for $\sigma(i)$ in each step i are made in such a way that finally, $C^*(\sigma) \leq 3\sqrt{r} \cdot \sum_{k=1}^{h} w_k$. Define the following quantities for any partial permutation τ_k ($1 \leq k \leq h$), step $0 \leq i \leq n$, and elements $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\phi\}$:

$$E_i^k(a_1, \cdots, a_i) \doteq \begin{array}{l}\text{expected value of the longest increasing subsequence} \\ \text{in } \langle a_1, \cdots, a_i, \tau \rangle, \text{ where } \tau \text{ is a permutation} \\ \text{on } \mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\} \text{ picked u.a.r.}\end{array}$$

$$U_i^k(a_1, \cdots, a_i) \doteq \begin{array}{l}\text{an efficiently computable upper bound on } E_i^k(a_1, \cdots, a_i) \\ \text{(exact definition later).}\end{array}$$

In the above definitions, the elements $a_1, \cdots, a_i$ represent the values $\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)$ respectively, obtained from the first i positions of permutation σ, that have been fixed thus far. We also define the expected value of function $C^*(\sigma)$ in step i as functions of the first i positions of permutation σ (that have been fixed):

$$E_i(\sigma(1), \cdots, \sigma(i)) : \begin{array}{l}\text{expected value } E_{\sigma(i+1)\cdots\sigma(n)}[C^*(\sigma)], \text{ with } \langle \sigma(i+1) \cdots \sigma(n) \rangle \\ \text{being a u.a.r. permutation on } [n] \setminus \{\sigma(1), \cdots, \sigma(i)\}\end{array}$$

Note that for any $1 \leq k \leq h$, since $\langle \sigma(i+1), \cdots, \sigma(n) \rangle$ is u.a.r. permutation on $[n] \setminus \{\sigma(1), \cdots, \sigma(i)\}$, we obtain that $\langle \tau_k \circ \sigma(i+1), \cdots, \tau_k \circ \sigma(n) \rangle$ is u.a.r. permutation on $\mathsf{Image}(\tau_k) \setminus \{\tau_k \circ \sigma(j) : 1 \leq j \leq i\}$. Thus we can rewrite E_i as:

$$E_i(\sigma(1), \cdots, \sigma(i)) \doteq \sum_{k=1}^{h} w_k \cdot E_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i))$$

We also define the efficiently computable upper bound on E_i as:

$$U_i(\sigma(1), \cdots, \sigma(i)) \doteq \sum_{k=1}^{h} w_k \cdot U_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i))$$

The precise definition of the upper bound functions U_i^k for $k = 1, \ldots, h$ and $i = 0, \ldots, n$ appears in the next subsection. In Lemmas 3.1 and 3.2, we prove some important properties of the functions U_i, which allow us to derandomize the algorithm of the previous section to obtain Theorem 3.1.

3.1 Properties of the pessimistic estimator Recall the definition of $E_i^k(a_1, \cdots, a_i)$; we now construct an upper bound
$U_i^k(a_1, \cdots, a_i)$ for this expected value. Fix a parameter $t = 3\sqrt{r}$, where $r = \max_{k=1}^{h} |\mathsf{Image}(\tau_k)|$ is an upper bound on the length of each partial permutation. Define $S_i^k(a_1, \cdots, a_i)$ to be the *expected*

number of t-length increasing subsequences in $\langle a_1, \cdots, a_i, \tau\rangle$, when τ is u.a.r. permutation on $\mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\}$. We can now upper bound $E_i^k(a_1, \cdots, a_i)$, the expected length of the longest increasing subsequence in $\langle a_1, \cdots, a_i, \tau\rangle$, as follows:

$$\begin{aligned} E_i^k(a_1, \cdots, a_i) &\leq t \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau\rangle \text{ has no } t\text{-length increasing subsequence}] \\ &\quad + r \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau\rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot Pr_\tau[\langle a_1, \cdots, a_i, \tau\rangle \text{ contains a } t\text{-length increasing subsequence}] \\ &\leq t + r \cdot S_i^k(a_1, \cdots, a_i) \end{aligned}$$

