

Dispatching and Scheduling in Semiconductor Manufacturing: Strategies for Deterministic vs. Stochastic Processing Times

First Author¹[0000–1111–2222–3333], Second Author^{2,3}[1111–2222–3333–4444], and
Third Author³[2222–3333–4444–5555]

¹ Princeton University, Princeton NJ 08544, USA

² Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
`lncs@springer.com`

<http://www.springer.com/gp/computer-science/lncs>

³ ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
`{abc,lncs}@uni-heidelberg.de`

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Scheduling production operations in complex manufacturing environments is one of the most challenging resource allocation tasks. The dynamic nature of real-world operations, subject to unpredictable process and demand fluctuations, supply chain delays, machine breakdowns, etc., further amplifies the complexity. Therefore, production scheduling needs not only be (near-)optimal, but also robust and flexible to adapt to frequent problem changes. To handle these phenomena, a scheduling method should be efficient enough to provide and update solutions in short time.

In the scheduling literature, the Job Shop Scheduling Problem (JSSP) [?] and the Flexible Job Shop Scheduling Problem (FJSSP) [?] stand out as popular and general scheduling models. Both JSSP and FJSSP are NP-hard combinatorial optimization problems, where the latter extends JSSP by introducing flexibility in assigning operations to machines. This flexibility is crucial in modern manufacturing environments, as the capability to respond to changing demands and utilize resources efficiently impacts the overall production performance.

In this paper, we address a large-scale Semiconductor Manufacturing Scheduling Problem (SMSP) [?], which can also be understood as FJSSP on instances that represent the diverse operations and high throughputs of semiconductor production. A semiconductor manufacturing facility, called fab for short, is a complex production environment characterized by customized job flows and high-tech machines. Particular challenges of SMSP include:

- **Long production routes:** Hundreds of operations to be performed on a lot can span over several months, which necessitates detailed scheduling to realize an efficient process flow.
- **Product-specific machine setups:** Production operations may require specific machine setups, which increases the complexity of operation sequencing and machine allocation.
- **Maintenance and stochastic events:** Preventive maintenance procedures and unpredictable yet frequent events, such as machine breakdowns or process deviations, demand for a highly adaptive and responsive scheduling system.

To meet the growing demands for semiconductors and manage the complexity of modern semiconductor production, manufacturers are increasingly striving for automated production scheduling and control systems. Such systems are conceived to optimize the resource utilization, regulate sophisticated sequences of manufacturing steps, and adjust to process deviations in real time. The overall goal is to minimize the production times and costs, while maximizing fab throughput and product quality.

In contrast to classical FJSSP benchmarks, where small to medium-scale instances include up to 500 operations and 60 machines [?], modern semiconductor fabs process tens of thousands of operations per day using more than 1000 machines [?]. These machines are organized in tool groups with specific functionalities, e.g., diffusion, etching, or metrology, and each manufacturing step can be flexibly allocated to a machine providing the required functionality. Hence, the scheduling of semiconductor production operations consists of the sub-tasks of assigning operations to machines and sequencing the operations on each machine. However, in case of machine breakdowns or process deviations, a schedule must be revised to adapt to the changed environment. To this end, we propose a Greedy Search based Ant Colony Optimization (GSACO) algorithm for (re-)scheduling semiconductor production operations. Our algorithm harnesses Ant Colony Optimization (ACO) [?] for exploration, while Greedy Search (GS) [?] enables responses in short time. In this way, GSACO overcomes limitations of a state-of-the-art Constraint Programming approach [?] on large-scale SMSP instances.

The paper is organized as follows. Section 2 provides literature reviews on FJSSP and ACO. In Section 3, we formulate SMSP in terms of the well-known FJSSP. Our GSACO algorithm is presented in Section 4. Section 5 provides and discusses experimental results. Finally, Section 6 concludes the paper.

