Climbing Depth-Bounded Adjacent Discrepancy Search for Solving Hybrid Flow Shop Scheduling Problems with Multiprocessor Tasks

A. LAHIMER$^1$, P. LOPEZ$^1$, M. HAOUARI$^2$

$^1$LAAS-CNRS ; Université de Toulouse, France
$^2$INSAT, Tunisie

{lahimer,lopez}@laas.fr, mohamed.haouari@ozyegin.edu.tr
Plan

1. Problem Definition
2. Discrepancy Methods
3. Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search
4. Computational Study
5. Conclusion
Multiprocessor Hybrid Flow shop
Multiprocessor Hybrid Flow shop
Multithreaded Hybrid Flow shop

Stage 1

Stage 2

Stage m

streams
Problem Definition

Multiprocessor Hybrid Flow shop

Stage 1

Stage 2

Stage m

\( F_m(Pm_1, \ldots, Pm_m) | size_{ij} | C_{max} \)
Some Applications

- Manufacturing: work-force assignment, transportation problem with recirculation...
- Operating Systems
- Real-time machine vision
Some Applications

- Manufacturing: work-force assignment, transportation problem with recirculation...

- Operating Systems

- Real-time machine vision

**Complexity**: NP-hard in the strong sense [J.A. Hoogeven, 1996]
Literature Review

Approaches

- Genetic Algorithm [C. Oğuz et al., 2003]
- Tabu Search [C. Oğuz et al., 2004]
- Ant Colony System [F.S. Şerifoğlu et al., 2006]
- Particle Swarm Optimization [M.F. Ercan et al., 2007]
- Constraint Programming [A. Jouglet et al., 2009]
Literature Review

**Approaches**

- Genetic Algorithm [C. Oğuz et al., 2003]
- Tabu Search [C. Oğuz et al., 2004]
- Ant Colony System [F.S. Şerifoğlu et al., 2006]
- Particle Swarm Optimization [M.F. Ercan et al., 2007]
- Constraint Programming [A. Jouglet et al., 2009]

**Lower Bounds**

- Specific to F2 [C. Oğuz et al., 2003]
- Adapted to Fm [C. Oğuz et al., 2004]
General Statement

- Genesis: LDS (Limited Discrepancy Search) [Harvey & Ginsberg, 1995]
General Statement

- **Genesis:** LDS (Limited Discrepancy Search) [Harvey & Ginsberg, 1995]
General Statement

- **Genesis:** LDS (Limited Discrepancy Search) [Harvey & Ginsberg, 1995]

- A discrepancy = any decision point in the search tree where the choice goes against the heuristic
ILDS: Improved LDS [R. Korf, 1996]

**Figure**: Improved Limited Discrepancy Search
DDS: Depth-bounded Discrepancy Search [T. Walsh, 1997]

**Figure**: DDS
CDS: Climbing Discrepancy Search [Milano & Roli, 2002]

\[ f_{\text{ref}} \geq f_1 \geq f_\text{ref} \geq f_2 \geq f_\text{ref} \ldots \geq f_5 < f_\text{ref} \]

**Figure:** A CDS scenario
DADS: Depth-bounded Adjacent Discrepancy Search

Figure: DADS
DADS: Depth-bounded Adjacent Discrepancy Search

**Figure:** DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS

SRef

X

X

X

X

X

X

X

X

X
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS

CPAIOR 2011
Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

CDADS

Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

CDADS

Climbing DADS
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Climbing DADS

Stopping Conditions

- CPU time (60 sec)
- Cost(Sol)=LB
CDADS is strongly based on the quality of the initial solution

An experimental comparison between various priority rules presented in the literature to consider the most effective
Heuristics Selection

- CDADS is strongly based on the quality of the initial solution
- An experimental comparison between various priority rules presented in the literature to consider the most effective

<table>
<thead>
<tr>
<th>Priority Rule</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPT_LastStage</td>
<td>27</td>
</tr>
<tr>
<td>Energy</td>
<td>25</td>
</tr>
<tr>
<td>SPT</td>
<td>17</td>
</tr>
<tr>
<td>SPR</td>
<td>14</td>
</tr>
</tbody>
</table>
Heuristics Selection

- CDADS is strongly based on the quality of the initial solution.
- An experimental comparison between various priority rules presented in the literature to consider the most effective.

**Table: Heuristics Selection**

<table>
<thead>
<tr>
<th>Priority Rule</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPT_LastStage</td>
<td>27</td>
</tr>
<tr>
<td>Energy</td>
<td>25</td>
</tr>
<tr>
<td>SPT</td>
<td>17</td>
</tr>
<tr>
<td>SPR</td>
<td>14</td>
</tr>
</tbody>
</table>

**Shortest Processing Requirement:** $size_{ij}$ increasing order
Heuristics Selection

- CDADS is strongly based on the quality of the initial solution.
- An experimental comparison between various priority rules presented in the literature to consider the most effective.

