

# A new algorithm for multidimensional scheduling problems <sup>\*</sup>

Thijs Urlings (Speaker) <sup>†</sup>

Rubén Ruiz <sup>‡</sup>

---

## 1 Introduction

In this work we consider the hybrid flexible flowline (HFFL) problem with a set of additional constraints that apply in real-world industrial environments. A set of  $n$  jobs has to be scheduled on a set of  $m$  ordered stages. All jobs are available at time 0, and job preemption is not allowed. Each stage  $i$  consists in  $m_i$  parallel unrelated machines. Jobs might skip stages; we denote the stages visited by job  $j$  as the set  $F_j$ . At each stage  $i$  in  $F_j$ , job  $j$  should be processed by exactly one machine in the set of eligible machines  $E_{ij}$ .

Precedence constraints among jobs refrain job  $j$  from starting in the first stage before ending the process of its predecessor jobs  $P_j$  in the last stage. Setup times  $S_{iljk}$  depend on both the previous job  $j$  and the next job  $k$ , and on the stage  $i$  and machine  $l$  where the setup is executed. These times can be anticipatory or non-anticipatory. Time lags  $lag_{ilj}$  between finishing a job  $j$  at stage  $i$  and starting the job at the next stage, can be positive or negative and depend on the machine  $l$  that job  $j$  is assigned to at stage  $i$ . Machines release dates are given by the input parameter  $rel_{ij}$ . The goal is to find a schedule that minimises the makespan, that is, the maximum job completion time.

The gap between HFFL theory and scheduling practice is named in two reviews on HFFL problems [1, 2]. For a more recent review, see [3].

## 2 Solution representations

Distinct solution representations can be applied for the HFFL problem. First of all, we exclude all non semi-active solutions: the solutions for which a job can be finished earlier without changing the order of processing on any of the machines [4]. For the makespan objective the optimal solution is guaranteed not to be semi-active.

In order to represent all semi-active solutions, for each machine the assigned jobs should be

---

<sup>\*</sup>This work is partially funded by the Spanish Ministry of Science and Innovation, under the project SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theoretical Advances with reference DPI2008-03511/DPI, and partially funded by the Polytechnic University of Valencia, under the project PPAR - Production Programming in Highly Constrained Environments: New Algorithms and Computational Advances with reference 3147.

<sup>†</sup>[thijs@iti.upv.es](mailto:thijs@iti.upv.es). Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain.

<sup>‡</sup>[rruiz@eio.upv.es](mailto:r Ruiz@eio.upv.es). Instituto Tecnológico de Informática, Departamento de Estadística e Investigación Operativa Aplicadas y Calidad, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain.

given in the processing order. This represents the problem in its full dimension: both the assignments and the permutations have to be decided. The disadvantage of this full solution representation, is the huge size of the solution space and the relatively long time needed to calculate makespan [6].

A more compact solution representation helps to thoroughly and efficiently scan a part of the solution space. To do so, we introduce a single job permutation, that gives the order in which the jobs are launched in the HFFL. Each time a job arrives at a stage, a machine assignment rule is used to assign the job to one of the eligible machines. We call the machine assignment rule we apply earliest preparation next stage (EPNS), which means that job  $j$  is assigned to the machine with the lowest sum of completion time for job  $j$  and time lag towards next stage. This reduces the size of the search space to  $n!$ .

The reduction of the search space, however, implies the risk to exclude the optimal solution or even an entire set of good solutions. This is the main motivation for an algorithm that uses both solution representations.

### 3 The proposed algorithm

The algorithm we propose with the name shifting representation iterated search (SRIS), starts with the compact solution representation. An initial solution is created applying an adaptation of the NEH heuristic [7]. We apply an iterated greedy (IG) algorithm to the solution: iteratively we perform a local search, followed by exclusion and greedy insertion of a number of randomly chosen jobs. When  $1/2 \cdot t \cdot n \cdot \sum_{i=1}^m m_i$  milliseconds of CPU time have passed ( $t$  an input parameter to control the running time), the best found solution is transferred to the full solution representation. Another  $1/2 \cdot t \cdot n \cdot \sum_{i=1}^m m_i$  milliseconds of CPU time are used for iterated local search (ILS) in the full solution space: iteratively we perform a local search, followed by a number of random mutations.

