Heat Treatment for Bearings: Scheduling Strategies and Distributed Decision Support System[1]

B.R.S.M. JOTHI and SACHIN B. PATKAR
Department of Mathematics,
Indian Institute of Technology, Bombay,
Mumbai - 400 076,
INDIA.

Abstract: - Scheduling and Production Planning problems are typically large in scale and fairly complex. A decision support system for such an operations research (OR) problem makes a good case study to be implemented using the web enabled, enterprise scale facilities of Java 2 Enterprise Edition (J2EE) and Simple Object Access Protocol (SOAP). We propose mathematical programming models as well as design and analyze certain specialized algorithms for special versions of these problems. In general, we use a powerful grade Mixed Integer Programming (MIP) solver, which we have web-enabled in our distributed decision support system, to solve the MIP models created by our system. Reusing web infrastructure can drastically lower the cost of setting up these application and allows us to reuse all kinds of tools originally built for the web.

Keywords: - Scheduling, Apache SOAP, EJB, Web Services, JMatLink, Matlab and FortMP solver.

1 Introduction

Scheduling and Production Planning problems are typically large in scale and fairly complex. Scheduling problems, concern the allocation of limited resources over time to perform some tasks to satisfy certain conditions. Scheduling problems exist almost everywhere in real-world situations, especially in the manufacturing industries. A decision support system for such a operations research (OR) problem makes a good case study to be implemented using the web enabled, enterprise scale facilities of Java 2 Enterprise Edition (J2EE) and Simple Object Access Protocol (SOAP) [1, 7].

A bearings manufacturer produce a large number of bearings of different kinds. It operates under many constraints, some of which involves the utilization of resources, productivity, heat treatment for bearings, downtime of machines and the target price - the price of the product specified by the customer. After the heat treatment process the bearings must be assembled. Hence, it requires to find out the optimum number of different bearing types for the heat treatment schedule. We have modeled two Mixed Integer Programming formulation's for the Scheduling in Bearing Heat Treatment Plant.

In this paper, we investigate a few variations of scheduling problems arising on the shop floor of a heat treatment plant of a bearings manufacturing industry. These problems arose in our collaborative efforts with the industry NRB Bearings Limited, Mumbai. We propose mathematical programming (in particular, mixed integer programming (MIP)) models as well as design and analyze certain specialized algorithms for special versions of these problems. However, in general, we use the powerful, industry-grade MIP solver such as FortMP[4], which we have web-enabled in our distributed decision support system, as the solver in the back-end to solve the MIP models created by our system. (FortMP is an industrial strength large scale optimization solver system developed by CARISMA, Brunel University as a research tool which is also used for teaching with many industrial applications [4]).

[1]Research supported by grant from project WebOPT (Asia IT & C contract ASI/B7-301/97/0126-73)

We need to choose a software platform that has a good support for distributed computing, concurrency, transaction, security, component based design, component based deployment, rapid development of interactive applications etc. Therefore, we decided to develop a prototype of this application using Enterprise Java Beans (EJB) on Java 2 Enterprise Edition (J2EE) platform and SOAP based Web Services.

Our Prototypical implementation is approximately 5,000 lines of J2EE code and 1000 lines of code for the deployment of Web Services. This was implemented and deployed on the free, open source JBoss server running on x86-linux platform [3]. Also, we use Apache SOAP implementation for the SOAP specifications [1]. FortMP solver is available as add-on to Matlab [4]. Tomcat Server is used as the Web Services container.

The roadmap of this paper is as follows. In section 2 we model the scheduling problems in our context and provide algorithmic strategies for the same. Later in section 3, a brief overview of our software system and the simulation results are provided. We end the paper with brief conclusions and a bibliography.

2 Problem Models and Solutions

In this section, we have modeled two Mixed Integer Programming Formulations (MIP) for the Scheduling Problem at Heat Treatment Plant.

