Universidad de los Andes

Limited Discrepancy Search for flexible shop scheduling

(présentation actualisée pour le séminaire MOGISA du 3 avril 2009)

Pierre López

LAAS-CNRS

Toulouse, France

(joint work with A. Ben Hmida, M.-J. Huguet, M. Haouari)

II SEMINARIO DE LOGÍSTICA INTEGRAL
19 y 20 de Febrero de 2009

Universidad de los Andes

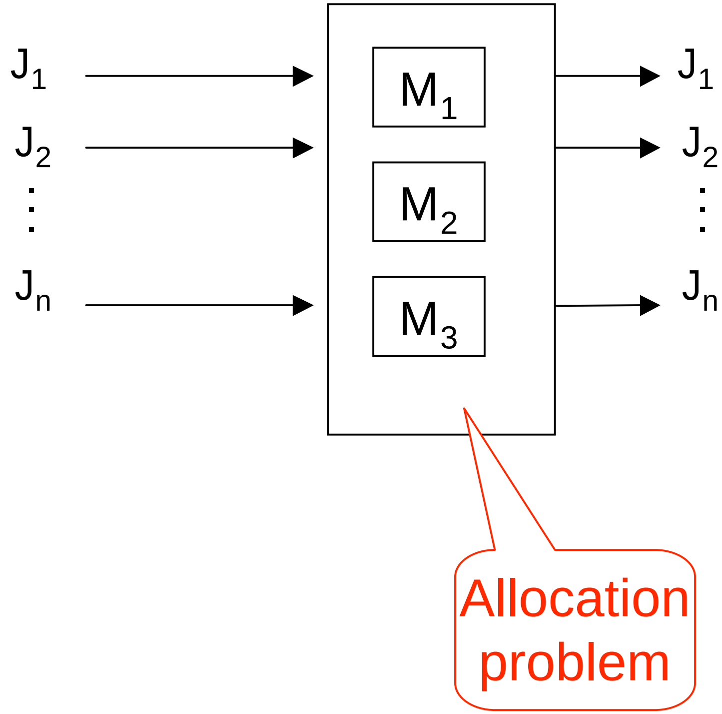
Objectives

- Some **research results** in solving **scheduling problems**, central in supply chain management
- Emphasis on **flexible shop problems**
- Adapting **discrepancy-based search methods** for the problems under study
- **Experimental evaluation** of the propositions

Scheduling Problems under study

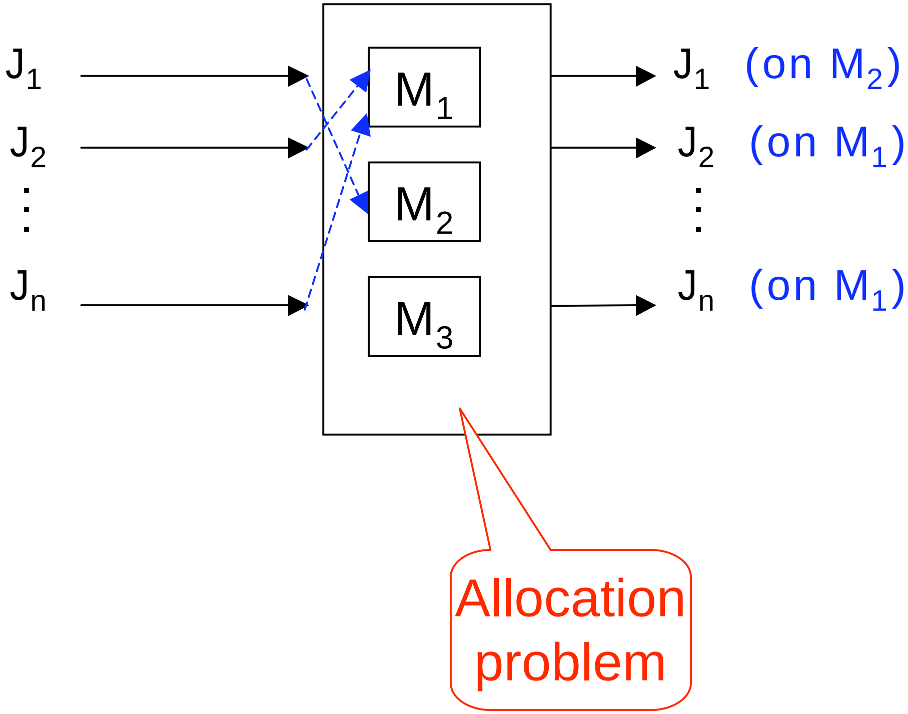
- Disjunctive scheduling
  - Resources = machines
  - One operation at a time on a machine
  - One machine at a time by operation
  - No preemption
    - Shop problems
      - Execution route (routing) = sequence of machines to follow to manufacture a product
      - Job = sequence of operations following a given route
- Focus on *flexible* shop problems
  - Resource assignment is **not** decided *a priori*
    - Parallel Machine
    - Hybrid Flow Shop (HFS)
    - Flexible Job Shop (FJS)

