TECHNICAL REPORT

CTR/41/05 November 2005

An Investigation into Approximate Solutions for Deterministic and Stochastic Multi-Dimensional Sequencing

S Patkar*, C.A. Poojari**, J Porwal***

*Department of Mathematics, Indian Institute of Technology Bombay, Mumbai
**CARISMA, School of Information Systems, Computing and Mathematics, Brunel University
***Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai

Brunel UNIVERSITY WEST LONDON

School of Information Systems, Computing and Mathematics
Brunel University, Uxbridge, Middlesex, UB8 3PH
www.carisma.brunel.ac.uk

CARISMA

An Investigation into Approximate Solutions for Deterministic and Stochastic Multi-Dimensional Sequencing.

Sachin Patkar[1], Chandra A Poojari[2], and Janak Porwal[3] *

[1] patkar@math.iitb.ac.in
Department of Mathematics, Indian Institute of Technology Bombay, Mumbai
[2] chandra.poojari@brunel.ac.uk
Centre for the Analysis of Risk and Optimisation Modelling Applications, School of Information Systems, Computing and Mathematics, Brunel University, London
[3] janak@cse.iitb.ac.in
Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Mumbai

Abstract. We investigate the problem that involves sequencing jobs having multiple components. Each component of the job needs to be processed independently on 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 an LP based and a greedy strategy. The LP based approach is an extension of (Sivasubramanian, 2003) to adapt Potts' formulation (Potts, 1980; Hall et al., 1996) for single component job scheduling. We propose bounds on the performance ratio of the two approaches for deterministic and stochastic formulation of the problem.

1 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. Manufacturing a composite bearing initially requires the individual components to be built. Such components include ball bearings, needle bearings and shaft bearings. The components are typically processed independently on different machines and then assembled together. 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 studied recently 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

* The work was funded by the Engineering and Physical Sciences Research Council's (U.K) grant number EP/C500148/1.

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.

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. 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 (Sivasubramanian, 2003; 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 discuss approximate solutions using a greedy and an LP based method for the deterministic and stochastic formulation of vector job scheduling problem.

1.1 Problem definition

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
p_i^k = processing time taken to complete the k^{th} component of i^{th} job, $p_i^k \geq 0$.
J_i = the i^{th} job, defined as $(p_i^1, p_i^2, \ldots, p_i^m)$,
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 preceeding and including over i^{th} job,

$$C_{\Pi_i} = \max_{k=1}^{m} \{ \sum_{j=1}^{i} p_{\Pi_j}^k \}. \tag{1}$$

We wish to determine an optimal schedule, Π^*, which minimizes the total completion time,

$$\mathcal{C} = \sum_{i=1}^{n} C_{\Pi_i}. \tag{2}$$

1.2 Terms

We define the following terminologies:
performance ratio of an algorithm = 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.

Claim. There exists an aligned schedule that minimises the sum of completion times of all jobs.

Therefore our focus would be on aligned schedules.

2 Approximate solutions to the deterministic problem

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

2.1 Linear programming based analysis

In (Sivasubramanian, 2003) an Integer Linear Programming (ILP) formulation of Potts ((Potts, 1980))) was extended to derive a 2-factor approximation algorithm for vector job scheduling.

Potts' formulation first appeared in (Potts, 1980) for the problem of single machine job scheduling with precedence constraints to minimize total completion time. The 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 is 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.

Formulation *VARIABLES*
δ_{ij} = a binary variable denoting the order between the i^{th} and the j^{th} job,

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

C_i = completion time for the i^{th} job. The complete ILP formulation can be written as:

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

Subject to

$$\delta_{ij} + \delta_{ji} = 1 \qquad \forall \{i, j = 1, \ldots, n;\ i < j\} \tag{3a}$$

$$C_i \geq p_i^k + \sum_{j \neq i} p_j^k \delta_{ji} \quad \forall \{i = 1, \ldots, n;\ k = 1, \ldots, m;\ p_i^k > 0\} \tag{3b}$$

$$\delta_{ij} \in \{0, 1\} \tag{3c}$$

The original formulation by Potts also had the inequalities

$$\delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2 \ \forall \{i, j, k = 1, \ldots, n;\ i < j < k \text{ or } k < j < i\} \tag{4}$$

for ensuring that the ordering of jobs does not result in any cycles. However, for a performance guarantee of factor 2, the algorithm of (Sivasubramanian, 2003) does not require these constraints to be included in the formulation. The above problem contains $\frac{n(n-1)}{2}$ constraints of type 3a and nm constraints of type 3b.

