Approximate Solutions for Deterministic and Stochastic Multi-Dimensional Sequencing

B Jothi [a], Sachin Patkar [a],
Chandra A Poojari [b] and Janak Porwal [c]

[a] *Department of Mathematics, Indian Institute of Technology Bombay, Mumbai*

[b] *Centre for the analysis of Risk and optimisation modelling applications (CARISMA), School of Information Systems, Computing and Mathematics, Brunel University,London, Uxbridge UB8 3PH, UK*

[c] *Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai*

Abstract

We investigate the problem of sequencing jobs that have multiple components. Each component of the job needs to be processed independently on a specified machine. We derive approximate algorithms for the problem of scheduling such vector jobs to minimize their total completion time in the deterministic as well as stochastic setting. In particular, we propose a Linear Programming and a Greedy heuristic based strategy to derive approximate solutions for deterministic and stochastic formulation of the problem.

Problem Description

The ideas for the approaches discussed in this paper arose during the investigation of a practical problem faced by a composite-bearings manufacturer. The individual components are required to be built prior to the manufacturing of a composite bearing. Such components include ball bearings, needle bearings and shaft bearings. The components are typically processed independently on different machines and then assembled together. Each job corresponding to the manufacturing of a composite-bearing can be looked upon as a vector. The completion time of a job is defined as the maximum time taken to complete its components. The problem is to schedule the vectors of jobs such that the sum of the completion times of all the jobs is minimised. Each machine requires to be configured uniquely to make a given component. The machines may be regarded as *suppliers* supplying basic commodities. The bearings-assembly

process can similarly be regarded as a *manufacturer* which "manufactures" end-products using the supplied basic commodities. Such models of interaction have been investigated in (Chen and Hall, 2005; Potts et al., 1995). The results of (Chen and Hall, 2005) throw light on the lower bound on the performance guarantees, whereas in this paper we study upper bounds on the worst case performance ratios. Potts' article (Potts et al., 1995) investigate different objective criteria, e.g. minimizing the makespan.

Outline of the paper

In this paper we consider deterministic and stochastic formulation of the vector job scheduling problem. In the stochastic formulation we assume that the completion time of the tasks on the machines are random variables. The results of (Hall, 1998) have already established the NP-hardness of this problem. Therefore, it is justified to study the approximation algorithms and general heuristic approaches to solve this problem.

We first discuss the Mathematical Programming formulation of the underlying optimisation problem. We, then develop approximate solutions using a Linear Programming and a Greedy heuristic based methods for the deterministic formulation of vector job scheduling problem. Finally, we derive approximate solutions for the stochastic formulation of the problem in which the processing time of the machines are assumed to be random.

Definition of the Problem

INDICES
n = number of jobs,
m = number of machines,
i,j= indices over the number of jobs,
k= index over the number of machines,
DATA
a_i^k = processing time taken to complete the k^{th} component of i^{th} job, $a_i^k \geq 0$.
J_i = the i^{th} job, defined as $(a_i^1, a_i^2, \ldots, a_i^m)$,
J = set of jobs = $\{J_1, J_2, \ldots J_n\}$
VARIABLES
Π = schedule for the vector jobs (described by a permutation of $\{1, 2, \ldots n\}$),
Π_i = the i^{th} job of the schedule given by the permutation Π,
C_{Π_i} = time taken to complete the i^{th} job in the schedule Π,
$\mathcal{C}$ = time taken to complete all the jobs.
OBJECTIVE

Minimise $\mathcal{C}$.
CONSTRAINTS
k^{th} machine can process only k^{th} component of each job.
The value C_{Π_i} is equal to the sum of the completion time for all the jobs preceding and including the i^{th} job, $C_{\Pi_i} = \max_{k=1}^{m} \left\{ \sum_{j=1}^{i} a_{\Pi_j}^{k} \right\}$. We wish to determine an optimal schedule, Π^*, which minimizes the total completion time, $\mathcal{C} = \sum_{i=1}^{n} C_{\Pi_i}$.

Terms

We define the following terminologies:
performance ratio of an algorithm is the ratio of the cost of the schedule obtained using the algorithm to the cost of the optimal schedule;
k-approximate algorithm is an algorithm whose performance ratio is at most k;
an *aligned schedule* is one in which the ordering of jobs in the schedule is the same on each machine. The following can be easily verified.

Theorem There exists an aligned schedule that minimises the sum of completion times of all jobs.
Proof Consider a non-aligned schedule, therefore there are at least 2 machines on which the jobs are ordered differently. Let Π^1 and Π^2 denote the schedule of the jobs on such 2 machines. By definition

- Π_i^1 & Π_i^2 denotes the i^{th} job in the two schedules, and
- $C_{\Pi_i^1}$ & $C_{\Pi_i^2}$ denotes the time taken to complete the i^{th} job.

Consider the pseudocode described in the algorithm 0.1,

Algorithm 0.1: AlignedSchedule(Π^1, Π^2)

$$
\textbf{for } i \leftarrow n \textbf{ to } 1
$$

$$
\textbf{do}
\begin{cases}
C_{\Pi_i^r} = min\left\{C_{\Pi_i^1}, C_{\Pi_i^2}\right\} \\
s \leftarrow 3 - r \\
\exists j \ (< i), \text{such that } \Pi_j^r = \Pi_i^s \\
\text{Let } \hat{\Pi}_i^r \leftarrow \Pi_j^r \\
\textbf{for } k = j \textbf{ to } i - 1 \\
\quad \textbf{do}
\begin{cases}
\hat{\Pi}_k^r \leftarrow \Pi_{k+1}^r \\
k \leftarrow k + 1
\end{cases} \\
\Pi^r \leftarrow \hat{\Pi}^r \\
\text{Update } C_{\Pi^r} \\
i \leftarrow i - 1
\end{cases}
$$

At the end of an execution of the algorithm 0.1, the resulting pair of permutations are identical. Furthermore, the cost of the updated schedule is less or equal to the original (as in every update the cost never increases). Therefore our focus would be on aligned schedules.

