Synchronous flow shop scheduling problems

Sigrid Knust, Stefan Waldherr

supported by DFG, KN 512/7-1

November 2022

UNIVERSITÄT OSNABRÜCK

https://schedulingseminar.com/

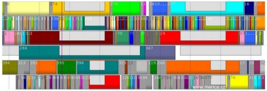
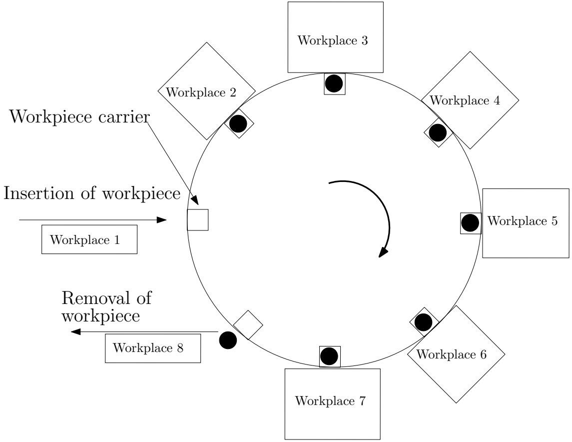


Table of Contents

Outline

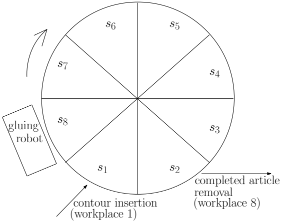
Production of kitchen elements (WK [14])

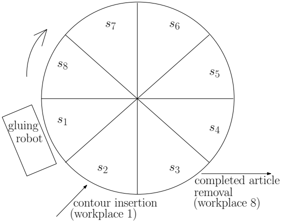
Production unit



Production environment

- three parallel production units
- rotating stations $S = \{s_1, \ldots, s_{24}\}$, 8 at each unit
- 8 fixed workplaces (machines), located around the units (insertion, gluing, drying, ..., removal)





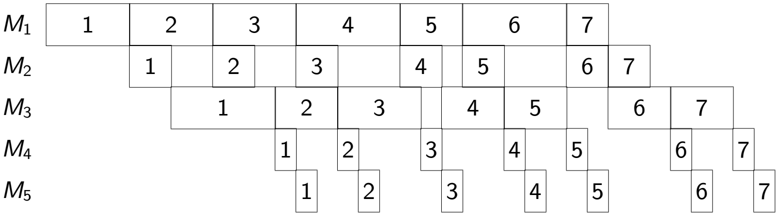
Products and resources

- different products
- insertion and gluing times relevant, all other times negligible
- orders with associated product, volume, due date
- limited resources: gluing forms of different types
- (constant) changeover time for change of gluing forms
- goal: find optimal production schedule
  - assign each product from the orders to a feasible gluing form
  - determine production sequence for each production unit
  - minimize number of late orders, total lateness and maximize number of produced items in specified time frame

Outline

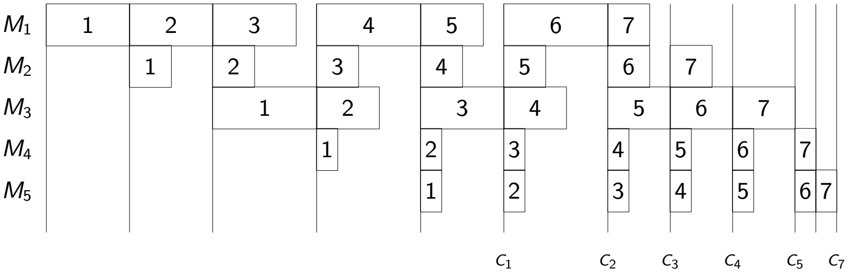
Classical permutation flow shop

- m machines $M_1, \ldots, M_m$
- n jobs $N = \{1, \ldots, n\}$, job j consists of m operations $O_{1j} \rightarrow O_{2j} \rightarrow \ldots \rightarrow O_{mj}$
- O_{ij} has to be processed on M_i for p_{ij} time units
- find job permutation (inducing completion times C_j) minimizing given objective function f