2 literature Review

We briefly survey the relevant literature on FJSSP, which we use as general scheduling model to represent semiconductor production processes and SMSP, as well as ACO and its application to FJSSP solving.

Table 1. Basic notations

Symbol	Description
J	Total number of <i>jobs</i>
T	Total number of <i>tool groups</i>
M	Total number of <i>machines</i>
N	Total number of <i>operations</i>
t_m	Tool group t of machine m
$O_{i,j,t}$	Operation i of job j on tool group t
$d_{i,j,t}$	<i>Duration</i> of operation $O_{i,j,t}$ on tool group t
$O_{i,j,t,m}$	Operation i of job j on machine m of tool group t
$s_{i,j,t,m}$	<i>Start time</i> of operation $O_{i,j,t}$ on machine m of tool group t

2.1 Flexible Job Shop Scheduling

Exact mathematical models of the NP-hard FJSSP allow for solving small-scale instances to optimality. For example, [?] devise Mixed-Integer Linear Programming (MILP) and Constraint Programming (CP) models for a distributed FJSSP setting. They observe that CP performs well at exploring feasible solutions, leading to high-quality solutions in relatively short computing time. Similarly, [?] make use of a CP model to effectively solve small to medium-scale FJSSP instances.

The scalability limits of exact optimization approaches motivate the design of approximation methods, where a large variety of heuristics and meta-heuristics have been proposed for FJSSP solving. Respective techniques include tabu search [?], simulated annealing [?], particle swarm optimization [?], genetic algorithms [?], artificial bee colony algorithms [?], and ACO [?] as well as hybrid approaches based on local search with meta-heuristics [?,?,?,?].

In their recent work, [?] train deep neural networks on historical FJSSP data to identify patterns and predict effective scheduling strategies. Related approaches based on deep reinforcement learning [?,?] or genetic algorithms [?] aim at optimizing the decision making and control within simulation models of semiconductor fabs. While these methods allocate production lots one by one based on their features, we address the task of scheduling the operations on given lots in advance.

2.2 Ant Colony Optimization

ACO is a meta-heuristic algorithm that mimics the foraging behavior of ants [?].

It was originally devised to solve the traveling salesperson problem [?], and meanwhile ACO has been adopted to various optimization problems, including routing [?], scheduling [?], task allocation [?], project planning [?], and network optimization [?].

The common idea is that artificial ants construct paths through a graph, making probabilistic decisions based on problem-specific heuristic information as well as temporary pheromone trails that indicate promising search directions.

Particular strengths of ACO lie in the high potential for parallelization, given that ants can be simulated in parallel, and a certain robustness against getting stuck in local optima, as the probabilistic decision rules of ants promote exploration. However, a specific difficulty in the ACO algorithm design concerns the tuning of hyperparameters, such as the number of ants to consider, the trade-off between heuristic information and pheromone trails, and the pheromone evaporation rate.

Among meta-heuristic FJSSP solving techniques, ACO has been shown to be a particularly promising approach [?]. The practical difficulty remains to escape local optima and reliably converge to high-quality solutions within a short computing time limit. This challenge has brought about a variety of extended ACO algorithms as well as hybrid approaches that combine ACO with local search methods [?,?,?,?]. While these methods have been designed and evaluated on small to medium-scale FJSSP benchmarks [?], our work addresses the large-scale SMSP instances encountered in the domain of semiconductor production scheduling [?]. Beyond FJSSP and SMSP investigated here, we note that hybrid optimization algorithms integrating meta-heuristics and local search have also been adopted in a variety of other application settings [?,?,?,?].

