May 23-27
CPAIOR 2011
Zuse Institute
BERLIN
8th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems

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

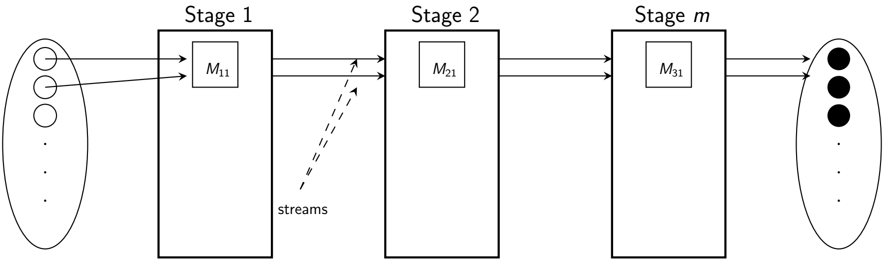
2 Discrepancy Methods

3 Proposal : Climbing Depth-Bounded Adjacent Discrepancy Search

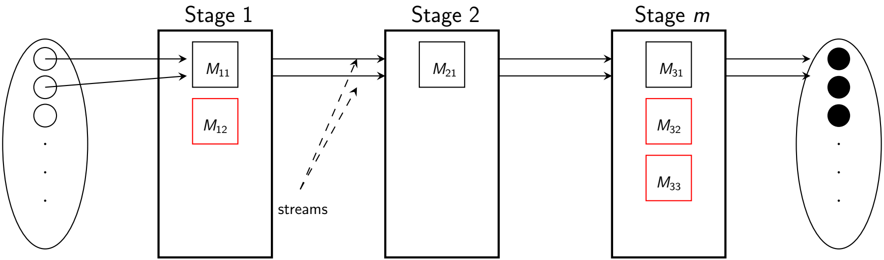
4 Computational Study

5 Conclusion

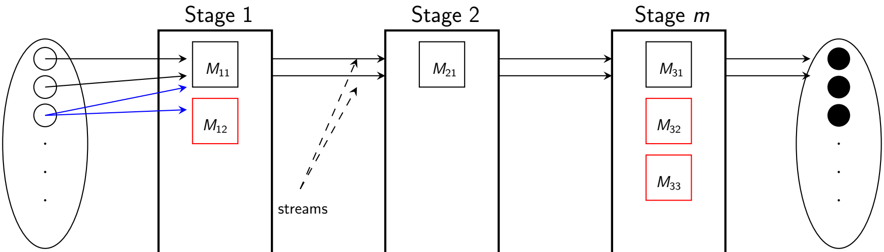
Multiprocessor Hybrid Flow shop



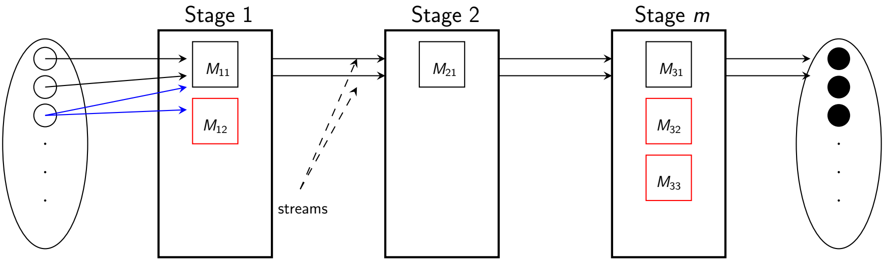
Multiprocessor Hybrid Flow shop



Multiprocessor Hybrid Flow shop



Multiprocessor Hybrid Flow shop



$$\text{Fm}(Pm_1,\ldots,Pm_m)|size_{ij}|C_{max}$$

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

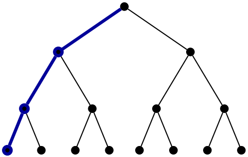
- Specific to F2 [C. Oğuz *et al.*, 2003]
- Adapted to Fm [C. Oğuz *et al.*, 2004]

General Statement

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

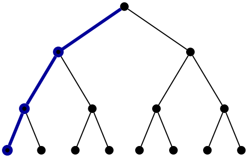
General Statement

- 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]