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 5,

$$\Pi_i = k \text{ where } \delta_{k.} = n - i \quad \forall i. \tag{5}$$

Integrality for single machine case Consider an LP relaxation of the problem 3 in which we replace the constraint 3c with

$$0 \leq \delta_{ij} \leq 1.$$

Claim. An optimal solution to the LP relaxation of problem 3 for the single machine case, is integral if $p_i \neq p_j$ for all i and j.

Proof. Replace δ_{ji} by $1 - \delta_{ij}$ for all $j > i$. For the single machine case, inequalities (3b) are satisfied exactly and hence the completion time of job i is:

$$C_i = p_i + \sum_{j<i} p_j \delta_{ji} + \sum_{j>i} p_j (1 - \delta_{ij}) = \sum_{j \geq i} p_j + \sum_{j<i} (p_j - p_i) \delta_{ji} \tag{6}$$

Thus the objective function to be minimized is $\sum_{i=1}^{n} \sum_{j<i} (p_j - p_i) \delta_{ji}$ plus a constant term. It is clear that the minimum value of this function is achieved when $\delta_{ji} = 0$ for $p_j > p_i$ and 1 for $p_j < p_i$. Thus, for the single machine case, there always exist an optimal integral solution to the LP relaxation of Potts' formulation and in particular, a unique optimal solution that is integral, when all processing times are distinct.

Note that the above claim in section 2.1 is just a different way of deriving the well known Smith's rule for single processor scheduling.

Number of non-zero entries Consider a basic feasible solution of (3). The rank of the coefficient matrix is at most the number of inequalities $(n(n-1)/2+nm)$. From a well known result of linear programming, the maximum number of non-zero variables in any basic feasible solution (bfs) of an LP is the rank of the coefficient matrix. Thus, the maximum number of non-zero variables in a bfs of (3) is $n(n-1)/2+nm$. It follows that a maximum of $n(n-1)/2+n(m-1)$ of the δ_{ij}'s are non-zero. Equivalently, a maximum of $n(m-1)$ equations in (3a) can have fractional δ_{ij} values. It may be possible to use this observation to devise efficient strategies to round-off the fractional optimal solution to get an integral solution. However, we have not investigated this idea in full detail.

Speeding-up the algorithm In this section we show how 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 (3a) 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 (3a). 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:

$$min \sum_{i=1}^{n} C_i \tag{7a}$$

such that :

$$C_i + \sum_{i<j} \delta_{ij} p_j^k - \sum_{j<i} p_i^k \delta_{ji} \geq \sum_{j=i}^{n} p_j^k \quad \forall\ i = 1, \ldots, n;\ k = 1, \ldots, m \tag{7b}$$

$$-\delta_{ij} \geq -1 \quad \forall\ i < j;\ i, j \in \{1, \ldots, n\} \tag{7c}$$

$$\delta_{ij} \geq 0 \quad \forall\ i < j;\ i, j \in \{1, \ldots, n\} \tag{7d}$$

The coefficient matrix of the LP comprising of (7a-7b) is an $(nm) \times (n(n+1)/2 + nm)$ matrix, in which the column for variable δ_{ij} contains $-p_i^k$ in the $((k-1)n+i)^{th}$ row, p_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.

2.2 Greedy heuristic based analysis

We discuss a greedy strategy for the deterministic version of our problem, which is based on intuitive extensions of the *shortest job first* rule for single component job scheduling to minimize total completion time. This algorithm generalizes those presented in (Sivasubramanian, 2003). We also derive bounds on its worst case performance. In the following, the q-norm of a vector $V = (v_1, v_2, \ldots, v_n)$, denoted by $||V||_q$, is the value $(\sum_{i=1}^{n}(v_i)^q)^{\frac{1}{q}}$.