Approximate Solutions to the Deterministic problem

In the deterministic version of the problem, the processing time of any job component is a fixed value, and is therefore known before computing the schedule.

Linear Programming based analysis

Potts' formulation first appeared in (Potts, 1980) for the problem of single machine job scheduling with precedence constraints to minimize total completion time. This Integer Linear Programming (ILP) formulation was used derive a lower bound on the value of an optimal solution, to be used in a branch and bound algorithm. The authors in the paper (Hall et al., 1996) used Potts' formulation to derive a 2-factor approximate algorithm for single

component job scheduling. The advantage of Potts' formulation is that the number of variables as well as inequalities are bounded by a polynomial in the input size. Moreover, as we later show, the special structure of the coefficient matrix of the LP can be used to speed up the algorithm to solve the LP using a sparse matrix implementation of the revised simplex algorithm or column generation.

Potts' Formulation

VARIABLES

Let δ_{ij} = a binary variable denoting the order between the i^{th} and the j^{th} job, and C_i = completion time for the i^{th} job. The variables

$$\delta_{ij} = \begin{cases} 1 \text{ if job } i \text{ is scheduled before job } j \\ 0 \qquad\qquad \text{otherwise.} \end{cases}$$

The complete ILP formulation can be written as:

$$\text{Minimize} \sum_{i=1}^{n} C_i$$

Subject to

$$\delta_{ij} + \delta_{ji} = 1 \qquad \forall \{i, j = 1, \ldots, n;\ i < j\} \tag{1a}$$

$$C_i \geq a_i^k + \sum_{j \neq i} a_j^k \delta_{ji} \quad \forall \{i = 1, \ldots, n;\ k = 1, \ldots, m;\ a_i^k > 0\} \tag{1b}$$

$$\delta_{ij} \in \{0, 1\} \tag{1c}$$

The original formulation by Potts also had the inequalities $\delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2\ \forall\ \{i, j, k = 1, \ldots, n;\ i < j < k \text{ or } k < j < i\}$ for ensuring that the ordering of jobs does not result in any cycles. The above problem contains $\frac{n(n-1)}{2}$ constraints of type 1a and nm constraints of type 1b.

Define a variable $\delta_{i.} = \sum_{j=1}^{n} \delta_{ij}$. This variable denotes the number of jobs following the i^{th} job. The schedule Π can be constructed using the equation 2,

$$\Pi_i = k \text{ where } \delta_{k.} = n - i \quad \forall i. \tag{2}$$

Integrality for single machine case

Consider an LP relaxation of the problem 1 in which we replace the constraint 1c with $0 \leq \delta_{ij} \leq 1$.

Theorem An optimal solution to the LP relaxation of problem 1 for the single machine case, is integral if $a_i \neq a_j$ for all i and j.

Proof Replace δ_{ji} by $1 - \delta_{ij}$ for all $j > i$. For the single machine case, inequalities (1b) are satisfied exactly and hence the completion time of job i is $C_i = a_i + \sum_{j \neq i} a_j \delta_{ji}$. The objective function is the summation of C_i's over all i's, which is, $\sum_i C_i = \sum_i a_i + \sum_i \sum_{j \neq i} a_j \delta_{ji}$. By combining terms involving δ_{ij} and δ_{ji}, this can be rewritten as follows: $\sum_i C_i = \sum_i a_i + \sum_{i<j} (a_j \delta_{ji} + a_i \delta_{ij})$.

Substituting $\delta_{ij} = 1 - \delta_{ji}$, we get $\sum_i C_i = \sum_i a_i + \sum_{i<j}((a_i - a_j)\delta_{ij} + a_j)$. As we have full freedom to choose values for δ_{ij} for $i < j$ we see that the minimization of the objective function (in the form derived above), occurs when we set $\delta_{ij} = 1$ whenever $p_i < p_j$ and $\delta_{ji} = 0$ whenever $p_j > p_i$ (please note that we assume a_j's to be distinct). Therefore the integrality for the single machine case follows when the processing times are all distinct.

Note that the above proof is just a different way of deriving the well known Smith's rule (Smith, 1956) for single processor scheduling.

Speeding-up the algorithm

The special structure of the coefficient matrix can be used to speed up the algorithm to obtain an optimal solution to the LP relaxation of Potts' formulation. We first eliminate constraints (1a) in the LP by replacing every δ_{ij} for $i > j$ by $1 - \delta_{ji}$. Note that now we need to add the constraints $\delta_{ij} \leq 1$ for all $i < j$ to ensure that $1 - \delta_{ij}$ is positive. However, constraints corresponding to upper (or lower) bounds on the variables can be handled more efficiently by the simplex algorithm than those of the type (1a). Thus we have effectively reduced the number of variables to $n(n-1)/2 + n$ and more importantly, the number of inequalities to nm (linear in number of jobs). The modified LP can be rewritten after suitably rearranging the terms as:

$$min \sum_{i=1}^{n} C_i \tag{3a}$$

such that :

$$C_i + \sum_{i<j} \delta_{ij} a_j^k - \sum_{j<i} a_i^k \delta_{ji} \geq \sum_{j=i}^{n} a_j^k \quad \forall\ i = 1, \ldots, n;\ k = 1, \ldots, m \tag{3b}$$

$$-\delta_{ij} \geq -1 \qquad \forall\ i < j;\ i, j \in \{1, \ldots, n\} \tag{3c}$$

$$\delta_{ij} \geq 0 \qquad \forall\ i < j;\ i, j \in \{1, \ldots, n\} \tag{3d}$$

