Discrepancy and backjumping heuristics for flexible job shop scheduling
Abir Ben Hmida, Mohamed Haouari, Marie-José Huguet, Pierre Lopez

To cite this version:

HAL Id: hal-00279752
https://hal.archives-ouvertes.fr/hal-00279752
Submitted on 15 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Discrepancy and backjumping heuristics for flexible job shop scheduling

A. Ben Hmida¹², M. Haouari¹, M.-J. Huguet² and P. Lopez²
¹Unité ROI, Ecole Polytechnique de Tunisie, Tunisie
e-mail: mohamed.haouari@ept.rnu.tn
²LAAS-CNRS, Université de Toulouse, France
e-mails: {abenhmid, huguet, lopez}@laas.fr

Keywords: Scheduling, job shop, discrepancy, makespan.

1. Problem statement

The Flexible Job Shop Problem (FJSP) is a generalization of the traditional Job Shop scheduling Problem (JSP), in which it is desired to process a set of \( n \) jobs on a set of \( m \) machines in the shortest amount of time. Every job \( J_i \) (\( i = 1, \ldots, N \)) consists of \( s_i \) operations \( O_{i1}, O_{i2}, \ldots, O_{is_i} \) which must be processed in the given order. Every operation must be assigned to a unique machine \( r \), selected among a given subset, which must process the operation during \( p_{ir} \) units.

Solving the flexible job shop consists in assigning a specific machine to each operation of each job as well as sequencing all operations assigned to each machine, such that successive operations of a job do not overlap and such that each machine processes at most one operation at a time. Job preemption and job splitting are not allowed. The objective is to find a schedule that minimizes the maximum completion time or makespan. As a generalization of the job shop problem, the FJSP is known to be strongly NP-Hard (Garey et al., 1976). Brucker and Schlie (1990) propose a polynomial algorithm for solving the FJSP with two jobs, in which the processing times are identical whatever the machine chosen to perform an operation. Brandimarte (Brandimarte, 1993) was the first to use a decomposition approach for the FJSP. He solved the assignment problem using some dispatching rules and then focused on the resulting job shop subproblems, which are solved using a tabu search heuristic. Hurink et al. (1994) propose to solve this problem with multiple capacities machines. Authors propose two neighborhoods which are based on the concept of block. Chambers et al. (1996) proposed a Tabu Search method to solve the problem. Mastrolilli et al. (2000) proposed two structures of neighborhood based on the displacement of an operation in the disjunctive graph. Authors showed that if a feasible solution does not have a neighbor according to first neighborhood, then it is an optimal solution. The second neighborhood is an extension of the first one. It preserves the property of optimality in the event of absence of neighbor. Authors showed the connexity of the second neighborhood. According to their experiments, in spite of the absence of the connexity of the first type of neighborhood, this last gives better results than the second one because of the higher speed of execution. Kacem et al. (2002) used a genetic algorithm (GA) to solve the FJSP and they adapted two approaches to solve jointly the assignment and the sequencing subproblems. The first one is to approach by localization and the second one is an evolutionary approach controlled by the assignment model and applying GA to solve the FJSP. Xia and Wu (2005) proposed a hybrid of particle swarm optimization and simulated annealing as a local search algorithm.

In this abstract, we propose to improve a discrepancy-based method, called CDDS, after being adapted to solve the flexible job shop problem in a precedent work (Ben Hmida et al., 2007b). So, we propose applying discrepancy on some pertinent variables chosen by using two types of heuristics. The remainder of this abstract is organized as follows. Section 2 introduces the principles of CDDS. Section 3 presents its adaptation for the problem under study and then proposes a discrepancy strategy to limit the tree search. Section 4 presents CDDS performance via an example and a series of tests. Finally, section 5 gives some concluding remarks and directions for future work.
2. Climbing Depth-bounded Discrepancy Search