3 Problem Formulation

We formulate SMSP in terms of the general FJSSP model, using the basic notations listed in Table 1. In detail, our setting for scheduling the production of a semiconductor fab is characterized as follows:

- The fab consists of M machines, which are partitioned into T tool groups, where $t_m \in \{1, \dots, T\}$ denotes the tool group to which a machine $m \in \{1, \dots, M\}$ belongs.
- There are J jobs, where each $j \in \{1, \dots, J\}$ represents a sequence $O_{1,j,t_1}, \dots, O_{n_j,j,t_{n_j}}$ of operations to be performed on a production lot. Note that $t_i \in \{1, \dots, T\}$ specifies the tool group responsible for processing an operation O_{i,j,t_i} , but not a specific machine of t_i , which reflects flexibility in assigning operations to machines. The total number of operations is denoted by $N = \sum_{j \in \{1, \dots, J\}} n_j$.
- For each operation $O_{i,j,t}$, the duration $d_{i,j,t}$ is required for processing $O_{i,j,t}$ on some machine of the tool group t .

Our SMSP model reflects several features originating from the semiconductor production scenarios of the SMT2020 dataset [?]. In the SMT2020 scenarios, jobs are associated with particular products, and their operation sequences, also called production routes, coincide for the same product. Since such production routes include hundreds of operations that are performed over several months in the physical fab, distinct lots of the same product are frequently at different steps of their routes when they get (re-)scheduled. Hence, we do not explicitly distinguish production routes by products but consider operation sequences of length n_j relative to a job j , which allows for separating the operations on lots that are at different manufacturing steps of the same route.

Table 2. GSACO parameters

Parameter	Description
l	Cycles/time limit
n	Planning horizon
k	Number of ants
τ_y	Initial pheromone level
τ_z	Minimum pheromone level
τ_e	Pheromone level on edge e
ρ	Evaporation rate
c	Contribution of best schedules

Moreover, the machines belonging to a tool group are assumed to be uniform, i.e., an operation requiring the tool group can be processed by any of its machines. This simplifying assumption ignores specific machine setups, which may be needed for some operations and take additional equipping time, as well as unavailabilities due to maintenance procedures or breakdowns. However, the greedy machine assignment performed by our GSACO algorithm in Section 4 can take such conditions into account for allocating an operation to the earliest available machine. In addition, some transportation time is required to move a lot from one machine to another between operations, which is not explicitly given but taken as part of the operation duration in the SMT2020 scenarios.

A schedule allocates each operation $O_{i,j,t}$ to some machine $m \in \{1, \dots, M\}$ such that $t_m = t$, and we denote the machine assignment by $O_{i,j,t,m}$. Each machine performs its assigned operations in sequence without preemption, i.e., $s_{i,j,t,m} + d_{i,j,t} \leq s_{i',j',t,m}$ or $s_{i',j',t,m} + d_{i',j',t} \leq s_{i,j,t,m}$ must hold for the start times $s_{i,j,t,m}$ and $s_{i',j',t,m}$ of operations $O_{i,j,t,m} \neq O_{i',j',t,m}$ allocated to the same machine m . The precedence between operations of a job $j \in \{1, \dots, J\}$ needs to be respected as well, necessitating that $s_{i,j,t,m} + d_{i,j,t} \leq s_{i+1,j,t',m'}$ when $i < n_j$. Assuming that $0 \leq s_{1,j,t,m}$ for each job $j \in \{1, \dots, J\}$, the makespan to complete all jobs is given by $\max\{s_{n_j,j,t,m} + d_{n_j,j,t} \mid j \in \{1, \dots, J\}\}$. We take makespan minimization as the optimization objective for scheduling, since it reflects efficient machine utilization and maximization of fab throughput.

An example schedule (generated by GSACO) for three jobs with three operations each is displayed in Figure ???. The start times for the operations of job 1 illustrate the scheduling constraints. That is, we have $s_{1,1,3,5} + d_{1,1,3} = 0 + 10 = s_{2,1,4,7}$ due to the precedence between the first and second operation, while the machine 7 performing the second operation is already available at time 9. Regarding the start of the third operation, $s_{2,1,4,7} + d_{2,1,4} = 10 + 2 < 13 = s_{3,1,1,1}$, i.e., the third operation needs to wait for machine 1 to complete the execution of another operation (of job 2). The completion time $s_{3,3,1,2} + d_{3,3,1} = 12 + 13$ of the third operation of job 3 constitutes the makespan 25 of the example schedule in Figure ??.