Synchronous flow shop

- jobs are processed in synchronized cycles
- synchronous movement of jobs to next machines
- more waiting for jobs and idle times on machines

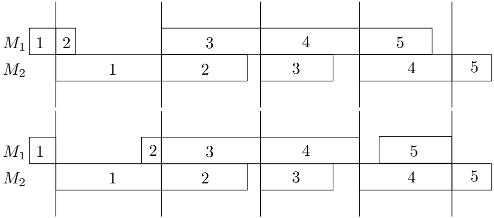


Literature

- Kouvelis & Karabati [99]: cyclic scheduling problem, MIP
- Karabati & Sayin [03]: cyclic assembly line balancing
- Soylu et al. [07]: branch-and-bound, heuristics
- Huang [08]: rotating production units, loading/unloading station, dynamic programming
- Panwalkar & Koulamas [19]: schematic representations
- Panwalkar & Koulamas [20]: complexity of ordered flow shops with $m = 3$
- Weiß et al. [17]: open shop with synchronization
- our papers [WK14], [WK15], [KKW16] [WK17], [WKB17], [BKW18]

Complexity ([WK15])

- $F2|synmv|C_{\max}$: equivalent to $F2|\text{no-wait}|C_{\max}$ polynomially solvable, $\mathcal{O}(n \log n)$ (Gilmore/Gomory [64])

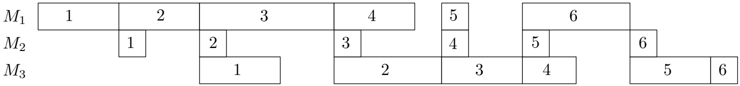


- $F3|synmv|C_{\max}$: strongly NP-hard, reduction from 3-PART
- $Fm|synmv|\sum C_j, L_{\max}$: strongly NP-hard for any fixed $m \geq 2$, reduction from $F2|\text{no-wait}|\sum C_j, L_{\max}$ (Röck [84])

Outline

Dominating machines ([WK15], [KKW16])

- M_k dominates M_l: $\min_{j \in N} p_{kj} \geq \max_{j \in N} p_{lj}$
- machine set $\{M_i \mid i \in \mathcal{I}\}$ dominating:
  $\min_{i \in \mathcal{I}} \min_{j \in N} p_{ij} \geq \max_{h \notin \mathcal{I}} \max_{j \in N} p_{hj}$



- processing times on non-dominating machines:
  - arbitrary values
  - job-independent $p_{ij} = p_i \ \forall i \notin \mathcal{I} \rightarrow p_{ij}^{ndom} = 0$

Problems with one dominating machine, $|\mathcal{I}| = 1$

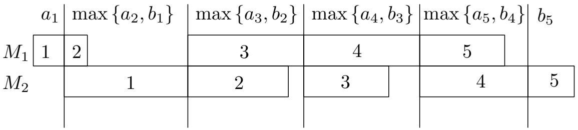
- $F|synmv, dom(\mathcal{I})|C_{\max}, \sum C_j$: strongly NP-hard
- $F|synmv, dom(\mathcal{I}), p_{ij}^{ndom} = 0|\sum C_j$: polynomially solvable, $\mathcal{O}(n \log n)$, SPT rule on dominating machine
- $F|synmv, dom(\mathcal{I}), p_{ij}^{ndom} = 0|L_{\max}$: polynomially solvable, $\mathcal{O}(n^3 \log n)$, consider feasibility problem, construct schedule from back to front
- $Fm|synmv, dom(\mathcal{I})|\sum C_j, L_{\max}$: polynomially solvable, additional factor $\mathcal{O}(n^{m-1})$
- $F2|synmv, dom(\mathcal{I}), p_{ij}^{ndom} = 0|\sum w_j C_j, \sum U_j$: open