Parallel machine



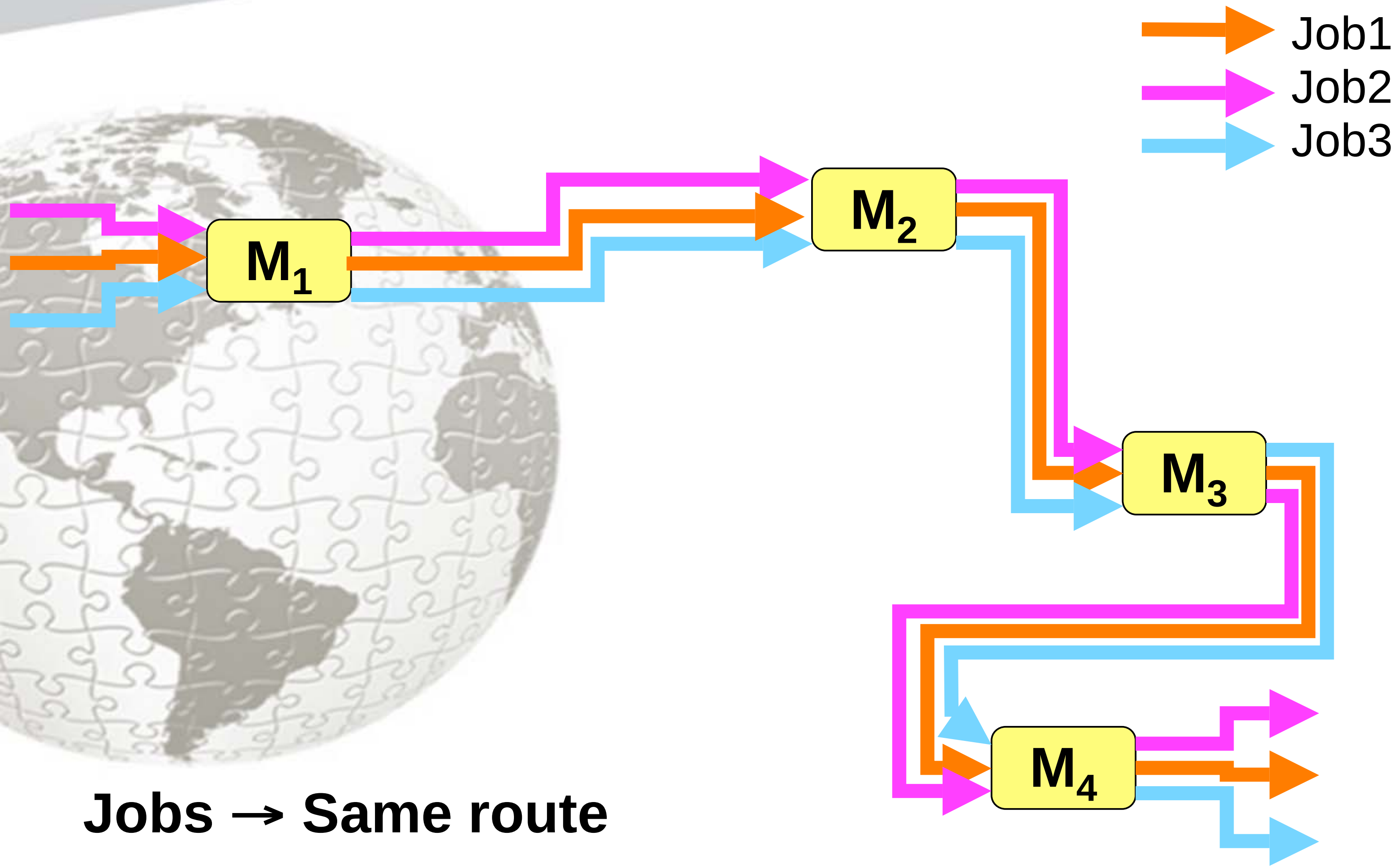
Job ≡ Operation

Parallel machine



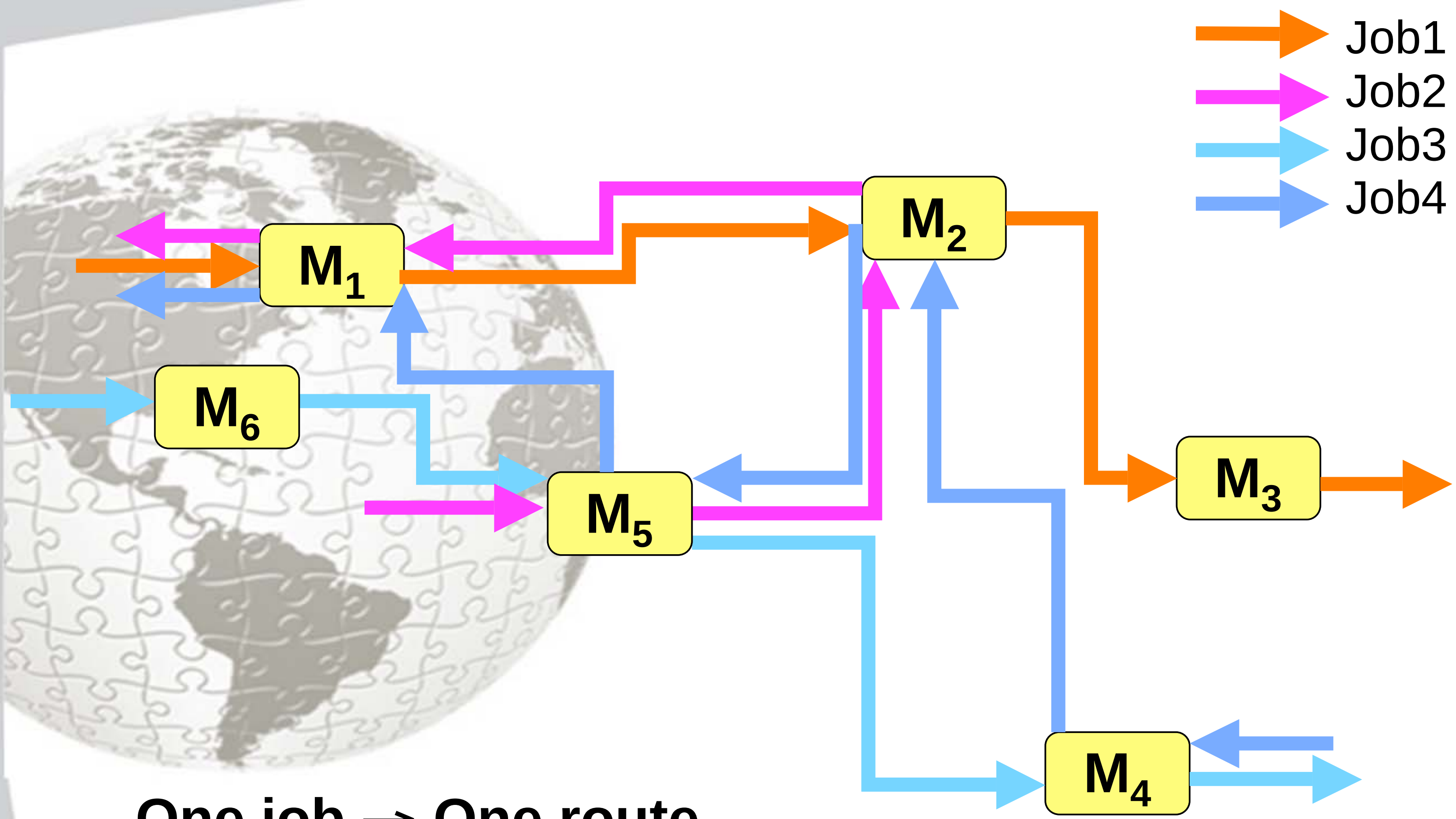
Job ≡ Operation

Shop problems: Flow Shop



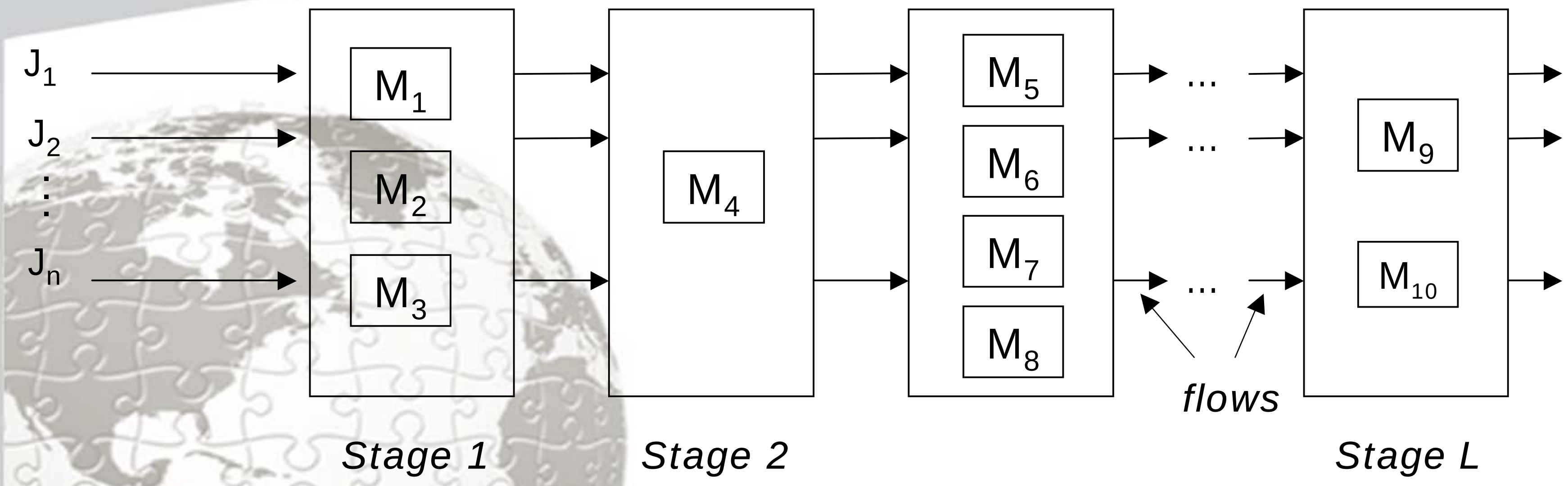
Jobs → Same route

Shop problems: Job Shop



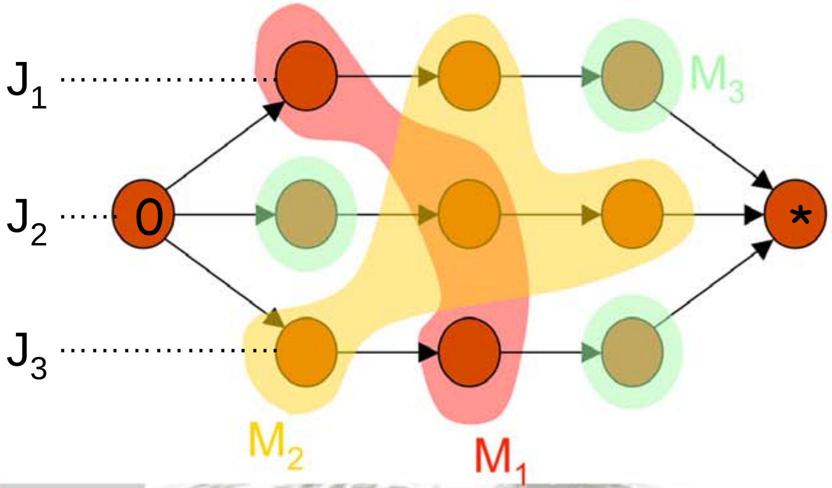
One job → One route

Shop problems: Hybrid Flow Shop



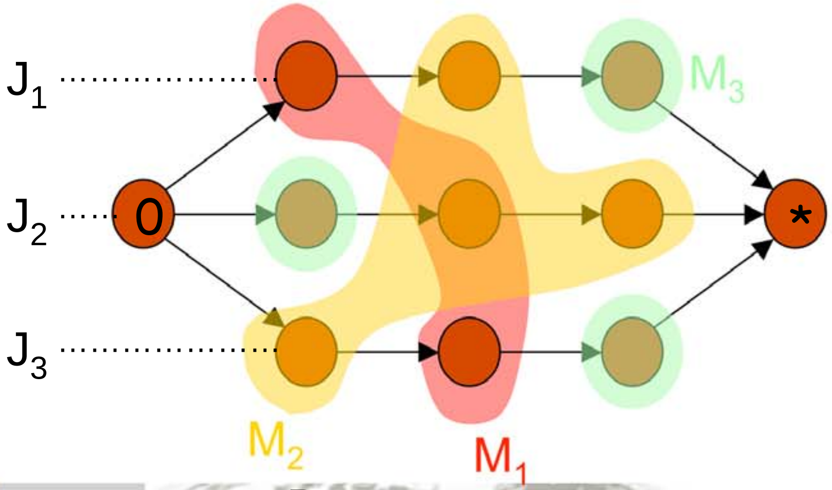
- $J=\{J_1,\ldots,J_i,\ldots,J_n\}$ jobs
- $E=\{1,\ldots,s,\ldots, L\}$ stages
- $M^{(s)}$: identical parallel machines / max $(M^{(s)})>1$
- Application: semiconductors

  (Printed Circuit Boards)



Shop pbs: Flexible Job Shop

- As in a Job Shop:
  - Resources = machines: $M=\{M_1,\ldots,M_m\}$
  - Job J_i= sequence of operations $O_{i1},\ldots,O_{im}$



Shop pbs: Flexible Job Shop

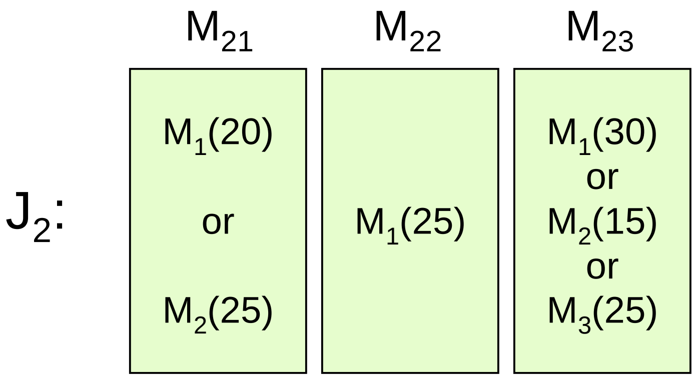
- As in a Job Shop:
  - Resources = machines: $M=\{M_1,\ldots,M_m\}$
  - Job J_i= sequence of operations $O_{i1},\ldots,O_{im}$
- But:
  - Alternative (unrelated) machines can process an operation
    - O_{is} on any machine among $M_{is} \subseteq M$;
    - $\forall i, \cap M_{is}$ may be non-empty ("recirculation")

Shop pbs: Flexible Job Shop

	M_{21}	M_{22}	M_{23}
J_2:	$M_1(20)$ or $M_2(25)$	$M_1(25)$	$M_1(30)$ or $M_2(15)$ or $M_3(25)$

- As in a Job Shop:
  - Resources = machines: $M=\{M_1,\ldots,M_m\}$
  - Job J_i= sequence of operations $O_{i1},\ldots,O_{im}$
- But:
  - Alternative (unrelated) machines can process an operation
    - O_{is} on any machine among $M_{is} \subseteq M$;
    - $\forall i, \cap M_{is}$ may be non-empty ("recirculation")

Shop pbs: Flexible Job Shop



- As in a Job Shop:
  - Resources = machines: $M=\{M_1,\dots,M_m\}$
  - Job J_i= sequence of operations $O_{i1},\dots,O_{im}$