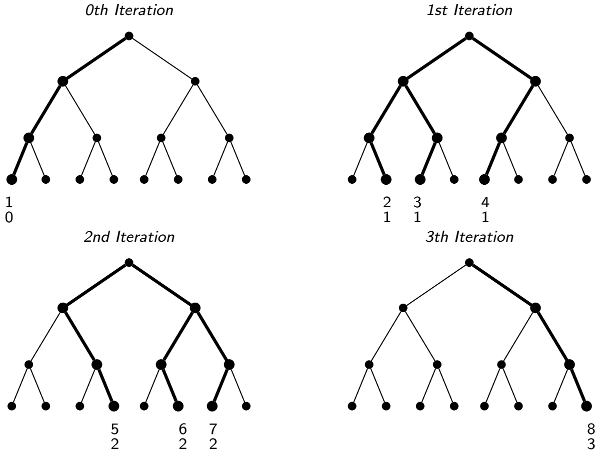


FIGURE: Improved Limited Discrepancy Search

DDS: Depth-bounded Discrepancy Search [T. Walsh, 1997]

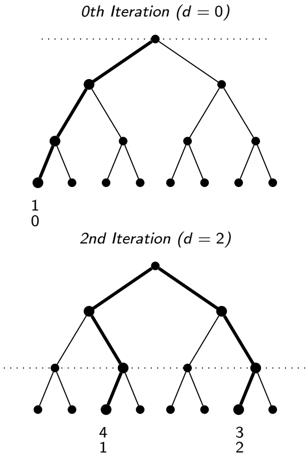


Figure: DDS

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

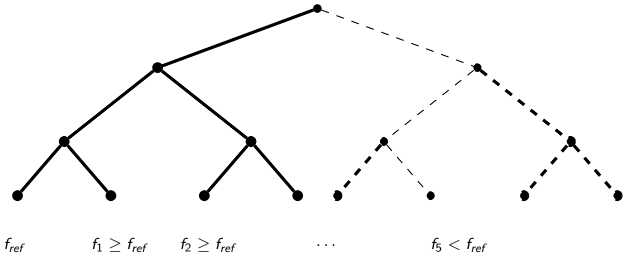


FIGURE: A CDS scenario

DADS: Depth-bounded Adjacent Discrepancy Search

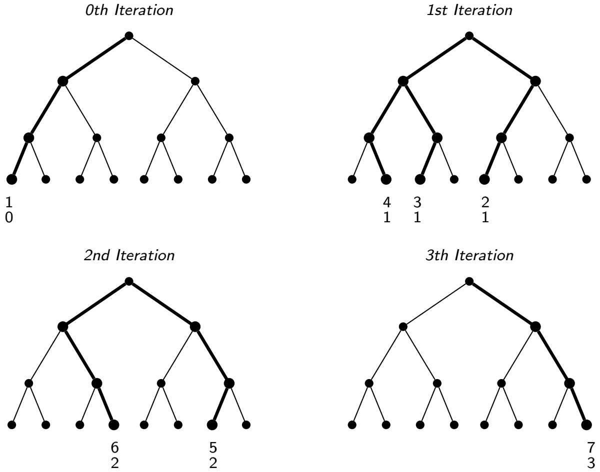


FIGURE: DADS

DADS: Depth-bounded Adjacent Discrepancy Search

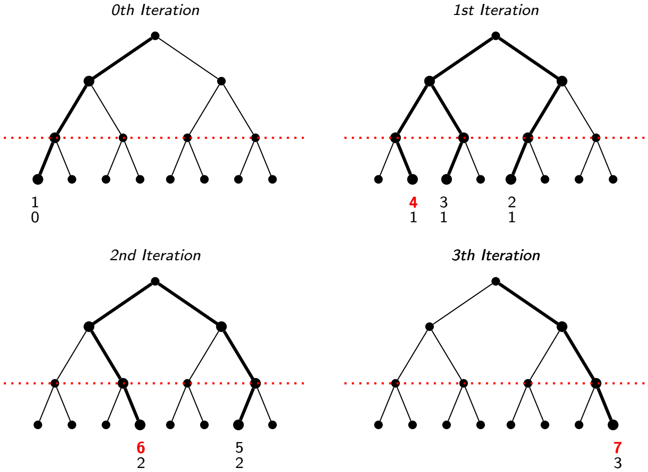
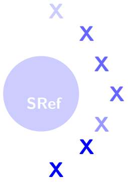