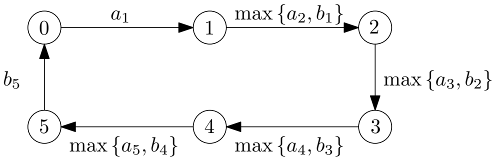
Problems with two dominating machines, $|\mathcal{I}| = 2$

- $F|synmv, dom(\mathcal{I}), p_{ij}^{ndom} = 0|C_{\max}$: strongly NP-hard, reduction from 3-PART
- $Fm|synmv, dom(\mathcal{I}), p_{ij}^{ndom} = 0|L_{\max}$: strongly NP-hard for any fixed $m \geq 2$ and each set $\mathcal{I}$ with $|\mathcal{I}| = 2$
- $Fm|synmv, dom(k, k+1), p_{ij}^{ndom} = 0|\sum C_j$: strongly NP-hard for any fixed $m \geq 2$ and each set of two adjacent dominating machines

Problems with two adjacent dominating machines

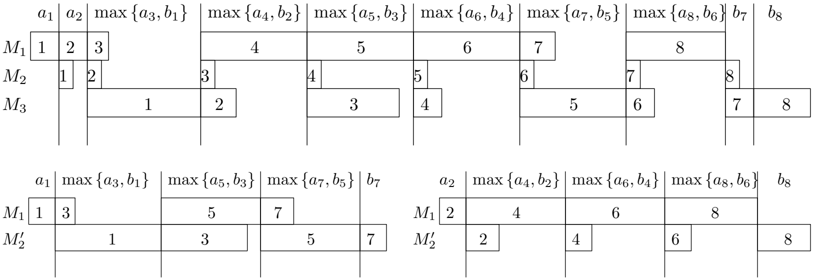
- $F|synmv, dom(k, k+1), p_{ij}^{ndom} = 0|C_{\max}$: $F2|no\text{-}wait|C_{\max}$
  "large TSP" with costs $c_{0j} = a_j$, $c_{ij} = \max\{a_j, b_i\}$, $c_{j0} = b_j$





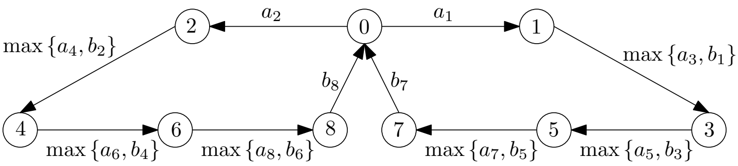
Problems with two non-adjacent dominating machines

- $F3|synmv, dom(1,3), p_{ij}^{ndom} = 0|C_{\max}$: complexity open



Problems with two non-adjacent dominating machines

- VRP with 2 vehicles and special arc costs, each route has to contain half of the nodes



$F|synmv, dom(k_1, k_2), p_{ij}^{ndom} = 0|C_{\max}$

- VRP with $\kappa = k_2 - k_1$ vehicles, special arc costs $c_{ij} = \max\{a_j, b_i\}$, each tour has to contain exactly $\frac{n}{\kappa}$ nodes
- MIP formulation based on VRP formulation:

$x_{ij} = 1$, if node j is visited directly after node i in some tour

$u_i =$ position of node i in its tour, $1 \leq u_i \leq \frac{n}{\kappa}$, MTZ subtour elimination

- important property: if partition of all jobs into subsets for the κ tours is given, optimal sequence for each subset can be calculated with algorithm of Gilmore/Gomory
- solution representation: κ disjoint subsets
- tabu search with swap neighborhood

Computational results of tabu search ([KKW16])

- Intel Pentium 4 with 3.2 GHz, CPLEX 12.5.0, 30 minutes
- 182 instances with $20 \leq n \leq 100$, $\kappa \in \{2, 3, 4, 5\}$, for which optimality could be verified by MIP