**Table: Heuristics Selection**

<table>
<thead>
<tr>
<th>Priority Rule</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPT_LastStage</td>
<td>27</td>
</tr>
<tr>
<td>Energy</td>
<td>25</td>
</tr>
<tr>
<td>SPT</td>
<td>17</td>
</tr>
<tr>
<td>SPR</td>
<td>14</td>
</tr>
</tbody>
</table>

*Shortest Processing Time: $p_{ij}$ increasing order*
Heuristics Selection

- CDADS is strongly based on the quality of the initial solution.
- An experimental comparison between various priority rules presented in the literature to consider the most effective.

<table>
<thead>
<tr>
<th>Priority Rule</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPT-LastStage</td>
<td>27</td>
</tr>
<tr>
<td>Energy</td>
<td>25</td>
</tr>
<tr>
<td>SPT</td>
<td>17</td>
</tr>
<tr>
<td>SPR</td>
<td>14</td>
</tr>
</tbody>
</table>

\[ energy_{ij} = size_{ij} \times p_{ij} \]
Heuristics Selection

- CDADS is strongly based on the quality of the initial solution
- An experimental comparison between various priority rules presented in the literature to consider the most effective

<table>
<thead>
<tr>
<th>Priority Rule</th>
<th>Performance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPT.LastStage</td>
<td>27</td>
</tr>
<tr>
<td>Energy</td>
<td>25</td>
</tr>
<tr>
<td>SPT</td>
<td>17</td>
</tr>
<tr>
<td>SPR</td>
<td>14</td>
</tr>
</tbody>
</table>

**NSPT**: Normalized SPT
Schedule Generation Scheme

- **Two Types of SGSs**
  - Serial SGS [Kelley et al., 1963]
  - Parallel SGS [Brooks et al., 1965]

- **Generated Schedules**
  - Serial SGSs generate active schedules.
  - Parallel SGSs generate non-delay schedules.

- According to our experimental studies, a parallel SGS is more adapted to our problem.
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Lower Bounds

Lower Bounds

\[ LB = \max(LB_s, LB_j) \]

- \( LB_s = \max_{i=1..m} LB(i) \)

- \( LB(i) = \min_{j \in J} \left( \sum_{l=1}^{i-1} p_{ij} \right) + \max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min_{j \in J} \left( \sum_{l=i+1}^{m} p_{ij} \right) \)

- \( M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} \cdot \text{size}_{ij} \right\rceil \)

- \( M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij} \)

- \( A_i = \{ j | \text{size}_{ij} > \frac{m_i}{2} \} \) and \( B_i = \{ j | \text{size}_{ij} = \frac{m_i}{2} \} \)

- \( LB_j = \max_{j \in J} \left( \sum_{i=1}^{m} p_{ij} \right) \).
Lower Bounds

\[ LB = \max(LB_s, LB_j) \]

- **\( LB_s \)** = \( \max_{i=1..m} LB(i) \)
  
  \[ LB(i) = \min_{j \in J} \left( \sum_{l=1}^{i-1} p_{ij} \right) + \max(M_1(i), M_2(i), \max(p_{ij})) + \min_{j \in J} \left( \sum_{l=i+1}^{m} p_{ij} \right) \]

- **\( M_1(i) \)** = \( \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} \text{size}_{ij} \right\rceil \)

- **\( M_2(i) \)** = \( \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij} \)

  \( A_i = \{ j | \text{size}_{ij} > \frac{m_i}{2} \} \), \( B_i = \{ j | \text{size}_{ij} = \frac{m_i}{2} \} \)

- **\( LB_j \)** = \( \max_{j \in J} \left( \sum_{i=1}^{m} p_{ij} \right) \).
Lower Bounds

\[ LB = \max(LB_s, LB_j) \]

- \( LB_s = \max_{i=1..m} LB(i) \)

- \( LB(i) = \min_{j \in J} \left( \sum_{l=1}^{i-1} p_{ij} \right) + \max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min_{j \in J} \left( \sum_{l=i+1}^{m} p_{lj} \right) \)

- \( M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} \text{size}_{ij} \right\rceil \)

- \( M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij} \)

- \( A_i = \{ j | \text{size}_{ij} > \frac{m_i}{2} \} \), \( B_i = \{ j | \text{size}_{ij} = \frac{m_i}{2} \} \).