Input: dataset, l
Output: best schedule found by ants
Parameters: $n, k, \tau_y, \tau_z, \rho, c$
Initialize adjacency, pheromone, and machine matrix;
 $makespan \leftarrow \infty$;
while cycle or time limit l is not reached **do**
 foreach ant **from** 1 **to** k **do**
 | Run GS procedure to find a schedule;
 end
 $new \leftarrow$ shortest makespan of ants' schedules;
 if $new < makespan$ **then**
 | $makespan \leftarrow new$;
 | $best \leftarrow$ an ant's schedule of $makespan$;
 end
 foreach edge e in pheromone matrix **do**
 | $\tau_e \leftarrow \max\{\rho \cdot \tau_e, \tau_z\}$; // evaporation
 end
 foreach edge e selected by best ant **do**
 | $\tau_e \leftarrow \tau_e + c$; // deposit pheromones
 end
end
return $best$;

Algorithm 1: Greedy Search based ACO (GSACO)

4 GSACO Algorithm

The framework of our GSACO algorithm, whose (input and internal) parameters are summarized in Table 2, is displayed in Figure ?? . Its four submodules, indicated in bold, are described in separate subsections below.

Moreover, Algorithm 1 provides a pseudo-code representation of GSACO. For a configurable cycle number or time limit l , each of the k ants applies greedy search using the GS procedure (detailed in Algorithm ??). That is, the first GS phase constructs an operation sequence, which is then taken as basis for greedily assigning the operations to machines in the second phase. Note that the ants run independently, so that their GS trials can be performed in parallel. As a result, k schedules along with edges between operations (described in Subsection 4.2) that have been selected for their construction are obtained. If some of these schedules improves the makespan over the best schedule found in previous iterations (if any), the best schedule gets updated. As common for ACO algorithms, pheromones τ_e on edges e are subject to evaporation, according to the formula $\rho \cdot \tau_e$, while edges selected to construct the best schedule obtained so far also receive a pheromone contribution, calculated as $\tau_e + c$. Such pheromone deposition increases the chance for edges contributing to the current best schedule to get re-selected in forthcoming iterations.

4.1 Input Module

This module reads in an SMSP instance, e.g., obtained from the SMT2020 dataset [?], specifying production routes, the tool groups with their machines, and the jobs to be performed. Moreover, a limit l on the number of cycles and/or the time to spend on optimization by GSACO is given as input.

4.2 Initialization Module

In view of long production routes with hundreds of operations in the SMT2020 dataset, we introduce a configurable planning horizon n as upper bound on the length n_j of the operation sequence for a job j . The planning horizon thus constitutes a scaling factor for the size and the resulting complexity of SMSP instances. In practice, unpredictable stochastic events make long-term schedules obsolete and necessitate frequent re-scheduling, where limiting the planning horizon upfront provides a means to control the search and enable short response times.

To express SMSP as a search problem on graphs, we identify an instance with the disjunctive graph

whose vertices V contain the operations $O_{i,j,t}$ plus a dummy start node 0, conjunctive edges

$$E_c = \{(0, O_{1,j,t_1}) \mid O_{1,j,t_1} \in V\} \cup \{(O_{i-1,j,t_{i-1}}, O_{i,j,t_i}) \mid O_{i,j,t_i} \in V, i > 1\} \quad (1)$$

connect the dummy start node 0 to the first operation and each operation on to its successor (if any) in the sequence for a job, and disjunctive edges

$$E_d = \{(O_{i,j,t}, O_{i',j',t}) \mid O_{i,j,t} \in V, O_{i',j',t} \in V, j \neq j'\} \quad (2)$$

link operations (of distinct jobs) sharing a common tool group, as such operations may be allocated to the same machine.