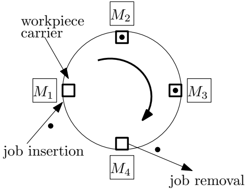
	$\kappa = 2$	$\kappa = 3$	$\kappa = 4$	$\kappa = 5$
Exactly solved	80/80	26/35	18/34	22/33
Deviation	0 %	0.005 %	0.038 %	0.049 %

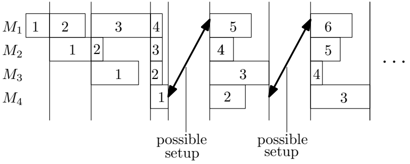
  average/maximum runtime: 6/32 seconds
- larger instances with $400 \leq n \leq 900$, $\kappa \in \{2, 3, 4, 5\}$:
  average deviation from LP relaxation (all instances): 0.0007 %
  average/maximum runtime: 9/28 minutes

Outline

Problem with additional resources ([WK17])

- renewable job resources $\mathcal{R}$ (pallet resources, gluing forms)
- every job j needs a single resource assigned from a subset $\mathcal{R}(j) \subseteq \mathcal{R}$ during its whole processing (from M_1 to M_m)
- change of resources needs setup time
- objective: minimize total production time = sum of all cycle and setup times





Different situations for resources

(R1) all jobs can be processed by all resources ($\mathcal{R}(j) = \mathcal{R}$ for all j)

- no setups necessary
- feasible solution exists $\Leftrightarrow |\mathcal{R}| \geq m$

(R2) the jobs are partitioned into disjoint families $\mathcal{F}$, where each job in a family can be processed by the same set of resources $(\mathcal{R}(j) \cap \mathcal{R}(h) \neq \emptyset \Rightarrow \mathcal{R}(j) = \mathcal{R}(h))$

- setup times s_{fg} between families f, g
- feasibility can be checked in $\mathcal{O}(n)$
- minimizing $C_{\max}$: $\mathcal{NP}$-hard even for $m = 2$ and $s_{fg} = s$

(R3) the sets $\mathcal{R}(j)$ are arbitrary subsets of $\mathcal{R}$

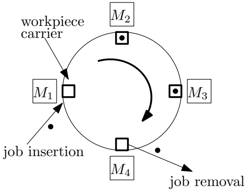
- feasibility can be checked with network flow problem

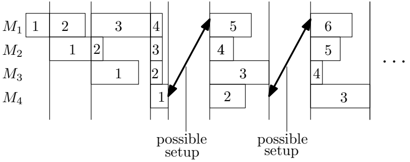
In the following focus on $(R2)$, company has even $s_{fg} = s$.

Decomposition approaches

solution representation: feasible schedule represented by

1. job permutation $\pi = (\pi_1, \ldots, \pi_n)$
2. corresponding resource sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ with $\varrho_i \in \mathcal{R}(\pi_i)$ for $i = 1, \ldots, n$ where no resource $r \in \mathcal{R}$ appears more than once in any m consecutive positions of ϱ





Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \ldots, \pi_n)$ with small sum of cycle times.

Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \ldots, \pi_n)$ with small sum of cycle times.
   [finding π^* minimizing sum of cycle times: $\mathcal{NP}$-hard for $m \geq 3$, therefore heuristic]

Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \dots, \pi_n)$ with small sum of cycle times.
   [finding π^* minimizing sum of cycle times: $\mathcal{NP}$-hard for $m \geq 3$, therefore heuristic]
2. Assign a feasible resource $\varrho_i \in \mathcal{R}(\pi_i)$ to each π_i such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.

Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \ldots, \pi_n)$ with small sum of cycle times.
   [finding π^* minimizing sum of cycle times: $\mathcal{NP}$-hard for $m \geq 3$, therefore heuristic]
2. Assign a feasible resource $\varrho_i \in \mathcal{R}(\pi_i)$ to each π_i such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [finding such an optimal resource sequence ϱ^*: $\mathcal{O}(n)$]

Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \ldots, \pi_n)$ with small sum of cycle times.
   [finding π^* minimizing sum of cycle times: $\mathcal{NP}$-hard for $m \geq 3$, therefore heuristic]
2. Assign a feasible resource $\varrho_i \in \mathcal{R}(\pi_i)$ to each π_i such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [finding such an optimal resource sequence ϱ^*: $\mathcal{O}(n)$]
   If no feasible ϱ exists, modify π.

Decomposition D1

1. Determine a job permutation $\pi = (\pi_1, \ldots, \pi_n)$ with small sum of cycle times.
   [finding π^* minimizing sum of cycle times: $\mathcal{NP}$-hard for $m \geq 3$, therefore heuristic]
2. Assign a feasible resource $\varrho_i \in \mathcal{R}(\pi_i)$ to each π_i such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [finding such an optimal resource sequence ϱ^*: $\mathcal{O}(n)$]
   If no feasible ϱ exists, modify π.

Local search: swap two jobs in π, reassign resources

Decomposition D2

1. Determine sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ of resources such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.

Decomposition D2

1. Determine sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ of resources such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
[$s_{fg} = s$: bin packing, polynomial in n for fixed m]

Decomposition D2

1. Determine sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ of resources such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [$s_{fg} = s$: bin packing, polynomial in n for fixed m]
2. Assign to each resource in the sequence a corresponding job which may be processed by this resource minimizing the sum of cycle times.

Decomposition D2

1. Determine sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ of resources such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [$s_{fg} = s$: bin packing, polynomial in n for fixed m]
2. Assign to each resource in the sequence a corresponding job which may be processed by this resource minimizing the sum of cycle times.
   [finding a corresponding optimal job permutation π^*: $\mathcal{NP}$-hard, even for $m = 2$, therefore heuristic]

Decomposition D2

1. Determine sequence $\varrho = (\varrho_1, \ldots, \varrho_n)$ of resources such that no resource appears more than once in m consecutive positions and the sum of setup times is minimized.
   [$s_{fg} = s$: bin packing, polynomial in n for fixed m]
2. Assign to each resource in the sequence a corresponding job which may be processed by this resource minimizing the sum of cycle times.
   [finding a corresponding optimal job permutation π^*: $\mathcal{NP}$-hard, even for $m = 2$, therefore heuristic]

Local search: modify ϱ, reassign jobs

Test instances

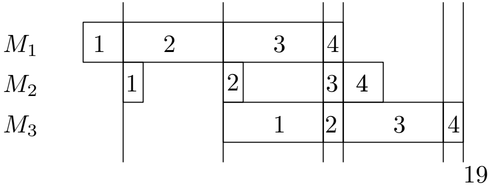
- 120 Taillard instances (20×5 up to 500×20), resources of type (R2), constant setup times $s_{fg} = s$
- different characteristics: number of job families, availability of resources, small/large setup time
- real-world data: $m = 8$, $n \in [4176, 8040]$, $|\mathcal{F}| \in [45, 67]$, constant setup time s
- time limit 10 minutes (1 hour) or 100 non-improving iterations

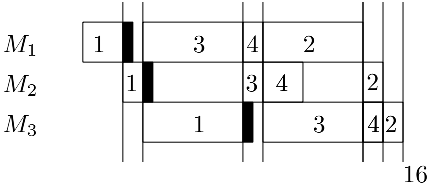
Computational results

- D2 usually outperfoms D1
- for some instances with small setup D1 better
- D2 can deal with setup times much better (bin packing achieves optimal solution minimizing number of setups)
- D1 easier to adapt for (R3)

Outline

Leaving machines idle ([WKB17])





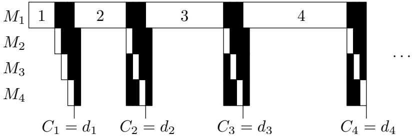
can be modeled by introducing dummy jobs in the sequence

How many dummy jobs are needed?