- But:
  - Alternative (unrelated) machines can process an operation
    - O_{is} on any machine among $M_{is} \subseteq M$;
    - $\forall i, \cap M_{is}$ may be non-empty ("recirculation")
- Application: semiconductor industry (wafer fabrication)
- Double problem:
  - Select a machine for each operation
  - Determine a start time for each operation

min C_{max}

Solving scheduling problems

- **Exact methods**
  - dynamic programming
  - integer programming
  - tree search
- **Heuristics**
  - dispatching rules
  - greedy algorithms
- **Metaheuristics**
  - Tabu search
  - genetic algorithms
  - ant colony optimization
- **Constraint programming**

Universidad de los Andes

Solving scheduling problems

- **Exact methods**
  - dynamic programming
  - integer programming
  - tree search
- **Heuristics**
  - dispatching rules
  - greedy algorithms
- **Metaheuristics**
  - Tabu search
  - genetic algorithms
  - ant colony optimization
- **Constraint programming**

Background

- *NP-hard* problems [Vaessens, 1995]
- Hybrid Flow Shop (HFS)
  - 2-HFS with $M^{(s)}=2$ for $s \in \{1,2\}$ is NP-hard: [Gupta, 1988]
  - Exact methods: [Brah & Hunsucker, 1991]; [Portmann *et al*., 1992]; [Moursli & Pochet, 2000]; [Carlier & Néron, 2000]; [Lin & Liao, 2003]
  - Lower bounds: [Santos *et al*., 1995]; [Moursli & Pochet, 2000]; [Carlier & Néron, 2000]
  - Heuristics: [Brah & Loo, 1999]; [Engin & Döyen, 2004]
- Flexible Job Shop (FJS)
  - First presented by [Brucker & Schlie, 1990]
  - FJSP is NP-Hard in general [Vaessens, 1995]
  - Greedy and GA algorithms were proposed (many references)
  - Best results obtained by Tabu Search [Mastrolilli & Gambardella, 2000]
  - JMPM | | C_{max} is strongly NP-hard [Brucker, 2004]

