University of Hagen
Department of Mathematics and Computer Science
Chair of Enterprise-wide Software Systems

Scheduling Problems in Semiconductor Wafer Fabrication Facilities: Part II

Lars Mönch
University of Hagen
Chair of Enterprise-wide Software Systems

John W. Fowler
Arizona State University
Department of Supply Chain Management

Motivation

- Manufacturing of integrated circuits is one of the key aspects of the electronics industry
- Complex scheduling problems attracted researchers and people from industry for the last three decades
- Causes: high degree of automation even 30 years ago
  - automated real-time data collection
  - manual production control leads often to an un-expected behavior of the system

- Increasing automation pressure by automated material handling systems (AMHS) and new requirements on production control (Industry 4.0)
- Deterministic machine scheduling approaches are crucial for wafer fabs

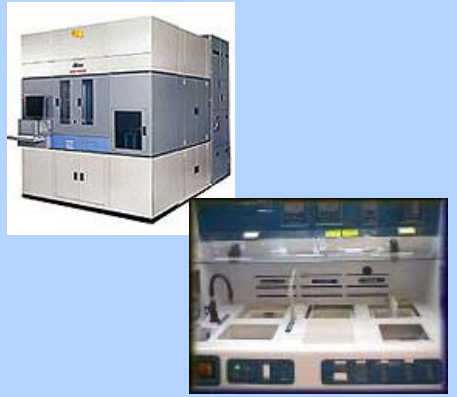
Outline

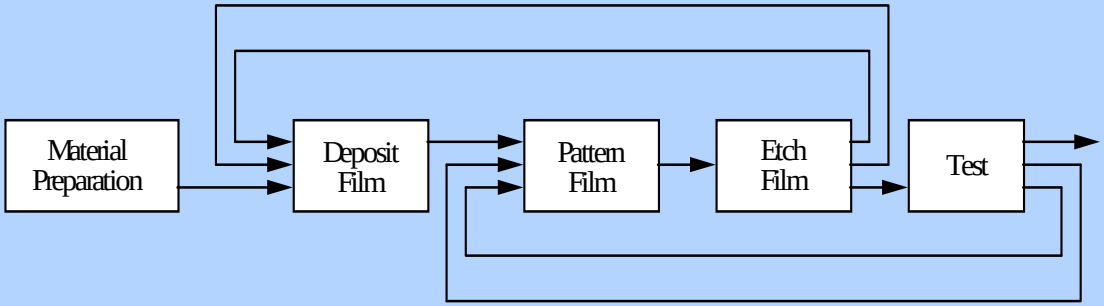
- Motivation
- Process Description
- Scheduling Problems
- Factory Scheduling **Part 1**

- Recap: Semiconductor Process Description **Part 2**
- Batch Scheduling Problems
- Scheduling Problems with Time Constraints
- Multiple Orders per Job Scheduling Problems
- Future Research Directions

Recap: Semiconductor Process Description

- many process flows
- each process flow contains up to 800 operations
- several hundreds of machines (often very expensive)
- unrelated parallel machines with dedications
- re-entrant flows





- time constraints for the processing of jobs to prevent native oxidation and contamination effects on the wafer surface
- some machines like implanters require significant sequence-dependent setup times
- dynamic bottlenecks depending on the product mix

Recap: Semiconductor Process Description (cont'd)

- some process steps require auxiliary resources such as reticles in photolithography

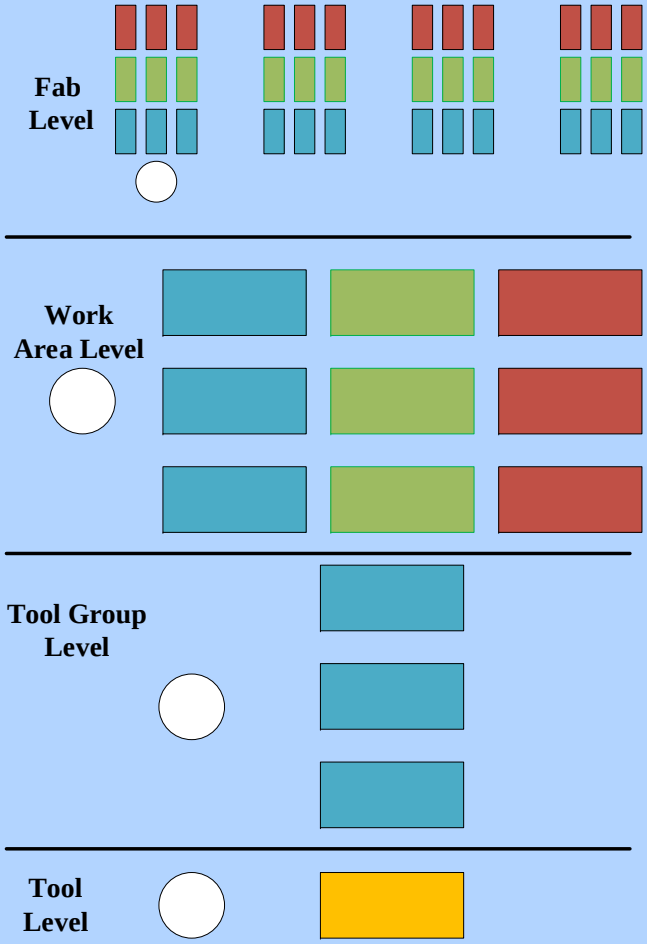
- different processing times
- **long operations often involve batch processes (one third of all operations might be batch operations)**
- non-linear flow because of mixture of batching and non-batching machines => long queues in front of the machines

- probabilistic occurrence of long machine failures leads to a large variability

Recap: Semiconductor Process Description (cont'd)