Define the upper bound U_i^k on the expected value E_i^k as:

$$U_i^k(a_1, \cdots, a_i) \doteq t + r \cdot S_i^k(a_1, \cdots, a_i) \qquad \forall 1 \leq k \leq h$$

Let $N_i^k = \mathsf{Image}(\tau_k) \setminus \{a_1, \cdots, a_i\} \subseteq [m]$; and for any set T, let $\mathcal{P}(T)$ denote the set of all permutations of the elements of T. We first show that each U_i^k can be efficiently computed, which implies the same for the functions $\{U_i\}_{i=0}^n$.

LEMMA 3.1 *For any $1 \leq k \leq h$, $i \in \{0, \cdots, n\}$ and $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\phi\}$, the value $U_i^k(a_1, \cdots, a_i)$ can be computed exactly in polynomial time.*

PROOF. Fix any values of $1 \leq k \leq h$, $0 \leq i \leq n$ and $a_1, \cdots, a_i \in \mathsf{Image}(\tau_k) \cup \{\phi\}$. Clearly it suffices to show that $S_i^k(a_1, \cdots, a_i)$ can be computed in polynomial time. We say that a t-length increasing subsequence s is *feasible* if there is some permutation $\tau \in \mathcal{P}(N_i^k)$ such that s is a subsequence in $\langle a_1, \cdots, a_i, \tau\rangle$. Let $\mathcal{I}$ denote the set of all such feasible t-length increasing subsequences. Then we can partition $\mathcal{I}$ as $\left(\sqcup \{\mathcal{I}_{j,q} \mid 1 \leq j \leq i \;\&\; 1 \leq q \leq t\}\right) \sqcup \mathcal{I}_{0,0}$ where:

$$\mathcal{I}_{0,0} = \{\tau^0 \mid \tau^0 \text{ is a } t\text{-length increasing sequence of numbers from } N_i^k\}$$

$$\mathcal{I}_{j,q} = \left\{ \langle \tau', \tau''\rangle \;\middle|\; \begin{array}{l} \tau' \text{ is a } q \text{ length increasing subsequence in } \langle a_1, \cdots, a_j\rangle \\ \text{ending at } a_j \neq \phi, \text{ and } \tau'' \text{ is a } t-q \text{ length increasing} \\ \text{sequence of numbers from } \{e \in N_i^k : e > a_j\} \end{array} \right\}$$

Note that given any $j \in \{1, \cdots, i\}$ and $q \in \{1, \cdots, t\}$, one can compute in polynomial time, the number of q-length increasing subsequences in $\langle a_1, \cdots, a_j\rangle$ that end at a_j; we denote this quantity by $\#I(j, q)$. The computation of $\#I(j, q)$ is based on a dynamic program using the following recurrence:

$$\#I(j,q) = \begin{cases} \sum\{\#I(j', q-1) \mid 1 \leq j' < j,\ a_{j'} < a_j\} & a_j \neq \phi,\ q \geq 2 \\ 1 & a_j \neq \phi,\ q = 1 \\ 0 & a_j = \phi \end{cases}$$

For ease of notation in the following, let $\#I(0, 0) = 1$. For every $1 \leq j \leq i$, denote the set $\{e \in N_i^k : e > a_j\}$ by L_j, and also let $L_0 = N_i^k$. Note that, for each part $\mathcal{I}_{j,q}$ (in the partition of $\mathcal{I}$), its size $|\mathcal{I}_{j,q}| = \#I(j, q) \cdot \binom{|L_j|}{t-q}$ (the first term corresponds to a q length increasing subsequence of $\langle a_1, \cdots, a_j\rangle$, and the second term corresponds to a $t - q$ length increasing sequence from L_j). When $\tau \in \mathcal{P}(N_i^k)$ is picked u.a.r., the induced permutation on each set L_j (for $0 \leq j \leq i$) is also u.a.r. Hence for each part $\mathcal{I}_{j,q}$, the probability that any particular subsequence $s \in \mathcal{I}_{j,q}$ appears in $\langle a_1, \cdots, a_i, \tau\rangle$ is exactly $1/(t-q)!$. (the last $t - q$ entries of s come from the random permutation τ). So we have:

$$\begin{aligned} & E_\tau\left[|\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau\rangle\}|\right] \\ &= \sum_{s \in \mathcal{I}_{j,q}} Pr_\tau\left[s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau\rangle\right] \\ &= |\mathcal{I}_{j,q}| \cdot \tfrac{1}{(t-q)!} \\ &= \#I(j,q) \cdot \tbinom{|L_j|}{t-q} \cdot \tfrac{1}{(t-q)!} \end{aligned}$$

Thus, we can write $S_i^k(a_1, \cdots, a_i)$ as

$$\begin{aligned} & E_{\tau \in \mathcal{P}(N_i^k)}\left[\ |\{s \in \mathcal{I} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau\rangle\}|\ \right] \\ &= \sum_{j,q} E_{\tau \in \mathcal{P}(N_i^k)}\left[\ |\{s \in \mathcal{I}_{j,q} : s \text{ is subsequence of } \langle a_1, \cdots, a_i, \tau\rangle\}|\ \right] \\ &= \sum_{j,q} \#I(j,q) \cdot \tbinom{|L_j|}{t-q} \cdot \tfrac{1}{(t-q)!} \end{aligned}$$

The lemma follows. □

LEMMA 3.2 *For any $0 \leq i \leq n$ and any prefix (possibly empty) $\sigma(1), \cdots, \sigma(i) \in [n]$ of a permutation σ,*

$$\min_{\sigma(i+1) \in [n] \setminus \{\sigma(1), \cdots, \sigma(i)\}} U_{i+1}(\sigma(1), \cdots, \sigma(i), \sigma(i+1)) \leq U_i(\sigma(1), \cdots, \sigma(i))$$

PROOF. Fix any i and a prefix $\sigma(1), \cdots, \sigma(i)$ of a permutation σ, and let $M = [n] \setminus \{\sigma(1), \cdots, \sigma(i)\}$. We first prove the following for an arbitrary $1 \leq k \leq h$:

$$S_i^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i)) = \frac{1}{n-i} \sum_{x \in M} S_{i+1}^k(\tau_k \circ \sigma(1), \cdots, \tau_k \circ \sigma(i), \tau_k(x)) \tag{1}$$

For ease of notation, let $a_j^k = \tau_k \circ \sigma(j)$ for all $1 \leq j \leq i$. Let $N_i^k = \mathsf{Image}(\tau_k) \setminus \{a_1^k, \cdots, a_i^k\} \subseteq [m]$ denote the remaining elements of $\mathsf{Image}(\tau_k)$, and $n_i^k = |N_i^k|$. Recall that,

$$S_i^k(a_1^k, \cdots, a_i^k) = E_\tau[\text{number of } t\text{-length increasing subsequences in } \langle a_1^k \cdots a_i^k, \tau \rangle]$$

where $\tau \in \mathcal{P}(N_i^k)$ is picked u.a.r. So multiplying both sides of 1 by $n_i^k! = |\mathcal{P}(N_i^k)|$, we can rewrite its left hand side as:

$$LHS' = n_i^k! \ \times \ S_i^k(a_1^k, \cdots, a_i^k) = \sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \cdots, a_i^k, \tau) \tag{2}$$

Above, for any sequence s, $\#I_t(s)$ denotes the number of t-length increasing subsequences in s. To compute the right hand side of 1, we split the summation into $M^{(k)} = \{x \in M \mid \tau_k(x) \neq \phi\}$ and $M \setminus M^{(k)} = \{x \in M \mid \tau_k(x) = \phi\}$. Note that $|M| = n - i$ and $|M^{(k)}| = n_i^k$. For any $x \in M \setminus M^{(k)}$, it is easy to see that $S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) = S_i^k(a_1^k, \cdots, a_i^k)$. Now the right hand side of 1 (scaled by $n_i^k!$) can be written as:

$$\begin{aligned} & n_i^k! \ \times \frac{n-i-n_i^k}{n-i} S_i^k(a_1^k, \cdots, a_i^k) + n_i^k! \ \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \\ = \ & (1 - \frac{n_i^k}{n-i}) LHS' + n_i^k! \ \times \frac{1}{n-i} \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \end{aligned}$$