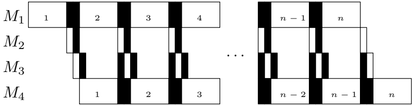
- $f(k)$: optimal objective value among all schedules where exactly k dummy jobs are introduced
- $k = 0$: normal problem, $k = \infty$: number of dummy jobs unlimited
- What is maximum value of k such that there is an instance with $f(k-1) > f(k)$ and $f(k) = f(\infty)$?
- $C_{\max}(k)$, $\sum C_j(k)$, $L_{\max}(k)$ are monotone non-increasing in k since additional dummy jobs can always be inserted at the end of a schedule without increasing the objective value

How many dummy jobs are needed?

- For any regular objective function there exists an optimal schedule with at most $(n-1)(m-1)$ dummy jobs.
  For the objectives $L_{\max}$ and $\sum C_j$ this bound is tight.



- For $C_{\max}$ there exists an optimal schedule with at most $(n-1)(m-2)$ dummy jobs and this bound is tight.



How much can we gain by leaving machines idle?

Theoretical bounds: k dummy jobs

- objective $C_{\max}$: the relative improvement is bounded by

$$C_{\max}(0)/C_{\max}(k) \leq \min\{k+1, \lceil m/2 \rceil\},$$

  the absolute improvement $C_{\max}(0) - C_{\max}(k)$ may be arbitrarily large

- objective $\sum C_j$: the relative improvement is bounded by

$$\sum C_j(0)/\sum C_j(k) \leq (k+1)\min\{k+1, \lceil m/2 \rceil\},$$

  the absolute improvement may be arbitrarily large

- objective $L_{\max}$: the relative and absolute improvement may be arbitrarily large

How much can we gain by leaving machines idle?

Computational experiments:

- 160 test instances with $n \in \{10, 15\}$ and $m \in \{2, 3, 4, 5\}$ solved to optimality by MIPs (once without dummy jobs, once with at most 4)
- 120 Taillard instances, 20×5 up to 500×20 solved heuristically

# inst.	# inst. improved			average % rel. impr. (among impr. inst.)					
	$C_{\max}$	$L_{\max}$	$\sum C_j$	$C_{\max}$		$L_{\max}$		$\sum C_j$	
160	6	41	79	0.06	(1.51)	2.79	(10.9)	0.47	(0.96)
120	31	0	10	0.27	(1.05)	0	(0)	0.10	(1.75)

results show only rather small gains when using dummy jobs

"Pliability" models ([BKW18])

- processing times p_{ij} of operations are not fixed in advance
- given only total processing times p_j for job j
- determine actual processing times $x_{ij} \geq 0$ satisfying

$$\sum_{i=1}^{m} x_{ij} = p_j \text{ for all } j$$

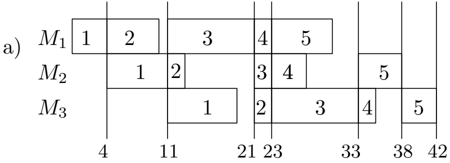
- more restricted scenario with lower/upper bounds $\underline{p}_{ij}, \overline{p}_{ij}$

$$\underline{p}_{ij} \leq x_{ij} \leq \overline{p}_{ij} \text{ for all } i, j$$

Example

$$n = 5,\ m = 3,\ \underline{p}_{ij} = 2$$

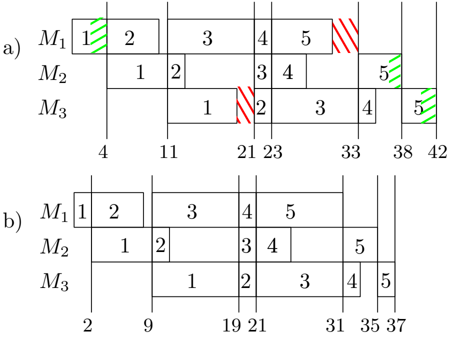
j	p_{1j}	p_{2j}	p_{3j}	p_j
1	4	7	8	19
2	6	2	2	10
3	10	2	10	22
4	2	4	2	8
5	7	5	4	16