Any feasible schedule induces an acyclic subgraph (V, E) of the disjunctive graph G such that $E_c \subseteq E$, and $(O_{i,j,t}, O_{i',j',t}) \in E_d \cap E$ iff $s_{i,j,t,m} + d_{i,j,t} < s_{i',j',t,m}$ for distinct jobs $j \neq j'$, i.e., the operation $O_{i,j,t}$ is processed before $O_{i',j',t}$ by the same machine m of tool group $t_m = t$. Conversely, the search for a high-quality solution can be accomplished by determining an acyclic subgraph (V, E) of G that represents a schedule of short makespan.

For example, Table 3 shows nine operations belonging to the operation sequences for three jobs, as they can be obtained with the parameter $n = 3$ for the planning horizon. Conjunctive edges connect the dummy start node 0 to the operations numbered 1, 4, and 7, which come first in their jobs, then operation 1 is connected on to 2 as well as 2 to 3, and similarly for the other two jobs. In addition, mutual disjunctive edges link operations to be processed on the same tool group, e.g., those numbered 4 and 7 have the tool group *Diffusion_FE_120* in common. The resulting $(N + 1) \times (N + 1)$ adjacency matrix, where $N = 9$ is

Table 3. Example operations

No.	Operation	Tool group name	Duration
1	$O_{1,1,3}$	Diffusion_FE_125	10
2	$O_{2,1,4}$	WE_FE_84	2
3	$O_{3,1,1}$	DefMet_FE_118	6
4	$O_{1,2,2}$	Diffusion_FE_120	8
5	$O_{2,2,4}$	WE_FE_84	1
6	$O_{3,2,1}$	DefMet_FE_118	4
7	$O_{1,3,2}$	Diffusion_FE_120	9
8	$O_{2,3,4}$	WE_FE_84	3
9	$O_{3,3,1}$	DefMet_FE_118	13

$$\begin{array}{c}
0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \\
\begin{array}{l}
0 \left[\begin{array}{cccccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0
\end{array} \right] \\
1 \left[\begin{array}{cccccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array} \right] \\
2 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{array} \right] \\
3 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{array} \right] \\
4 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0
\end{array} \right] \\
5 \left[\begin{array}{cccccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0
\end{array} \right] \\
6 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1
\end{array} \right] \\
7 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
\end{array} \right] \\
8 \left[\begin{array}{cccccccccc}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array} \right] \\
9 \left[\begin{array}{cccccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0
\end{array} \right]
\end{array}
\end{array}$$

Fig. 1. Adjacency matrix for the operations in Table 3

the total number of operations, 0 entries indicate the absence, and 1 entries the existence of edges, is given in Figure 1.

As initial pheromone level on edges $e \in E_c \cup E_d$, we take $\tau_y = 1$ by default. In general, representing pheromone levels by an $(N+1) \times (N+1)$ matrix similar to the adjacency matrix, the entries τ_e are initialized according to the following condition:

$$\tau_e = \quad (3)$$

With $\tau_y = 1$, this reproduces the adjacency matrix in Figure 1 as initial pheromone matrix for our example.

We additionally represent the possible machine assignments by an $(N+1) \times M$ machine matrix, where M is the total number of machines. For example, with two machines per tool group and the mapping $t_m = \lceil \frac{m}{2} \rceil$ from machine identifiers $m \in \{1, \dots, 8\}$ to the tool groups $t \in \{1, \dots, 4\}$, responsible for processing the operations in Table 3, we obtain the machine matrix shown in Figure 2. While the dummy start node 0 needs no machine to process it, the operation $O_{1,1,3}$ numbered 1 can be allocated to either the machine 5 or 6 of tool group 3, and corresponding 1 entries indicate the machines available to perform the other operations as well.