- Resource hierarchy in a wafer fab

Semiconductor manufacturing	Deterministic scheduling
tool	single-machine problem
tool group/work center	parallel-machine problem
work area	flexible flow-shop/job-shop problem
wafer fab	complex job-shop problem



- In this talk: **tool group/work area/tool problems**

Batch Scheduling Problems

- **Batch** = group of jobs that have to be processed together
- Completion time of a batch = completion time of the last job of the batch

- Two types of batching problems (with batch availability):
  - s-batching: processing time of the batch is the **sum** of the processing times of the jobs
  - p-batching: processing time of the batch is the maximum processing time of the jobs that form the batch
  - compatible vs. incompatible setting

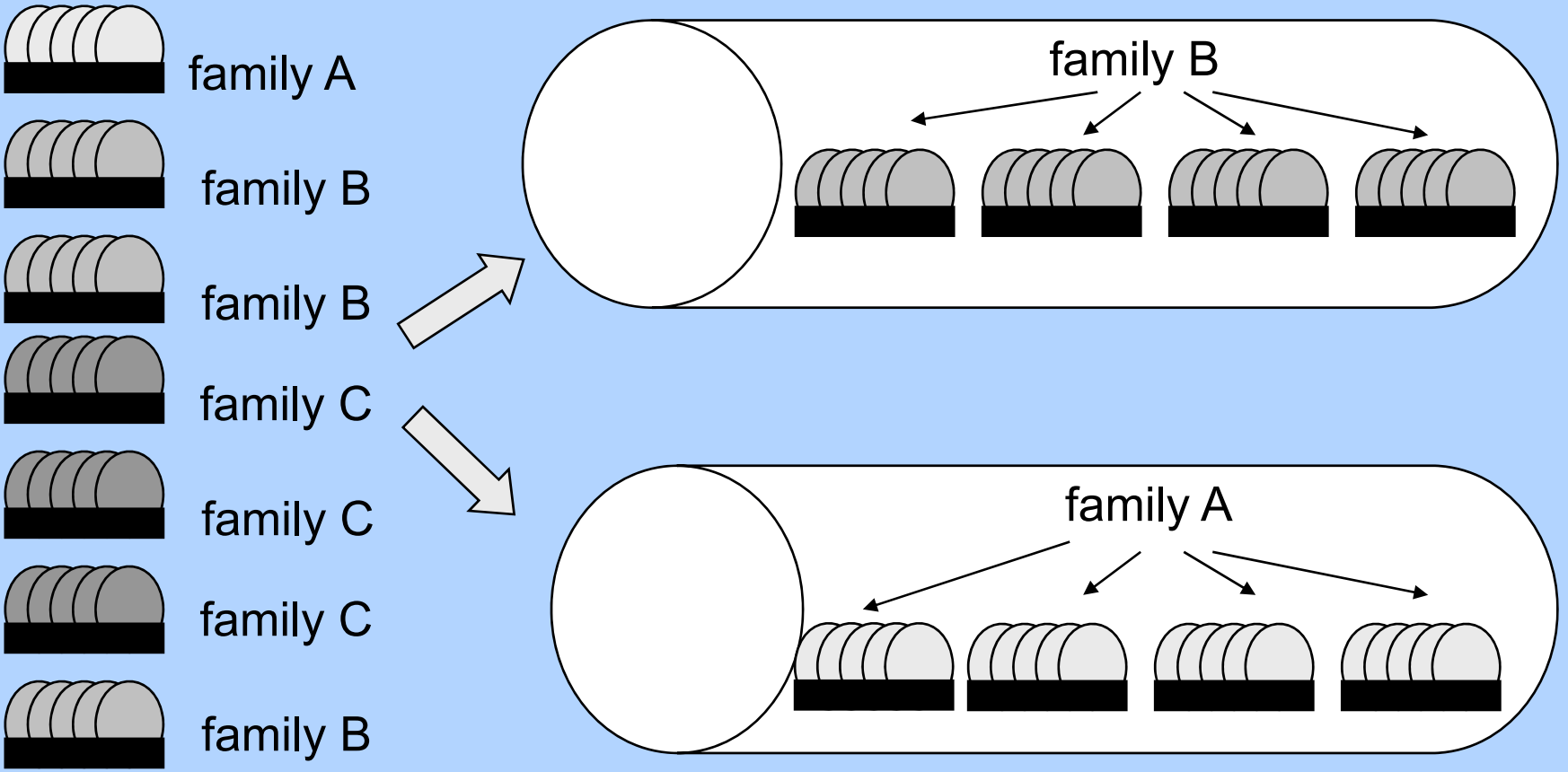
- p-batching is very important in semiconductor manufacturing
- maximum batch size B

- **Examples:**
  - burn-in ovens are used to heat-stress test chips
  - diffusion furnaces => incompatible job families, only jobs of one job family can be batched together due to the chemical nature of the process

- Fowler, J., Mönch, L. (2022): **A survey of scheduling with parallel batch (p-batch) processing**. EJOR, 298(1), 1-24.

Batch Scheduling Problems (cont'd)

Example: three incompatible families, B=4



Batch Scheduling Problems (cont'd)

- joint work with Sebastian Roob
- **Scheduling problem:**

$$P_m \mid p-batch, incompatible \mid \sum_{j=1}^{n} f_j\left(C_j\right) \quad (1)$$

- f regular sum objective function
- Incompatible families: all jobs of the same family have a common processing time

- **Property:**

  There exists an optimal schedule for each instance of problem (1) that does not contain partially full batches except possibly the last batch of each family to be processed in the schedule.

- The number of batches for each family can be precomputed.