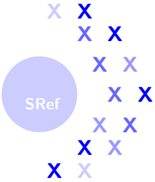
FIGURE: DADS

Climbing DADS

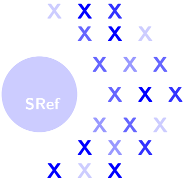
Climbing DADS



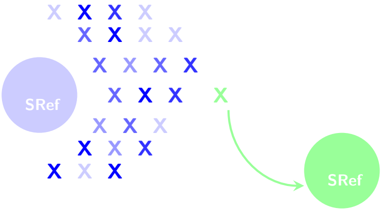
Climbing DADS



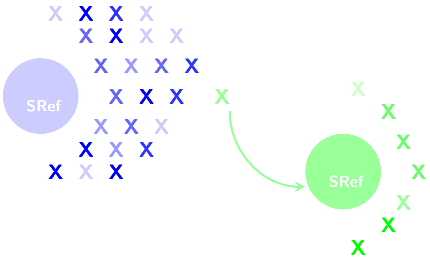
Climbing DADS



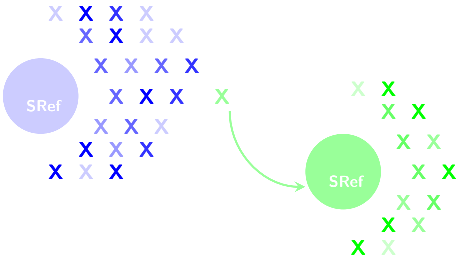
Climbing DADS



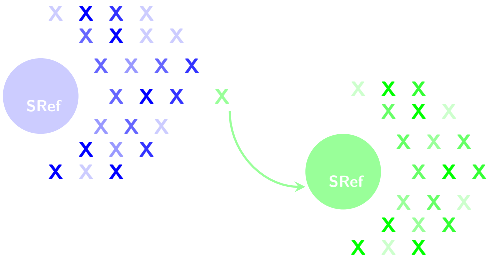
Climbing DADS



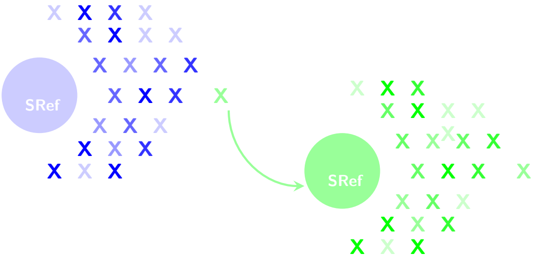
Climbing DADS



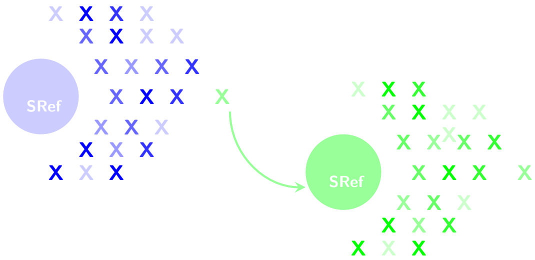
Climbing DADS



Climbing DADS



Climbing DADS



Stopping Conditions

- CPU time (60 sec)
- Cost(Sol)=LB

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

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

Priority Rule	Performance (%)
NSPT_LastStage	27
Energy	25
SPT	17
SPR	14

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

Priority Rule	Performance (%)
NSPT_LastStage	27
Energy	25
SPT	17
SPR	14

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

Priority Rule	Performance (%)
NSPT_LastStage	27
Energy	25
SPT	17
SPR	14

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: Heuristics Selection

Priority Rule	Performance (%)
NSPT_LastStage	27
Energy	25
SPT	17
SPR	14

$energy_{ij} = size_{ij} \times p_{ij}$

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: Heuristics Selection

Priority Rule	Performance (%)
NSPT_LastStage	27
Energy	25
SPT	17
SPR	14

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.

Lower Bounds

$$LB = \max(LB_s, LB_j)$$

- $LB_s = \max\limits_{i=1..m} LB(i)$
    - $LB(i) = \min\limits_{j \in J}(\sum_{l=1}^{i-1} p_{lj}) + max(M_1(i), M_2(i), \max\limits_{j \in J}(p_{ij})) + \min\limits_{j \in J}(\sum_{l=i+1}^{m} p_{lj})$
    - $M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} size_{ij} \right\rceil$

      $$M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij}$$
    - $A_i = \{j | size_{ij} > \frac{m_i}{2}\}$, $B_i = \{j | size_{ij} = \frac{m_i}{2}\}$
- $LB_j = \max\limits_{j \in J}(\sum_{i=1}^{m} p_{ij})$.