	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	1	0	0
2	0	0	0	0	0	0	1	1
3	1	1	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0
5	0	0	0	0	0	0	1	1
6	1	1	0	0	0	0	0	0
7	0	0	1	1	0	0	0	0
8	0	0	0	0	0	0	1	1
9	1	1	0	0	0	0	0	0

Fig. 2. Machine matrix for the operations in Table 3

4.3 GS Module

The general goal of greedy search methods consists of using heuristic decisions to find high-quality, but not necessarily optimal solutions in short time. Within GSACO, each ant applies greedy search to efficiently construct some feasible schedule for a given SMSP instance. The respective GS procedure, outlined by the pseudo-code in Algorithm ??, includes two phases: operation sequencing and machine assignment.

The first phase constructs a sequence comprising all operations of an SMSP instance. To this end, a probabilistic decision rule based on the pheromone matrix selects edges $(O', O_{i,j,t})$ and adds their target operations $O_{i,j,t}$ to the sequence one by one. As invariant ensuring the feasibility of a resulting schedule, the predecessor operation $O_{i-1,j,t'}$ of the same job j must already belong to the sequence in case $i > 1$. This is accomplished by maintaining a set *next* of selectable conjunctive and disjunctive edges $(O', O_{i,j,t})$ such that O' is the dummy start node 0 or already in sequence, while $O_{i,j,t}$ is the first yet unsequenced operation of its job j .

For example, starting with an empty sequence for the operations in Table 3, the initially selectable edges are $(0, O_{1,1,3})$, $(0, O_{1,2,2})$, and $(0, O_{1,3,2})$. Assuming that $(0, O_{1,2,2})$ gets selected, the first operation $O_{1,2,2}$ of job 2 is added to the sequence, and the conjunctive edge $(O_{1,2,2}, O_{2,2,4})$ as well as the disjunctive edge $(O_{1,2,2}, O_{1,3,2})$ replace $(0, O_{1,2,2})$ among the selectable edges. The process of selecting some edge to a yet unsequenced operation is repeated until each operation along with an edge targeting it has been processed. Note that selected disjunctive edges link operations based on tool groups, i.e., they do not reflect a machine assignment to be made in the second phase.

With a sequence of operations at hand, the second phase allocates operations one by one to an earliest available machine. Reconsidering the operations $O_{1,2,2}$ and $O_{1,3,2}$ as well as the machine matrix in Figure 2, where these operations are numbered 4 and 7, we can first allocate $O_{1,2,2}$ to the free machine 3 with the start time $s_{1,2,2,3} = 0$. This means that machine 3 becomes available again at time $a_3 = s_{1,2,2,3} + d_{1,2,2} = 8$. However, the other machine 4 of tool group 2 is still free, so that $O_{1,3,2}$ is greedily assigned to machine 4 with the start time

	0	1	2	3	4	5	6	7	8	9
0	0	1.3	0	0	1.3	0	0	1.3	0	0
1	0	0	1.3	0	0	0	0	0	0	0
2	0	0	0	1.3	0	4.9	0	0	4.9	0
3	0	0	0	0	0	0	4.9	0	0	4.9
4	0	0	0	0	0	1.3	0	4.9	0	0
5	0	0	4.9	0	0	0	1.3	0	4.9	0
6	0	0	0	4.9	0	0	0	0	0	4.9
7	0	0	0	0	4.9	0	0	0	1.3	0
8	0	0	4.9	0	0	4.9	0	0	0	1.3
9	0	0	0	4.9	0	0	4.9	0	0	0

Fig. 3. Updated pheromone matrix for the operations in Table 3 after a few GSACO iterations

$s_{1,3,2,4} = 0$ and the updated availability $a_4 = s_{1,3,2,4} + d_{1,3,2} = 9$. As we have that $a_3 = 8 < 9 = a_4$, if a third operation were to be allocated to either the machine 3 or 4 of tool group 2, our greedy assignment strategy would decide for machine 3. The described machine assignment process allocates operations according to the sequence from the first phase, yielding a feasible schedule that assigns the machines and start times for all operations.