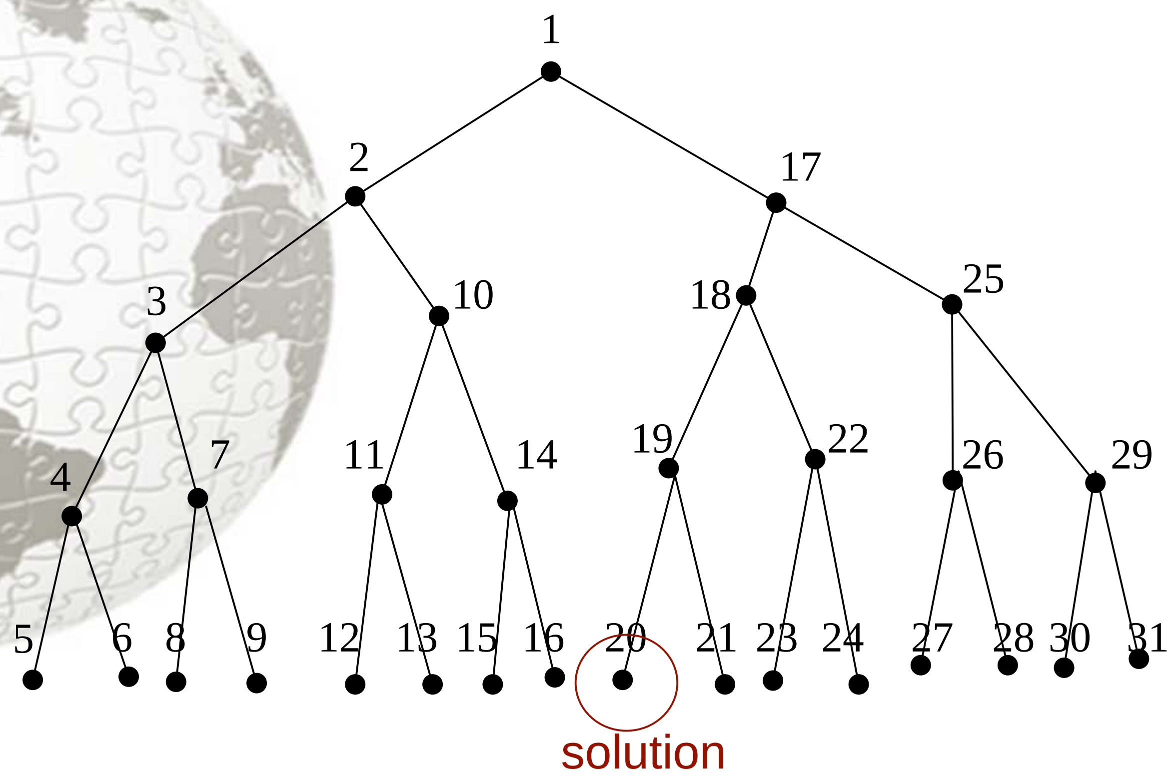
Discrepancy-based search methods (1)

- Limited Discrepancy Search - LDS [Harvey & Ginsberg, 1995]
  - Is a problem satisfiable? → *Satisfaction*
  - Iterative tree search method
  - Instantiation heuristic to guide the search
    (the initial global instantiation is not necessarily a solution)
  - When the heuristic does not find a good solution, it is probably because it made a few poor choices → *discrepancy* then makes a choice different than heuristically top-ranked
  - Hope to find a solution before *Depth-First Search*

Depth First Search (DFS)

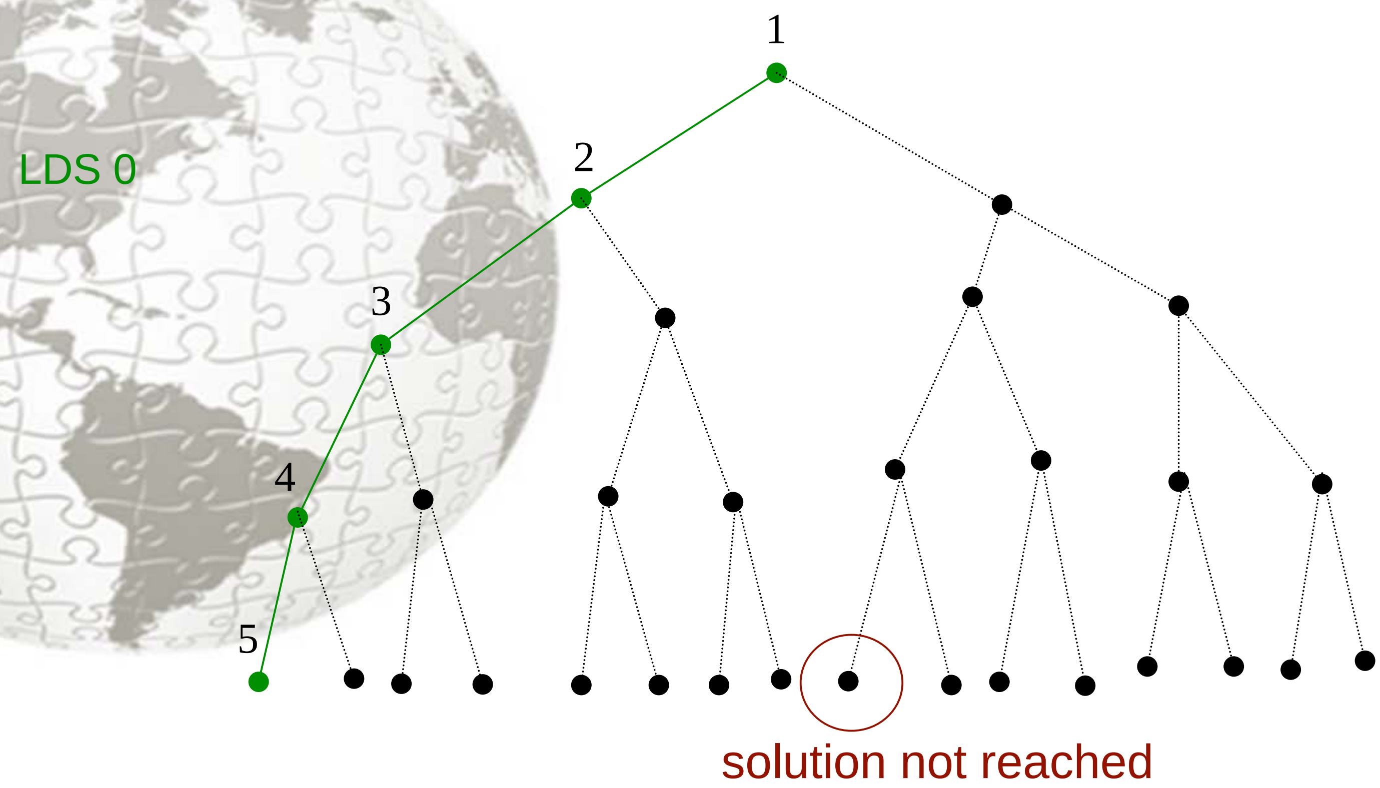
- Search principle
  - Example: binary tree