Lower Bounds

$$LB = \max(LB_s, LB_j)$$

- $LB_s = \max\limits_{i=1..m} LB(i)$
  - $LB(i) = \min\limits_{j \in J}(\sum_{l=1}^{i-1} p_{lj}) + max(M_1(i), M_2(i), \max\limits_{j \in J}(p_{ij})) + \min\limits_{j \in J}(\sum_{l=i+1}^{m} p_{lj})$
  - $M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} size_{ij} \right\rceil$

    $M_2(i) = \sum\limits_{j \in A_i} p_{ij} + \frac{1}{2} \sum\limits_{j \in B_i} p_{ij}$
    - $A_i = \{j | size_{ij} > \frac{m_i}{2}\}$, $B_i = \{j | size_{ij} = \frac{m_i}{2}\}$.
- $LB_j = \max\limits_{j \in J}(\sum_{i=1}^{m} p_{ij})$.

Lower Bounds

$$LB = \max(LB_s, LB_j)$$

- $LB_s = \max_{i=1..m} LB(i)$
  - $LB(i) = \min_{j \in J}(\sum_{l=1}^{i-1} p_{lj}) + max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min_{j \in J}(\sum_{l=i+1}^{m} p_{lj})$
  - $M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} size_{ij} \right\rceil$

    $M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij}$
    - $A_i = \{j | size_{ij} > \frac{m_i}{2}\}$, $B_i = \{j | size_{ij} = \frac{m_i}{2}\}$.
- $LB_j = \max_{j \in J}(\sum_{i=1}^{m} p_{ij})$.

Lower Bounds

$$LB = \max(LB_s, LB_j)$$

- $LB_s = \max_{i=1..m} LB(i)$
  - $LB(i) = \min_{j \in J}(\sum_{l=1}^{i-1} p_{lj}) + max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min_{j \in J}(\sum_{l=i+1}^{m} p_{lj})$
  - $M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} size_{ij} \right\rceil$

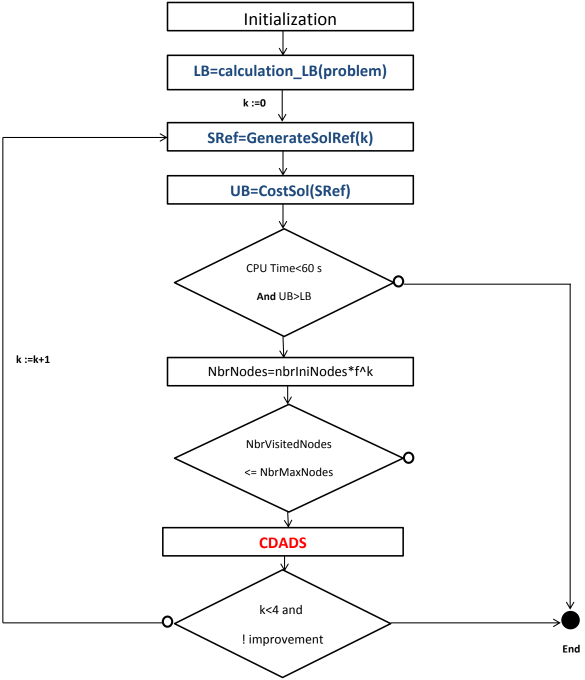
    $M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij}$
    - $A_i = \{j | size_{ij} > \frac{m_i}{2}\}$, $B_i = \{j | size_{ij} = \frac{m_i}{2}\}$.
- $LB_j = \max_{j \in J}(\sum_{i=1}^{m} p_{ij})$.

Lower Bounds

$$LB = \max(LB_s, LB_j)$$

- $LB_s = \max_{i=1..m} LB(i)$
  - $LB(i) = \min_{j \in J}(\sum_{l=1}^{i-1} p_{lj}) + max(M_1(i), M_2(i), \max_{j \in J}(p_{ij})) + \min_{j \in J}(\sum_{l=i+1}^{m} p_{lj})$
  - $M_1(i) = \left\lceil \frac{1}{m_i} \sum_{j \in J} p_{ij} size_{ij} \right\rceil$

    $M_2(i) = \sum_{j \in A_i} p_{ij} + \frac{1}{2} \sum_{j \in B_i} p_{ij}$
    - $A_i = \{j | size_{ij} > \frac{m_i}{2}\}$, $B_i = \{j | size_{ij} = \frac{m_i}{2}\}$.
- $LB_j = \max_{j \in J}(\sum_{i=1}^{m} p_{ij})$.