The coefficient matrix of the LP comprising of (3a-3b) is an $(nm) \times (n(n+1)/2 + nm)$ matrix, in which the column for variable δ_{ij} contains $-a_i^k$ in the $((k-1)n + i)^{th}$ row, a_j^k in the $((k-1)n + j)^{th}$ row and 0 elsewhere. The column for variable C_i contains a 1 in rows $((k-1)n + i)$ and 0 elsewhere while columns corresponding to slack variables have a single non-zero entry 1. It can be shown that the number of operations needed to compute the reduced cost coefficient of any column is at most *2m*, if carried out separately (without applying direct matrix multiplication). Hence, computing the vector of reduced cost coefficients separately requires $O(mn^2)$ operations per iteration in the sparse matrix implementation of the revised simplex algorithm. (as opposed to $O(mn^3)$ operations in a normal implementation). The same speed-up of factor n can be achieved by introducing a separate subroutine of column

generation in the normal implementation of the simplex algorithm.

Greedy heuristic

Suppose Π is a permutation of $\{1,2,\cdots,n\}$ such that for each i=1,2,$\cdots$,n,

$$||\sum_{j=1}^{i+1} J_{\Pi_j}||_\infty = \min_{l=i+1,....,n} \left\{ ||\sum_{j=1}^{i} J_{\Pi_j} + J_{\Pi_l}||_\infty \right\}.$$

Then Π is called *max-norm* based greedy (aligned) ordering. That is, at every step we choose the next job as the one that along with the previously chosen ones has smallest completion time. On the other hand if Π denotes the permutation of jobs that orders jobs in the increasing order of their *1-norms*. That is,
$\sum_{1\le k\le m} a_1^k \le \sum_{1\le k\le m} a_2^k \le \cdots \le \sum_{1\le k\le m} a_n^k$.
Then we call Π as the *1-norm* based (aligned) greedy ordering.
In general, we call a permutation Π, a *q-norm* based greedy ordering if Π_{i+1} is chosen as $l \in \{1,2,...,n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$ that minimizes

$$f_q(l,\Pi,i) = ||\sum_{1\le j\le i} J_{\Pi_j} + J_l||_q - ||\sum_{1\le j\le i} J_{\Pi_j}||_q.$$

Performance Guarantees for Greedy Strategies

To find an aligned optimal schedule, we attempt to find an ordering of the jobs, Π. Let's suppose we already have computed the first i indices in Π (ie. $\Pi_1, \Pi_2, ..., \Pi_i$). Now to compute Π_{i+1}, we consider the following function that evaluates each choice for Π_{i+1}.

For each $l \in \{1,2,...,n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$, we compute $f(l,\Pi,i)$ that depends upon the greedy choices already made, ie. $\Pi_1, \Pi_2, ..., \Pi_i$ and $l \in \{1,2,...,n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$. We set Π_{i+1} to that l that minimizes the function $f(l,\Pi,i)$ over $l \in \{1,2,...,n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$. Clearly, when we define $f(l,\Pi,i) = \sum_{1\le k\le m} a_l^k$, we get the 1-norm based greedy ordering. Similarly, if we define $f(l,\Pi,i) = ||(\sum_{1\le j\le i} J_{\Pi_j}) + J_l||_\infty$, we get the max-norm based greedy ordering.

We analyze the performance of a class of greedy strategies based on various norms, *viz.*, q-norm ($q = 1,2,...,\infty$). We call a permutation Π, a *q-norm greedy ordering* if Π_{i+1} is chosen as $l \in \{1,2,...,n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$ that minimizes $f_q(l,\Pi,i) = ||\sum_{1\le j\le i} J_{\Pi_j} + J_l||_q$. Let

- $\hat{l}$ denote the index l that minimizes the above, that is, $\Pi_{i+1} = \hat{l}$;

- l' denote $l \in \{1, 2, ..., n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$ that minimizes $||J_l||_1$;
- $incrCost_{i+1}(\Pi) = ||\sum_{1 \le j \le i+1} J_{\Pi_j}||_\infty - ||\sum_{1 \le j \le i} J_{\Pi_j}||_\infty$.