CDDS is a tree search method based on the discrepancy principle to expand the search for visiting the neighborhood of the initial solution. It combines the Climbing Discrepancy Search (CDS) method (Milano et al., 2002) and the Depth-bounded Discrepancy Search (DDS) method (Walsh, 1997). CDDS method has been developed initially to solve Hybrid Flow Shop problems (Ben Hmida et al., 2007a) and has proved its efficiency in this domain. Then, it has been adapted to solve the flexible job shop problem and has provided promising results, especially with instances of a higher degree of flexibility (Ben Hmida et al., 2007b). The CDDS method starts from an initial solution suggested by a given heuristic. Hence nodes with discrepancy equal to 1 are first explored then those having a discrepancy equal to 2, and so on. When a leaf with improved value of the objective function is found, the reference solution is updated, the number of discrepancy is reset to 0, and the process for exploring the neighborhood is restarted. To limit the tree search expansion, CDDS strategy applies discrepancies only at the top of the tree to correct early mistakes in the instantiation heuristic (for more details see Ben Hmida et al., 2007a). This method can be improved by using constraint propagation, e.g. the forward checking strategy (Haralick et al., 1980) which suppresses inconsistent values in the domain of not yet instantiated variables involved in a constraint with the assigned variable; one can also use a more refined mechanism. Although this method showed its efficiency for the resolution of the Hybrid Flow Shop problems (Ben Hmida et al., 2007a), it remains, nevertheless, difficult to adapt to the FJSP (Ben Hmida et al., 2007b). This is especially due to the considerable number of parameters to define: initial solution, search heuristics, discrepancy strategy, and tree search expansion. To improve our CDDS method for FJSP and more precisely for discrepancy strategy, we introduce some specific heuristics for applying discrepancies.

3. Adaptation of CDDS for Flexible Job Shop Problem

3.1. Instantiation Heuristics

It seems reasonable that the efficiency of the discrepancy-based methods depends closely on the quality of the initial solution (Harvey, 1995). In our approach, the initial solution is determined by the use of several heuristics: (1) Selection of operations: We first give the priority to the operation belonging to the job with the earliest start time (EST) and in case of equalities we consider the operation belonging to the job with the longest duration (LDJ). (2) Assignment of machines to operation: The operation previously chosen is assigned to the machine such that the task completes as soon as possible. This heuristic is called Earliest Completion Time (ECT). Heuristic is dynamic; the machine with the highest priority depends on the machines previously loaded. After both instantiations, we use a simple Forward Checking constraint propagation mechanism to update the finishing time of the selected operation as well as the starting time of the successor operation. We also maintain the availability date of the chosen resource.

3.2. Tree search expansion

To limit the tree search expansion, we propose to introduce a lower bounding strategy. In fact, a lower bounding strategy is useful to speed-up the search for the optimal solution and to improve the quality of the first solution found in the tree. The following trivial lower bound is computed after a variable instantiation:

\[ LB = c_y + \sum_{j=1}^{t} \min_j p_j \] (where \( C_{ij} \) is the completion time of \( O_{ij} \))

3.3. Discrepancy strategy

In our problem, the initial leaf (with 0 discrepancy) is a solution since we do not constrain the makespan value. We may use the discrepancy principle to expand the tree search for visiting the neighborhood of this initial solution. In a previous work, we have developed three strategies to apply discrepancy:

- Considering discrepancy only on operation selection variables;
- Considering discrepancy only on resource allocation variables;
- Mix the two kinds of discrepancies.
This latter strategy gives best solutions (Ben Hmida et al., 2007b), but all of the three strategies lead to a huge computing time since they visit the entire neighborhood and recalculate starting times of operations and their assignments following the dynamic heuristic (ECT). To restrict it, we propose to backjump on promising choice points (Huguet et al., 2004). We therefore decide to apply discrepancy on some relevant variables chosen by using two types of heuristics:

- Permutation of two adjacent critical operations carried out by the same resource (discrepancy on selection variable).
- Replacement of a critical operation on another resource (discrepancy on allocation variable but restricted to critical operations).

This led us to recalculate only the starting times of a subset of operations who are actually concerned with the discrepancy.

4. Computational results

The CDDS procedure described in Section 3 has been tested on different problem instances from literature (Brandimarte, 1993; Hurink et al. 1994).

- Brandimarte: The data set consists of 10 problems with number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15, and number of operations for each job ranging from 5 to 15.
- Hurink: The data set consists of 129 test problems created from 43 classical JSP instances. They divide the test problems into three subsets, Edata, Rdata and Vdata, depending on the average number of alternative machines for each operation. The number of jobs ranges from 6 to 30, the number of machines ranges from 5 to 15.