2.1 The General Model

2.1.1 The Problem Specification

Let M_1,M_2,...,M_k,... be the machines available at the heat treatment plant and Md_1, Md_2,...,Md_l,... be the different modes of treatments which bearing types require. Examples of basic bearing types are subtypes of Cage bearings, Needle bearings, Ring bearings and Shaft bearings. Composite bearings have these as their constituents. Note that it may not be possible for all modes to be run on all machines. Let T_k denote the time available on machine M_k for processing. The different types of basic bearings that need to be processed are denoted by B_i.

In this general setup we assume that a bearing type may be possibly treated in several different modes on several different machines. These basic bearings are to be assembled to produce composite bearings. We denote the orders by O_j. An order from a client is a collection of (bearing type, quantity) pairs (called "suborder"). For example an order from a client could be **Order 1 : 7 units of Cage A, 20 units of Shell B, 10 units of Shell C.**

The pair *(machine, {list of modes for given machine})* uniquely determines the machine and mode specifications i.e, the specified machine will work under these modes. The pair *(bearing type, {list of modes for given bearing type})* uniquely determines the bearing type and mode specifications i.e, the specified bearing type will work under these modes.

2.1.2 Mixed Integer Programming Formulation of the Problem

In this formulation, the suffixes i, j, k and l will consistently be associated with the bearing type B_i, order O_j, machine M_k and mode Md_l respectively. Let C_{ij} be the quantity of bearing type B_i required in order O_j.

Let b_{ikl} denote the duration of a session of bearing type B_i in mode Md_l on machine M_k. That is, for every bearing type B_i, every machine M_k and mode Md_l, there must be a fixed duration of processing time b_{ikl} per session. Let a_{ikl} denote the quantity of B_i that can be processed in mode Md_l on machine M_k in a session. Let x_{ikl} denote the number of sessions required for processing bearings of type B_i in mode Md_l on machine M_k. Also, let y be the makespan. ie., the completion time of all the jobs in the system. The constraints are as follows:

Time Constraint: For each M_k, the amount of time taken by the allocation in a machine should not exceed T_k. That is, $\sum_{i,l} b_{ikl}.x_{ikl} \leq T_k, \ \ \forall k$.

Makespan Constraint: Makespan must be atleast as large as the processing time assigned to any machine: $y - \sum_{i,l} b_{ikl}.x_{ikl} \geq 0, \ \ \forall k$.

Demand Constraint: The problem requires an allocation which will saturate the demand for resources. The total requirement for B_i is $\sum_j C_{ij}$. Hence, we have the constraint, $\sum_{k,l} a_{ikl}.x_{ikl} \geq \sum_j C_{ij}, \ \ \forall i$.

Since the heat treatments are based on chemicals it may be possible that the treatment turns ineffective after a certain period of being in use. Hence we have an upper bound for the number of sessions. Likewise, for effective utilization of resources a lower bound on this duration may be desirable.

Also $x_{ikl} \leq u$, u is an upper bound
$x_{ikl} \geq 0$ and x_{ikl} are integers.

Objective Function : let p_{ikl} denote the processing cost per session for bearing type B_i on machine M_k in mode Md_l. The objective function will be allocation of jobs on machines-mode combinations that cost lesser as well as minimize the makespan.

$$min \;\; y + \sum_{i,k,l} p_{ikl}.x_{ikl}$$

2.2 A Specialization of the General Model

The following model is a specialization of the general model discussed above.

2.2.1 Problem Specification

$M_1, M_2, ..., M_{k_{max}}$ are **identical** machines. Each machine is capable of operating in any mode. Each bearing type can be processed by only a **unique** mode (however, on any machine). Each session in any mode on any machine has the same duration, which we will conveniently assume to be **one unit** long. A charge of B_i is the quantity of that bearing type that can be processed in a session. C_i is the number of charges of the type B_i to be processed. s and $2s$, respectively, are the lower and upper bounds of the number of sessions in a shift. T_k is the number of sessions available on machine M_k.

