

Flowshop hybride : de nouvelles perspectives en mêlant algorithmes génétiques et propagation de contraintes

Antoine Jouglet¹ Ceyda Oğuz² Marc Sevaux³

¹Université de Technologie de Compiègne, France
antoine.jouglet@hds.utc.fr

²Koç University, Turkey
coguz@ku.edu.tr

³Université de Valenciennes, France
marc.sevaux@univ-valenciennes.fr

ROADEF
14-16 Février 2005
Tours, France

Hybrid Flowshop Problem

Flowshop hybride

A. Jouglet
C. Oğuz
M. Sevaux

$J = \{1, 2, \dots, n\}$ independent jobs

$K = \{1, 2, \dots, k\}$ stages

$M_i = \{1, 2, \dots, m_i\}$ set of identical processors

A job is a sequence of k tasks
one task (T_{ij}) for each stage.

Each task, within a job, requires one or several processors simultaneously.

$size_{ij}$: # of processors required for the i -th task of job j .

p_{ij} : processing time of the i -th task of job j .

Objective: minimise the makespan C_{\max}

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

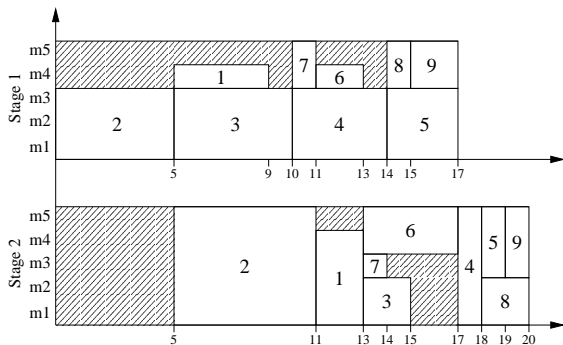
Memetic Algorithm

Computational
Experiments

Conclusion

Example

Jobs j	1	2	3	4	5	6	7	8	9
p_{1j}	4	5	5	4	3	2	1	1	2
$size_{1j}$	1	3	3	3	3	1	2	2	2
p_{2j}	2	6	2	1	1	4	1	2	1
$size_{2j}$	4	5	2	5	3	2	1	2	3



Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion

Encoding: Each individual is coded as a permutation representing the sequence of jobs at the first stage

Decoding: Use a generalisation of a list algorithm

- ▶ Construct the schedule at the first stage by iteratively assigning the jobs to the processors according to the permutation order
- ▶ Create a new list on the basis of the job's completion time from the current stage (non-decreasing)
- ▶ Construct the schedule of the next stage by iteratively assigning the jobs to the processors according to the new list
- ▶ Repeat the previous two items for every stage

Initial population size: $pop_size = n$

Four priority rules (Oğuz *et al.*, 2003):

1. Sort tasks in non-increasing order of last stage's processing times multiplied by last state's processor requirements
2. Sort tasks in non-decreasing order of stage 1 processing times multiplied by stage 1 processor requirements
3. Sort tasks in non-decreasing order of stage 1 processor requirements
4. Sort tasks in non-decreasing order of last stage's processor requirements

The rest of the population is generated randomly

Stopping criterion: Time limit = 900 seconds

Fitness: C_{\max} value of the corresponding schedule

Selection: by binary tournament
(biased random selection)

- ▶ Select two individuals at randoms from the population and keep the best one as the first parent
- ▶ do the same for the second parent

Crossover: NXO operator (Oğuz & Ercan, 2005)

Keep the best characteristics of the parents in terms of the neighboring jobs

- ▶ if two jobs are adjacent to each other in both parents, which are having good fitness values, then we want to keep this structure in the offspring
- ▶ if this is not possible, then we choose the next jobs which fit well in terms of the processor allocations

Mutation: Insertion operator

- ▶ one of the jobs is selected at random and then inserted at a random position
(preserves the structure of several consecutive jobs)

CP-based B&B algorithm characteristics

$start(T_{ij})$: start time of task T_{ij}

$end(T_{ij})$: end time of task T_{ij}

Temporal relations between tasks:

$$end(T_{ij}) \leq start(T_{i+1,j})$$

propagation: arc-B-consistency algorithm (Lhomme, 93)

Cumulative resource constraints:

$$\sum_{\substack{j \in J \\ \exists start(T_{ij}) \leq t \leq end(T_{ij})}} size_{ij} \leq m_i, \quad i = 1, 2, \dots, k; \quad \forall t$$

propagated through explicit data structure (Le Pape, 94)

