Quasi Assignment Algorithms in Job Shops

Eui-Seok Byeon, Jang-Yong Lee

Department of Industrial Engineering, SunMoon University
#100, Kalsanri, Tangeongmyeon, Asansi, Chungnam 336-840, Korea

Abstract

Production scheduling has been one of the most critical issues in a manufacturing environment. Job-shop scheduling problems (JSP) are well known from the standpoint of production planning and operations control. In this research scheduling against due date is a measure of performance and the objective is minimizing total weighted tardiness. This paper presents an idea of decomposition of the problem and shows robustness of the schedule under various disturbances along with exact and approximation methods. The proposed method can indeed handle shop disturbances more effectively when compared with traditional and dynamic scheduling methods.

1. Introduction

Job shop scheduling often has to face various disturbances in real world applications. Disturbances are unplanned events such as processing time deviations, machine breakdowns, etc. These events undoubtedly cause scheduling performance to degrade or rendered useless. Clearly the less a schedule is sensitive to disturbances, the more robust it is. In this paper, improving schedule robustness is done by increasing its flexibility and reducing its sensitivity to disturbances. The ability of a schedule to handle disturbances depends on flexibility that exists in the schedule and the information available about the disturbances. Leon et al. [5] defined a robust schedule as one that is insensitive to unforeseen shop floor disturbances. Moreover, it maintains good performance in the presence of these disruptions.

Other research in job-shop scheduling problem has focused on the development of dispatching rules which are not sensitive to varying shop conditions or performance measures [4,6,7]. Dispatching in a job-shop is a decision of selecting from a machine queue a job to be processed next. As a result, a scheduling heuristic determines how the jobs at each machine should be sequenced. In earlier researches [3,8], the method which decomposes scheduling decisions into two different stages has been proposed: a first stage makes "partial" scheduling decision by partitioning the operations into subsets. The second stage completes the scheduling decision over time in a dynamic fashion. The method provides an additional flexibility in scheduling and thereby
increases its robustness. In this paper, we show a
detailed simulation study of the above decomposition
scheme and demonstrate the robustness of the method
based on its performance under a wide range of
processing time variations.

2. Problem Statement

We have developed a decomposition
scheme where the operation of a Jobshop Scheduling
Problem (JSP) is assigned to a number of subsets(p)
each with a size of αk. Each subset does not
necessarily have the same size. Associated with a
JSP is a disjunctive graph G(N, A, E) where A is a set
of conjunctive arcs representing the precedence
constrains among operations, and is a set of
disjunctive arcs representing the possible processing
orders on machine m, E = UαkEm.

If we try to assign each of |N| operations into a
predetermined number of subsets p, the
corresponding formulation is for variation of
assignment problem (VAP) as follows:

(VAP): Minimize \( \sum \alpha_k \sum X_{ik} \) \( C_{ik} \)

s.t. \( \sum kX_{ik} = 1 \) \( \quad i \in N \) \( \quad (2) \)
\( \alpha_k \) \( \quad k=1,...,p \) \( \quad (3) \)
\( \sum X_{ik} \leq \sum \alpha_k (i,j) \in A \) \( \quad (4) \)
\( X_{ik} \in \{0,1\} \) \( \quad i \in N \) \( \quad k=1,...,p \) \( \quad (5) \)

In this formulation, \( C_{ik} \) is called a cost incurred for
assigning operation i to subset k. \( X_{ik} \) is the decision
variable: \( X_{ik} = 1 \) if operation i is assigned to subset k,
\( X_{ik} = 0 \) otherwise. The above problem is a variation
of the classical assignment problem (VAP). Solving
VAP resolves a part of the disjunctive arcs in the set
E. Specially, for a disjunctive arc (i,j) with i,j not
assigned to a same subset will have its direction
resolved. Suppose \( E_{mn} \) represents the remaining,
unresolved set of disjunctive arcs in subset k, on
machine m, then \( E_{mn} = \{ (i,j) \in E_m \mid X_{ik}=X_{jk}=1 \} \).
After solving VAP on the original graph, G(N,A,E),
we have a new disjunctive graph, \( \overline{G} (N, \overline{A}, \overline{E}) \),
where \( \overline{A} \) consists of the newly resolved disjunctive
arcs as well as the original conjunctive arcs. \( \overline{E} \) is
the union of \( E_{km} \) for all m and k. Therefore, the new
graph \( \overline{G} \) represents the original JSP with an
additional set of precedence constraints. This new
JSP makes global scheduling decision while the
remaining sequencing decisions are allowed great
flexibility through the use of dynamic dispatching.
In the rest of the paper, the robustness of “VAP
schedules” via intensive simulation is demonstrated.

3. Experiments

The objective of this study is to examine
how the proposed decomposition scheme performs
when operation processing times vary. The
proposed method is simulated and compared with a
static and a dynamic scheduling method. We
measure the mean weighted tardiness (wT) of the
schedules after 100 simulation runs, and use it as the
robustness measure. Disturbances are generated by
altering the processing time by an exponential
distribution. We have tested a wide range of
disturbances by varying the mean (\( \tau \)) of the
exponential from 5 to 60, incremented by 5. The
actual processing times (\( \text{Pi}^* \)) for each operation are
generated as follows: \( \text{Pi}^* = \text{Pi} + \text{Exp}(1/\tau) \), where
\( \tau = 5, 10, 15, \ldots, 60 \). Note that since \( \text{Pi}^* \) could be
negative, assume an unit processing time when \( \text{Pi}^* \leq 0 \).