Initialization
LB=calculation_LB(problem)
k :=0
SRef=GenerateSolRef(k)
UB=CostSol(SRef)
CPU Time<60 s
And UB>LB
k :=k+1
NbrNodes=nbrIniNodes*f^k
NbrVisitedNodes
<= NbrMaxNodes
CDADS
k<4 and
! improvement
End

Test beds

Implementation

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

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': $m_i = 1, \dots, 5$
'Type_2': $m_i = 5$

Indicators

Deviation (%):

- $100 \times \frac{C_{\max} - LB}{LB}$
- $100 \times \frac{C_{\max} - C^*_{\max}}{C^*_{\max}}$

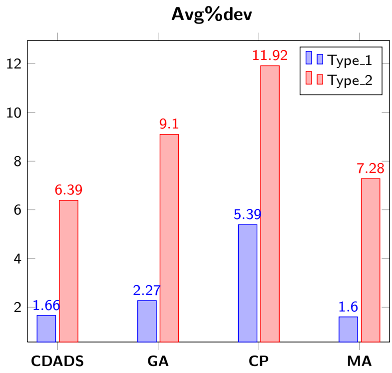
CPU time (sec)

CDADS Performance

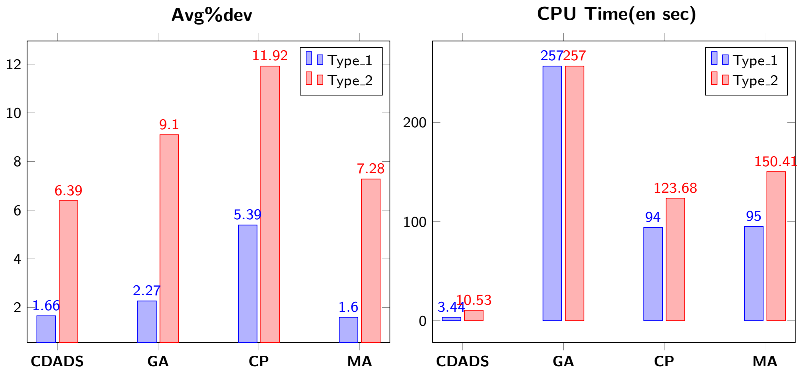
TABLE: CDADS Performance

		'Type_1' Problems		'Type_2' Problems	
n	**m**	*Avg %dev*	*CPU (s)*	*Avg %dev*	*CPU (s)*
5	2	0.00	< 0.1	0.00	< 0.1
	5	0.21	< 0.1	0.46	< 0.1
	8	1.71	< 0.1	0.50	< 0.1
10	2	0.00	< 0.1	1.72	< 0.1
	5	0.66	0.40	6.44	< 0.1
	8	8.47	< 0.1	9.61	< 0.1
20	2	0.05	0.10	3.34	3.10
	5	2.57	1.10	7.97	1.30
	8	5.11	0.20	15.00	1.30
50	2	0.49	2.30	1.74	4.20
	5	0.54	5.00	8.20	13.50
	8	1.62	6.80	12.42	33.40
100	2	0.08	11.10	3.32	22.80
	5	1.50	13.60	10.75	40.90
	8	1.86	11.00	14.33	47.30
Avg %dev		**1.66**		**6.39**	
CPU (s)			**3.44**		**10.53**

CDADS Vs literature



CDADS Vs literature



CDADS Vs literature

The rate of improvement reaches 25

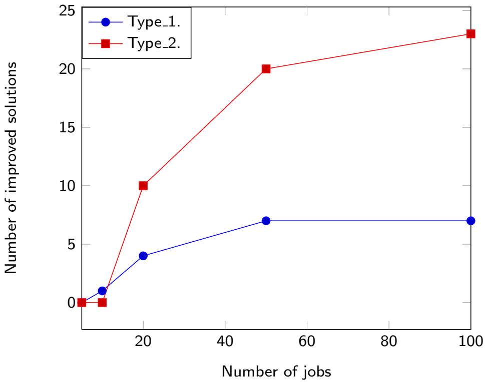


FIGURE: Variation of the number of improved solutions with the number of jobs

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 $(size_{ij} = 1, \forall i, j)$;
- 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 ($size_{ij} = 1, \forall i, j$) ;
- 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.