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

$$f(l, \Pi, i) = \sum_{1 \leq k \leq m} p_l^k,$$

we get the 1-norm based greedy ordering. Similarly, if we define

$$f(l, \Pi, i) = ||(\sum_{1 \leq j \leq i} J_{\Pi_j}) + J_l||_\infty,$$

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

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

Let $\hat{l}$ denote the index l that minimizes the above. That is, $\Pi_{i+1} = \hat{l}$.
Let l' denote $l \in \{1, 2, ..., n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$ that minimizes $||J_l||_1$.
Let

$$incrCost_{i+1}(\Pi) = || \sum_{1 \leq j \leq i+1} J_{\Pi_j}||_\infty - || \sum_{1 \leq j \leq i} J_{\Pi_j}||_\infty.$$

We shall show that

$$incrCost_{i+1}(\Pi) \leq ||J_{l'}||_1.$$

Once we establish

$$incrCost_{i+1}(\Pi) \leq ||J_{l'}||_1,$$

we obtain the following performance guarantee.

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

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

$$\sum_{1\le k\le m} (X^k + p^k_{\hat{l}})^q > \sum_{1\le k\le m} (X^k + p^k_{l'})^q. \tag{8}$$

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,

$$\sum_{1\le k\le m} (X^k + p^k_{l'})^q \le (\sum_{1\le k\le m, k\ne \underline{k}} (X^k)^q) + (X^{\underline{k}} + ||J_{l'}||_1)^q, \tag{9}$$

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

Also

$$\sum_{1\le k\le m} (X^k + p^k_{\hat{l}})^q \ge (\sum_{1\le k\le m, k\ne \bar{k}} (X^k)^q) + (X^{\bar{k}} + p^{\bar{k}}_{\hat{l}})^q, \tag{10}$$

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

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

$$((X^{\bar{k}} + p^{\bar{k}}_{\hat{l}})^q - (X^{\bar{k}})^q) > ((X^{\underline{k}} + ||J_{l'}||_1)^q - (X^{\underline{k}})^q). \tag{11}$$

Note that, due to our assumption,

$$incrCost_{i+1}(\Pi) > ||J_{l'}||_1$$

that is,

$$||\sum_{1\le j\le i+1} J_{\Pi_i}||_\infty - ||\sum_{1\le j\le i} J_{\Pi_i}||_\infty = (X^{\bar{k}} + p^{\bar{k}}_{\hat{l}}) - X^{\underline{k}} > ||J_{l'}||_1$$

we get,

$$(X^{\bar{k}} + p^{\bar{k}}_{\hat{l}}) > (X^{\underline{k}} + ||J_{l'}||_1). \tag{12}$$

This and

$$X^{\bar{k}} < X^{\underline{k}},$$

together yield the following contradiction.

$$((X^{\bar{k}} + p_{\hat{l}}^{\bar{k}})^q - (X^{\bar{k}})^q) > ((X^{\underline{k}} + ||J_{l'}||_1)^q - (X^{\underline{k}})^q). \quad (13)$$

Indeed, the same performance guarantee holds even for the following "static"-variant of q-norm based greedy orderings. We call a permutation Π, a *q-norm (static)* greedy ordering if Π_{i+1} is chosen as $l \in \{1, 2, ..., n\} - \{\Pi_1, \Pi_2, ..., \Pi_i\}$ that minimizes

$$f'_q(l, \Pi, i) = ||J_l||_q.$$

Note that 1-norm based *static* greedy ordering is same as the 1-norm based greedy ordering.

Theorem 1. *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.*

The worst case examples we could construct had a performace ratio of factor 1.5 for $m = 2$ and $\sqrt{m}$ for $m > 2$. Details of the example can be seen in (Patkar, 2004).

2.3 Combination-norm

Though the greedy strategies have a performance ratio between $\sqrt{m}$ and m, the examples on which the *sum-norm* and *max-norm* (dynamic or static) strategies perform badly are of different nature. In fact, we have not been able to come up with an example for which both the strategies give a schedule with cost significantly more than the optimal. The manuscript (Patkar, 2004) describes an example for which the *sum-norm* and *max-norm* strategies give identical but not the optimal (though very close to optimal) schedules.