Figure ?? displays a sequence for the operations in Table 3, where sequence numbers indicate the order in which edges to the operations are selected, as it can be generated by the GS procedure. The edges running vertically are conjunctive and connect the dummy start node 0 to the first operations of the three jobs as well as predecessors to their successor operations. In addition, two disjunctive edges are selected between the second or third operation, respectively, of job 2 and the corresponding operation of job 1, considering that these operations have the tool group 4 or 1 in common.

The resulting greedy machine assignment, computed in the second phase of the GS procedure, is denoted by m within the node labels $O_{i,j,t,m}$ in Figure ?. In view of two machines available per tool group and three operations sharing each of the tool groups 1 and 4, the first two operations according to the sequence from the first phase, i.e., $O_{2,2,4}$ and $O_{2,3,4}$ or $O_{3,2,1}$ and $O_{3,3,1}$, respectively, get distributed among the two machines of each tool group, which leads to the machine assignments $O_{2,2,4,7}$ and $O_{2,3,4,8}$ as well as $O_{3,2,1,1}$ and $O_{3,3,1,2}$. Which machines of the tool groups 4 and 1 then become available first to allocate the operations $O_{2,1,4}$ and $O_{3,1,1}$ can be read off the corresponding schedule shown in Figure ?. That is, the machine 7 of tool group 4 is available from time 9, yielding the greedy assignment $O_{2,1,4,7}$. Similarly, $O_{3,1,1,1}$ is obtained because the machine 1 gets free at time 13, while the other machine 2 of tool group 1 processes the third operation of job 3 until time 25, which is the makespan of the constructed schedule. Given that this makespan coincides with the cumulative duration of job 3 and its operations must be processed in sequence, the feasible schedule in Figure ? happens to be optimal for the operations in Table 3.

4.4 Evaluation Module

While each ant independently constructs a feasible schedule by means of the GS procedure, the evaluation module collects the results obtained by the ants in a GSACO iteration. Among them, a schedule of shortest makespan is determined as outcome of the iteration and stored as new best solution in case it improves over the schedules found in previous iterations (if any). After evaporating pheromones τ_e by $\rho \cdot \tau_e$, where τ_z remains as minimum pheromone level if the obtained value would be smaller, the edges e that have been selected by the GS procedure for constructing the current best schedule receive a pheromone contribution and are updated to $\tau_e + c$.

Note that our contribution parameter c is a constant, while approaches in the literature often take the inverse of an objective value [?], i.e., of the makespan in our case. The latter requires careful scaling to obtain non-marginal pheromone contributions, in particular, when makespans get as large as for SMSP instances. We instead opt for pheromone contributions such that the edges selected to construct best schedules are certain to have an increased chance of getting re-selected in forthcoming iterations. For our example in Table 3, Figure 3 provides the updated pheromone matrix after a few iterations of GSACO with the parameter values $\rho = 0.7$ and $c = 0.5$, showing a clear separation between the more frequently selected and futile edges.

5 EXPERIMENTAL EVALUATION

We implemented our GSACO algorithm in Python using PyTorch, as it handles tensor operations efficiently and provides a multiprocessing library for parallelization, thus significantly speeding up the ants’ execution of the GS procedure in each GSACO iteration.⁴ The first challenge consists of determining suitable values for the input parameters listed in Table 2, i.e., all parameters but the internally calculated pheromone level τ_e on edges e . We commit to the values given in Table 4, where a time limit l of 10 minutes and a planning horizon n of up to 5 operations per job are plausible for SMSP instances based on the SMT2020 dataset, whose stochastic events necessitate frequent re-scheduling in practice. For the initial pheromone level τ_y , we start from value 1, and take 0.00001 as the minimum τ_z to avoid going down to 0, considering that the GS procedure can only select edges with non-zero entries in the pheromone matrix. The values for the number k of ants, the evaporation rate ρ , and the pheromone contribution c are more sophisticated to pick. That is, we tuned these parameters in a trial-and-error process that, starting from a baseline, inspects deviations of the final makespan and convergence speed obtained with iterative modifications. Certainly, an automated approach would be desirable to perform this task efficiently for new instance sets.