- \( LB_j = \max_{j \in J} \left( \sum_{i=1}^{m} p_{ij} \right) \).
Lower Bounds

\[ LB = \max(LB_s, LB_j) \]

- **\( LB_s = \max_{i=1..m} LB(i) \)**
  \[ LB(i) = \min_{j \in J} \left( \sum_{l=1}^{i-1} p_{ij} \right) + \max(M_1(i), M_2(i), \max(p_{ij})) + \min_{j \in J} \left( \sum_{l=i+1}^{m} p_{ij} \right) \]
  \[ M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} \cdot \text{size}_{ij} \right\rceil \]
  \[ M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij} \]
  \[ A_i = \{ j | \text{size}_{ij} > \frac{m_i}{2} \} \]
  \[ B_i = \{ j | \text{size}_{ij} = \frac{m_i}{2} \} . \]
- **\( LB_j = \max_{j \in J} \left( \sum_{i=1}^{m} p_{ij} \right) \).**
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Proposal: Climbing Depth-Bounded Adjacent Discrepancy Search

Lower Bounds

\[ LB = \max(LB_s, LB_j) \]

- \( LB_s = \max_{i=1..m} LB(i) \)
  \[ LB(i) = \min \left( \sum_{j \in J} p_{ij} \right) + \max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min \left( \sum_{l=i+1}^{m} p_{lj} \right) \]

- \( M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} \cdot \text{size}_{ij} \right\rceil \)

- \( M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij} \)
  
  \( A_i = \{ j | \text{size}_{ij} > \frac{m_i}{2} \} \), \( B_i = \{ j | \text{size}_{ij} = \frac{m_i}{2} \} \).

- \( LB_j = \max_{j \in J} \left( \sum_{i=1}^{m} p_{ij} \right) \).
Initialization

LB = calculation_LB(problem)

k := 0

SRef = GenerateSolRef(k)

UB = CostSol(SRef)

CPU Time < 60 s
And UB > LB

NbrNodes = nbrIniNodes * f^k

NbrVisitedNodes <= NbrMaxNodes

CDADS

k < 4 and ! improvement

k := k + 1

End
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Computational Study

Test beds

Implementation

PC Intel Centrino 2 Duo 2 GHz
OS: Ubuntu
language: C++

CPU time (sec)
Test beds

Implementation

PC Intel Centrino 2 Duo 2 GHz
OS: Ubuntu
language: C++

Oğuz et al.’s Benchmark, 2004

Size: 300 instances
number of jobs: \{5, 10, 20, 50, 100\}
number of stages: \{2, 5, 8\}
2 Categories: ‘Type_1’ and ‘Type_2’
‘Type_1’: \(m_i = 1, \ldots, 5\)
‘Type_2’: \(m_i = 5\)
Test beds

Implementation

PC Intel Centrino 2 Duo 2 GHz
OS: Ubuntu
language: C++

Oğuz et al. ’s Benchmark, 2004

Size: 300 instances
number of jobs: {5, 10, 20, 50, 100}
number of stages: {2, 5, 8}
2 Categories: ‘Type_1’ and ‘Type_2’
‘Type_1’: \( m_i = 1, \ldots, 5 \)
‘Type_2’: \( m_i = 5 \)

Indicators

Deviation (%):

- \( 100 \times \frac{C_{\text{max}} - LB}{LB} \)
- \( 100 \times \frac{C_{\text{max}} - C^{*}_{\text{max}}}{C^{*}_{\text{max}}} \)

CPU time (sec)
CDADS Performance

**Table: CDADS Performance**

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>Avg %dev</th>
<th>CPU (s)</th>
<th>Avg %dev</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>0.00</td>
<td>&lt;0.1</td>
<td>0.00</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.21</td>
<td>&lt;0.1</td>
<td>0.46</td>
<td>&lt;0.1</td>
<td>0.50</td>
</tr>
<tr>
<td>8</td>
<td>1.71</td>
<td>&lt;0.1</td>
<td>0.00</td>
<td>&lt;0.1</td>
<td>1.72</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.00</td>
<td>&lt;0.1</td>
<td>1.72</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>5</td>
<td>0.66</td>
<td>0.40</td>
<td>6.44</td>
<td>&lt;0.1</td>
<td>9.61</td>
</tr>
<tr>
<td>8</td>
<td>8.47</td>
<td>&lt;0.1</td>
<td>9.61</td>
<td>&lt;0.1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>0.05</td>
<td>0.10</td>
<td>3.34</td>
<td>3.10</td>
</tr>
<tr>
<td>5</td>
<td>2.57</td>
<td>1.10</td>
<td>7.97</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5.11</td>
<td>0.20</td>
<td>15.00</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>0.49</td>
<td>2.30</td>
<td>1.74</td>
<td>4.20</td>
</tr>
<tr>
<td>5</td>
<td>0.54</td>
<td>5.00</td>
<td>8.20</td>
<td>13.50</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.62</td>
<td>6.80</td>
<td>12.42</td>
<td>33.40</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>0.08</td>
<td>11.10</td>
<td>3.32</td>
<td>22.80</td>
</tr>
<tr>
<td>5</td>
<td>1.50</td>
<td>13.60</td>
<td>10.75</td>
<td>40.90</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.86</td>
<td>11.00</td>
<td>14.33</td>
<td>47.30</td>
<td></td>
</tr>
</tbody>
</table>