Thus in order to prove 1, it suffices to show:

$$LHS' = (n_i^k - 1)! \ \times \sum_{x \in M^{(k)}} S_{i+1}^k(a_1^k, \cdots, a_i^k, \tau_k(x)) \tag{3}$$

Note that τ_k induces a bijection between $M^{(k)}$ and N_i^k: $|M^{(k)}| = |N_i^k|$ and $\tau_k(M^{(k)}) = N_i^k$. Thus we can rewrite the right hand side in 3 as:

$$(n_i^k - 1)! \sum_{y \in N_i^k} S_{i+1}^k(a_1^k, \cdots, a_i^k, y) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \cdots, a_i^k, y, \tau')$$

To prove 3, using the expression for LHS' from 2, it suffices to show that:

$$\sum_{\tau \in \mathcal{P}(N_i^k)} \#I_t(a_1^k, \cdots, a_i^k, \tau) = \sum_{y \in N_i^k} \sum_{\tau' \in \mathcal{P}(N_i^k \setminus y)} \#I_t(a_1^k, \cdots, a_i^k, y, \tau')$$

Now observe that $\mathcal{P}(N_i^k) = \sqcup_{y \in N_i^k} \{\langle y, \tau' \rangle \mid \tau' \in \mathcal{P}(N_i^k \setminus y)\}$. Thus the summations in the two expressions above run over exactly the same set of sequences, and this implies equality 3 which in turn gives equation

1. We are now ready to complete the proof of the lemma.

$$\begin{aligned}
&\min_{\sigma(i+1)\in M} U_{i+1}(\sigma(1),\cdots,\sigma(i),\sigma(i+1)) \le \frac{1}{n-i}\sum_{x\in M} U_{i+1}(\sigma(1),\cdots,\sigma(i),x)\\
&= \frac{1}{n-i}\sum_{x\in M}\sum_{k=1}^{h} w_k\big[t + r\cdot S_{i+1}^k(\tau_k\circ\sigma(1),\cdots,\tau_k\circ\sigma(i),\tau_k(x))\big]\\
&= \frac{|M|}{n-i}\cdot t\sum_{k=1}^{h} w_k + \frac{1}{n-i}\sum_{x\in M} r\cdot\sum_{k=1}^{h} w_k\cdot S_{i+1}^k(\tau_k\circ\sigma(1),\cdots,\tau_k\circ\sigma(i),\tau_k(x))\\
&= t\sum_{k=1}^{h} w_k + r\cdot\sum_{k=1}^{h} w_k\cdot\frac{1}{n-i}\sum_{x\in M} S_{i+1}^k(\tau_k\circ\sigma(1),\cdots,\tau_k\circ\sigma(i),\tau_k(x))\\
&= t\sum_{k=1}^{h} w_k + r\cdot\sum_{k=1}^{h} w_k\cdot S_i^k(\tau_k\circ\sigma(1),\cdots,\tau_k\circ\sigma(i)) \qquad \text{(Using equation 1)}\\
&= \sum_{k=1}^{h} w_k\cdot U_i^k(\tau_k\circ\sigma(1),\cdots,\tau_k\circ\sigma(i))\\
&= U_i(\sigma(1),\cdots,\sigma(i))
\end{aligned}$$

Thus we have the lemma. □

3.2 Applying the pessimistic estimators We now use the upper-bound functions U_i for $i = 1,\ldots,n$ described in the previous subsection to obtain a deterministic approximation algorithm for the permutation flow shop problem. This algorithm follows the general framework of the method of pessimistic estimators.