Batch Scheduling Problems (cont'd)

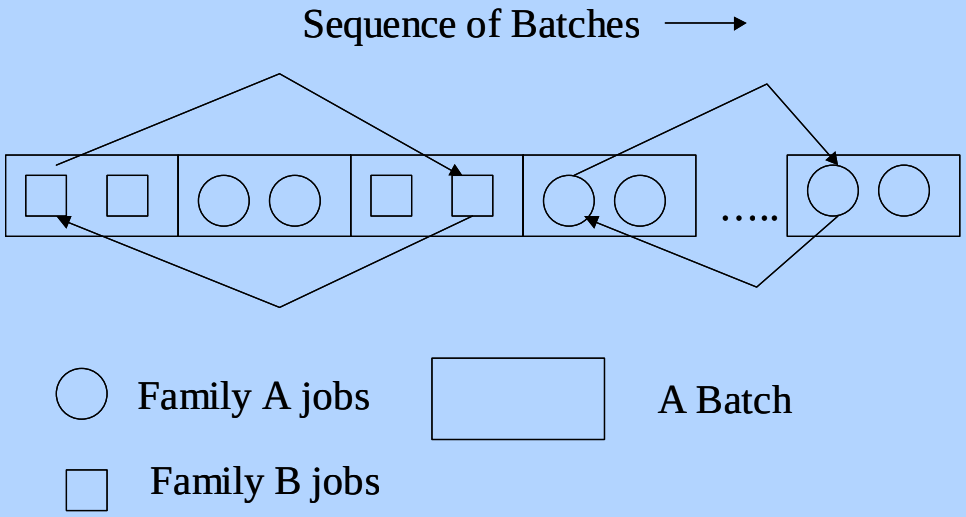
1. How to form batches?
2. How to assign the batches to one of the parallel machines?
3. How to sequence the batches on each of the parallel machines?

- Swap is used in Balasubramanian et al. (2004), Almeder and Mönch (2011), Lausch and Mönch (2016) for correcting the content of batches
- **Swap-Iter:**

  1. Start with the first job of the batch with the smallest start time.
  2. Try to interchange the current job with each job that is found in a batch of the same family that starts later.
  3. If an objective function value reduction occurs because of the job swap, exchange the two jobs and start over with the first job of the batch.

Batch Scheduling Problems (cont'd)

- Works for single and parallel machines, only starting times of the batches of a given family are of interest



- **Observation:** If starting time of batches are known, the cost (objective function value) for assigning a job to any batch can be calculated.

Batch Scheduling Problems (cont'd)

- Swap can be modeled as the following assignment-type problem for each incompatible family *f*:

$$\min \sum_{b=1}^{b_f} \sum_{j=1}^{n_f} f_j(C_b) x_{jb}$$

subject to

$$\sum_{b=1}^{b_f} x_{jb} = 1, \quad j = 1, \ldots, n_f$$

$$\sum_{j=1}^{n_f} x_{jb} = B_b, \quad b = 1, \ldots, b_f$$

$$x_{jb} \in \{0, 1\}, \quad j = 1, \ldots, n_f, b = 1, \ldots, b_f$$

Batch Scheduling Problems (cont'd)

- **Observation 1:** Since the coefficient matrix is totally unimodular, the IP is easy to solve by the Simplex algorithm.

- **Observation 2:** IP can be reformulated as a minimum cost flow (MCF) problem =>

  more efficient solution methods compared to the conventional Simplex algorithm are available in solver software, i.e., the network simplex algorithm or the out-of-kilter algorithm

  => **Swap-MCF**

Batch Scheduling Problems (cont'd)

- It is enough to assign and sequence "empty" batches
- Computing the optimal content of the batches is "easy" task

- **Matheuristic:** biased random-key genetic algorithm (BRKGA) to assign and sequence batches
- Representation: RK = [rk_1, rk_2,…, r_b]
- Random keys
  - integer part: determines machine
  - fractional part: determines sequence

- Uniform crossover
- Immigration strategy for mutation

Batch Scheduling Problems (cont'd)

- Applying Swap-MCF for each chromosome time-consuming
- Reference heuristic *I* based on list scheduling
- Heuristic *H* to compute the content of batches for each chromosome, followed by a workload balancing procedure

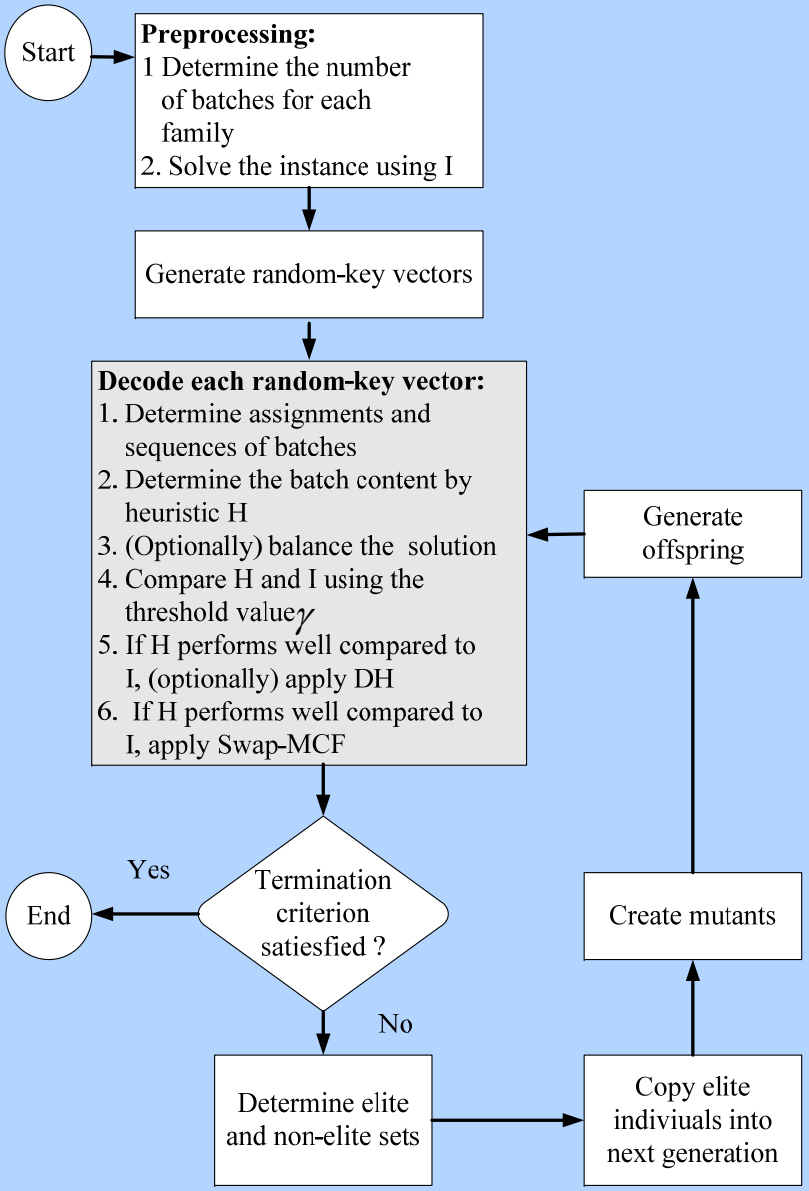
- Consider threshold value $\gamma > 0$:

$$\sum_{j=1}^{n} f_j\left(C_j(H)\right) \leq \gamma \sum_{j=1}^{n} f_j\left(C_j(I)\right) \quad (2)$$

- If (2) holds => improve the schedule by
  - (optionally) a decomposition heuristic (DH) for each single machine
  - Swap-MCF

Batch Scheduling Problems (cont'd)

▷ Overall algorithm:



Batch Scheduling Problems (cont'd)

- Application of the framework to:

$$1 \mid p-batch, incompatible \mid \sum_{j=1}^{n} w_j U_j \quad (3)$$

- Dauzere-Peres & Mönch (2013):
  - specific MILP formulations
  - RKGA based on job sequences

- Tailoring of the framework:
  - I: list scheduling based on weighted modified due date (WMDD) rule
  - H: modification of this approach for given batch set

Batch Scheduling Problems (cont'd)

- Application of the framework to:

$$P_m \mid p-batch, incompatible \mid \sum_{j=1}^{n} w_j T_j \quad (4)$$

- Almeder & Mönch (2011)/Lausch & Mönch (2016):
  - ACO
  - VNS
  - LNS

- Tailoring of the framework:
  - I: list scheduling based on apparent tardiness cost (ATC) rule
  - H: modification of this approach for given batch set

- Implementation issues:
  - brkga-API framework, C++ programming language
  - lp_solve 5.5

Batch Scheduling Problems (cont'd)

- 144 small-sized and 144 large-sized instances from Dauzere-Peres & Mönch (2013)
- Up to 144 jobs, up to 6 families, up to 90% tardy jobs
- Computational results (60 sec computing time per instance):
  - 72 small-sized unweighted instances: optimal solutions known from Dauzere-Peres & Mönch (2013) are found
  - 72 small-sized weighted instances: only **0.97%** worst compared to the best performing MILP approach from Dauzere-Peres & Mönch (2013)
  - 72 large-sized unweighted instances: **0.31%**
  - 72 large-sized weighted instances: **1.82%**

Batch Scheduling Problems (cont'd)

- 288 problem instances from Almeder & Mönch (2011)
- Up to 300 jobs, up to 12 families, up to 60% tardy jobs
- Average deviation from results obtained by VNS/LNS is 1.33%
- different schedules:
  - **low-quality** solutions
  - **medium-quality** solutions
  - **fairly high-quality** solutions
- Solution quality of Swap-Iter: percentage of optimal solutions

Low-quality Schedules	Medium-quality Schedules	High-quality Schedules
16.94%	36.39%	36.18%

- TWT ratios are 0.98 for low-quality and 0.99 for high-quality schedules
- Swap-MCF requires only 70% computing time of Swap-Iter

Scheduling Problems with Time Constraints

- **Definition:** We consider job J_i from product *j*. A time constraint holds for the operations O_{ir} and O_{is} if we have for the corresponding start times for $r < s$:

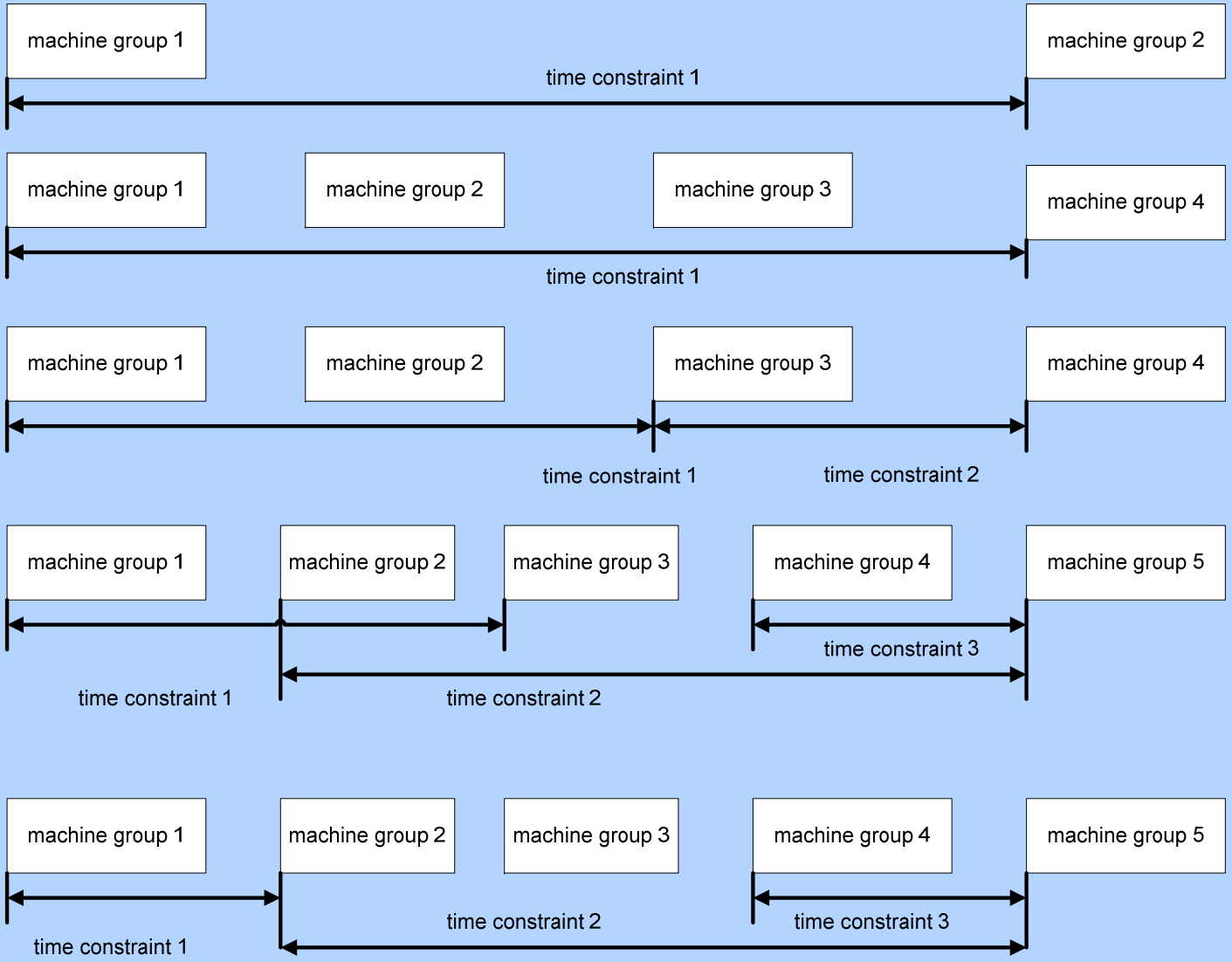
$$s_{is} \leq s_{ir} + t_{irs}.$$

- **scheduling literature:**
  - only the very special case: $s = r + 1$ maximal time lag
  - minimal time lag: modeling of transportation times
  - Caumond *et al.* (2008), Grimes and Hebrard (2010), Jung *et al.* (2014), **Klemmt and Mönch (2012)**

- different classes of time constraints can be found in semiconductor manufacturing
- nested time constraints

Scheduling Problems with Time Constraints (cont'd)

- classes of time constraints found in wafer fabs



Scheduling Problems with Time Constraints (cont'd)

▷ simplified model problem: $FF_m \mid r_i, t_{ip}, t_{jop} \mid TWT$

where $T := \{(j, o, p) \mid 1 \le o < p \le m\}$ is the set of all time constraints and t_{ip} is an initial time constraint.
$T^{(i)} := \{(i, p) \mid 1 \le p \le m\}$ is the corresponding set.

▷ notation:

- d_i: due date of job J_i
- r_i : release date of job J_i
- C_i : completion time of job J_i
- T_i : tardiness of job J_i
- r_{ol} : availability of machine M_{ol}

Scheduling Problems with Time Constraints (cont'd)

▷ MILP formulation: (main) decision variables:

- $s_{io} \in \mathrm{IR}_+$: starting time of operation *o* of job J_i
- $x_{igo} \in \{0,1\}$: 1 if operation *o* of J_i is scheduled before operation *o* of J_g
- $w_{iol} \in \{0,1\}$: 1 if operation *o* of J_i is scheduled on machine *l*

$$\min \sum_{i=1}^{n} w_i T_i$$

subject to

$$C_i - T_i \le d_i, \qquad i = 1,\ldots,n$$

$$s_{io} + p_{jo} \le C_i, \quad i = 1,\ldots,n,\ o = 1,\ldots,m,\ j = f(i)$$

$$r_{ol} w_{iol} \le s_{io}, \qquad i = 1,\ldots,n,\ o = 1,\ldots,m,\ l = 1,\ldots,m_o$$

$$r_i \le s_{i1}, \qquad i = 1,\ldots,n$$

Scheduling Problems with Time Constraints (cont'd)

$$\sum_{l=1}^{m_o} w_{iol} = 1, \qquad i = 1,\ldots,n,\ o = 1,\ldots,m$$

$$s_{io} + p_{jo} \le s_{i,o+1}, \qquad i = 1,\ldots,n,\ o = 1,\ldots,m-1,\ j = f(i)$$

$$K(w_{iol} - x_{igo} - 1) + s_{go} + p_{jo} w_{gol} \le s_{io}, \quad o = 1,\ldots,m, l = 1,\ldots,m_o, 1 \le i < g \le n,\ j = f(g)$$

$$K(w_{gol} + x_{igo} - 2) + s_{io} + p_{jo} w_{iol} \le s_{go}, \quad o = 1,\ldots,m, l = 1,\ldots,m_o, 1 \le i < g \le n,\ j = f(i)$$

$$s_{iq} \le s_{io} + t_{joq}, \quad (j,o,q) \in T,\ i = 1,\ldots,n,\ j = f(i)$$

$$s_{ip} \le t_{ip}, \qquad (i,p) \in T^{(i)}, i = 1,\ldots,n.$$

- Up to 10 jobs can be solved on a small number of machines to optimality in a reasonable amount of time.

