Comparison of priority rules, machine allocation, and stage allocation strategies for hybrid flow shop instances using combinatorial logic and a standard trace format

Dominik Mäckel[1][0000−0002−7866−7927], Benedikt Kordus[2][0009−0006−7432−0332], and Christin Schumacher[3][0000−0002−9847−0443]

[1] TU Dortmund University, Institute of Transport Logistics, Germany
dominik.maeckel@tu-dortmund.de
[2] TU Dortmund University, Institute of Transport Logistics, Germany
bendikt.kordus@tu-dortmund.de
[3] TU Dortmund University, Institute of Transport Logistics, Germany
christin.schumacher@tu-dortmund.de

Abstract. The configuration of heuristics strongly affects the resulting objective function values for given problem instances. Finding the most suitable algorithm components is essential for efficient production scheduling. This paper aims to automatically compare and combine different priority rules, machine allocation, and stage allocation strategies on hybrid flow shop instances with different sets of real-world constraints. To test and tune algorithm configurations and parameters for scheduling problems, an approach with combinatorial logic using the combinatory logic synthesizer is applied. To automatically test scheduling instances, the trace formats in the literature are first reviewed. A trace format is proposed to harmonize the data sets for a large number of beta constraints. In the second step, 32 constructive heuristic configurations are automatically constructed using componentization and recombination with four priority rules, four machine assignment strategies, and two-stage assignment strategies in a general framework of a constructive heuristic. Finally, three sets of test instances with different numbers of jobs by Wittrock [1], Naderi et al. [2], and Ruiz et al. [3] are used to evaluate the resulting constructive heuristics.

Keywords: priority rules · machine allocation · stage allocation · constructive heuristics · makespan · algorithm synthesis

1 Introduction

To meet the diverse planning requirements of different industrial environments and to provide assistance systems for planners, most algorithms for scheduling software are currently built as custom solutions. The selection and composition of heuristics strongly affect the scheduling results and key performance indicators in the use cases of scheduling. To optimize production processes and scheduling efficiency, we need to provide flexible software tools for job scheduling.

The objective of this paper is to develop a framework that combines and compares different priority rules, machine allocation, and stage allocation strategies on hybrid flow shop instances. Therefore, the paper first proposes a unified way of representing instances for hybrid flow shop problems with different beta constraints. Second, constructive heuristics are compared for hybrid flow shops with unrelated machines and the makespan objective. Jobs are allowed to skip stages. The heuristics are generated from each combination of a much smaller collection of rules and strategies using algorithm synthesis. The synthesis

framework for algorithm compositions can be easily extended by adding new strategies that are automatically used in the existing framework. The developed algorithm components are compared by using five data sets. Evaluations show which combination of algorithms performs best on the data sets. First, a literature review of existing datasets for (hybrid) flow shops is conducted. The trace formats used in these data sets are examined and used to create a new and flexible trace format for scheduling. Then, three instances from Wittrock [1], Ruiz et al. [3], and Naderi et al. [2] are converted into the new trace format and are automatically tested using componentization of algorithm components and program synthesis.

2 Literature review

The following table 1 gives an overview of the data sets containing (hybrid) flow shop problems found in the literature. The Graham notation column largely follows the problem notation introduced by Graham et al. [4] with extensions made by Ruiz et al. [3] and Pinedo [5]. Data selected for the evaluation is marked in bold face. The machines here are either all the same (PM) or have different job-specific processing times (RM). Among the beta constraints, jobs are allowed to skip stages ($skip$). Further beta constraints included in the data remain unused. This overview clearly shows an increase in the complexity of