We shall show that $incrCost_{i+1}(\Pi) \le ||J_{l'}||_1$. Once we establish $incrCost_{i+1}(\Pi) \le ||J_{l'}||_1$, we obtain the following performance guarantee.

$$\begin{aligned} cost(\Pi) &= \sum_{1 \le i \le n} \sum_{i \le j \le i} incrCost_j(\Pi) \\ &\le \sum_{1 \le i \le n} \sum_{1 \le j \le i} ||J_{l'}||_1 \\ &\le m \times cost(\Pi_{opt}). \end{aligned}$$

Suppose the contrary, that is, $incrCost_{i+1}(\Pi) > ||J_{l'}||_1$. We will now arrive at a contradiction that says, $f_q(\hat{l}, \Pi, i) > f_q(l', \Pi, i)$. For the sake of convenience, let X denote $\sum_{1 \le j \le i} J_{\Pi_j}$. By raising both sides to q^{th} power, the task reduces to establishing

$$\sum_{1 \le k \le m} (X^k + a_{\hat{l}}^k)^q > \sum_{1 \le k \le m} (X^k + a_{l'}^k)^q. \tag{4}$$

Using the repeated application of the inequality

$$(u + \beta)^q + (v + \gamma)^q \le (u)^q + (v + \beta + \gamma)^q,$$

for $q \ge 1$, $u, v, \beta, \gamma \ge 0$ and $u \le v$, it is easy to see that,

$$\sum_{1 \le k \le m} (X^k + a_{l'}^k)^q \le (\sum_{1 \le k \le m, k \ne \underline{k}} (X^k)^q) + (X^{\underline{k}} + ||J_{l'}||_1)^q, \tag{5}$$

where $\underline{k}$ is k that maximizes X^k. Also

$$\sum_{1 \le k \le m} (X^k + a_{\hat{l}}^k)^q \ge (\sum_{1 \le k \le m, k \ne \bar{k}} (X^k)^q) + (X^{\bar{k}} + a_{\hat{l}}^{\bar{k}})^q, \tag{6}$$

where $\bar{k}$ is k that maximizes $X^k + a_{\hat{l}}^k$.

To arrive at the required contradiction, it suffices to prove that

$$((X^{\bar{k}} + a_{\hat{l}}^{\bar{k}})^q - (X^{\bar{k}})^q) > ((X^{\underline{k}} + ||J_{l'}||_1)^q - (X^{\underline{k}})^q). \tag{7}$$

Note that, due to our assumption, $incrCost_{i+1}(\Pi) > ||J_{l'}||_1$ that is,
$||\sum_{1 \le j \le i+1} J_{\Pi_i}||_\infty - ||\sum_{1 \le j \le i} J_{\Pi_i}||_\infty = (X^{\bar{k}} + a_{\hat{l}}^{\bar{k}}) - X^{\underline{k}} > ||J_{l'}||_1$ we get,

$$(X^{\bar{k}} + a_{\hat{l}}^{\bar{k}}) > (X^{\underline{k}} + ||J_{l'}||_1). \tag{8}$$

This and $X^{\bar{k}} < X^{\underline{k}}$, together yield the following contradiction.

$$((X^{\bar{k}} + a_{\hat{l}}^{\bar{k}})^q - (X^{\bar{k}})^q) > ((X^{\underline{k}} + ||J_{l'}||_1)^q - (X^{\underline{k}})^q). \tag{9}$$

Therefore, we have the following theorem:
Theorem For the problem of scheduling m-dimensional jobs to minimize total completion time, the q-norm based greedy ordering strategy (or its static variant) finds an aligned schedule whose cost is within m times the optimal.

Lower Bound on Performance Guarantee of Greedy Schedule

Though we have not been able to prove that the above bound is tight or derive a tighter bound, we provide an example for which, the *1-norm* and *max-norm* based strategy finds a schedule with cost $\sqrt{m}$ times the cost of an optimal schedule, where m is the number of machines.

Now, we will compute a lower bound on the worst case performance guarantee of the max-norm and 1-norm based greedy schedules.

Consider the following input to the 2-dimensional job sequencing problem.

$J_i = (1,1) \qquad \forall i = 1,2,\ldots,n$

$J_i = (1,0) \quad \forall i = n+1, n+3,\ldots,n+2m-1$

$J_i = (0,1) \qquad \forall i = n+2, n+4,\ldots,n+2m$

Consider the following ordering (expressed as a permutation of $[1,2,3,...,n+2m]$) that is one of those induced by max-norm based greedy ordering strategy, $\Pi = [1,2,3,\ldots,n+2m]$. It can be easily seen that the cost of this ordering is

$$\begin{aligned} cost(\Pi) &= \sum_{i=1,2,\ldots,n} i + 2 \times \sum_{j=1,2,\ldots,m} (n+j) \\ &= (\frac{n}{2})(n+1) + 2mn + m(m+1) \end{aligned}$$

On the other hand, consider the permutation $\Pi' = [n+1, n+2, \ldots, n+2m, 1, 2, 3, \ldots, n]$.

$$\begin{aligned} cost(\Pi') &= 2 \times \sum_{i=1,2,\ldots,m} i + \sum_{j=1,2,\ldots,n} (m+j) \\ &= m(m+1) + mn + (\frac{n}{2})(n+1) \end{aligned}$$

Let $m = \alpha \times n$. It is easy to show that,

$$\lim_{n\to\infty} \frac{cost(\Pi)}{cost(\Pi')} = \frac{1+4\alpha+2\alpha^2}{1+2\alpha+2\alpha^2}.$$

Using differential calculus, it is easy to show that $\frac{1+4\alpha+2\alpha^2}{1+2\alpha+2\alpha^2}$ maximizes at $\alpha = \frac{1}{\sqrt{2}}$. And for this value of α, the ratio

$$\begin{aligned}\lim_{n\to\infty} \frac{cost(\Pi)}{cost(\Pi')} &= \frac{1+4\alpha+2\alpha^2}{1+2\alpha+2\alpha^2} \\ &= 1 + \frac{\sqrt{2}}{2+\sqrt{2}}. \\ &= \sqrt{2}\end{aligned}$$

Thus,
Theorem The max-norm based greedy ordering strategy finds an aligned schedule whose cost is within c times the optimal for some $c \in [\sqrt{2}, 2]$.