THEOREM 3.1 *There is a deterministic polynomial time* $3\sqrt{\min\{m,n\}}$ *approximation algorithm for the permutation flow shop scheduling problem with makespan objective.*

PROOF. We now describe our final deterministic algorithm:

(i) Decompose the matrix P of processing times according to Lemma 2.1, to obtain $h \le mn$ permutation-matrices with corresponding weights $\{\Pi_k, w_k\}_{k=1}^h$, such that $P = \sum_{k=1}^h w_k\cdot\Pi_k$ and $\sum_{k=1}^h w_k$ equals the trivial lower-bound for the permutation flow shop instance.
(ii) For each $1 \le k \le h$, τ_k denotes the partial permutation corresponding to permutation-matrix Π_k.
(iii) For each $i = 1,\cdots,n$: set $\sigma(i) \leftarrow x$ for the value $x \in [n]\setminus\{\sigma(1),\cdots,\sigma(i-1)\}$ that minimizes the function value $U_i(\sigma(1),\cdots,\sigma(i-1),x)$.

As mentioned earlier, the decomposition in step (i) can be carried out in polynomial time using an edge-coloring algorithm. In step (iii), the algorithm uses the efficiently computable functions $\{U_i\}_{i=0}^n$ (see Lemma 3.1) to fix the solution σ step by step. Hence the above algorithm runs in polynomial time. The rest of this proof shows that it achieves the desired approximation guarantee.

We claim that for each $i \in \{0,\cdots,n\}$, $U_i(\sigma(1),\cdots,\sigma(i)) \le W\cdot(t+2)$ where $W = \sum_{k=1}^h w_k$ is the trivial lower-bound (recall that $t = 3\sqrt{r} \le 3\sqrt{\min\{m,n\}}$). Assuming that the base case (i.e. $i = 0$) for this claim holds, using Lemma 3.2 and induction, we obtain that the claim is true for all values of $i \ge 1$. It remains to prove the claim for $i = 0$: here U_0 takes no arguments and is a fixed value $U_0 = tW + r\sum_{k=1}^h w_k\cdot S_0^k$. From the definition of the S_i^ks, we have that each S_0^k is the expected number of t-length increasing subsequences in a u.a.r. permutation of $\mathsf{Image}(\tau_k)$. Since $\mathsf{Image}(\tau_k)$ has at most r elements, using linearity of expectation, it follows that $S_0^k \le \binom{r}{t}\cdot\frac{1}{t!}$ for every $k = 1,\cdots,h$. We have,

$$\begin{aligned}
U_0 &\le tW + rW\binom{r}{t}\frac{1}{t!} = tW + rW\frac{r!}{(r-t)!t!}\frac{1}{t!} \le tW + rW\frac{r^t}{(t!)^2}\\
&\le tW + rW\left(\frac{re^2}{t^2}\right)^t = tW + rW\left(\frac{e^2}{9}\right)^t = tW + rW\left(\frac{e}{3}\right)^{6\sqrt{r}} \le W\cdot(t+2)
\end{aligned}$$

Now observe that after the last step, $E_n(\sigma(1),\cdots,\sigma(n))$ is exactly the value $C^*(\sigma)$ (at this point all positions have been fixed, so there is no randomness left in the expected value E_n). Since the function

U_n upper bounds E_n, we have $C^*(\sigma) = E_n(\sigma(1), \cdots, \sigma(n)) \leq U_n(\sigma(1), \cdots, \sigma(n)) \leq W \cdot (t+2)$. Now the theorem follows from the fact that W equals the trivial lower-bound for the permutation flow shop instance and $r \leq \min\{m, n\}$. □

4. Weighted sum of completion times In this section, we consider the permutation flow shop problem with the objective being the weighted sum of completion times. We show that our algorithm for the makespan objective can be used within an LP-based approach to obtain an $O(\sqrt{\min\{m,n\}})$ approximation algorithm for weighted completion time. This approach is similar to that used in Queyranne and Sviridenko [23] (and many other papers on scheduling with the weighted completion time objective [11, 1, 10]), where the authors considered a class of job shop problems (these do not have the permutation constraint). We consider the following linear relaxation for the permutation flow shop problem with weighted completion time as objective. In fact this LP is a relaxation for even the usual flow shop problem (without the permutation constraint).