Table 1: Data sets found in the literature

Year	Author(s)	Graham notation	# Instances
1960	Heller [6]	F10 | | C_{max}	2
1978	Carlier [7]	Fm | | C_{max}	8
1988	**Wittrock [1]**	FHm, $((PM^{(k)})_{k=1}^{m})$ | block, lag | C_{max}	6
1993	Taillard [8]	Fm | prmu | C_{max}	120
1995	Reeves [9]	Fm | prmu | C_{max}	126
1998	Demirkol et al. [10]	Fm | | C_{max}; Fm | | L_{max}	40
2001	Néron et al. [11]	FHm, $((PM^{(k)})_{k=1}^{m})$ || C_{max}	271
2008	**Ruiz et al. [3]**	FHm, $((RM^{(k)})_{k=1}^{m})$ | skip, r_m, lag, S_{jk}, M_j, prec | C_{max}	12288
2010	Urlings [12]	FHm, $((RM^{(k)})_{k=1}^{m})$ | | T_{max}, C_{max}	720
2010	**Naderi et al. [2]**	FHm, $((PM^{(k)})_{k=1}^{m})$ | skip, S_{jk} | C_{max}	960
2015	Vallada et al. [13]	Fm | prmu | C_{max}	480
2017	Pan et al. [14]	FHm, $((PM^{(k)})_{k=1}^{m})$ | ddw | T_{max}, E_{max}	3060
2020	Lang et al. [15]	FH2, ($P4^1$, $P5^2$) | fmls, prmu | T_{max}, C_{max}	4
2022	Missaoui et al. [16]	FHm, $((PM^{(k)})_{k=1}^{m})$ | S_{jk}, ddw | T_{max}, E_{max}	9180
2022	Laroque et al. [17]	FHm, $((RM^{(k)})_{k=1}^{m})$ | M_j, batch, prec, skip, fmls | C_{max}	20

the problems, with hybrid flow shops becoming increasingly important, and an increase in the number of beta constraints used. Therefore, the reusability of algorithms and data sets for hybrid flow shops with different constraints would be of great benefit for the comparability of results on newly developed heuristics. CLS is a program synthesis framework that enumerates all possible combinations with the guarantee of completeness. All results of the inhabitation are valid compositions. It has been used in the past for several applications, such as the automated generation of construction plans [18] and algorithm decomposition [19].

3 Trace format

The trace format defines how an instance is stored, such as the file format, the layout of the file, and the rules used to extend this layout in case of new beta constraints that need to be added. Two main types of trace formats for hybrid flow shop problem data sets are found in the literature. In addition, there are unique trace formats used, for example, by Wittrock [1] or Naderi et al. [2].

The first group of trace formats combines a large job table, which contains all the job-dependent information, with smaller auxiliary tables that provide additional details that are not job-dependent. This trace format is used, for example, by Lang et al. [15] or Laroque et al. [17]. The main problem with this type is the job dependency. If we consider beta constraints that depend on two or more jobs, such as sequence-dependent setup times, the job table becomes exponentially large with the number of jobs because each job needs a value for at least every other job.

The second format was already used by Taillard [8] but is still applied by Ruiz et al. [3], Urlings [12], and Missaoui et al. [16]. This type uses separate tables for each type of data, for example, the processing times of jobs on machines are represented in one table, while the values for each beta constraint are stored in separate tables, sometimes even more than one per constraint. The main drawback here, especially on quadratic tables, is the inconsistent usage of rows and columns where the order of reading is undefined. However, the weakness of the latter type can be fixed by introducing the following rules to define the new trace format.

- An instance is a text file where the first line is the number of jobs and the second line is the number of machines in each stage separated by a space.
- The data is given starting with the third line. Each data group starts with a line containing the full data type name, written in 'SHOUTING_SNAKE_CASE', followed by a data line or table.
- Scalar- and one-dimensional data can be written on one line. Data indexing on two dimensions, like jobs and machines for processing times, require tables. In this case, the rows correspond to the first index and the columns correspond to the second index. Three-dimensional data is split at the first index into two-dimensional tables.
- Machine numbers are continuously counted across stages, truncating indexation on the stage-machine-pair into one dimension. Instead of 2 stages with 2 machines each, the table would only consider 4 machines, resulting in a dimension reduction and the ability to display normally four-dimensional data.

Fig. 1: Example data in the trace format, including additional explanation in gray.

```
5                                          Five jobs.
3 3                                        Two stages with 3 machines each.
PROCESSING_TIMES                           Processing times indexed by
74 35 45 0 82                              machine (rows 1-6)
25 22 93 0 54                              and job (column 1-5).
47 89 36 0 95                              Processing time zero
19 75 0 55 0                               reflects stage skipping.
66 93 0 87 0
78 11 0 32 0
SKIP
0 0 0 1 0                                  Skip constraint indexed by
0 0 1 0 1                                  stage (rows 1-2) and job (column 1-5).
```

An example trace of a two-stage hybrid flow shop with five jobs and three machines on both stages from the data set by Ruiz et al. [3] including processing times and skip data converted into the new format is given in Figure 1. Additional explanation is given in gray, which is not part of the trace.

4 Synthesis of scheduling algorithms

The test conducted in this paper uses four different priority rules to determine the initial order of the jobs to be assigned in the first stage:

- *Shortest/Longest processing time first (SPT/LPT)*: Sort jobs in ascending/descending order of average processing time on the first stage.
- *Johnson's rule (JNS)*: Schedules jobs according to their minimum processing times across all stages. If the lowest average processing time across all machines on a stage is found in the first stage, the job is placed at the beginning of the priority queue, otherwise at the end.
- *Random (RND_O)*: Assign jobs in random order.