We shall assume that the scheduling/resource allocation problem is of large enough scale. In particular, we shall assume the following. $T_k \geq 6s \;\; \forall k$, $C_i \geq 6s \;\; \forall i$, $\sum T_k \geq \sum C_i$ and that the number of machines is less than or equal to the number of different basic bearing types. These are reasonable assumption since we are not worried about the time complexity of small scale problems. The problem that we wish to solve is to check for feasibility of assigning shifts of acceptable lengths (within the specified bounds) such that all bearings are processed in the specified units of time on all the machines. We also wish to output such a feasible schedule.

The following algorithm provides a feasible solution for the problem. Though much simpler that the general problem, it is not really intuitive in nature. Thus, even though scheduling problems may be simplified, an optimal solution is not always evident.

2.2.2 Algorithm

Initialize { $avail(i) = C_i$ for all bearing types B_i;
  $tavail(k) = T_k$ for all machines M_k; $i = 1$;
}
for $k = 1, ..., k_{max}$ {
  if ($avail(i) \geq 6s$) {
    allocate $2s$ charges of B_i to M_k;
    $tavail(k) = tavail(k) - 2s$;
    $avail(i) = avail(i) - 2s$;
    mark B_i as in the "cap" of schedule
      of machine M_k.
  }
  else if ($4s \leq avail(i) < 6s$) {
    allocate $2s$ charges of B_i to M_k;
    allocate $2s$ charges of B_{i+1} to M_k;
    mark B_i and B_{i+1} as in the "cap" of schedule
      of machine M_k.
    $tavail(k) = tavail(k) - 4s$;
    $avail(i) = avail(i) - 2s$;
    $avail(i+1) = avail(i+1) - 2s$; i++;
  }
}
$k = 1$;
for $i = 1, 2, ..., i_{max}$ {
  let kl be the largest k' (could be $k-1$) such that
    $\sum_{k'=k}^{kl} tavail(k') \leq avail(i)$
  allocate $tavail(k')$ charges of B_i to $M_{k'}$ for
    $k' = k, .., kl$
  $avail(i) = avail(i) - \sum_{k'=k}^{kl}(tavail(k'))$;
  k = kl+1; if ($avail(i) == 0$) break;
  if ($avail(i) < s$) {
    allocate $avail(i)$ charges of B_i to M_k;
    swap $s - avail(i)$ charges of B_i in M_{k-1} and
      $s - avail(i)$ charges of some B_j that is in
      the "cap" of both M_k and M_{k-1} from M_k.
    $tavail(k) = tavail(k) - avail(i)$;
  } else {
    allocate $avail(i)$ charges of B_i to M_k;
    $tavail(k) = tavail(k) - avail(i)$;
  }
  if ($tavail(k) < s$) {
    move $tavail(k)$ charges of some B_j that is in
      the "cap" of both M_k and M_{k+1} from M_{k+1}
      to M_k.
    $tavail(k+1) = tavail(k+1) + tavail(k)$;
    k++;
  }

}

Proof that the schedule is feasible: Note that $T_k \geq 6s \ \forall k$ and $C_i \geq 6s \ \forall i$ and the number of machines less or equal to the number of basic bearing types. The "cap formation" phase clearly ensures that for every adjacent pair of machines, there exist at least $2s$ sessions which are scheduled to process bearings of the same type. After the "cap formation" phase, there exists at least $2s$ units of each bearing type.

The "non-cap" phase of the algorithm allocates the remaining units of bearing types in such a manner that for each bearing type B_i, the remaining units of bearings of type B_i are scheduled on contiguous sequence of machines such that with the possible exception of only the first and the last machines in this sequence, the non-cap portion of the machines is completely filled with bearings of type B_i.

Let the non-cap portion of B_i (ie. the units of B_i not scheduled in the cap of any of the machines) be scheduled on the contiguous sequence of machines M_k to $M_{k'}$. As the non-cap portion of each machine is at least $6s - 4s$ ie. $2s$, the machines intermediate between M_k to $M_{k'}$ satisfy the shift length requirement.