A strategy based on the previous greedy strategies is to order the jobs according to a combination of their *sum-norm* and *max-norm*. Thus, jobs are scheduled in order of their combination-norms $\alpha||J_i||_1 + (1 - \alpha)||J_i||_\infty$, where $0 \leq \alpha \leq 1$ is a variable parameter. Details of the analysis of the strategy are described in (Patkar, 2004). Though we have not been able to prove a guarantee better than factor m on the performance ratio of the combination-norm strategy, experiments conducted by B. Jothi (Jothi, 2005) show that this strategy performs almost as good as an algorithm based on Potts' LP formulation and two genetic algorithms indicating that the combination norm strategy gives results very close to optimal in practice.

Analyzing the combination norm strategy (and its extensions that include the q-norms for arbitrary values of q) is an interesting problem. It is also interesting to investigate if the strategy can actually be used to compute optimal schedules for special cases of the problem. Though the Potts' formulation can be used to find a schedule with cost guaranteed to be less than twice the optimal, a greedy strategy is much faster and hence, more practical than any LP based approach.

3 Approximate solutions to the stochastic problem

In the stochastic version of the problem, 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. In particular, Mohring et al. extend Queyranne's inequalities for deterministic job scheduling to the stochastic case in (Mohring et al., 1999). They derive 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.

In the general version of stochastic vector job scheduling, the processing times of job components are independent random variables. There is no correlation between the processing time of various components of the same or different jobs. We have not been able to extend the results of the deterministic case to the *general* stochastic version of the *vector* job scheduling problem. The main difficulty is that unlike for single component job scheduling, the linear programs for vector 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.

3.1 LP-based Approximation in Stochastic Setting

In this section, 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 conside 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.

$$\begin{array}{c} \min \sum_{i=1}^{n}(pC_i^1 + qC_i^2) \\ C_i^1 \geq a_i^1 + \sum_{j \neq i} x_{ji} a_j^1 \ \ \forall\, i = 1, 2, \ldots, n \\ C_i^1 \geq b_i^1 + \sum_{j \neq i} x_{ji} b_j^1 \ \ \forall\, i = 1, 2, \ldots, n \\ C_i^2 \geq a_i^2 + \sum_{j \neq i} x_{ji} a_j^2 \ \ \forall\, i = 1, 2, \ldots, n \\ C_i^2 \geq b_i^2 + \sum_{j \neq i} x_{ji} b_j^2 \ \ \forall\, i = 1, 2, \ldots, n \\ x_{ij} + x_{ji} = 1 \ \forall\ i, j = 1, 2, \ldots, n \text{ and } i \neq j \\ x_{ij} + x_{jk} + x_{ki} = 2 \ \forall\ i, j, k = 1, 2, \ldots, n \text{ and } i \neq j \neq k \\ x_{ij} = 0 \text{ or } 1 \ \forall\ i \neq j \end{array} \tag{14}$$

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

$$\begin{array}{c} \min \sum_{i=1}^{n}(pC_i^1 + qC_i^2) \\ pC_i^1 + qC_i^2 \geq \overline{a_i} + \sum_{j \neq i} x_{ji} \overline{a_j} \ \ \forall\, i = 1, 2, \ldots, n \\ pC_i^1 + qC_i^2 \geq \overline{b_i} + \sum_{j \neq i} x_{ji} \overline{b_j} \ \ \forall\, i = 1, 2, \ldots, n \\ x_{ij} + x_{ji} = 1 \ \forall\ i, j = 1, 2, \ldots, n \text{ and } i \neq j \\ x_{ij} + x_{jk} + x_{ki} = 2 \ \forall\ i, j, k = 1, 2, \ldots, n \text{ and } i \neq j \neq k \\ x_{ij} = 0 \text{ or } 1 \ \forall\ i \neq j \end{array} \tag{15}$$

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.