$$\min \sum_{j=1}^{n} w_j \cdot C_j, \tag{4}$$

$$z_{1,j} \geq p_{1,j}, \qquad \forall 1 \leq j \leq n \tag{5}$$

$$z_{i,j} \geq z_{i-1,j} + p_{i,j}, \qquad \forall 2 \leq i \leq m, 1 \leq j \leq n \tag{6}$$

$$\sum_{j \in A} p_{i,j} \cdot z_{i,j} \geq \frac{1}{2}(\sum_{j \in A} p_{i,j})^2 + \frac{1}{2}\sum_{j \in A} p_{i,j}^2, \qquad \forall A \subseteq [n], 1 \leq i \leq m, \tag{7}$$

$$C_j = z_{m,j}, \qquad \forall 1 \leq j \leq n \tag{8}$$

$$z_{i,j} \geq 0, \qquad \forall 1 \leq i \leq m, 1 \leq j \leq n \tag{9}$$

Here each variable $z_{i,j}$ denotes the time when job j's operation on machine i is completed; and $C_j = z_{m,j}$ is the completion time of job j. Constraints (6) ensure that the operations of each job are performed in the flow shop order. Constraints (7) are a relaxation of the machine resource constraints. The weighted completion time objective is captured in (4). As shown in Queyranne [22], this LP can be solved in polynomial time using the Ellipsoid algorithm (the separation oracle for the exponential-sized constraints (7) reduces to a submodular function minimization). The algorithm first obtains an optimal solution (z, C) to the above LP. Then it reduces the weighted completion time problem to one of minimizing makespan as outlined below.

(i) Group the jobs $[n]$ based on their fractional completion times C_j. For each integer $a \geq 0$, group G_a consists of all jobs $1 \leq j \leq n$ such that $C_j \in (2^a, 2^{a+1}]$. Note that there are at most n non-empty groups.

(ii) For each non-empty group G_a, run the $O(\sqrt{\min\{m,n\}})$ approximation algorithm for minimizing makespan, to obtain a permutation π_a of G_a.

(iii) Output the permutation of jobs $[n]$ given by $\pi_0, \pi_1, \cdots$.

The remaining analysis is identical to the one in Queyranne and Sviridenko [23]; however it is presented here in the context of permutation flowshop, for the sake of completeness. For any group G_a, let $l_a = \max\{\sum_{i=1}^{m} p_{i,j} \mid j \in G_a\}$ denote the maximum job length, and $L_a = \max\{\sum_{j \in G_a} p_{i,j} \mid 1 \leq i \leq m\}$ the maximum machine load. We first prove the following auxiliary lemma.

LEMMA 4.1 *For each group G_a, $\max\{l_a, L_a\} \leq 2^{a+2}$*

PROOF. For any job j, constraints (6) imply $C_j = z_{m,j} \geq z_{m-1,j} + p_{m,j} \geq \cdots \geq \sum_{i=1}^{m} p_{i,j}$. Hence $l_a = \max\{\sum_{i=1}^{m} p_{i,j} \mid j \in G_a\} \leq \max\{C_j \mid j \in G_a\} \leq 2^{a+1}$.

For any machine i, constraint (7) applied to subset $A = G_a$ implies $\sum_{j \in G_a} p_{i,j} \cdot z_{i,j} \geq \frac{1}{2}(\sum_{j \in G_a} p_{i,j})^2$. Furthermore, constraints (6) imply $z_{i,j} \leq C_j$ for all machines i and jobs j. Hence, $\frac{1}{2}(\sum_{j \in G_a} p_{i,j})^2 \leq \sum_{j \in G_a} p_{i,j} \cdot C_j \leq 2^{a+1} \sum_{j \in G_a} p_{i,j}$. In other words, $\sum_{j \in G_a} p_{i,j} \leq 2^{a+2}$ for all machines i. Thus $L_a \leq 2^{a+2}$, and the lemma follows. □

THEOREM 4.1 *There is a polynomial time $O(\sqrt{\min\{m,n\}})$ approximation algorithm for minimizing weighted completion time in the permutation flow shop problem.*