Thus for each bearing type B_i, only the first and the last machines in the sequence, ie. possibly only M_k and $M_{k'}$ can have an infeasible schedule for the non-cap portion of B_i in them. That is the number of units of B_i in the non-cap portion is less than s.

We show that the algorithm ensures that this does not happen. First we handle the case of $M_{k'}$ facing the allocation of less than s units of B_i, ie. the case $avail(i) < s$. Note that in this case, the previous machine in this sequence (that is, machine $M_{k'-1}$) has at least $2s - avail(i)$ units of B_i, ie. at least $s - avail(i)$ units of B_i more than what is required for feasibility. Furthermore, the cap of the machine $M_{k'}$ has $2s$ units of some bearing type $B_{i'}$ that is also in the cap of $M_{k'-1}$ thus swapping $s - avail(i)$ units of B_i from the non-cap portion of $M_{k'-1}$ with $s - avail(i)$ units of $B_{i'}$ from the cap of $M_{k'}$ would satisfy the shift length constraints.

Next, it remains to handle the case of M_k (that is the first machine in the above mentioned sequence) facing the allocation of B_i to less than s sessions remaining in the non-cap portion. However, it can be seen that the algorithm does not permit this to happen, as whenever the number of sessions remaining in the non-cap portion of machine M_k is less than s, these sessions are filled up by moving the required number of bearings of some type $B_{i''}$ that is present in the cap of both M_k and M_{k+1}, from the cap of M_{k+1} to these sessions on M_k. This lengthens the number of sessions of $B_{i''}$ on M_k without reducing the number of sessions of $B_{i''}$ on $M_{k''}$ below s. Fortunately, the modified cap of machine M_{k+1} does not present a problem as the dwindled cap portion of machine M_{k+1} would not need to donate any further portion of it, for the reason that M_{k+1} would be used for scheduling the non-cap portion of the next bearing type and thus would have plenty of these to allocate sessions to. **q.e.d**

2.3 Vector Job Scheduling

Here we consider another specialized scheduling problem that considers "jobs" that have "components" that can only be processed on machines dedicated to unique different components. For example, such a job represents a certain quantity of composite bearings whose components are proportionate amount of quantities of its basic constituent bearing types. For the sake of simplicity of the presentation of the ideas, we assume that all such jobs have exactly two components, processed by machine 1 and machine 2 respectively. Machine 1 processes the first components and the second machine processes the second components. The specified quantity of the composite bearing is available only after its components are processed on the respective machines. Therefore, the completion time of a job may be defined naturally as the maximum of the completion times of the components. Therefore, a natural optimization criterion is to schedule (sequence) the jobs so as to minimize the sum of the completion times of the individual jobs (this would minimize average completion times of the jobs). We solve this problem using *Potts' formulation* and alternately using greedy approaches based on *sum-norm* and *max-norm* (for further details see [8]).

2.3.1 Definitions

We are given two machines M^1 and M^2. Also given is a collection of jobs $J_i = [a_i, b_i], \quad i = 1, 2, ..., n$, where the first component a_i has to be processed on M^1 and the second component b_i has to be processed

on M^2. A schedule is given by the permutation of $[1, 2, ..., n]$, say, Π. Completion time of J_i on M^1 is defined as $\sum_{\Pi[j]=1,...,i} a_{\Pi[j]}$. Similarly, the completion time of J_i is defined on M^2 in terms of b's and Π. The completion time of J_i on M^1 and M^2 is defined as the maximum of its completion times on M^1 and M^2. The total (cumulative) completion time of the schedule is the sum of the completion times of all the jobs.

Objective: The objective is to find a permutation Π of {1,2,...,n} such that $cost(\Pi)$, which is, $\sum_{i=1}^{n} \max\{\sum_{j=1}^{i} a_{\Pi_j}, \sum_{j=1}^{i} b_{\Pi_j}\}$ is minimum.