- ▶ from resources to tasks to update time bounds of tasks according to the availability of resources
- ▶ from tasks to resources to update the resource utilisation and resource availability according to the time bounds of tasks

Objective: makespan criterion

$$C_{\max} = \max_{i,j} (end(T_{ij}))$$

Decision problem:

- ▶ New constraint: $C_{\max} \leq UB$
- ▶ Each time a feasible solution is found
Update UB and decrease UB by one unit
- ▶ Stop when no feasible solutions are found

Improve the algorithm:

- ▶ At each node \rightarrow lower bound (Oğuz & Ercan, 2005)
- ▶ Disjunctive constraints $[size_{ij} + size_{ik} \leq m_i] \vee [end(T_{ij}) \leq start(T_{ik})] \vee [end(T_{ik}) \leq start(T_{ij})]$
- ▶ edge-finding propagation (task vs set of tasks)

CP-B&B algorithm as a local search operator

Application: the CP Search is started with a fixed probability p_{hs} .

Priority sequence: the permutation (representing the sequence at the first stage) is given by the GA to the B&B algorithm as a priority rule for the branching strategy.

Improvement: the new C_{\max} value found by CP Search is returned to the GA as the new fitness value of the solution. When no better solutions are found the solution is left unchanged.

Time limit: one second + an additional second each time the solution is improved.

The proposed memetic algorithm

Flowshop hybride

A. Jouglet
C. Oğuz
M. Sevaux

Algorithm 1: MA

input: Population size n_{pop} , mutation rate p_m , hybrid search rate p_{hs}
Generate an initial population P of n_{pop} chromosomes
Identify best (b) and worst (w) chromosomes
while *Stopping conditions are not met* **do**
 Selection: p_1 and p_2 from P by binary tournament
 Crossover: $p_1 \otimes p_2 \rightarrow c$ //apply NXO crossover operator
 if $f(c) \geq f(b)$ **then**
 | *Mutation: mutate c under probability p_m*
 endif
 CP Search: apply the CP Search to c with probability p_{hs}
 if $f(c) \geq f(w)$ **then**
 | *Discard c*
 else
 | *Select r by reverse binary tournament*
 | *c replaces r in P*
 endif
endw

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion

300 instances from (Oğuz & Ercan, 2005)

Type 1: # of processors for each stage $\in [1, 5]$

Type 2: # of processors for each stage = 5

of stages: $\{2, 5, 8\}$

of jobs: $\{5, 10, 20, 50, 100\}$

Four algorithms compared: GA_OE, GA, CP, MA.

Parameters of the algorithms

Parameter	value	Algorithm
p_m	0.1	GA, MA
p_{hs}	0.0001	MA
CPU_Max	900s	GA, CP, MA
MaxIterImp	10 000	GA, MA
CPU_HS	1s+1s+...	MA

Comparing the two GAs

k	n	Type 1 instances				Type 2 instances			
		Av. C_{\max}		# best		Av. C_{\max}		# best	
		GA_OE	GA	GA_OE	GA	GA_OE	GA	GA_OE	GA
2	5	267.60	267.60	10	10	256.40	256.40	10	10
	10	451.10	451.10	10	10	409.50	426.30	9	4
	20	876.70	876.50	9	9	757.40	809.50	10	5
	50	2049.40	2048.50	6	8	1671.10	1731.80	9	4
	100	4355.00	4351.50	4	10	3205.30	3242.50	9	1
5	5	472.10	472.10	10	10	423.80	423.80	10	10
	10	639.10	648.40	8	4	616.30	606.80	8	3
	20	1072.10	1077.70	9	4	948.50	950.90	8	2
	50	2604.00	2574.80	4	8	2074.60	1971.70	2	8
	100	4755.00	4755.90	4	6	4145.30	3822.30	0	10
8	5	641.60	641.60	10	10	614.20	614.20	10	10
	10	836.90	850.50	9	2	813.10	840.40	6	3
	20	1319.30	1319.90	6	4	1148.00	1144.70	6	4
	50	2669.60	2634.30	2	8	2321.80	2292.20	5	5
	100	5327.70	5260.20	1	9	4216.60	4412.10	9	1

The two methods are almost equivalent:

GA provides better results for Type 1 instances

GA_OE provides better results for Type 2 instances

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion

Comparing C_{\max} average values

k	n	Type 1 instances			Type 2 instances		
		GA	CP	MA	GA	CP	MA
2	5	267.6	267.0	267.0	256.4	253.5	253.5
	10	451.1	451.1	451.1	426.3	422.0	422.1
	20	876.5	889.8	877.7	809.5	832.5	807.6
	50	2048.5	2087.0	2046.1	1731.8	1794.1	1722.2
	100	4351.5	4417.7	4348.6	3242.5	3325.9	3215.0
5	5	472.1	466.6	466.6	423.8	418.4	418.4
	10	648.4	637.8	637.8	606.8	596.8	594.9
	20	1077.7	1151.0	1070.3	950.9	1066.5	945.3
	50	2574.8	2680.7	2571.7	1971.7	2134.1	1960.7
	100	4755.9	4868.5	4746.9	3822.3	3965.2	3802.0
8	5	641.6	616.7	616.7	614.2	599.9	599.9
	10	850.5	854.0	840.3	840.4	820.9	824.0
	20	1319.9	1468.2	1315.2	1144.7	1256.2	1133.4
	50	2634.3	2950.2	2623.1	2292.2	2569.2	2315.7
	100	5260.2	5643.9	5267.2	4412.1	4525.8	4330.7

Small size \rightarrow CP finds optimal solutions

CP and MA provide best results for small size

of best solution found \rightarrow same conclusion

Problem definition

Genetic Algorithm

 Constraint-Based
Scheduling

Memetic Algorithm

 Computational
Experiments

Conclusion

Deviation of C_{\max} value from the optimal value or from the lower bound (in %)

Flowshop hybride

A. Jouglet
C. Oğuz
M. Sevaux

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion

k	n	<i>Type 1 instances</i>			<i>Type 2 instances</i>		
		GA	CP	MA	GA	CP	MA
2	5	0.29	0.00	0.00	1.23	0.00	0.00
	10	0.00	0.00	0.00	10.45	9.33	9.36
	20	0.44	2.59	0.66	9.63	12.90	9.34
	50	0.63	2.79	0.49	5.33	9.29	4.70
	100	0.15	1.96	0.07	2.67	5.30	1.77
5	5	1.35	0.00	0.00	1.44	0.00	0.00
	10	1.64	0.00	0.00	3.71	1.92	1.60
	20	3.49	10.85	2.78	7.83	20.78	7.15
	50	0.59	5.30	0.51	4.90	13.61	4.35
	100	2.50	5.19	2.33	10.67	14.89	10.12
8	5	4.15	0.00	0.00	2.38	0.00	0.00
	10	9.38	10.32	8.02	9.32	6.80	7.24
	20	5.69	17.98	5.32	17.26	28.52	16.02
	50	2.17	14.42	1.71	15.62	29.62	16.87
	100	2.02	9.49	2.18	18.85	21.97	16.64

Very good performance of MA

Comparing CPU times (in seconds)

Flowshop hybride

A. Jouglet
C. Oğuz
M. Sevaux

		<i>Type 1 instances</i>			<i>Type 2 instances</i>		
<i>k</i>	<i>n</i>	GA	CP	MA	GA	CP	MA
2	5	598.8	0.03	0.8	586.3	0.02	0.7
	10	900	0.05	0.9	900	91.19	763.1
	20	900	450.54	450.4	900	648.35	541.2
	50	900	540.69	454.9	900	725.09	629.8
	100	900	451.62	458.8	900	900	900
5	5	900	0.04	1.1	900	0.04	1.7
	10	900	60.55	541.6	900	411.13	843.6
	20	900	900	634.6	900	900	900
	50	900	721.31	467.8	900	900	900
	100	900	812.03	551.7	900	900	900
8	5	900	0.11	2.2	900	0.06	1.2
	10	900	386.8	720.4	900	560.7	900
	20	900	720.88	721	900	900	900
	50	900	900	816.1	900	900	900
	100	900	900	865.1	900	900	900

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion

MA much more efficient than GA and CP

Presented here:

- ▶ A GA based on previous components
- ▶ A new CP-based B&B algorithm for HFS (first exact method for this problem)
- ▶ An efficient combination in a memetic algorithm

Results obtained:

- ▶ GA equivalent to the previous method
- ▶ CP solves efficiently small size instances
- ▶ Combination in a memetic algorithm
→ necessary to obtain very good results

Problem definition

Genetic Algorithm

Constraint-Based
Scheduling

Memetic Algorithm

Computational
Experiments

Conclusion