$$\begin{array}{c} \min \sum_{i=1}^{n} \overline{C_i} \\ \overline{C_i} \geq \overline{a_i} + \sum_{j \neq i} x_{ji} \overline{a_j} \ \ \forall\, i = 1, 2, \ldots, n \\ \overline{C_i} \geq \overline{b_i} + \sum_{j \neq i} x_{ji} \overline{b_j} \ \ \forall\, i = 1, 2, \ldots, n \\ x_{ij} + x_{ji} = 1 \ \forall\ i, j = 1, 2, \ldots, n \text{ and } i \neq j \\ \\ x_{ij} = 0 \text{ or } 1 \ \forall\ i \neq j \end{array} \tag{16}$$

Now by solving the LP relaxation of the above and using it to sequence the jobs, we would obtain a 4-factor approximation algorithm for our stochastic scheduling problem as follows.

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 16. Without loss of generality, we assume that the jobs are numbered such that

$$\overline{C_1}^* \leq \overline{C_2}^* \ldots \leq \overline{C_n}^*. \tag{17}$$

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 16 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

$$\sum_{i=1}^{n} E[\max\left\{\sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j\right\}]. \tag{18}$$

Note that, for any two random variables, say X and Y, the following holds

$$E[\max\{X, Y\}] \leq E[X + Y] = E[X] + E[Y] \leq 2 \times max(E[X], E[Y]). \tag{19}$$

Therefore,

$$\sum_{i=1}^{n} E[\max\left\{\sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j\right\}] \leq \sum_{i=1}^{n} 2 \times \max\left\{\sum_{j=1}^{i} \overline{a_j}, \sum_{j=1}^{i} \overline{b_j}\right\}. \tag{20}$$

Next, we will show that

$$\sum_{i=1}^{n} \max\left\{\sum_{j=1}^{i} \overline{a_j}, \sum_{j=1}^{i} \overline{b_j}\right\} \leq 2 \times \text{opt(LP relaxation of problem 16)}. \tag{21}$$

That is,

$$\sum_{i=1}^{n} \max\left\{\sum_{j=1}^{i} \overline{a_j}, \sum_{j=1}^{i} \overline{b_j}\right\} \leq 2 \times \sum_{i=1}^{n} \overline{C_i}^*. \tag{22}$$

Therefore, it suffices to prove that

$$\max\left\{\sum_{j=1}^{i} \overline{a_j}, \sum_{j=1}^{i} \overline{b_j}\right\} \leq 2 \times \overline{C_i}^*. \tag{23}$$

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

$$\begin{aligned}
(\sum_{j=1}^{i} \overline{a_j})] \times \overline{C_i}^* &\geq \sum_{j=1}^{i} (\overline{a_j} \times \overline{C_j}^*) \\
&\geq \sum_{j=1}^{i} \overline{a_j}^2 + \sum_{l \neq j, l, j=1,2,\ldots,n} ((x^*_{lj} + x^*_{jl})\overline{a_l a_j}) \\
&\geq \frac{1}{2} \sum_{j=1}^{i} \overline{a_j}^2 + \frac{1}{2} (\sum_{j=1}^{i} \overline{a_j})^2 \\
&\geq \frac{1}{2} (\sum_{j=1}^{i} \overline{a_j})^2.
\end{aligned}$$

Therefore

$$\max\left\{\sum_{j=1}^{i} \overline{a_j}, \sum_{j=1}^{i} \overline{b_j}\right\} \leq 2 \times \overline{C_i}^*. \tag{24}$$

Noting that the LP relaxation of the problem 16 is a relaxation of the problem 14, we get

$$\sum_{i=1}^{n} E[\max\left\{\sum_{j=1}^{i} a_j, \sum_{j=1}^{i} b_j\right\}] \leq 4 \times \text{opt(problem 14)}. \tag{25}$$

Note that the above factor 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 performace 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.

3.2 Greedy heuristic based analysis

Now we consider the problem of computing an optimum schedule among the aligned schedules. Let's consider the 2-machine problem only.

Let the job J_l be described by the 2-dimensional vector of its components, ie. (a_l, b_l).

Let $||J_l||_q$ denote the q-norm of the vector J_l. Let $||J_l||_\infty$ denote the max-norm of the vector J_l.

Our task is to find a permutation Π of {1,2,...,n} such that

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

is minimum.