2.3.2 Potts' formulation [6]

One can extend, in a straightforward manner, a formulation due to Potts (see [6, 8] for a known application of Potts' formulation) for a single machine scheduling problem, to get the following. Given a collection of jobs $J_i = [a_i, b_i]$, i=1,2,$\cdots$,n. Let C_i be the completion time of job i and *n* be the number of jobs. Also let $x_{ij} = 1$ if job i precedes job j and 0 otherwise. Thus an MIP model of our problem based on Potts' idea is given by $min \quad \sum_{i=1}^{n} C_i$ subject to the constraints,

$$x_{ij} + x_{ji} = 1, \qquad 1 \leq i < j \leq n$$

$$x_{ij} + x_{jk} + x_{ki} \leq 2, \qquad 1 \leq i < j < k \leq n$$

$$C_j \geq a_j + \sum_{1 \leq i \leq n,\ i \neq j} x_{ij} a_i, \qquad j = 1, 2, ..., n.$$

$$C_j \geq b_j + \sum_{1 \leq i \leq n,\ i \neq j} x_{ij} b_i, \qquad j = 1, 2, ..., n.$$

Here the variables x_{ij} (i $\neq$j, 1$\leq$ i, j $\leq$n) are binary variables and $C_i \geq 0$ (i=1,$\cdots$,n).
In this Potts' formulation, for n jobs, we use n^2 variables and $nC_2 + 2 * nC_3 + 2 * n$ constraints.

2.3.3 Greedy Strategies: sum-norm and max-norm

Let Π denote the permutation of jobs that orders jobs in the increasing order of their *sum-norms*. That is, $a_{\Pi_1} + b_{\Pi_1} \leq a_{\Pi_2} + b_{\Pi_2} \leq \cdots \leq a_{\Pi_n} + b_{\Pi_n}$. Then we call Π as the **sum-norm** based greedy ordering. Similarly, if Π' is a permutation of $\{1, 2, \cdots, n\}$ $\max\{a_{\Pi_1}, b_{\Pi_1}\} \leq \max\{a_{\Pi_2}, b_{\Pi_2}\} \leq \cdots \leq \max\{a_{\Pi_n}, b_{\Pi_n}\}$, then Π' is called **max-norm** based greedy ordering.

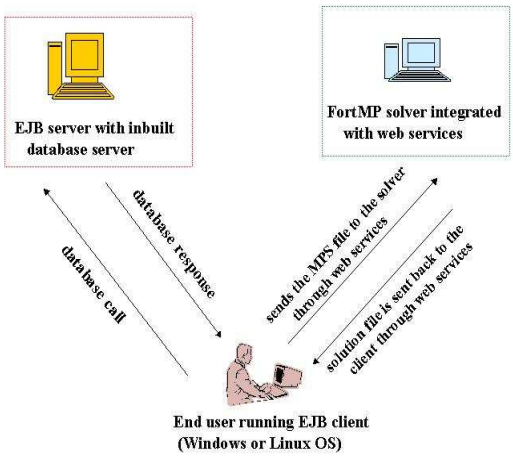

Figure 1: Our Decision Support System

Theorem 2.1 *For the problem of scheduling 2-dimensional jobs to minimize total completion time, the sum-norm based greedy ordering strategy finds a schedule whose cost is within 2 times the optimal.*
(Proof:) Let Π^* denote the permutation that orders jobs in the increasing order of their sum-norms. Let Π^{opt} denote an optimal ordering. We use the following notation. $sum_l(\Pi)$ denotes $\sum_{i=1}^{l}(\sum_{j=1}^{i}(a_{\Pi_j} + b_{\Pi_j}))$ and $cost_l(\Pi)$ denotes $\sum_{i=1}^{l}(\max\{(\sum_{j=1}^{i} a_{\Pi_j}, \sum_{j=1}^{i} b_{\Pi_j})\})$.
Informally, let $cost_l(\Pi)$ denote the cumulative completion time elapsed upto the l^{th} job.
Clearly, $cost_n(\Pi) = cost(\Pi)$.
We also have, $cost_l(\Pi^*) <= sum_l(\Pi^*)$ and $cost_l(\Pi_{opt}) >= sum_l(\Pi_{opt})/2 >= sum_l(\Pi^*)/2$. So, $cost_l(\Pi^*) <= 2 * cost_l(\Pi_{opt})$. For $i = n$, we get $cost(\Pi^*) <= 2 * cost(\Pi_{opt})$. **q.e.d**

Similarly one can also establish the same performance guarantee for max-norm based greedy ordering strategy.