Example

$n = 5, \; m = 3, \; \underline{p}_{ij} = 2$

j	p_{1j}	p_{2j}	p_{3j}	p_j
1	4	7	8	19
2	6	2	2	10
3	10	2	10	22
4	2	4	2	8
5	7	5	4	16



"Pliability" models

- problem NP-hard, even for 2 machines and no bounds
- distinction: x_{ij} arbitrary real values, integers required
  only lower bounds: always optimal integer-valued solution
- decomposition approach:
  1. local search on set of job permutations π
  2. for each π calculate corresponding (optimal) x_{ij}
     - subproblem of 2nd stage polynomially solvable as LP
     - only lower bounds: direct combinatorial algorithm
     - integers required: NP-hard

Outline

Conclusion

- synchronous flow shop problems
- practical application of production planning
- dominating machines
- additional job resources and setup times
- leaving machines idle, pliability
- complexity results, polynomially solvable subcases useful for efficient algorithms
- decomposition approaches, using problem-specific properties
- further research: problems with open complexity

References

[BKW18] M. Bultmann, S. Knust, S. Waldherr (2018): Synchronous flow shop scheduling with pliable jobs, EJOR 270, 943-956.

[GG64] P. Gilmore, R. Gomory (1964): Sequencing a one state-variable machine: A solvable case of the traveling salesman problem, OR 12,655-679.

[H08] K.-L. Huang (2008): Flow shop scheduling with synchronous and asynchronous transportation times, Ph.D. Thesis, The Pennsylvania State University.

[KKW16] M. Kampmeyer, S. Knust, S. Waldherr (2016): Solution algorithms for synchronous flow shop problems with two dominating machines, COR 74, 42-52.

[KS03] S. Karabati, S. Sayin (2003): Assembly line balancing in a mixed-model sequencing environment with synchronous transfers, EJOR 149, 417-429.

[KK99] P. Kouvelis, S. Karabati (1999): Cyclic scheduling in synchronous production lines, IIET 31, 709-719.

References

[PK19] S.S. Panwalkar, C. Koulamas (2019): The evolution of schematic representations of flow shop scheduling problems. JOS 22, 379-391.

[PK20] S.S. Panwalkar, C. Koulamas (2020): Three-stage ordered flow shops with either synchronous flow, blocking or no-idle machines, JOS 23, 145-154.

[R84] H. Röck (1984): Some new results in flow shop scheduling, MMOR 28, 1-16.

[SKA07] B. Soylu, Ö. Kirca, M. Azizoglu (2007): Flow shop-sequencing problem with synchronous transfers and makespan minimization, IJPR 45, 3311-3331.

[W15] S. Waldherr (2015): Scheduling of flow shops with synchronous movement, Ph.D. Thesis, University of Osnabrück.

[WK14] S. Waldherr, S. Knust (2014): Two-stage scheduling in shelf-board production: A case study, IJPR 52, 4078-4092.

References

[WK15] S. Waldherr, S. Knust (2015): Complexity results for flow shop problems with synchronous movement, EJOR 242, 34-44.

[WK17] S. Waldherr, S. Knust (2017): Decomposition algorithms for synchronous flow shop problems with additional resources and setup times, EJOR 259, 847-863.

[WKB17] S. Waldherr, S. Knust, D. Briskorn (2017): Synchronous flow shop problems: How much can we gain by leaving machines idle? Omega 72, 15-24.

[WKS1] C. Weiß, S. Waldherr, S. Knust, N.V. Shakhlevich (2017): Open shop scheduling with synchronization, JOS 20, 557-581.