PROOF. Recall that the approximation guarantee of our algorithm for minimizing makespan (Section 3) is relative to the trivial lower bound. Along with the Lemma 4.1, we obtain that for each group G_a, the resulting makespan $C_{max}(\pi_a) \leq \rho \cdot \max\{l_a, L_a\} \leq \rho \cdot 2^{a+2}$, where $\rho = O(\sqrt{\min\{m,n\}})$ is the approximation guarantee for the makespan problem. Under the final permutation $\langle \pi_0, \pi_1, \cdots \rangle$, the completion time of each job in group G_a is at most $\sum_{b=0}^{a} C_{max}(\pi_b) \leq 4\rho \sum_{b=0}^{a} 2^b \leq 8\rho \cdot 2^a$. But in the LP solution, $C_j \geq 2^a$ for all $j \in G_a$ and groups G_a. This implies that the weighted completion time of the final permutation is at most 8ρ times the optimal LP value. □

Integrality gap of the linear program (4)-(9). We observe that the example of Potts et al. [21] (comparing permutation and non-permutation schedules) also gives an $\Omega(\sqrt{\min\{m,n\}})$ lower bound on the integrality gap of the linear program (4)-(9). So our algorithm is the best possible approximation algorithm based on this linear programming relaxation. For any $n \in \mathbb{N}$, let $\mathcal{I}_n$ denote the following instance of permutation flow shop: there are n jobs and $2n$ machines; for each $j = 1, \cdots, n$, job j has processing time 1 on machines j and $2n+1-j$, and 0 elsewhere. It was shown in [21] that the optimal makespan $C^*_{max}(\mathcal{I}_n) \geq \sqrt{2n}$ for all $n \geq 1$. Consider the objective of minimizing the *total completion time* for instance $\mathcal{I}_n$; i.e. the weighted completion time objective with all weights $w_j = 1$ $(1 \leq j \leq n)$. Note that for any $1 \leq k \leq n$, *any* set of k jobs in the instance $\mathcal{I}_n$ is equivalent to the instance $\mathcal{I}_k$. Hence, for any permutation of the jobs, the completion time of the k-th job in the permutation is at least $C^*_{max}(\mathcal{I}_k) \geq \sqrt{2k}$, for all $1 \leq k \leq n$. Thus the optimal total completion time of instance $\mathcal{I}_n$ is at least $\sum_{k=1}^{n} \sqrt{2k} = \Omega(n^{3/2})$. We now construct a fractional feasible solution (z, C) to the linear program (4)-(9) having objective value $O(n)$, which would establish the claimed integrality gap. The z-variables of each job $j \in [n]$ are set as follows: $z_{i,j} = 0$ for $1 \leq i < j$, $z_{i,j} = 1$ for $j \leq i < 2n-j+1$, and $z_{i,j} = 2$ for $2n-j+1 \leq i \leq 2n$. This fixes the fractional completion-time $C_j = z_{2n,j} = 2$ for all jobs j, and the objective value is $2n$. The only non-trivial constraints to check are (6) and (7). From the construction of solution (z, C), it follows that constraints (6) are satisfied. To see that constraints are (7) are satisfied, consider any machine i: for every $A \subseteq [n]$, the right-hand-side of (7) is either 0 or 1; moreover whenever it is 1, the left-hand-side of (7) is 1 as well.

Acknowledgments. The work of the first author was supported in part by NSF grant CCF-0728841.

References

[1] S. Chakrabarti, C. Phillips, A. Schulz, D. Shmoys, C. Stein and J. Wein, Improved Scheduling Algorithms for Minsum Criteria, In Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming (ICALP'96), 646–657.

[2] B. Chazelle, The discrepancy method. Randomness and complexity. Cambridge University Press, Cambridge, 2000.

[3] B. Chen, C. Potts and G. Woeginger, A review of machine scheduling: complexity, algorithms and approximability, in Handbook of combinatorial optimization, Vol. 3, Edited by Ding-Zhu Du and Panos M. Pardalos, Kluwer Academic Publishers, Boston, MA, (1998), 21-169.