  => decomposition approaches

Scheduling Problems with Time Constraints (cont'd)

I. List scheduling approach with backtracking

▷ job-by-job decomposition approach to determine "canonical" schedules

1. **Initialization:** Discretize the time axis. Sort all jobs with respect to a (global) ATC dispatching rule.
2. **Schedule** the first job in ATC order taking into account the time constraints.
3. **Repeat** for the remaining jobs:
   a) Repeat for all stages.
   b) Determine the machine with the smallest availability time for the current stage. If the current operation can be started without time constraint violation then schedule it and go to a).
   c) Otherwise, do a right shift of the starting time of the earlier operation that contributes to the time constraint by increasing it by one unit and try to schedule the operations taking into account other time constraints until the current one that leads to the infeasibility.

Scheduling Problems with Time Constraints (cont'd)

II. Decomposition approach based on the MILP/CP formulation

1. **Initialization:** Sort the jobs with respect to the ATC rule. Denote the set of scheduled jobs by S. Set $S := \varnothing$. *L* is the set of unscheduled jobs.
2. **Repeat** for *I* that contains the first $\min(|L|, \bar{n})$ unscheduled jobs in ATC order
   a) Schedule the job set *I* using the MILP/CP with an objective function that contains the sum of the start times as a penalty term.
   b) Update the machine availability and the sets *S* and *L* and go to Step 2.

- ▷ $\bar{n} \leq 6$
- ▷ Problem instances with up to 100 machines that are organized in up to 20 stages can be solved.
- ▷ All classes of nested time constraints can be treated by this approach.

Scheduling Problems with Time Constraints (cont'd)

III. Decomposition approach based on genetic algorithm

1. Replace ATC ordering by sequencing based on BRKGA, random-key representation
2. Use list scheduling approach with backtracking for each chromosome to determine fitness
3. Use MILP/CP decomposition approach to improve best sequence obtained from BRKGA

▷ Implementation based on brkgaAPI (Toso & Resende 2014)

Scheduling Problems with Time Constraints (cont'd)

- design of experiments
- randomly generated problem instances
- performance depends on
  - number of jobs
  - number of stages
  - number of parallel machines on each stage
  - ready time and due date setting
  - time constraint class and tightness of the time constraints
- two scenarios:
  - class 3
  - class 4, but only a single overlap of time constraints is allowed
- totally, 40 problem instances with up to 100 jobs and 72 instances with up to 750 jobs
- $\overline{n} := 5$
- maximum computing time per MILP/CP instance 5 seconds (ILOG CPLEX 12.7)

Scheduling Problems with Time Constraints (cont'd)

- Improvement in % over list scheduling with backtracking

Factor level	ATC+ CP(5)	BRKGA(300s) +MILP(5)	BRKGA(300s) +CP(5)
n=50	11.78	**21.68**	21.54
n=100	12.58	24.71	**24.79**
n=125	2.03	8.84	**8.88**
n=150	5.85	19.72	**19.94**
n=250	7.53	15.95	**18.00**
n=375	5.41	11.44	**12.93**
n=500	2.48	10.32	**11.14**
n=750	1.96	7.28	**7.94**
Overall	8.43	17.99	**18.50**

Scheduling Problems with Time Constraints (cont'd)

- BRKGA+CP slightly outperforms BRKGA+MILP approach
- large number of jobs => improvements are smaller
- list scheduling with backtracking is outperformed by BRKGA-type approaches, post-optimization by CP beneficial
- Improvement for the TWT setting is much smaller compared to improvement for TT setting

- extensions for flexible flow-shops with batch processing machines by Cailloux and Mönch (2019)

Multiple Orders per Job Scheduling Problems

Joint work with Jens Rocholl

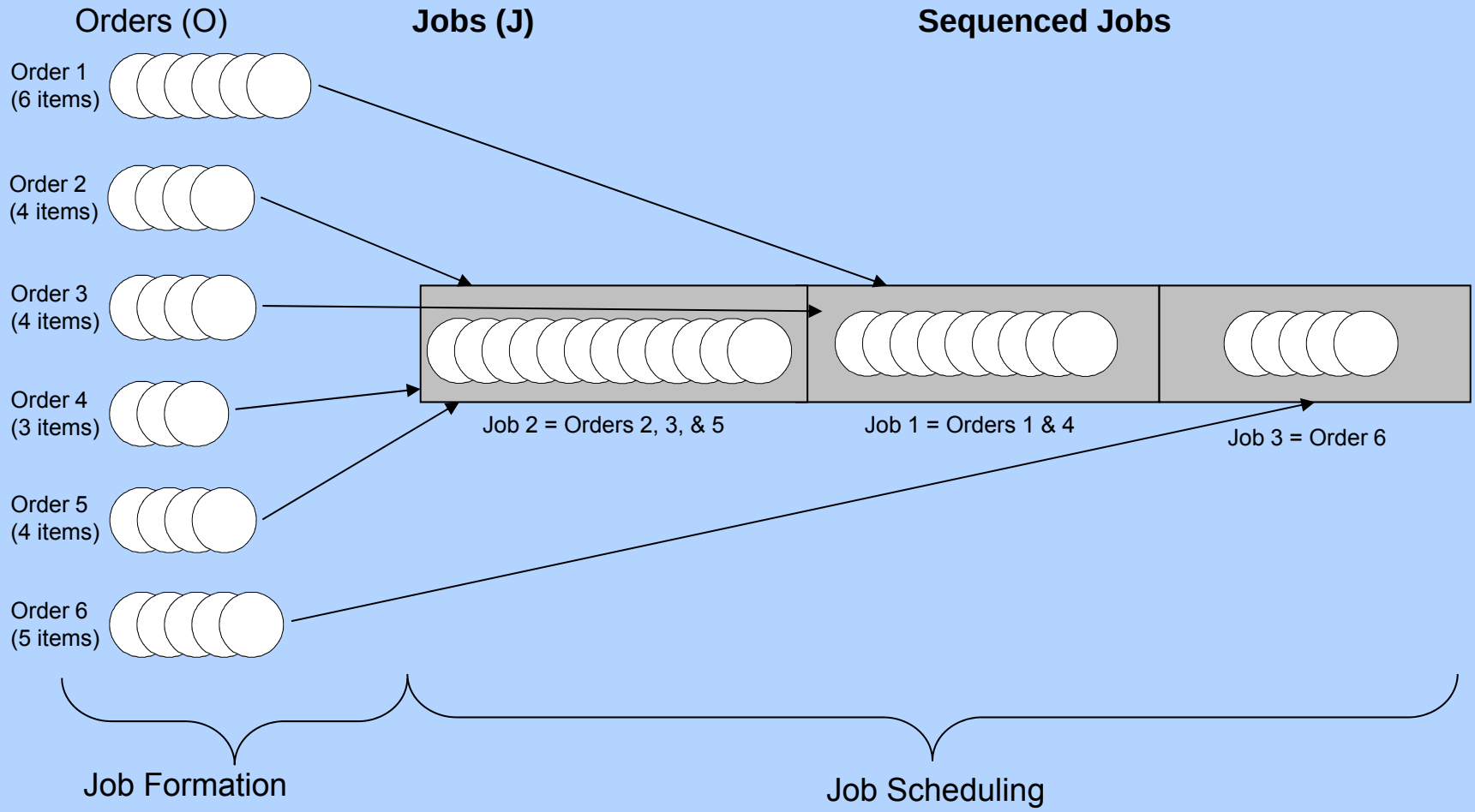
- Front-opening unified pod (FOUP) = standard unit of lot transfer between machines, container that holds up to 25-wafer lots of 300-mm wafers in an inert, nitrogen atmosphere