A similar argument (with the help of a slightly different construction of an input compared to the one above) can be provided to show the following.
Theorem The 1-norm based greedy ordering strategy finds an aligned schedule whose cost is within c times the optimal for some $c \in [\sqrt{2}, 2]$.

We can tighten the above result for 1-norm based greedy schedule, with the help of the following example.

Consider the following collection of jobs.

$J_i = (1, 0) \qquad \forall i = 1, 2, \ldots, n$

$J_i = (\epsilon, 1) \ \forall i = n+1, n+2, \ldots, 2n$

It can be easily seen that the 1-norm based greedy ordering is asymptotically (as $n \to \infty$ and $\epsilon \to 0$) greater or equal to 1.5 times the optimal.

Theorem The 1-norm based greedy ordering strategy finds an aligned schedule whose cost is within c times the optimal for some $c \in [1.5, 2]$.
Consider the following input to the m-dimensional job sequencing problem.

$J_i = (1, 1, \ldots, 1) \qquad \forall i = 1, 2, \ldots, n$

$J_i = (1, 0, 0, \ldots, 0) \ \forall i = n+1, n+m+1, \ldots, n+mp-(m-1)$

$J_i = (0, 1, 0, \ldots, 0) \ \forall i = n+2, n+m+2, \ldots, n+mp-(m-2)$

$\ldots \qquad \ldots$

$J_i = (0, 0, 0, \ldots, 1) \qquad \forall i = n+m, n+2m, \ldots, n+mp$

Consider the following ordering (expressed as a permutation of $[1, 2, 3, ..., n + mp]$) that is one of those induced by max-norm based greedy ordering strategy, $\Pi = [1, 2, 3, \ldots, n + mp]$. It can be easily seen that the cost of this ordering is

$$\begin{aligned} cost(\Pi) &= \sum_{i=1,2,\ldots,n} i + m \times \sum_{j=1,2,\ldots,p} (n + j) \\ &= (\frac{n}{2})(n + 1) + mpn + (\frac{m}{2})p(p + 1) \end{aligned}$$

On the other hand, consider the permutation $\Pi' = [n + 1, n + 2, \ldots, n + mp, 1, 2, 3, \ldots, n]$.

$$\begin{aligned} cost(\Pi') &= m \times \sum_{i=1,2,\ldots,p} i + \sum_{j=1,2,\ldots,n} (p + j) \\ &= (\frac{m}{2})p(p + 1) + pn + (\frac{n}{2})(n + 1) \end{aligned}$$

Let $p = \alpha \times n$. It is easy to show that,

$$\lim_{n\to\infty} \frac{cost(\Pi)}{cost(\Pi')} = \frac{1 + 2m\alpha + m\alpha^2}{1 + 2\alpha + m\alpha^2}.$$

Using differential calculus, it is easy to show that $\frac{1+2m\alpha+m\alpha^2}{1+2\alpha+m\alpha^2}$ attains maximum value at $\alpha = \frac{1}{\sqrt{m}}$. And for this value of α, the ratio becomes $\sqrt{m}$.

$$\begin{aligned} \lim_{n\to\infty} \frac{cost(\Pi)}{cost(\Pi')} &= \frac{1 + 2m\alpha + m\alpha^2}{1 + 2\alpha + m\alpha^2} \\ &= \sqrt{m}. \end{aligned}$$

Theorem For the problem of scheduling m-dimensional jobs on m machines, the max-norm based greedy ordering strategy finds an aligned schedule whose cost is within c times the optimal for some $c \in [\sqrt{m}, m]$.
A similar argument (with the help of a slightly different construction of an input compared to the one above) can be provided to show the following.
Theorem For the problem of scheduling m-dimensional jobs on m machines, the 1-norm based greedy ordering strategy finds an aligned schedule whose cost is within c times the optimal for some $c \in [\sqrt{m}, m]$.

We show using an example that even if, in an input the *1-norm* and *max-norm* based greedy orderings are identical, the greedy solution need not be optimal.

Consider the following jobs:

$J_1 = (10, 0)$

$J_2 = (1, 19)$

$J_3 = (9 + \epsilon, 11)$

$J_4 = (9 + \epsilon, 11)$.

Clearly, the ordering (J_1, J_2, J_3, J_4) is both 1-norm based as well as max-norm based greedy ordering. However, its cost is $10 + 19 + 30 + 41$. On the other hand, the cost of the ordering (J_1, J_3, J_4, J_2) is $10+19+\epsilon+28+2\epsilon+41$, which is lower than that of the greedy ordering described above.

Approximate Solutions to the Stochastic problem

In the stochastic version, job completion times are random variables instead of fixed constants. Stochastic versions of various optimization problems including processor scheduling have been studied by many researchers in the past Mohring et al. (1999); Stougie (1987). Mohring et al. extended Queyranne's inequalities for deterministic job scheduling to the stochastic case in Mohring et al. (1999). They derived a $3-(1/m)+max\{1,((m-1)/m)\Delta\}$ approximate algorithm for single component stochastic job scheduling, where m is the number of identical parallel machines and Δ is an upper bound on the co-efficient of variation of the job processing times.