[4] R. Conway, W. Maxwell and L. Miller, Theory of scheduling, Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1967.

[5] A. Czumaj and C. Scheideler, A New Algorithmic Approach to the General Lovasz Local Lemma with Applications to Schedulung and Satisfiability Problems, Proc. 32 ACM Symposium on Theory of Computing (STOC) , 2000.

[6] U. Feige and C. Scheideler, "Improved bounds for acyclic job shop scheduling." In: *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (STOC'98, 1998), ACM Press, New York, NY, 624-633.

[7] A. Fishkin, K. Jansen and M. Mastrolilli, On minimizing average weighted completion time: a PTAS for the job shop problem with release dates. Algorithms and computation, 319–328, Lecture Notes in Comput. Sci., 2906, Springer, Berlin, 2003.

[8] J. Framinan, J. Gupta and R. Leisten, A review and classification of heuristics for permutation flow-shop scheduling with makespan objective, Journal of the Operational Research Society (2004) 55, 1243-1255.

[9] A. Frieze, On the length of the longest monotone subsequence of a random permutation, The Annals of Applied Probability 1(2), 301-305, 1991.

[10] L.A. Hall, D.B. Shmoys, and J. Wein, Scheduling to Minimize Average Completion Time: Off–line and On–line Algorithms, *Proceedings of the 7th Symposium on Discrete Algorithms* (1996) 142–151.

[11] L.A. Hall, A.S. Schulz, D.B. Shmoys, and J. Wein, Scheduling to Minimize Average Completion Time: Off–Line and On–Line Approximation Algorithms, *Mathematics of Operations Research* **22** (1997) 513–544.

[12] K.Jansen, R.Solis-Oba and M. Sviridenko, Makespan Minimization in Job Shops: a Linear Time Approximation Scheme, SIAM Journal of Discrete Mathematics **16** (2003), 288-300.

[13] S. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quartely **1** (1954), pp. 61-68.

[14] M. Hofri, Probabilistic Analisys of Algorithms: On Computing Metodologies for Computing Algorithms Performance Evaluation, Springer Verlag, 1987.

[15] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, in: Handbook in Operations Research and Management Science, Vol. 4, North-Holland, 1993, 445-522.

[16] B.F. Logan and L.A. Shepp, A Variational Problem for Random Young Tableaux, Advances in Mathematics **26** (1977), 206-222.

[17] M. Nawaz, E. Enscore Jr. and I. Ham, A heuristic algorithm for the m-machine n-job flow-shop sequencing problem, OMEGA International J. Management Sci. 11 (1983), 91-95.

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

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

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

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

[22] M. Queyranne, Structure of a simple scheduling polyhedron. Math. Programming 58 (1993), no. 2, Ser. A, 263–285.

[23] M. Queyranne and M. Sviridenko, Approximation Algorithms for Shop Scheduling Problems with Minsum Objective, Journal of Scheduling **5** (2002), pp. 287–305.

[24] P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs, J. Comput. System Sci. **37** (1988), 130-143.

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

[26] S. Sevast'janov, On some geometric methods in scheduling theory: a survey, Discrete Applied Mathematics **55** (1994), 59-82.

[27] A. Schrijver, Combinatorial optimization. Polyhedra and efficiency. Algorithms and Combinatorics, 24,B. Springer-Verlag, Berlin, 2003.

[28] D. Shmoys, C. Stein, and J. Wein. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing* 23:3, 617-632, 1994.

[29] D. Sotelo and M. Poggi de Aragao, An Approximation Algorithm for the Permutation Flow Shop Scheduling Problem via Erdos-Szekeres Theorem Extensions, manuscript, 2008.

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

[31] C. Smutnicki, Some results of the worst-case analysis for flow shop scheduling, European Journal of Operational Research **109** (1998), 6687.

[32] A.M. Vershik and S.V. Kerov, Asymptotics of the Plancherel measure of the symmetric group and the limit form of Young tableaux, Dokl. Akad. Nauk SSSR **233** (1977), 1024-1027.