Avg %dev 1.66 6.39

CPU (s) 3.44 10.53
CDADS Vs literature

<table>
<thead>
<tr>
<th>Type</th>
<th>Avg%dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDADS</td>
<td>1.66</td>
</tr>
<tr>
<td>GA</td>
<td>2.27</td>
</tr>
<tr>
<td>CP</td>
<td>5.39</td>
</tr>
<tr>
<td>MA</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Legend:
- Type_1
- Type_2
Climbing Depth-Bounded Adjacent Discrepancy Search for Solving HFS Scheduling Problems with Multiprocessor Tasks

Computational Study

CDADS Vs literature

CDADS Vs literature

Avg%dev

CPU Time(en sec)

<table>
<thead>
<tr>
<th>Type</th>
<th>CDADS</th>
<th>GA</th>
<th>CP</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type_1</td>
<td>1.66</td>
<td>2.27</td>
<td>5.39</td>
<td>1.6</td>
</tr>
<tr>
<td>Type_2</td>
<td>6.39</td>
<td>9.1</td>
<td>11.92</td>
<td>7.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>CDADS</th>
<th>GA</th>
<th>CP</th>
<th>MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type_1</td>
<td>3.44</td>
<td>10.53</td>
<td>257257</td>
<td>123.68</td>
</tr>
<tr>
<td>Type_2</td>
<td>94</td>
<td>95</td>
<td>150.41</td>
<td></td>
</tr>
</tbody>
</table>
CDADS Vs literature

The rate of improvement reaches 25

**Figure:** Variation of the number of improved solutions with the number of jobs
Contributions

- CDADS provides better solutions in little CPU time;
- CDADS excels on large instances;
- The proposed LB is efficient [Oğuz & Ercan, 2005];
- Experimental study shows the most adapted heuristics to the studied problem.
Contributions

- CDADS provides better solutions in little CPU time;
- CDADS excels on large instances;
- The proposed LB is efficient [Oğuz & Ercan, 2005];
- Experimental study shows the most adapted heuristics to the studied problem.
Contributions

- CDADS provides better solutions in little CPU time;
- CDADS excels on large instances;
- The proposed LB is efficient [Oğuz & Ercan, 2005];
- Experimental study shows the most adapted heuristics to the studied problem.
Contributions

- CDADS provides better solutions in little CPU time;
- CDADS excels on large instances;
- The proposed LB is efficient [Oğuz & Ercan, 2005];
- Experimental study shows the most adapted heuristics to the studied problem.
Prospects

- Explore the impact of adjacent discrepancies vs. other strategies for limiting the search space;
- Consider the application of CDADS to simpler problems like classical hybrid flow shop ($size_{ij} = 1, \forall i, j$);
- Adapt the proposed implementation of discrepancy search to more general scheduling problems, in particular the Resource-Constrained Project Scheduling Problem;
- Propose a new lower bound based on linear relaxation of the RCPSP.
Prospects

- Explore the impact of adjacent discrepancies vs. other strategies for limiting the search space;
- Consider the application of CDADS to simpler problems like classical hybrid flow shop ($\text{size}_{ij} = 1, \forall i,j$);
- Adapt the proposed implementation of discrepancy search to more general scheduling problems, in particular the Resource-Constrained Project Scheduling Problem;
- Propose a new lower bound based on linear relaxation of the RCPSP.
Prospects

- Explore the impact of adjacent discrepancies vs. other strategies for limiting the search space;
- Consider the application of CDADS to simpler problems like classical hybrid flow shop ($size_{ij} = 1, \forall i, j$);
- Adapt the proposed implementation of discrepancy search to more general scheduling problems, in particular the Resource-Constrained Project Scheduling Problem;
- Propose a new lower bound based on linear relaxation of the RCPSP.
Prospects

- Explore the impact of adjacent discrepancies vs. other strategies for limiting the search space;
- Consider the application of CDADS to simpler problems like classical hybrid flow shop ($\text{size}_{ij} = 1$, $\forall i, j$);
- Adapt the proposed implementation of discrepancy search to more general scheduling problems, in particular the Resource-Constrained Project Scheduling Problem;
- Propose a new lower bound based on linear relaxation of the RCPSP.