DFS



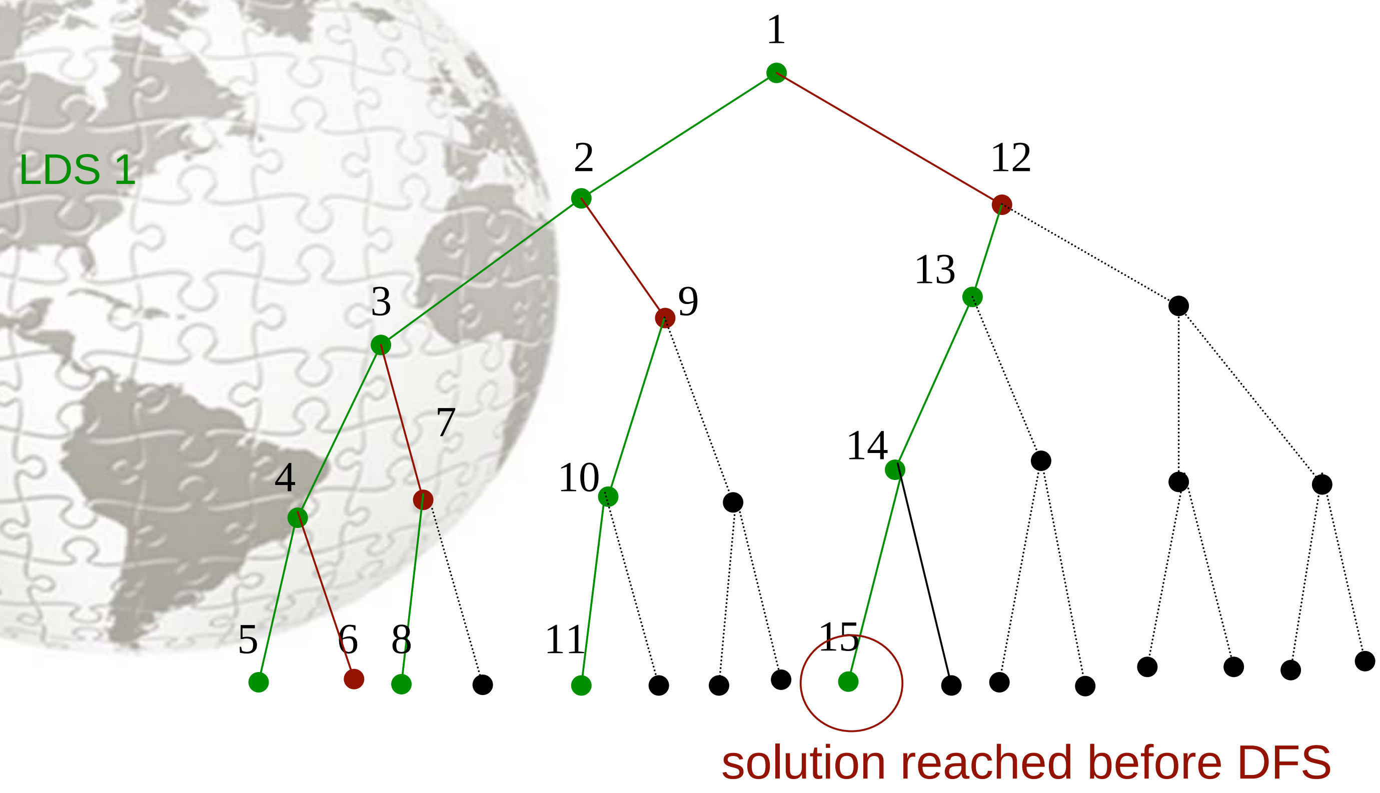
Limited Discrepancy Search (LDS)

- Search principle
  - Example: binary tree
  - LDS 0: The choices of the heuristic are satisfied



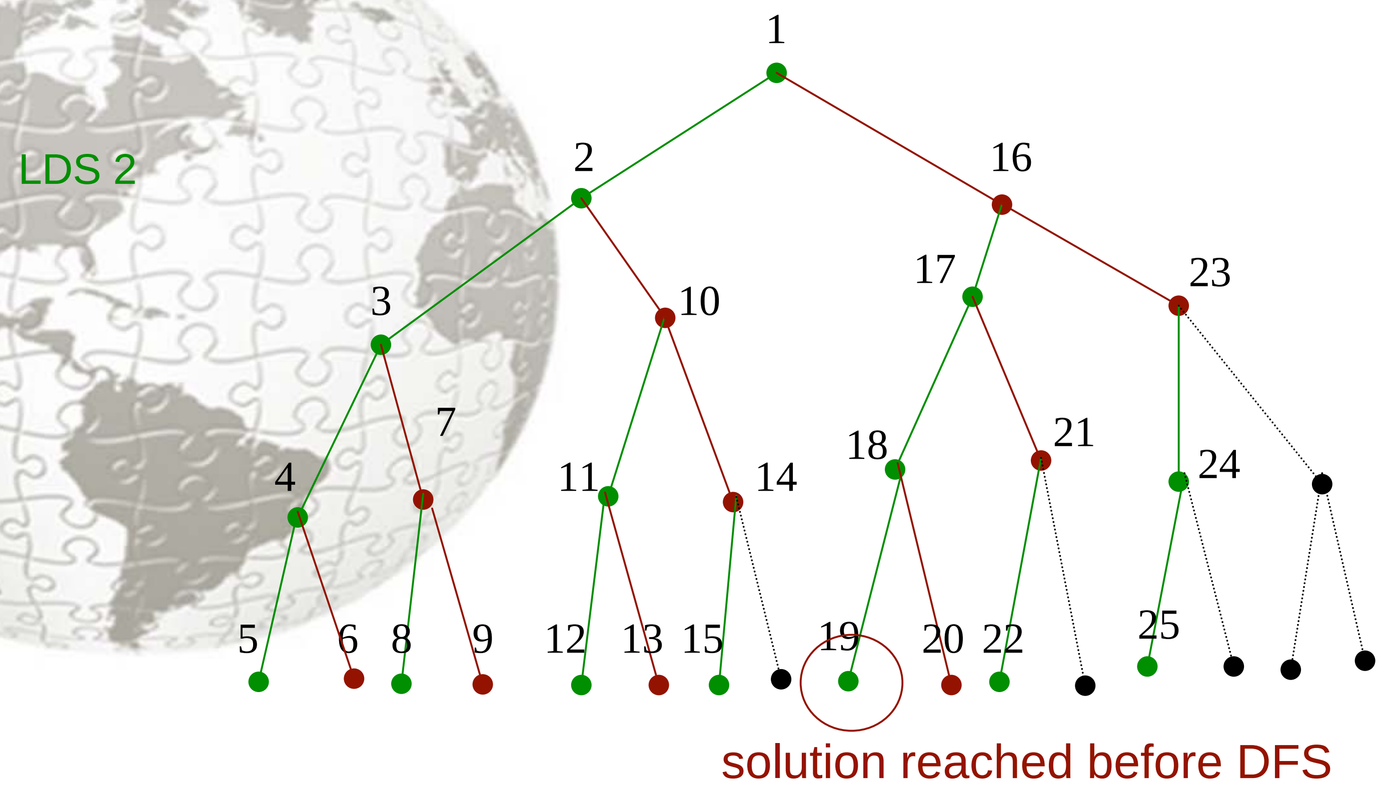
Limited Discrepancy Search (LDS)

- Search principle
  - Example: binary tree
  - LDS 1: All the paths differ of ONE decision from LDS 0



Limited Discrepancy Search (LDS)

- Search principle
  - Example: binary tree
  - LDS 2: All the paths differ of TWO decisions (for TWO variables) from LDS 0

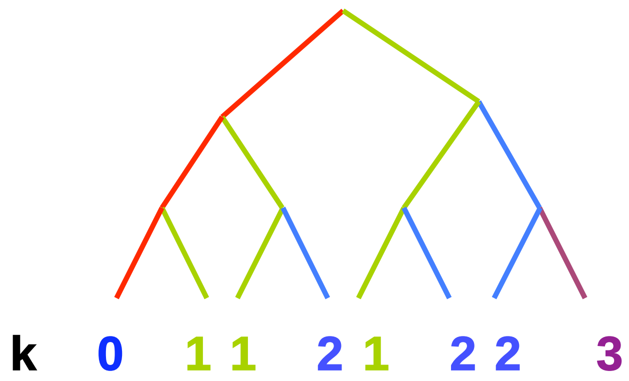


(Improved) Limited Discrepancy Search (I-LDS)