Four machine assignment strategies are tested to distribute the jobs among the machines:

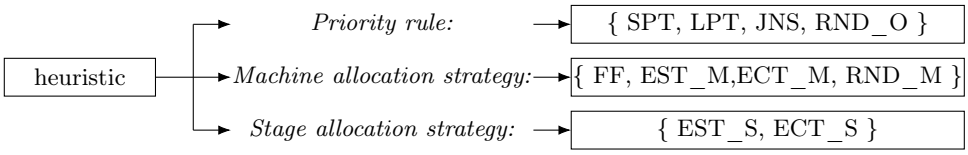
- *FirstFit (FF)*: Assign the job to the first machine with a possible start time earlier than on machine 0.
- *Earliest starting time (EST_M)*: Assign the job to the machine that is available first.
- *Earliest completion time (ECT_M)*: Assign the job to the machine that can finish first.
- *Random (RND_M)*. Assign the job to a random machine.

To assign jobs to machines in the following stages, two assignment strategies are tested:

- *Earliest starting time (EST_S)*: Assign the job that can start on a machine first.
- *Earliest completion time (ECT_S)*: Assign the job that can finish on a machine first.

ECT_S aims at dealing with unrelated machines and sequence-dependent set-up times and reducing the completion time, while EST_S minimizes the machine idle times. The algorithm components are combined using program synthesis by applying the Combinatory Logic Synthesizer (CLS) [20]. CLS is a type-based framework for software synthesis from components declared in a repository. Each component is assigned a unique semantic type as well as an abstract supertype such as "Machine allocation strategy". CLS automatically selects fitting components of such type by answering the inhabitation question $\Gamma \vdash e : \tau$: Does there exist a composition e of the algorithm components in the repository Γ that constructs a scheduling heuristic τ? CLS automatically composes the algorithms to run on the given instances. Figure 2 displays the three needed algorithm components (supertypes) and the sets of available components, of which one has to be selected for each of the 32 possible combinations.

Fig. 2: Possible algorithm components in the repository.



5 Computational evaluation

Using CLS as well as the data by Ruiz et al. [3], Naderi et al. [2] and Wittrock [1] the 32 possible combinations of the described algorithm components are evaluated. On the data sets by Ruiz et al. [3] and Naderi et al. [2], those with four and eight stages and two machines per stage were chosen to match the data set by Wittrock [1] with three stages and three machines per stage best. All data sets are evaluated individually since the data characteristics, for example, processing times, vary between the sets. First, the second-stage allocation strategies are compared. It can be shown by paired confidence intervals that ECT_S dominates EST_S with a confidence level of 95% comparing their ranks for the data set by Ruiz et al. [3] while EST_S dominates ECT_S for the other sets. Among the machine allocation strategies, FirstFit dominates the other strategies and RND_M performs worst on all data sets except for Wittrock's. Here, FirstFit outperforms EST_M and the difference between RND_M and ECT_M was not significant. Among the priority rules, LPT performs worst on all tested instances. Johnson's algorithm dominates LPT, SPT, and RND_O on the data by Ruiz et al. [3] while on the data by Naderi et al. [2] significant dominance could only be proven against LPT and RND_O. The results show that the best combination of algorithm components heavily depends on the data, which strengthens the importance of automatic composition and evaluation of different solution methods. The used data sets and computational results are available at [21].

6 Conclusion and Outlook

In this paper, an overview of benchmark data for hybrid flow shop problems has been given. For reusability and comparability of results in heuristics development, a unified trace format has been proposed. Data of up to four dimensions can be displayed. Current limitations could be extended in the future to also use the trace format for job shop and open shop problems. A variety of solution strategies including priority rules, machine, and stage allocation strategies have been applied to synthesize 32 different constructive heuristics for solving three traces from the literature using the Combinatory Logic Synthesizer. The synthesis of scheduling algorithms proved to be a useful tool for testing combinations of different scheduling algorithm components with easy extendability which simplifies the evaluation process also for further research. In the future, more algorithmic components and also metaheuristics can be integrated into the framework.

References

[1] Robert J. Wittrock. "An Adaptable Scheduling Algorithm for Flexible Flow Lines". In: *Operations Research* 36.3 (1988), pp. 445–453.