- FOUPs are expensive
- A large number of FOUPs can cause a congested AMHS
- Combination of decreased line width and increased area per wafer => fewer wafers are required to fill the ICs of a certain customer

- Need to group orders of different customers into one FOUP => **Multiple Orders per Job (MOJ)** scheduling problem

Multiple Orders per Job Scheduling Problems

▷ MOJ problems



Multiple Orders per Job Scheduling Problems (cont'd)

- FOUPs: F
  - capacity K

- Orders $o=1,\ldots,N$
  - s_o: size (number of wafers), $s_o \leq K$
  - d_o: due date

- Jobs: $j=1,\ldots,|J|$
  - $|J| \leq \min(F,N)$
  - capacity K, $\sum s_o \leq K$
  - processing time p_j

Multiple Orders per Job Scheduling Problems (cont'd)

- Two environments:
  - **Lot processing**: all wafers of a job will be processed simultaneously, $p_j = \rho$ processing time of job j
  - **Single item processing**: product of wafer processing time and number of wafers, $p_j = \rho \sum s_o$

Problem (1): $E/T = \sum |C_o - d| = \sum E_o + \sum T_o$

$1|moj(lot), d_o = d \,|E/T$, d common due date,

$\sum p_o \leq d$, nonrestrictive

Multiple Orders per Job Scheduling Problems (cont'd)

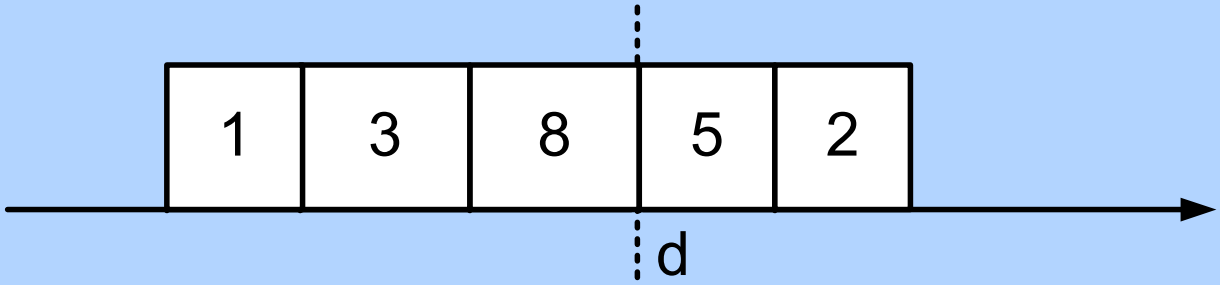
▷ Properties of problem (1)

1. **Property:** There is an optimal schedule where one job is completed at the common due date *d*.

   A: set of early or on-time jobs (= early or on-time orders)
   B: set of tardy jobs (= tardy orders)
   wj: number of orders in job *j*

2. **Property:** There is an optimal schedule where all jobs in A are sorted with respect to increasing w_j/p_j and all jobs in B with respect to decreasing w_j/p_j.

=> V-shape schedule



NP-hard problem!