[Harvey & Ginsberg, 1995] and [Korf, 1996]

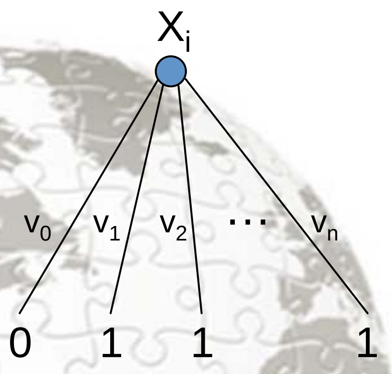
Algorithm

```
k ← 0
kmax ← N
I ← Initial_instantiation()
While no_solution() and (k ≤ kmax) do
    k ← k+1
      -- Generate leaves at discrepancy k from I
      -- Stop when a solution is found
     I ← compute_Leaves (I, k)
End while
```

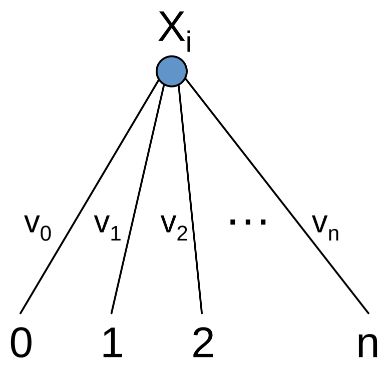


Discrepancy-based search methods (2)

- Non-binary trees: 2 ways for counting discrepancies

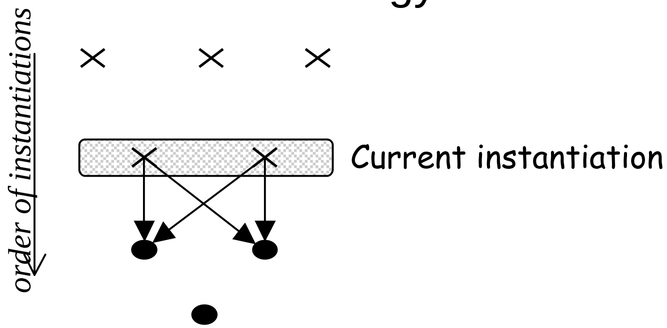


binary counting



non-binary counting

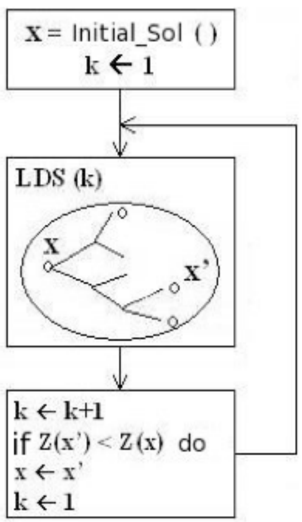
- The binary counting is adopted in our search strategy
- Local propagation by *Forward-Checking*



Prospective scheme

Discrepancy-based search methods (3)

- Depth-bounded Discrepancy Search - DDS [Walsh, 1997]
  - → *Satisfaction*
  - To correct "early mistakes" (the most important)
  - Principle: discrepancies on top of the search tree (given depth)
  - Stop: a solution is found
- Climbing Discrepancy Search - CDS [Milano & Roli, 2005]
  - To improve current solution → *Optimization*
  - Principle:
    - `Initial solution (Reference)`
    - `Apply LDS principle to explore the neighborhood from this reference`
    - `Reference ← Improved_Solution`
    - `Restart with #discrepancy ← 0`
  - Stop: no more improvement, limit on time or #iterations reached
  - CDS is close to VNS [Hansen & Mladenovic, 2001]



Universidad de los Andes

Proposed method: CDDS (1)

- To combine 2 discrepancy-based methods
  - Climbing DS (neighborhood search)
  - Depth-bounded DS (neighborhood restricted at the top of the tree)
  - → *Climbing Depth-bounded Discrepancy Search (CDDS)*

- Optimization method: approximate solutions
  - Criterion = *makespan* minimization
  - A solution = UB

$$\min C_{\max}$$

- LBs to fathom nodes in the search tree

- Example: HFS

$$LB_l(i) = ct_{il} + \sum_{j=l+1}^{k} p_{ij} \quad \text{(where } ct_{il} \text{ is the completion time of } O_{il}\text{)}$$

$$LBM[Sch^{(s)}(Y)] = \begin{cases} ACT[Sch^{(s)}(Y)] + \min_{i \in J-Y} \{ \sum_{s'=s+1}^{L} p_i^{(s')} \} & \text{if } ACT[Sch^{(s)}(Y)] > MCT[Sch^{(s)}(Y)] \\ MCT[Sch^{(s)}(Y)] + \min_{i \in Y} \{ \sum_{s'=s+1}^{L} p_i^{(s')} \} & \text{if } ACT[Sch^{(s)}(Y)] < MCT[Sch^{(s)}(Y)] \\ ACT[Sch^{(s)}(Y)] + \max\{ \min_{i \in J-Y} \sum_{s'=s+1}^{L} p_i^{(s')}, \min_{i \in Y} \sum_{s'=s+1}^{L} p_i^{(s')} \} & \text{if } ACT[Sch^{(s)}(Y)] = MCT[Sch^{(s)}(Y)] \end{cases} \quad (7)$$

Proposed method: CDDS (2)

- Exploration strategy
  - Instantiation heuristics
    1. Job selection: $X_i \in \{O_{11}, O_{12}, \ldots, O_{1s_1}, O_{21}, \ldots, O_{n1}, \ldots, O_{ns_n}\}$
       11. Earliest Start Time (EST)
       12. SPT or EDD or LDJ (job of longest duration)
    2. Machine selection (allocation): $A_i \in \{M_1, \ldots, M_m\}$
       Earliest Completion Time (ECT)
  - Propagation = Forward-Checking over:
    - Start time of subsequent operations
    - Availability date of selected machine
  - Discrepancies
    - On job selection for HFS
    - On both types of variables (job and machine selection) for FJS