Our main difficulty in extending the deterministic multi-dimensional job sequencing problem to stochastic setting (as done in Mohring et al. (1999) for single component job scheduling) is that unlike for single component job scheduling, the linear programs for multi-dimensional job scheduling cannot be extended in any obvious way for the stochastic case. This is because the expected completion time of a vector job is equal to the expectation of the maximum of its completion time on all machines (and not the maximum of the expectations of the completion times). Expectations of maximum of a set of random variables are difficult to analyze and in general, cannot be expressed using linear inequalities. Moreover, even for special cases of processing time distributions, the problem looks hard as we need to compute not just the expectation of the maximum of a set of random variables, but the expectation of the maximum of sums of random variables. Very little work on computing the expectation of maximum of sums of random variables exactly or approximately can be found in the literature and this is one possible direction for further research.

We extend Potts' LP formulation for the deterministic case to a formulation for a special case of stochastic vector job scheduling. In this setting, there are different possible scenarios, each of which can occur with a specified probability. The processing time of all job components in each scenario is known. For the sake of simplicity of discussion, we consider the special case of 2 machines and 2 scenarios. Let p and q denote the probabilities of the 2 scenarios. Let a_i^1 and a_i^2 denote the processing time requirements for i^{th} job on the first machine in the first and second scenario respectively. Similarly, let b_i^1 and b_i^2 denote the processing time requirements for i^{th} job on the second machine in the first and second scenario respectively. We use C_i^1 and C_i^2, respectively, to denote the completion times of the i^{th} job in the first and second scenarios. Let $\overline{a}_i$ and $\overline{b}_i$ denote the expected processing time of i^{th} job on the two machines. Consider the following stochastic integer program that models our problem of minimizing the expected total completion time.

$$\begin{gathered}
\min \ \textstyle\sum_{i=1}^{n}(pC_i^1 + qC_i^2) \\
C_i^1 \geq a_i^1 + \textstyle\sum_{j\neq i} x_{ji}a_j^1 \ \ \forall\ i = 1, 2, \ldots, n \\
C_i^1 \geq b_i^1 + \textstyle\sum_{j\neq i} x_{ji}b_j^1 \ \ \forall\ i = 1, 2, \ldots, n \\
C_i^2 \geq a_i^2 + \textstyle\sum_{j\neq i} x_{ji}a_j^2 \ \ \forall\ i = 1, 2, \ldots, n \\
C_i^2 \geq b_i^2 + \textstyle\sum_{j\neq i} x_{ji}b_j^2 \ \ \forall\ i = 1, 2, \ldots, n \\
x_{ij} + x_{ji} = 1 \ \forall\ i, j = 1, 2, \ldots, n \text{ and } i \neq j \\
x_{ij} + x_{jk} + x_{ki} = 2 \ \forall\ i, j, k = 1, 2, \ldots, n \text{ and } i \neq j \neq k \\
x_{ij} = 0 \text{ or } 1 \ \forall\ i \neq j
\end{gathered} \tag{10}$$

By aggregating corresponding inequalities using scale factors p and q, we get the following relaxation of the above.

$$\begin{gathered}
\min \ \textstyle\sum_{i=1}^{n}(pC_i^1 + qC_i^2) \\
pC_i^1 + qC_i^2 \geq \overline{a_i} + \textstyle\sum_{j\neq i} x_{ji}\overline{a_j} \ \ \forall\ i = 1, 2, \ldots, n \\
pC_i^1 + qC_i^2 \geq \overline{b_i} + \textstyle\sum_{j\neq i} x_{ji}\overline{b_j} \ \ \forall\ i = 1, 2, \ldots, n \\
x_{ij} + x_{ji} = 1 \ \forall\ i, j = 1, 2, \ldots, n \text{ and } i \neq j \\
x_{ij} + x_{jk} + x_{ki} = 2 \ \forall\ i, j, k = 1, 2, \ldots, n \text{ and } i \neq j \neq k \\
x_{ij} = 0 \text{ or } 1 \ \forall\ i \neq j
\end{gathered} \tag{11}$$

By introducing the variables $\overline{C}_i$ for $pC_i^1 + qC_i^2$ for $i = 1, 2, \ldots, n$, we get the following relaxation of our original stochastic integer program.

$$\begin{aligned} &\min \ \textstyle\sum_{i=1}^{n} \overline{C}_i \\ &\overline{C}_i \geq \overline{a}_i + \textstyle\sum_{j \neq i} x_{ji} \overline{a}_j \ \ \forall \ i = 1, 2, \ldots, n \\ &\overline{C}_i \geq \overline{b}_i + \textstyle\sum_{j \neq i} x_{ji} \overline{b}_j \ \ \forall \ i = 1, 2, \ldots, n \\ &x_{ij} + x_{ji} = 1 \ \forall \ i, j = 1, 2, \ldots, n \text{ and } i \neq j \end{aligned} \tag{12}$$

$$x_{ij} = 0 \text{ or } 1 \ \forall \ i \neq j$$

Theorem The LP relaxation of the problem 12 provides a 4-factor approximation algorithm for our stochastic original stochastic formulation.
Proof Let $x_{ij}^*, \overline{C}_i^*$ for $i, j = 1, 2, \ldots, n$ and $i \neq j$ denote an optimal solution to the LP relaxation of the problem 12. Without loss of generality, we assume that the jobs are numbered such that
$\overline{C}_1^* \leq \overline{C}_2^* \ldots \leq \overline{C}_n^*$. Therefore the completion time of i^{th} job in the above ordering, denoted by $\hat{C}_i$, would be the random variable given by $\max \left\{ \sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j \right\}$.