Table 2. Comparison with the Tabu Search of Mastrolilli and Gambardella (M.G.) on 10 FJSP instances from Brandimarte

<table>
<thead>
<tr>
<th>instances</th>
<th>n</th>
<th>m</th>
<th>LB</th>
<th>M.G.</th>
<th>CDDS</th>
<th>%dev</th>
<th>CPU(M.G.)</th>
<th>CPU(CDDS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mk01</td>
<td>10</td>
<td>6</td>
<td>36</td>
<td>40</td>
<td>40</td>
<td>0.0</td>
<td>0.01</td>
<td>0.1</td>
</tr>
<tr>
<td>Mk02</td>
<td>10</td>
<td>6</td>
<td>24</td>
<td>26</td>
<td>26</td>
<td>0.0</td>
<td>0.73</td>
<td>0.2</td>
</tr>
<tr>
<td>Mk03</td>
<td>15</td>
<td>8</td>
<td>204</td>
<td>204*</td>
<td>204*</td>
<td>0.0</td>
<td>0.01</td>
<td>0.2</td>
</tr>
<tr>
<td>Mk04</td>
<td>15</td>
<td>8</td>
<td>48</td>
<td>60</td>
<td>60</td>
<td>0.0</td>
<td>0.08</td>
<td>0.03</td>
</tr>
<tr>
<td>Mk05</td>
<td>15</td>
<td>4</td>
<td>168</td>
<td>173</td>
<td>182</td>
<td>5.2</td>
<td>0.96</td>
<td>0.2</td>
</tr>
<tr>
<td>Mk06</td>
<td>10</td>
<td>15</td>
<td>33</td>
<td>58</td>
<td>60</td>
<td>3.4</td>
<td>3.26</td>
<td>0.1</td>
</tr>
<tr>
<td>Mk07</td>
<td>20</td>
<td>5</td>
<td>133</td>
<td>144</td>
<td>139</td>
<td>-3.5</td>
<td>8.91</td>
<td>0.3</td>
</tr>
<tr>
<td>Mk08</td>
<td>20</td>
<td>0</td>
<td>523</td>
<td>523*</td>
<td>523*</td>
<td>0.0</td>
<td>0.02</td>
<td>0.8</td>
</tr>
<tr>
<td>Mk09</td>
<td>20</td>
<td>10</td>
<td>299</td>
<td>307</td>
<td>307</td>
<td>0.0</td>
<td>0.15</td>
<td>0.4</td>
</tr>
<tr>
<td>Mk10</td>
<td>20</td>
<td>15</td>
<td>165</td>
<td>198</td>
<td>212</td>
<td>7.1</td>
<td>7.69</td>
<td>0.3</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.2</td>
<td>2.18</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 2 compares our CDDS algorithm with the Tabu Search algorithm proposed by Mastrolilli and Gambardella (2000) on 10 FJSP problem instances from Brandimarte (1993). The second and third columns report the number of jobs and the number of machines for each instance, respectively. The fourth column reports the best known lower-bound (Mastrolilli and Gambardella, 2000). The fifth column reports the best results of TS. The sixth and the seven columns report our makespan with the relative deviation with respect to TS algorithm. The remaining columns report the CPU time. Results show that solutions are comparable in time and quality.

Table 3 shows computational results over two instance classes. The first column reports the data set, the second column the number of instances for each class, the third column the average number of alternative machines per operation. The next column reports the percentage deviation of the best solution obtained by our CDDS, with respect to the best known lower bound. The table shows that our algorithm is stronger with a higher degree of flexibility (Hurink Vdata).

Table 3. Deviation percentage over the best known lower bound

<table>
<thead>
<tr>
<th>Data set</th>
<th>num</th>
<th>alt</th>
<th>CDDS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brandimarte</td>
<td>10</td>
<td>2.59</td>
<td>17.02</td>
</tr>
<tr>
<td>Hurink Edata</td>
<td>43</td>
<td>1.15</td>
<td>15.81</td>
</tr>
</tbody>
</table>
5. Conclusions and further works

In this abstract a Climbing Depth-bounded Discrepancy Search (CDDS) method is presented to solve Flexible Job Shop Scheduling problems with the objective of minimizing makespan. Our CDDS approach is based on ordering heuristics and involves a backjumping heuristic to apply two types of discrepancies. The test problems are benchmarks used in the literature. Our results are not better compared with those obtained using a Tabu Search, but in terms of makespan, we can consider that the CDDS method provides promising results. Developments can still be done to improve the solution’s quality of CDDS algorithm. Moreover, other variants of CDDS algorithm may be envisaged for instance by including efficient lower bounds for the FJSP.

References


Hurink Rdata | 43 | 2 | 9.85
Hurink Vdata | 43 | 4.31 | 1.11