Multiple Orders per Job Scheduling Problems (cont'd)

- ▷ IP formulation for problem (1)
- ▷ Basic idea (Hall and Possner 1991):

  $C_j = d - \lceil |J|/2 \rceil \rho + j\rho \Rightarrow$

  $E/T_j = \rho \max(\lceil |J|/2 \rceil - j, 0) + \rho \max(-\lceil |J|/2 \rceil + j, 0)$

$$\min \sum_{o=1}^{N} E/T_j X_{oj}$$

subject to

$$\sum_{j \in J} X_{oj} = 1, \qquad \forall o \in O$$

$$\sum_{o=1}^{N} s_o X_{oj} \leq K, \qquad \forall j \in J$$

$$X_{oj} \in \{0,1\}, \forall o \in O, \forall j \in J$$

Multiple Orders per Job Scheduling Problems (cont'd)

- **MOJ scheduling problems:**
  - Mason *et al.* a series of papers starting in 2002
  - Qu & Mason (2005) for problem (1) => two GAs
  - Sobeyko & Mönch (2015) => GGA

- **nonrestrictive common due date and p-batching:**
  - Kanet (1984), polynomial-time algorithm for $1|d_j=d|E/T$
  - Hall & Posner (1991), total weighted earliness and tardiness, DP
  - Mönch & Unbehaun (2007) $P_m|p\text{-}batch,d_j=d|E/T$ => GAs hybridized with DP
  - Li *et al.* (2015) $1|p\text{-}batch,s_j,d_j=d|E/T$ => GA hybridized with DP

Multiple Orders per Job Scheduling Problems (cont'd)

- **Simple reference heuristic** similar to Mason *et al.* (2004):

1. Sort the orders according to a priority rule (i.e. Smallest Size (SS), i.e. $1/s_o$ or Largest Size (LS), i.e. s_o)
2. Assign orders to jobs according to First Fit Decreasing FFDn, i.e., assign an order to the first job with available capacity.
3. Sort jobs in non-increasing order of the number of orders in the jobs and determine V-shaped schedule by assigning jobs to the sets A and B

   (**|J|-1,..,3,1**#2,4,…,|J|) or (**|J|,..,3,1**#2,4,…,|J|-1)

Multiple Orders per Job Scheduling Problems (cont'd)

- **First GA (GA-H)**
  - Idea: GA assigns orders to A and B
  - Representation: $c:=(a_1,\ldots,a_N)$ and

$$a_j := \begin{cases} 1, \text{if } o_j \in A \\ 0, \text{if } o_j \in B \end{cases}$$

  - Compute solution by applying SS-FFDn or LS-FFDn to A, B
  - One-Point Crossover + Flip Mutation

  => GA-H1 and GA-H2

Multiple Orders per Job Scheduling Problems (cont'd)

- **Second GA (RKGA)**
  - Idea: GA produces permutations of the orders
  - Random-key representation: $c:=(a_1,\ldots,a_N)$ with $a_j \in [0,1]$
  - order sequence by $o_i \trianglelefteq o_k$ if $a_i \leq a_k$
  - Parameterized uniform crossover, i.e. biased coin for each gene to determine which gene contributes to the child
  - No mutation, but immigration of new randomly generated individuals
  - Permutation of orders => FFDn + V-shaped schedule

Multiple Orders per Job Scheduling Problems (cont'd)

Factor	Level	Count
s_o	$\sim DU\left[\nu - \frac{\nu+1}{2}, \nu + \frac{\nu+1}{2}\right], \nu \in \{3,5\}$	2
w_o	$\sim DU[1,15]$	1
K (in wafer)	$12\beta + 1,\ \beta \in \{1,2\}$	2
N	10,25,50,100	4
F	$\lceil N\nu / 12\beta \rceil + 1$	1
Processing time	$\rho = 10$	1
Number of independent problem instances	10 per factor combination	10
Total number of problem instances		160

Multiple Orders per Job Scheduling Problems (cont'd)

▷ 160 instances

TABLE I COMPUTATIONAL RESULTS

N	β	ν	GA-H1	GA-H2	RKGA
10	1	3	**35**	38	**35**
	1	5	n/a	**91**	**91**
	2	3	**6**	**6**	**6**
	2	5	**41**	**41**	**41**
25	1	3	221	227	**219**
	1	5	n/a	515	**505**
	2	3	**118**	121	**118**
	2	5	248	248	**246**
50	1	3	908	951	**902**
	1	5	n/a	2023	**1987**
	2	3	**479**	490	**479**
	2	5	n/a	**1019**	1020
100	1	3	**3700**	3924	3718
	1	5	n/a	**8173**	8057
	2	3	**1865**	1935	1935
	2	5	n/a	n/a	**4080**

Multiple Orders per Job Scheduling Problems (cont'd)

- SS-FFDn => often not able to provide feasible solution
- GA-H1 similar problems
- RKGA works well for small- and medium-sized problem instances
- GA-H2 is best performer for large-sized instances
- CPLEX is able to solve instances up to 25 orders to optimality
- RKGA was always able to find these optimal solutions

Future Research Directions

▷ **New problem settings:**

1. borderline between dispatching and scheduling must be explored in more detail
2. assumption of deterministic data is not always true, stochastic settings must be considered
3. scheduling problems with interfering job sets since production lots compete with engineering lots for scarce resources
4. better interaction of AMHS decisions and job scheduling decisions, storing and transporting lots must be done consistently with their production schedule
5. interaction of scheduling decisions and Advanced Process Control (APC), new constraints for scheduling to improve quality and yield
6. better understanding of the interplay between production planning and scheduling decisions for wafer fabs is desirable

Future Research Directions (cont'd)

▷ **New solution techniques:**

1. more theoretical insights are necessary related to hierarchical and non-hierarchical decomposition schemes, i.e., consistency between global wafer fab scheduling and detailed work area scheduling
2. hybrid approaches based on MILP, CP, and metaheuristic approaches, especially for fab-wide scheduling
3. common benchmarks are desirable like for traditional job shop scheduling problems
4. ideas from stochastic programming and robust optimization can be used to tackle scheduling problems in a stochastic setting
5. promising trend is the use of genetic programming to discover dispatching rules
6. more research is needed to investigate whether the solution process can be accelerated using the Compute Unified Device architecture (CUDA) platform or Graphics Processing Unit (GPU) programming approaches

Future Research Directions (cont'd)

- **Survey paper:**
  - Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., Rose, O. 2011. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6), 583-599.
  - Updated version is planned for 2022.

Discussion