We need to show that, the total expected completion time under the above ordering induced by an optimal solution of the LP relaxation of the problem 12 is within a constant factor of the optimal solution to the expected completion time problem. The total expected completion time under the above ordering is
$\sum_{i=1}^{n} E[\max \left\{ \sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j \right\}]$.
Note that, for any two random variables, say X and Y, the following holds

$$E[\max\{X, Y\}] \leq E[X + Y] = E[X] + E[Y] \leq 2 \times max(E[X], E[Y]).$$

Therefore,

$$\sum_{i=1}^{n} E[\max \left\{ \sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j \right\}] \leq \sum_{i=1}^{n} 2 \times \max \left\{ \sum_{j=1}^{i} \overline{a}_j, \sum_{j=1}^{i} \overline{b}_j \right\}.$$

Next, we will show that

$$\sum_{i=1}^{n} \max \left\{ \sum_{j=1}^{i} \overline{a}_j, \sum_{j=1}^{i} \overline{b}_j \right\} \leq 2 \times \text{opt(LP relaxation of problem 12)}.$$

That is,

$$\sum_{i=1}^{n} \max\left\{\sum_{j=1}^{i} \overline{a}_j, \sum_{j=1}^{i} \overline{b}_j\right\} \leq 2 \times \sum_{i=1}^{n} \overline{C}_i^*.$$

Therefore, it suffices to prove that

$$\max\left\{\sum_{j=1}^{i} \overline{a}_j, \sum_{j=1}^{i} \overline{b}_j\right\} \leq 2 \times \overline{C}_i^*.$$

Fix an i. W.l.o.g. assume that $\sum_{j=1}^{i} \overline{a}_j \geq \sum_{j=1}^{i} \overline{b}_j$.

$$\begin{aligned}
(\sum_{j=1}^{i} \overline{a}_j)] \times \overline{C}_i^* &\geq \sum_{j=1}^{i} (\overline{a}_j \times \overline{C}_j^*) \\
&\geq \sum_{j=1}^{i} \overline{a}_j^2 + \sum_{l \neq j, l, j=1,2,\ldots,n} ((x_{lj}^* + x_{jl}^*) \overline{a}_l \overline{a}_j) \\
&\geq \frac{1}{2} \sum_{j=1}^{i} \overline{a}_j^2 + \frac{1}{2} (\sum_{j=1}^{i} \overline{a}_j)^2 \\
&\geq \frac{1}{2} (\sum_{j=1}^{i} \overline{a}_j)^2.
\end{aligned}$$

Therefore

$$\max\left\{\sum_{j=1}^{i} \overline{a}_j, \sum_{j=1}^{i} \overline{b}_j\right\} \leq 2 \times \overline{C}_i^*.$$

Noting that the LP relaxation of the problem 12 is a relaxation of the problem 10, we get

$$\sum_{i=1}^{n} E[\max\left\{\sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j\right\}] \leq 4 \times \text{opt(problem 10)}.$$

In the quality of the approximation in the proof is influenced by the ratio of expectation of maximum of random variables to the maximum of their expectations. In the above case it was pessimistically taken as 2. It may be lower in practice. The above analysis can be extended to the general case, giving a performance guarantee of $2 \times \Gamma$ for the case of m machines, where Γ (in the asymptotic sense) equals the ratio of expectation of max of m normal random variables to that of maximum of their expectations.

Greedy heuristic

Now we consider the problem of computing an optimum schedule among the aligned schedules. For the sake of simplicity of analysis we consider the 2-machine problem only. Our goal is to find a permutation Π of {1,2,...,n} such that

$$E(\Pi) = E[\sum_{i=1}^{n} \max \left\{ \sum_{j=1}^{i} a_{\Pi_j}, \sum_{j=1}^{i} b_{\Pi_j} \right\}]$$

is minimum.

Let

- C_{opt} denote the cost of such an optimum schedule, say Π^{opt} (i.e., $C_{opt} = E(\Pi^{opt})$),
- Π^* denote the permutation of jobs that orders jobs in the increasing order of the expected value of their 1-norms.

Clearly,

$$\begin{aligned} C_{opt} &= E(\Pi^{opt}) \\ &\geq \sum_{i=1}^{n} \frac{1}{2} (\sum_{j=1}^{i} (\overline{a}_{\Pi_j^{opt}} + \overline{b}_{\Pi_j^{opt}})) \\ &\geq \sum_{i=1}^{n} \frac{1}{2} (\sum_{j=1}^{i} (\overline{a}_{\Pi_j^{*}} + \overline{b}_{\Pi_j^{*}})) \end{aligned} \tag{13}$$

Now let Π' be a permutation such that for each *i=1,2, …, n*

$$E[\max \left\{ \sum_{j=1}^{i+1} a_{\Pi'_j}, \sum_{j=1}^{i+1} b_{\Pi'_j} \right\}] = \min_{l=i+1,\ldots,n} \{E[\max \left\{ \sum_{j=1}^{i} a_{\Pi'_j} + a_{\Pi'_l}, \sum_{j=1}^{i} b_{\Pi'_j} + b_{\Pi'_l} \right\}]\}$$

In other words, $\Pi^{'}$ is max-norm based greedy (aligned) schedule. That is, at every step we choose the next job as the one that along with the previously chosen ones has smallest completion time.

