A GRASP and Branch-and-Bound Metaheuristic for the Job-Shop Scheduling Problem

Susana Fernandes
Universidade do Algarve
Faro, Portugal
sfer@ualg.pt

Helena R. Lourenço
Universitat Pompeu Fabra
Barcelona, Spain
helena.ramalhinho@upf.edu

The work of S. Fernandes is supported by the program POCI2010 of the Portuguese Fundação para a Ciência e Tecnologia. The work of Helena R. Lourenço is supported by Ministerio de Educación y Ciencia, Spain, SEC2003-01991/ECO.

Outline of the Presentation

• The Job-Shop Scheduling Problem
• Optimized Search Heuristic – GRASP_B&B
• Computational Results
• Conclusions
The Job-Shop Scheduling Problem

• Definition
  – It considers a set of jobs to be processed on a set of distinct machines.
  – Each job is defined by an ordered set of operations.
  – Each operation is assigned to a different machine with a predefined constant processing time.
  – The order of the operations within the jobs and its correspondent machines are fixed a priori and are independent from job to job.

• Objective
  – Find a sequence of time slots for each operation on each machine, minimizing the maximum of the completion time of all jobs – the makespan.

• Constraints
  – Each machine can only process one operation at a time.
  – Different machines cannot process the same job simultaneously.
  – Preemption is not allowed when processing operations.
The Job-Shop Scheduling Problem

- Instance

<table>
<thead>
<tr>
<th></th>
<th>Job 1</th>
<th>Job 2</th>
<th>Job 3</th>
<th>Job 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>operations</td>
<td>1 2 3</td>
<td>4 5 6</td>
<td>7 8 9</td>
<td>10 11 12</td>
</tr>
<tr>
<td>machines</td>
<td>1 2 3</td>
<td>1 3 2</td>
<td>1 3 2</td>
<td>1 2 3</td>
</tr>
<tr>
<td>proc. time</td>
<td>1 1 2</td>
<td>4 2 2</td>
<td>1 1 1</td>
<td>4 2 1</td>
</tr>
</tbody>
</table>

- A feasible solution

- Disjunctive Graph
Optimized Search Heuristic – GRASP_B&B

• GRASP – applied to job shop

\[
\text{GRASP (total\_runs)} \\
\text{for (run = 1 to total\_runs)} \\
\text{while (solution not complete) do} \\
\text{Greedy randomized building step} \\
\text{Local search}
\]

– Elements to join the solution - the sequence of operations at each machine.

– Each element is evaluated by a heuristic function and incorporated (or not) in a restricted candidate list (RCL) according to its evaluation.

– Greedy function to evaluate the elements – the makespan of each one-machine problem.

Optimized Search Heuristic – GRASP_B&B

• Building step – RCL

– Find the optimal solution for the one-machine problems for every machine not yet scheduled
  • branch-and-bound algorithm of Carlier (1982).

– identify the best (lowest) and worse (biggest) makespans.

– A machine \(k\) is included in the RCL if

\[
f(s_k) \leq \overline{f} - \alpha(\overline{f} - f)
\]

– Machines with low values of makespan have less probability of being included in the RCL.

– The machine to join the solution is chosen randomly from the RCL.
Optimized Search Heuristic – GRASP_B&B

• Building step – instance

Optimized Search Heuristic – GRASP_B&B

• Building step – instance

Optimized Search Heuristic – GRASP_B&B

• Building step – instance
Optimized Search Heuristic – GRASP_B&B

• Building step – instance

Optimized Search Heuristic – GRASP_B&B

• Local search – neighborhood structure
  – Forward moves over forward critical pairs of operations
  – Backward moves over backward critical pairs of operations
    • Block of critical operations - maximal ordered set of consecutive operations on a critical path sharing the same machine
Optimized Search Heuristic – GRASP_B&B

• Local search – neighborhood structure

  – Two operations $u$ and $v$ form a **forward critical pair** $(u,v)$ if:
    • a) they both belong to the same block;
    • b) $v$ is the last operation of the block;
    • c) the job successor of $v$ also belongs to the same critical path;
    • d) the length of the critical path from $v$ to the end is not less than the length of the critical path from the job successor of $u$ to the end.

  – A forward move is executed by moving operation $u$ to be processed immediately after operation $v$.

Optimized Search Heuristic – GRASP_B&B

• Local search – neighborhood structure

  – **forward move** – example (10,7)
Optimized Search Heuristic – GRASP_B&B

• Local search – neighborhood structure
  
  – Two operations \( u \) and \( v \) form a **backward critical pair** \((u,v)\) if:
    - a) they both belong to the same block;
    - b) \( u \) is the first operation of the block;
    - c) the job predecessor of \( u \) also belongs to the same critical path;
    - d) the length of the critical path from node 0 to \( u \), including the processing time of \( u \), is not less than the length of the critical path from node 0 to the job predecessor of \( v \), including the processing time of the job predecessor of \( v \).
  
  – A backward move is executed by moving operation \( v \) to be processed immediately before operation \( u \).