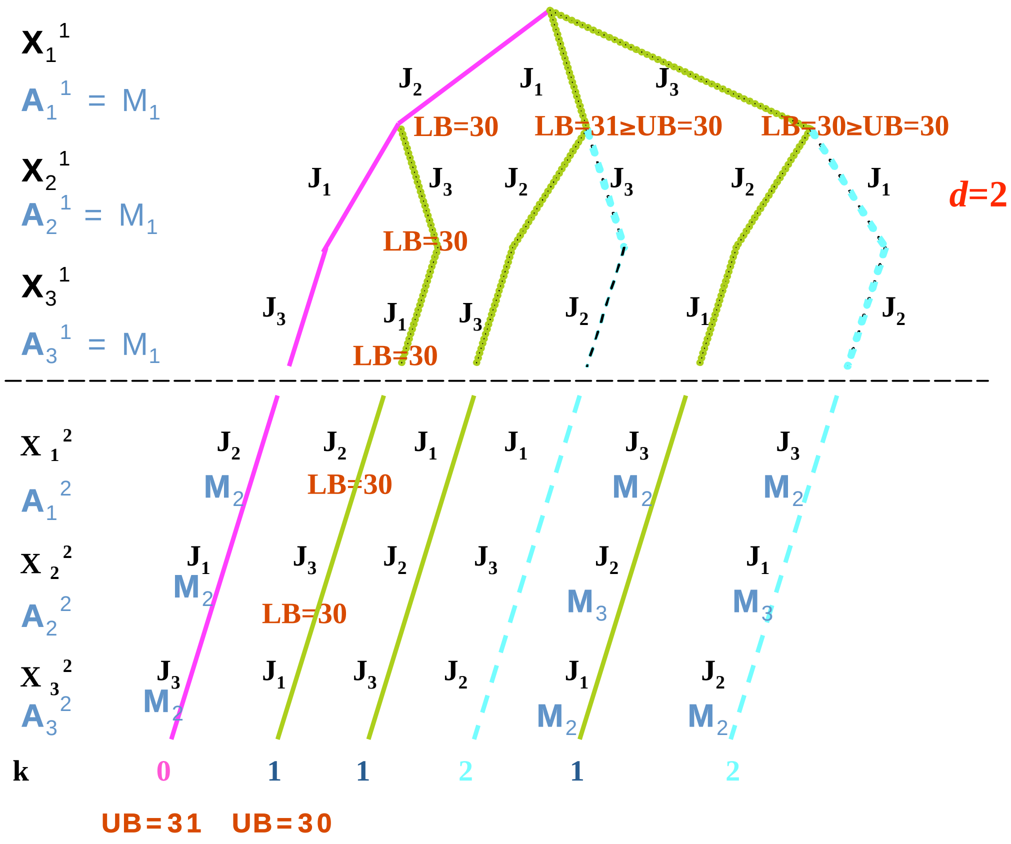
HFS: Example

Processing times of a 3×2 Hybrid Flow Shop

Jobs	*Stage 1*		*Stage 2*	
1	O_{11}	8	O_{12}	7
2	O_{21}	7	O_{22}	8
3	O_{31}	8	O_{32}	8

$M^{(1)} = 1;\ M^{(2)} = 2$

EST-SPT (1st stage):
J_2 ; J_1 ; J_3

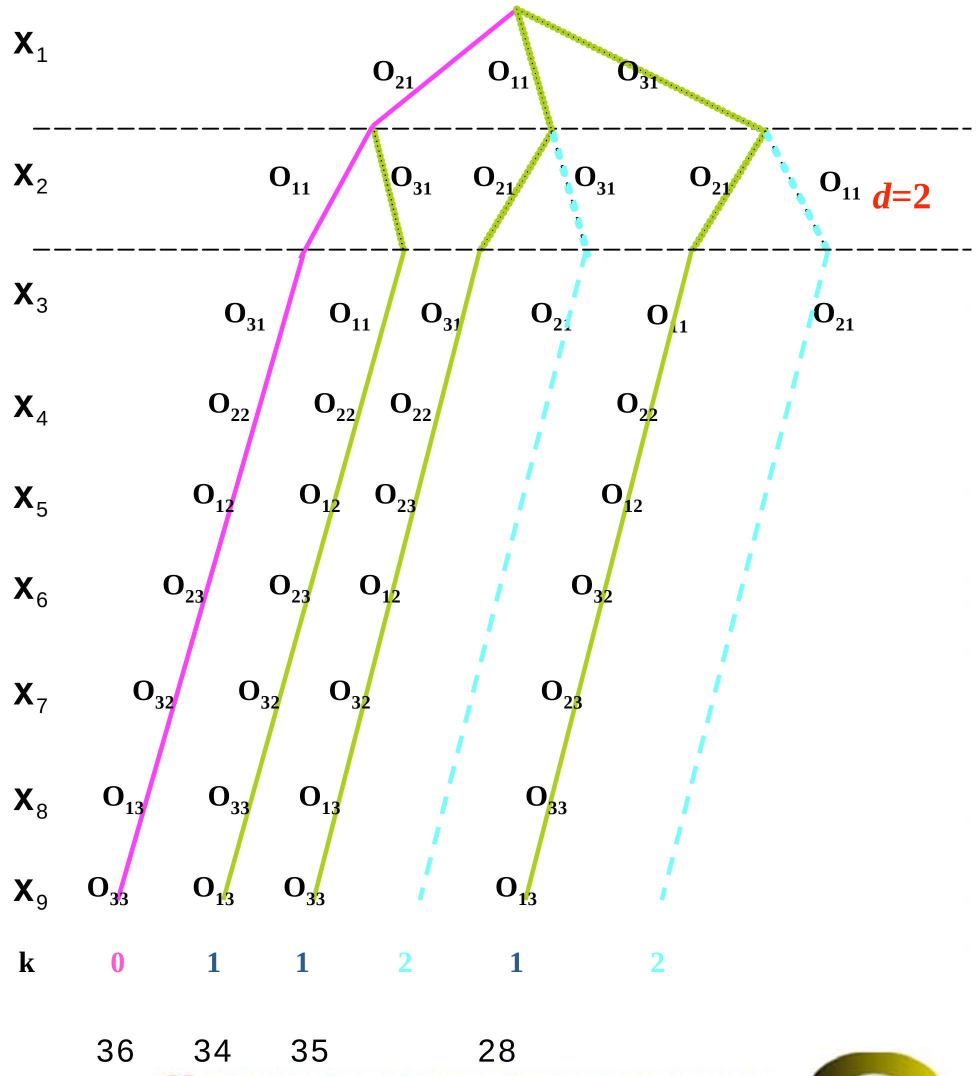


FJS: Example of discrepancy on job selection variable

Processing times of a 3×3 Flexible Job Shop

	M_1	M_2	M_3
O_{11}	--	8	--
O_{12}	--	8	--
O_{13}	--	6	--
O_{21}	11	2	--
O_{22}	--	3	10
O_{23}	11	--	10
O_{31}	12	--	12
O_{32}	4	--	12
O_{33}	12	--	--

EST-SPT: O_{21} ; O_{11} ; O_{31}



HFS: Experiments

- Instances:
  - Néron & Carlier
    - 52 easy problems
    - 24 hard problems
- Comparison:
  - LDJ is the best rule
  - B&B of [Néron & Carlier, 2000]
  - AIS
- Stop:
  - limit on CPU time=30 sec.

Relative performance of methods

Method	*easy*	*hard*	*all*
	% deviation / LB		
B&B	*2.21*	*6.88*	*3.68*
DDS	*1.42*	*8.01*	*3.58*
CDDS	*1.1*	*5.0*	*2.32*
AIS	***1.01***	***3.12***	***1.68***
$CDDS^L$	***0.96***	***3.06***	***1.62***

$$\% \text{ deviation} = \frac{C\max\_best - LowerBound}{LowerBound} \times 100$$

Universidad de los Andes

2 stage-HFS: Experiments

- Instances:
  - Three sets generated in a similar way as [Lee & Vairaktarakis, 1994]
    - Set A: $S_1[1-20]$; $S_2[1-40]$
    - Set B: $S_1[1-40]$; $S_2[1-20]$
    - Set C: $S_1[1-40]$; $S_2[1-40]$