Theorem

$$\text{cost}(\Pi^{'}) \leq 2 \times C_{opt},$$

Proof Consider $cost(\Pi^{'})$,

$$cost(\Pi^{'}) = E[\sum_{i=1}^{n} \max \left\{ \sum_{j=1}^{i+1} a_{\Pi'_j}, \sum_{j=1}^{i+1} b_{\Pi'_j} \right\}]$$

for *i=1* it can be easily verified that

$$E[\max\left\{\sum_{j=1}^{i} a_{\Pi'_j}, \sum_{j=1}^{i} b_{\Pi'_j}\right\}] \leq \sum_{j=1}^{i} (\overline{a}_{\Pi^*_j} + \overline{b}_{\Pi^*_j}) \tag{14}$$

Assume that the equation 14 holds for *i=2 ... p*. Now consider *i = p+1*.

$$\begin{aligned}
E[\max\left\{\sum_{j=1}^{i} a_{\Pi'_j}, \sum_{j=1}^{i} b_{\Pi'_j}\right\}] &= \min_{l=p+1,\ldots,n} \{E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j} + a_{\Pi'_l}, \sum_{j=1}^{p} b_{\Pi'_j} + b_{\Pi'_l}\right\}]\} \\
&\leq \min_{l=p+1,\ldots,n} \{E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j}, \sum_{j=1}^{p} b_{\Pi'_j}\right\}] + E[\max\left\{a_{\Pi'_l}, b_{\Pi'_l}\right\}]\} \\
&\leq E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j}, \sum_{j=1}^{p} b_{\Pi'_j}\right\}] + \min_{l=p+1,\ldots,n} \{E[\max\left\{a_{\Pi'_l}, b_{\Pi'_l}\right\}]\} \\
&\leq E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j}, \sum_{j=1}^{p} b_{\Pi'_j}\right\}] + \min_{l=p+1,\ldots,n} E[a_{\Pi'_l} + b_{\Pi'_l}] \\
&\leq E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j}, \sum_{j=1}^{p} b_{\Pi'_j}\right\}] + \min_{l=p+1,\ldots,n} (\overline{a}_{\Pi'_l} + \overline{b}_{\Pi'_l}) \\
&\leq E[\max\left\{\sum_{j=1}^{p} a_{\Pi'_j}, \sum_{j=1}^{p} b_{\Pi'_j}\right\}] + (\overline{a}_{\Pi^*_{p+1}} + \overline{b}_{\Pi^*_{p+1}}) \\
&\leq \sum_{j=1}^{p} (\overline{a}_{\Pi^*_j} + \overline{b}_{\Pi^*_j}) + (\overline{a}_{\Pi^*_{p+1}} + \overline{b}_{\Pi^*_{p+1}}) \\
&\leq \sum_{j=1}^{p+1} (\overline{a}_{\Pi^*_j} + \overline{b}_{\Pi^*_j})
\end{aligned}$$

$\therefore$ by induction hypothesis,

$$cost(\Pi') \leq \sum_{j=1}^{n} (\overline{a}_{\Pi^*_j} + \overline{b}_{\Pi^*_j}). \tag{15}$$

Using the results from the equations 13 and 15, we have

$$\text{cost}(\Pi') \leq 2 \times C_{opt}$$

Conclusion and Discussion

We discussed and developed approximate solutions for scheduling vector jobs using an LP based approach and a greedy heuristic. We analysed the bounds obtained for the deterministic and the stochastic formulation of the problem. The LP based approach was the extension of Potts' formulation for single component job scheduling.

For deterministic formulation of the problem, Potts' formulation is a practical approach with a performance guarantee of factor *2*. Moreover, the LP involves a polynomial number of variables and inequalities and the special structure of its coefficient matrix can be used to speed up the algorithm to solve it. The greedy approaches, which be computationally much faster than any LP based approach, have a performance guarantee of factor m (the number of machines).

It is difficult to extend the LP based approaches to the general version of Stochastic vector job scheduling. However, Potts' LP approach can be extended to handle a special case of stochastic vector job scheduling to give a performance guarantee of $2m$(twice the number of machines). The greedy strategies on the other hand can be used to solve the general version of stochastic vector job scheduling with the same performance guarantee of m, the number of the machines, as for the deterministic case.

Acknowledgment

The work was funded by the Engineering and Physical Sciences Research Council's (U.K) grant number EP/C500148/1. This grant allowed Prof. Patkar to visit Brunel university and carry out this joint work.

References

Chen, Z., Hall, N., 2005. Supply chain scheduling: Conflict and cooperation in assembly systems, working paper.

Hall, L., Schulz, A., Shmoys, D., Wein, J., 1996. Scheduling to minimize average completion time: Off-line and on-line algorithms. In: Proc. of the 7th ACMSIAM Symp. on Discrete Algorithms. pp. 142–151.
URL `citeseer.ist.psu.edu/hall97scheduling.html`

Hall, N., 1998. A comparison of inventory replenishment heuristics for minimizing maximum storage. American Journal of Mathematical and Management Sciences 18, 245–258.

Mohring, R., Schulz, A., Uetz, M., 1999. Approximation in stochastic scheduling: the power of lp-based priority policies. J. ACM 46 (6), 924–942.

Potts, C., 1980. An algorithm for the single machine sequencing problem with precedence constraints. Mathematical Programming Studies 13, 78–87.

Potts, C., Sevastjnov, S., Strusevich, V. A., van Wassenhove, L. N., Zwaneveld, C., 1995. The two-stage assembly scheduling problem: complexity and approximation. Operations Research 42 (2), 346–355.

Smith, W., 1956. Various optimizers for single-stage production. Naval Reserch Logistics Quaterly 3, 59–66.

Stougie, L., 1987. Design and analysis of algorithms for stochastic integer programming. CWI Tract 37, 48–62.