Optimized Search Heuristic – GRASP_B&B

• Local search – neighborhood structure
  
  – backward move – example (8,3)

![Diagram](image1)

**Diagram Description:**
- Node 0 to 13 represents the operations with their corresponding processing times.
- Edges indicate dependencies between operations.
- The makespan is 14 for both diagrams.
Optimized Search Heuristic – GRASP_B&B

- Local search – inspecting the neighborhood
  - Acceptance strategy – first accept
    - Given solution \( x \) with \( M \) machines already scheduled
    - stop whenever we find a neighbor with a best evaluation value than the makespan of \( x \).
  - Evaluation function
    - Compute the length of all the longest paths through the operations that were between operations \( u \) and \( v \) of the critical pair \( (u,v) \) on the critical path of solution \( x \).
    - Use the same subroutine of Balas and Vazacopoulos (1998) algorithm.

Computational Results

- Instances
  - abz5-9 (Adams et al. 1988)
  - ft6, ft10, ft20 (Fisher and Thompson 1963)
  - la01-40 (Lawrence 1984)
  - orb01-10 (Applegate and Cook 1991)
  - swv01-20 (Storer et al. 1992)
  - ta01-70 (Taillard 1993)
  - yn1-4 (Yamada and Nakano 1992)

- Platform
  - Pentium 4 CPU 2.80 GHz
  - Code in Visual C

- Runs
  - 100 runs for each instance.
Computational Results

- Boxplots of $RE_{c_f}(x) = 100 \times \frac{f(x) - UB}{UB}$

Upperbounds (UB) gathered from Jain and Meeran (1999) and Nowicki and Smutnicki (1996, 2005).

- Fisher and Thompson (1963) instances

- Lawrence (1984) instances

la 01 – 05: 10*5
la 06 – 10: 15*5
la 11 – 15: 20*5
la 16 – 20: 10*10
la 21 – 25: 15*10
la 26 – 30: 20*10
la 31 – 35: 30*10
la 36 – 40: 15*15
Computational Results

• Comparison to other algorithms

  – The Shifting Bottleneck Procedure of Adams, Balas and Zawack (1988)
  
  – The GRASP Procedure of Binato, Loewenstern and Resende (2001)

Computational Results

• GRASP_B&B  Comparison to other algorithms

  – Lawrence (1984) instances 31-40

<table>
<thead>
<tr>
<th>name</th>
<th>GRASP_B&amp;B</th>
<th>btime (s)</th>
<th>ttime (s)</th>
<th>GRASP</th>
<th>time (s)</th>
<th>Shifting Bottleneck</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>la31</td>
<td>1784</td>
<td>0.0702</td>
<td>7.0160</td>
<td>1784</td>
<td>231290</td>
<td>1784</td>
<td>38.3</td>
</tr>
<tr>
<td>la32</td>
<td>1850</td>
<td>0.5612</td>
<td>6.2350</td>
<td>1850</td>
<td>241390</td>
<td>1850</td>
<td>29.1</td>
</tr>
<tr>
<td>la33</td>
<td>1719</td>
<td>1.265</td>
<td>7.9060</td>
<td>1719</td>
<td>241390</td>
<td>1719</td>
<td>25.6</td>
</tr>
<tr>
<td>la34</td>
<td>1721</td>
<td>3.8093</td>
<td>8.2810</td>
<td>1753</td>
<td>240380</td>
<td>1721</td>
<td>27.6</td>
</tr>
<tr>
<td>la35</td>
<td>1888</td>
<td>0.2844</td>
<td>5.6880</td>
<td>1888</td>
<td>22200</td>
<td>1888</td>
<td>21.3</td>
</tr>
<tr>
<td>la36</td>
<td>1325</td>
<td>0.0853</td>
<td>4.2650</td>
<td>1334</td>
<td>115360</td>
<td>1351</td>
<td>46.9</td>
</tr>
<tr>
<td>la37</td>
<td>1479</td>
<td>4.0295</td>
<td>4.7970</td>
<td>1457</td>
<td>1115360</td>
<td>1485</td>
<td>61.4</td>
</tr>
<tr>
<td>la38</td>
<td>1274</td>
<td>0.7153</td>
<td>5.1090</td>
<td>1267</td>
<td>118720</td>
<td>1280</td>
<td>57.5</td>
</tr>
<tr>
<td>la39</td>
<td>1309</td>
<td>2.9835</td>
<td>4.4530</td>
<td>1290</td>
<td>115360</td>
<td>1321</td>
<td>71.8</td>
</tr>
<tr>
<td>la40</td>
<td>1291</td>
<td>3.5581</td>
<td>5.3910</td>
<td>1259</td>
<td>123200</td>
<td>1326</td>
<td>76.7</td>
</tr>
</tbody>
</table>
Conclusions

• We have designed a very simple optimized search heuristic
  – GRASP
  – Branch-and-Bound

• Although this is a very simple (and fast) algorithm, it achieves the best
  known upper bound in 23 of the 152 instances used in this study.

• Compared with other initial solution heuristics:
    • Much faster
    • Quality of solution slightly worse in 60% of all the instances tested.
    • Faster
    • Better solutions for all comparable instances (except 2).

• Good starting point for a more elaborated metaheuristic.

References