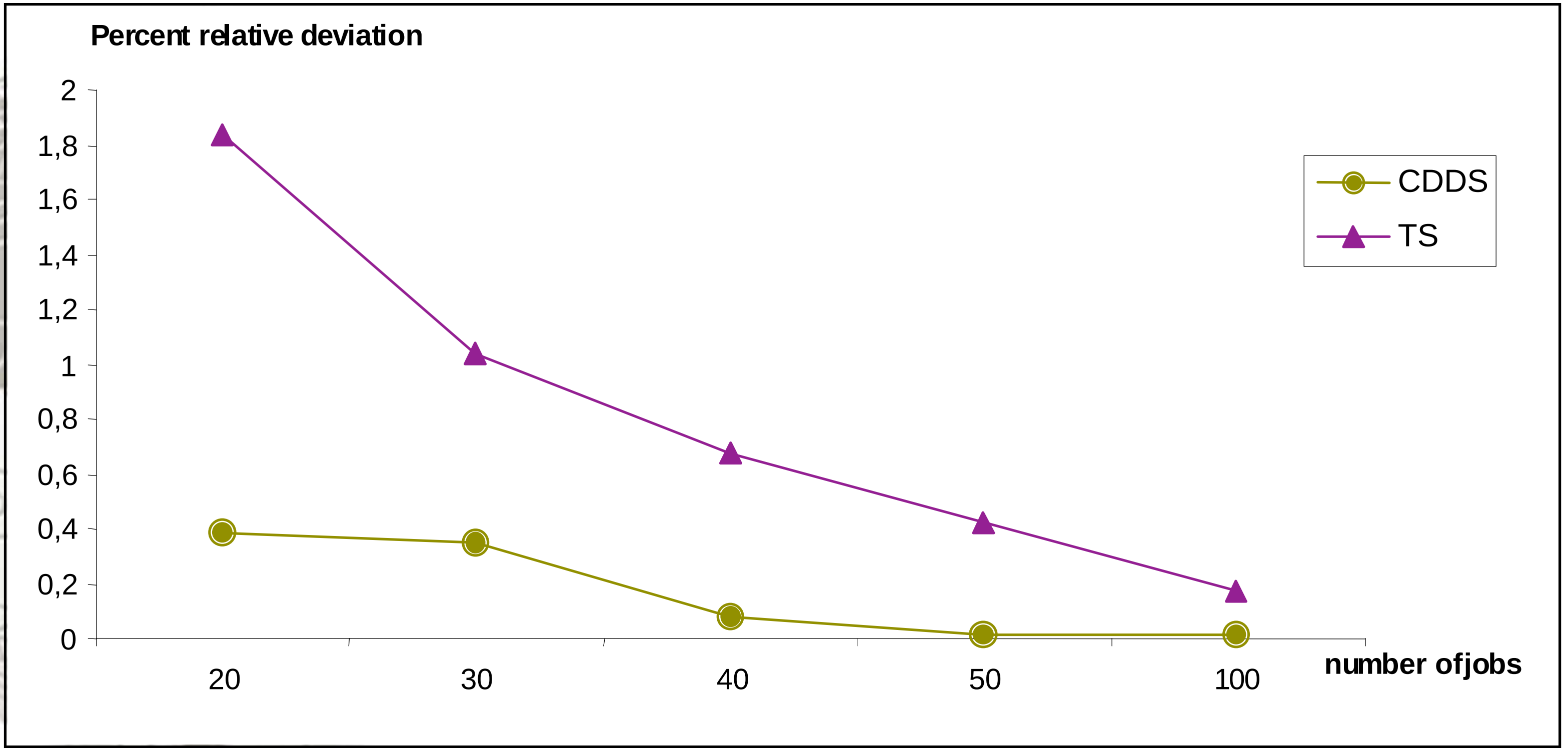
    n={10,20,30,40,50,100,150} : 1680 instances.
- Comparison:
  - LBs [Haouari *et al.*, 2006]
  - TS and LBs [Haouari & M'Hallah, 1997]
- Stop:
  - limit on CPU time=15 sec.

Relative performance of methods

Method	Set A	Set B	Set C
	% deviation / LBs_{2006}		
$CDDS^L$	0.82	0.33	0.34
$CDDS^2$	0.17	0.13	0.22
	% deviation / LBs_{1997}		
TS	0.63	0.97	0.86
$CDDS^2$	0.16	0.12	0.26

$$\% \text{ deviation} = \frac{C\max\_best - LowerBound}{LowerBound} \times 100$$

2 stage-HFS: Experiments



FJS: Experiments

- Instances:
  - Brandimarte's benchmarks
  - 10 problems
  - n=[10 – 20]; m=[4 – 15]; n_i=[5 – 15]
- Comparison:
  - EDD is the best rule
  - Brandimarte's LBs
  - TS of [Mastrolilli & Gambardella, 2000] *(M.G.)*
- Stop: limit on CPU time=30 sec.

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

FJS: Experiments

- Instances:
  - Hurink's benchmarks
  - 129 problems (43 JSP): EData
    RData
    Vdata
    (*Degree of flexibility* ↓)
  - n=[6 – 30]; m=[5 – 15]
- Comparison:
  - EDD is the best rule
  - [Pezella *et al.*, 2007]
  - Tabu + GA + LBs
- Stop:
  - limit on CPU time=30 sec.

- Mean relative error / best_LB):

Problems	***Tabu (%)***	***CDDS (%)***	***GA (%)***
EData	*2.2*	*5.3*	*6.0*
RData	*1.2*	*2.5*	*4.4*
VData	*0.1*	*0.6*	*2.0*

$$\%\ \text{deviation} = \frac{C\max\_best - LowerBound}{LowerBound} \times 100$$

Universidad de los Andes

FJS: Experiments

Deviation percentage over the best known lower bound

Data set	num	alt	CDDS (%)
Brandimarte	10	2.59	17.0
Hurink Edata	43	1.15	5.3
Hurink Rdata	43	2	2.5
Hurink Vdata	43	4.31	0.6

num: number of instances; alt: machine's number per job

Parallel machine scheduling

- Comparisons on $Pm|r_i,q_i|C_{max}$ problems [Néron *et al.*, 2008]
  (50 hard instances; n = 100, m = 10, p_i = [1 − 10])
- Stop: limit on CPU time=30 sec.

$CPU_{limit} = 30\ s$	*Best Solution*	*Best Sol. Strict*	*CPU(s)*
$LDS^{TW}_{z=1}$	1	0	29.64
$LDS^{CHR}_{z=2}$	7	0	28.40
$BS^{TW}_{\omega=3}$	25	3	20.37
$BS^{CHR}_{\omega=4}$	22	0	28.40
CDS	35	6	30 (8.03)
HD-CDDS	38	9	30 (7.02)

Universidad de los Andes

Conclusions

- Novel method to solve Flexible Shop Problems:

  - CDDS: Climbing Depth-bounded Discrepancy Search

    – Hybrid Flow Shop
      → Excellent results - [Ben Hmida *et al.*, 2007; *EJIE*]
      → 2-stages - [Ben Hmida *et al.*, 2009; *JOS, under review*]

    – Flexible Job Shop (results to confirm)

    – Parallel machine (with precedence constraints and setup times, L_{max} ; ΣC_i)
      → Excellent results - [Gacias *et al.*, 2009; *COR, under review*]

Universidad de los Andes

Further works

- FJS:
  - *Backjumping heuristic* on promising choice points for making discrepancies → concept of *Block* neighborhoods [Jurish, 1992]
    1. Permutation of two adjacent critical operations carried out by the same resource (discrepancy on selection variable)
    2. Re-assignment of a critical operation on another resource (discrepancy on allocation variable but restricted to critical operations)
  - → Results improved [Ben Hmida *et al.* 2009, *in preparation*]

instances	#	CDDS (old)	GA	TS	hGA
Brandimarte	10	15.0 (17.0)	17.5	15.1	14.9
Barnes/Chambers	21	22.5 (nil)	29.6	22.5	22.6
Hurink Edata	43	2.3 (5.3)	6.0	2.2	2.1
Hurink Rdata	43	1.3 (2.5)	4.4	1.2	1.2
Hurink Vdata	43	0.1 (0.6)	2.0	0.1	0.08

Limit on CPU time=15 sec.

- Even better results on FJS (adapted lower bounds?)
- Extension to Multimode Resource-Constrained Project Scheduling Problems (MRCPSPs)
- Multicriteria FJS (ΣC_i ; L_{max} – [Vilcot & Billaut, 2007])