3 Architecture of Our Distributed System

Our present decision support system has three tier architecture where the EJB server-clients run on two different desktops, but a third desktop hosts web services integrated with the FortMP solver.

Our EJB client has the following features: multiple end-users supported, security and concurrency,

transaction management, distributed framework, intuitive GUI, plugging of deployable components supported.

Due to the feature of our EJB based system supporting pluggable component, we managed to integrate the MIP model component which generates the MPS file describing the MIP model for the given set of jobs/orders and calls the web services further to solve the MIP problem with the help of FortMP.

The jobs/orders are submitted through EJB client. The jobs/orders information will be stored in the Hypersonic database that is embedded in the JBoss server. Later the MPS file is created using the problem data obtained from the above database. Using the SOAP attachment feature of web services the MPS file is sent over eventually to the FortMP solver whose solution is also received back as a file using SOAP attachment feature.

In the server side, web services which perform the task of transferring files and invoking Matlab session using JMatLink [5]. JMatLink provides programmatic connectivity between Java and Matlab. The MIP problem gets solved by FortMP solver. Server side also includes a matlab script which is called through JMatLink with necessary parameters and it generates the solution and log file as the output.

3.1 Results Comparison for Vector Job Scheduling

Next, we compare the results of *Potts' formulation* with *sum-norm* and *max-norm* based greedy strategies for the Vector Job Scheduling problem. We have experimented with five problems of different size. Table 1 shows the comparison of total completion times of *Potts' formulation* and greedy strategies based on *sum-norm* and *max-norm*. Here, n denotes the number of jobs.

Table 1
Comparison of total completion times

n	Potts' formulation	sum-norm	max-norm
100	26458	27324	27279
200	109037	115350	111296
300	226090	229111	234529
400	424683	441359	434975
500	619845	624419	641497

From the above table, it is observed that *Potts' formulation* performed well compared to *sum-norm* and *max-norm* based greedy strategies.

4 Conclusion

The problem of scheduling ball bearings at a Heat Treatment plant is large scale and complex. Several mathematical formulations were formulated and solved using the powerful solver through Web Services. Special efficient algorithms were also devised and analyzed. Our EJB/SOAP based decision support system allows efficient security, concurrency, transaction management and persistence at a little extra programming effort. Further developments or generalizations, as and when required, should also be easy since EJB is scalable and maintainable. The business logics of various scheduling heuristics can be used as "plug-ins".

References

[1] Apache Foundation, "*Apache SOAP v.2.3.1*", http://ws.apache.org/soap, 2002.

[2] S. French, "*Sequencing and Scheduling, An Introduction to the Mathematics of the Job-Shop*", Ellis Horwood Limited, 1990.

[3] "*JBoss Server*", http://www.jboss.org, 2002.

[4] "FortMP Solver", *CARISMA*, http://www.optirisk-systems.com, 2004.

[5] S. Muller, "*JMatLink version 1.00*", http://www.held-mueller.de/JMatlink, 2001.

[6] 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 ACM-SIAM Symp. on Discrete Algorithms*, pages 142-151, 1996.

[7] E. Roman, S.W. Ambler and T. Jewell, "*Mastering Enterprise JavaBeans*", John Willey & Sons, 2002.

[8] S. Sivaramakrishnan, S.B. Patkar and B.R.S.M. Jothi, "Results on Multidimensional Job Sequencing", unpublished manuscript.