[2] B. Naderi, Rubén Ruiz, and M. Zandieh. "Algorithms for a realistic variant of flowshop scheduling". In: *Computers & Operations Research* 37.2 (2010), pp. 236–246.

[3] Rubén Ruiz, Funda Sivrikaya Şerifoğlu, and Thijs Urlings. "Modeling realistic hybrid flexible flowshop scheduling problems". In: *Computers & Operations Research* 35.4 (2008), pp. 1151–1175.

[4] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". In: *Discrete Optimization II*. Vol. 5. Annals of Discrete Mathematics. Elsevier, 1979, pp. 287–326.

[5] Michael L Pinedo. *Scheduling*. Vol. 29. Springer, 2012.

[6] J. Heller. “Some Numerical Experiments for an M× J Flow Shop and Its Decision-Theoretical Aspects”. In: *Operations Research* 8.2 (1960), pp. 178–184.

[7] Jacques Carlier. “Ordonnancements à contraintes disjonctives”. In: *RAIRO - Operations Research - Recherche Opérationnelle* 12.4 (1978), pp. 333–350.

[8] E. Taillard. “Benchmarks for basic scheduling problems”. In: *European Journal of Operational Research* 64.2 (1993). Project Management anf Scheduling, pp. 278–285.

[9] Colin R. Reeves. “A genetic algorithm for flowshop sequencing”. In: *Computers & Operations Research* 22.1 (1995), pp. 5–13.

[10] Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. “Benchmarks for shop scheduling problems”. In: *European Journal of Operational Research* 109.1 (1998), pp. 137–141.

[11] Emmanuel Néron, Philippe Baptiste, and Jatinder N.D Gupta. “Solving hybrid flow shop problem using energetic reasoning and global operations”. In: *Omega* 29.6 (2001), pp. 501–511.

[12] Thijs Urlings. “Heuristics and metaheuristics for heavily constrained hybrid flowshop problems”. PhD thesis. Universitat Politècnica de València, 2010.

[13] Eva Vallada, Rubén Ruiz, and Jose M. Framinan. “New hard benchmark for flowshop scheduling problems minimising makespan”. In: *European Journal of Operational Research* 240.3 (2015), pp. 666–677.

[14] Quan-Ke Pan, Rubén Ruiz, and Pedro Alfaro-Fernández. “Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows”. In: *Computers & Operations Research* 80 (2017), pp. 50–60.

[15] Sebastian Lang, Tobias Reggelin, Fabian Behrendt, and Abdulrahman Nahhas. “Evolving Neural Networks to Solve a Two-Stage Hybrid Flow Shop Scheduling Problem with Family Setup Times”. In: *Hawaii International Conference on System Sciences* 53 (2020), pp. 1298–1307.

[16] Ahmed Missaoui and Rubén Ruiz. “A parameter-Less iterated greedy method for the hybrid flowshop scheduling problem with setup times and due date windows”. In: *European Journal of Operational Research* 303.1 (2022), pp. 99–113.

[17] Christoph Laroque, Madlene Leißau, Pedro Copado Méndez, Christin Schumacher, Javier Panadero, and Angel Juan. “A Biased-Randomized Discrete-Event Algorithm for the Hybrid Flow Shop Problem with Time Dependencies and Priority Constraints”. In: *Algorithms* 15(2):54 (Feb. 2022).

[18] Lisa Theresa Lenz, Julian Graefenstein, Jan Winkels, and Mike Gralla. “Smart Factory Adaptation Planning by means of BIM in Combination of Constraint Solving Techniques”. In: *Proceedings of the International Council for Research and Innovation in Building and Construction (CIB), World Building Congress 2019 – Constructing Smart Cities* (2019).

[19] Dominik Mäckel, Jan Winkels, and Christin Schumacher. “Synthesis of Scheduling Heuristics by Composition and Recombination”. In: *Optimization and Learning*. Cham: Springer International Publishing, 2021, pp. 283–293.

[20] Jan Bessai, Andrej Dudenhefner, Boris Düdder, Moritz Martens, and Jakob Rehof. “Combinatory Logic Synthesizer”. In: *Leveraging applications of formal methods, verification and validation*. Vol. 8802. LNCS sublibrary. SL 1, Theoretical computer science and general issues. Heidelberg: Springer, 2014, pp. 26–40.

[21] Dominik Mäckel, Benedikt Julian Kordus, and Christin Schumacher. *Data from: Comparison of priority rules, machine allocation, and stage allocation strategies for hybrid flow shop instances using combinatorial logic and a standard trace format*. Zenodo. DOI: 10.5281/zenodo.10100868.