3.1 VAP Heuristic
As a solution procedure for VAP, use a heuristic which assigns operations to subsets based on a priority index. Specifically, we use an “ATC-index” due to Vepsalainen and Morton [7]. Given a job due date \( dd_i \) and weight \( w_i \), the index at the current time \( t \) is computed as follows:

\[
\frac{W_j}{P_i} \exp \left( \frac{dd_j-t-P_i-\sum g \in \Psi_i, \psi_i(w_g+p_g)}{kp} \right)
\]

Where, operation \( i \) belongs to job \( J \) while having a set of successors \( \Psi_i \). The \( p \) is the average processing time, \( k \) is a look-ahead parameter, and set \( k=3 \). \( \Psi \) is the lead time estimation parameter as in Vepsalainen and Morton [7], and set to 2. Based on the above index, assign the first highest \( \alpha_1 \) operations to subset 1, and the second highest \( \alpha_2 \) operations to subset 2, and so on. Each heuristic assignment is evaluated based on the lower bound of its corresponding scheduling problem. The procedure is as follows:

**Iterative Searching Procedure**

Step 0. Set \( \text{Itm}=1 \), and denote \( NTI \) be the number of total iterations. Compute ATC index using original due date and weight. Goto Step 2.


Step 2. Assign the first highest \( \alpha_1 \) operations to subset 1, and the second highest \( \alpha_2 \) operations to subset 2, and so on. Compute Lower Bound of above graph.

Step 3. If \( \text{Itm} < NTI \), then goto Step 4. Otherwise, stop.

Step 4. Update due dates as follows:

\[
dd^{'i}=dd^{''}(j) + \text{stepsize} \times dd^{'}(j) \times \left( \frac{\text{SLACK}_j}{SS} \right)
\]

Goto Step 1. End.

The stepsize ranges from 0.5% to 3.0% through extensive empirical testing. The SS is total job slacks, where \( \text{SLACK}_j= dd^j - \text{completion time}(C_j) \).

The above heuristic generates half completed schedules while the detailed schedule is generated dynamically. We complete this dynamic schedule using the ATC dispatching rule. The ATC heuristic by Vepsalainen and Morton [7] is a dynamic dispatching rule with priority updated each time a job is scheduled. Their priority index for each unscheduled job is a function of weight, due date and processing time. In this paper, the robustness of the VAP heuristic in the presence of processing time variations is tested. Simulation results are compared with pre-generated static schedule and Morton’s dynamic schedule as follows.

**3.2 Static Schedule**

We generate a static schedule using the ATC heuristic at the beginning of the planning horizon, before disturbances occur.

**3.3 Dynamic Schedule**

Here, a schedule using ATC-rule is generated, but all the scheduling decisions are made dynamically. That is, the dynamic schedule updates its parameters over time based on varied processing times. We use three sets of test problems for the simulation study. All test problems are from Applegate and Cook[1]. The first set contains 10x10 problems. Knowing the optimal makespan schedule for the above problems, subtract a constant (i.e. 100) from each job’s completion time, and set them as job
Due date. The second and third sets are larger sized problems, i.e. 20x15 and 30x10. In these cases, the optimal makespan schedule is not available. Thus set due date as follows: first generate a non-delay schedule [2] and obtain the job completion times. Subtract 100 from each job completion time, and use them as job due dates.

4. Results

The VAP heuristic simulated is characterized by two primary factors; the number of subsets, and the size of each subset $\alpha_n$. We tested a various number of subsets including 2, 3, 4, 5 and 10. Then, given a specific number of subsets, we varied run. The number after “W” stands for the number of subset used. For example, in A the W5 column represents the simulation result of VAP with 5 subsets of identical sizes (i.e., 20 operations per subset). For subsets with non-identical sizes, we used additional numbers after “W”. For example, W225 represents 2 subsets where 25 operations are in the first subset and 75 operations in the second. Similarly, W212 in B and C represents 100 operations in the first subset and 200 in the second.

5. Conclusions

In summary, more robust schedule could be indeed generated when compared to the traditional

<table>
<thead>
<tr>
<th>Problem (10x10)</th>
<th>Static</th>
<th>VAP Heuristic ($\tau=15$)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>W2</td>
<td>W225</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>11770</td>
<td>10595</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4480</td>
<td>4904</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5598</td>
<td>6019</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8368</td>
<td>8592</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12175</td>
<td>8554</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem (30x10)</th>
<th>Static</th>
<th>VAP Heuristic ($\tau=30$)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>W2</td>
<td>W212</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>91459</td>
<td>43504</td>
</tr>
<tr>
<td></td>
<td></td>
<td>75855</td>
<td>33685</td>
</tr>
<tr>
<td></td>
<td></td>
<td>79567</td>
<td>48930</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem (30x10)</th>
<th>Static</th>
<th>VAP Heuristic ($\tau=10$)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>W2</td>
<td>W212</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>20501</td>
<td>14288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23165</td>
<td>21041</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23012</td>
<td>14822</td>
</tr>
</tbody>
</table>

<Table 1> Mean Weighted Tardiness after 100 simulation Runs

the subset sizes. For example, for 10x10 problems with two subsets, we consider the following splits: [50, 50], [25, 75] and [75, 25]. Table 1 shows the mean weighted tardiness (WT) from 100 simulations static and the dynamic methods. We have demonstrated through simulation that the VAP scheduling heuristic can manage a wide range of disturbances due to its embedded flexibility.
6. References