Let C_{opt} denote the cost of such an optimum schedule, say Π^{opt} (i.e., $C_{opt} =$ exp_cost(Π^{opt}))

Let Π^* denote the permutation of jobs that orders jobs in the increasing order of the expected value of their 1-norms. Clearly,

$$C_{opt} = \text{exp\_cost}(\Pi^{opt}) \geq \sum_{i=1}^{n} \frac{1}{2}(\sum_{j=1}^{i}(\overline{a_{\Pi_j^{opt}}} + \overline{b_{\Pi_j^{opt}}})) \geq \sum_{i=1}^{n} \frac{1}{2}(\sum_{j=1}^{i}(\overline{a_{\Pi_j^*}} + \overline{b_{\Pi_j^*}})). \tag{26}$$

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

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

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).

We shall prove that

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

that is,

$$cost(\Pi^{'}) \leq 2 \times cost(\Pi^{opt}).$$

We will now show that

$$cost(\Pi^{'}) \leq \sum_{i=1}^{n} \sum_{j=1}^{i} (\overline{a_{\Pi_j^*}} + \overline{b_{\Pi_j^*}})$$

It suffices to show that

$$E[\max \left\{ \sum_{j=1}^{i} a_{\Pi_j'}, \sum_{j=1}^{i} b_{\Pi_j'} \right\}] \leq \sum_{j=1}^{i} (\overline{a_{\Pi_j^*}} + \overline{b_{\Pi_j^*}}).$$

We shall prove this by induction on i. For i=1, it can be easily verified.

Now suppose

$$E[\max \left\{ \sum_{j=1}^{i} a_{\Pi_j'}, \sum_{j=1}^{i} b_{\Pi_j'} \right\}] \leq \sum_{j=1}^{i} (\overline{a_{\Pi_j^*}} + \overline{b_{\Pi_j^*}})$$

holds for i=1,2,...,p.

It remains to show that

$$E[\max \left\{ \sum_{j=1}^{p+1} a_{\Pi_j'}, \sum_{j=1}^{p+1} b_{\Pi_j'} \right\}] \leq \sum_{j=1}^{p} (\overline{a_{\Pi_j^*}} + \overline{b_{\Pi_j^*}}).$$

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

Thus we have established

$$cost(\Pi^{'}) \leq \sum_{i=1}^{n}\sum_{j=1}^{i}(\overline{a_{\Pi^*_j}} + \overline{b_{\Pi^*_j}}).$$

This along with the inequality 26 gives,

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

4 Summary

The approaches for the vector job scheduling problem can be classified into two types: greedy and LP based. Potts' formulation is one such LP based approach, which is an extension of a previous approach for single component job scheduling. Potts' formulation is a practical approach with a performance guarantee of factor 2. Morever, 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 are much faster any LP based approach, have a performance guarantee of factor m, the number of machines. However, the bound may not be tight and the examples on which the greedy strategies perform badly are sensitive to the input data. The combination-norm strategy gives solutions as good as the approach based on Potts' LP formulation or genetic algorithms in practice. Further work on analyzing the performance of the greedy strategies, especially the combination-norm strategy is of theoretical as well as practical interest.

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 . Analyzing the applicability of the LP based and other strategies to other special cases of stochastic vector job scheduling is a possible area of future work in this area.

Bibliography

Zhi-Long Chen and Nicholas G. Hall. Supply chain scheduling : Conflict and cooperation in assembly systems. 2005.

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

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

B. Jothi. Unpublished manuscript, Department of Mathematics, Indian Institute of Technology Bombay, May 2005.

Rolf H. Mohring, Andreas S. Schulz, and Marc Uetz. Approximation in stochastic scheduling: the power of lp-based priority policies. *J. ACM*, 46(6):924–942, 1999. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/331524.331530.

Sachin Patkar. A note on multi-dimensional job sequencing. Unpublished manuscript, Department of Mathematics, Indian Institute of Technology Bombay, June 2004.

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

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

Sivaramakrishnan Sivasubramanian. A multiprocessor scheduling problem. Unpublished manuscript, STCS, Tata Institute of Fundamental Research, Mumbai, India, November 2003.