Both in IG and ILS, acceptance of the solution obtained in each iteration, depends on the simulated annealing (SA) aspiration criterion: if the new solution is better than the previous solution it is accepted directly, otherwise it is accepted with a probability given by  $e^{(C_{max}^* - C_{max})/temp}$ , where  $C_{max}^*$  is the previous makespan value,  $C_{max}$  the new makespan and  $temp$  an algorithm parameter that influences on how easily worse solutions are permitted.

### 4 Experimental results

We use an algorithm calibration in order to fix the parameters. The number of jobs that is excluded and inserted in each iteration of the compact representation phase, is fixed to four. The number of mutations done in each iteration of the full representation phase, is fixed to two.  $temp$  is fixed at 0.01 and 0.001 for the compact and the full representation phase respectively. The SRIS algorithm is compared to several algorithms presented in earlier work: an iterated greedy algorithm, iterated local search, a memetic algorithm and a steady-state genetic algorithm [5]. The code is compiled with Delphi 2007 in Windows XP on a computer with one 3.0GHz processor and 1 Gbyte internal memory. The algorithm is tested on a set of 192 instances with 50 to 100 jobs and 8 to 32 machines. The input parameter  $t$  is set to 5, 25 and 125. For each instance, each algorithm and each value of  $t$ , five replicates are done.

Since the optimum solution values are unknown for the instance set, and no tight lowerbounds are available, the results are measured as a percentage over the best found solution value. An ANOVA is used to analyse the statistical significance of the results. The means are given in Table 1, where one can see that the 99% Tukey confidence intervals do not overlap for any of the algorithms.

Table 1: Table of means with 99% confidence intervals

Algorithm	Count	Mean	Standard Error	Lower Limit	Upper Limit
SRIS	2880	5.81	0.052	5.68	5.95
IG	2880	6.84	0.052	6.71	6.98
ILS	2880	8.11	0.052	7.98	8.25
MA	2880	7.25	0.052	7.12	7.39
SGA	2880	7.62	0.052	7.48	7.75

## 5 Conclusion

To the best of our knowledge, the use of two solution representations in one algorithm is new in the field of machine scheduling. We have designed a SRIS algorithm using this novel technique and we compare the new algorithm to the state of the art for the HFFL problem. The SRIS is statistically better than all algorithms in the comparison.

The use of various solution representations in a single algorithm is likely to be interesting for other multidimensional problems as well. In future research we plan to work on multiobjective problems with two dimensions.

## References

- [1] A. VIGNIER, J. C. BILLAUT AND C. PROUST (1999). Hybrid flowshop scheduling problems: State of the art. *Rairo-Recherche Operationnelle-Operations Research*, 33(2):117-183.
- [2] R. LINN AND W. ZHANG (1999). Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering*, 37(1-2):57-61.
- [3] D. QUADT AND H. KUHN (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178(3):686-698.
- [4] M. PINEDO (2008). Scheduling: Theory, Algorithms and Systems. *Springer*, Third Edition.
- [5] T. URLINGS AND R. RUIZ (2007). Local Search in Complex Scheduling Problems. *SLS 2007*, 202-206, [http://dx.doi.org/10.1007/978-3-540-74446-7\\_18](http://dx.doi.org/10.1007/978-3-540-74446-7_18).
- [6] T. URLINGS, R. RUIZ AND F. S. SERIFOGLU (2009). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, Submitted.
- [7] M. NAWAZ, E. E. ENSCORE AND I. HAM (1983). A Heuristic Algorithm for the M-Machine, N-Job Flowshop Sequencing Problem. *Omega-International Journal of Management Science*, 11(1):91-95.