⁴ The source code is publicly available in our GitHub repository: <https://github.com/prosysscience/GSACO>

Table 4. GSACO input parameter values

Parameter	Value
l	10
n	1–5
k	10
τ_y	1
τ_z	0.00001
ρ	0.7
c	0.5

To compare GSACO with the state of the art in FJSSP solving, we also run the CP solver OR-Tools (version 9.5) [?], while heuristic and meta-heuristic methods from the literature could not be reproduced due to the inaccessibility of source code or tuned hyperparameters. Note that CP approaches have been shown to be particularly effective among exact optimization techniques for FJSSP [?], where the free and open-source solver OR-Tools excels as serial winner of the MiniZinc Challenge.⁵

We performed our experiments on a TUXEDO Pulse 14 Gen1 machine equipped with an 8-core AMD Ryzen 7 4800H processor at 2.9GHz and onboard Radeon graphics card.

Table 5 illustrates the strengths of CP models on a benchmark set of small to medium-scale FJSSP instances [?], where the instances MK1–MK4 are due to [?] and the remaining eight instances have been introduced in [?]. In view of the small numbers J and M of jobs or machines, respectively, in these classical FJSSP instances, OR-Tools solves all of them to optimality within a few seconds, so that the CP column shows the minimal makespan for each instance. Since GSACO is an approximation method, its best schedules are not necessarily optimal, as it can be observed on instances other than SFJSSP1–SFJSSP4. This reconfirms that exact models, particularly those based on CP [?], are the first choice for FJSSP instances of moderate size and complexity.

To evaluate large-scale SMSP instances, we consider two semiconductor production scenarios of the SMT2020 dataset: Low-Volume/High-Mix (LV/HM) and High-Volume/Low-Mix (HV/LM). As indicated in Table 6, both scenarios include more than 2000 jobs and more than 1300 machines, modeling the production processes of modern semiconductor fabs. The main difference is given by the number of products and associated production routes for jobs, where LV/HM considers 10 production routes varying between 200–600 steps in total, while HV/LM comprises 2 production routes with about 300 or 600 steps, respectively. Originally, the LV/HM and HV/LM scenarios have been designed to represent fab load at the start of simulation runs, so that the jobs are at different steps of their production routes. We here instead focus on scheduling for planning horizons n from 1 to 5, standing for the up to 5 next operations to be performed per job. Hence, the operations O to schedule gradually increase from

⁵ <https://www.minizinc.org/challenge>

Table 5. FJSSP results obtained with CP and GSACO

Instance	J	M	CP	GSACO
MK1	10	6	40	44
MK2	10	6	27	40
MK3	15	8	204	239
MK4	15	8	60	83
SFJSSP1	2	2	66	66
SFJSSP2	2	2	107	107
SFJSSP3	3	2	221	221
SFJSSP4	3	2	355	355
MFJSSP1	5	6	468	498
MFJSSP2	5	7	446	470
MFJSSP3	6	7	466	523
MFJSSP4	7	7	554	664

Table 6. Number of jobs, machines, and operations for SMSP instances

Scenario	J	M	O
LV/HM	2156	1313	up to 10747
HV/LM	2256	1443	up to 11218

the number J of jobs, in case of the planning horizon $n = 1$, to more than 10000 operations for the longest horizon $n = 5$.

While running with a time limit l of 10 minutes, Table 7 reports the makespans of current best schedules found by the CP solver OR-Tools and our GSACO implementation at 1, 3, 5, 7, and 9 minutes of computing time. Considering that the SMSP instances are large, OR-Tools now takes 3 minutes to come up with the first solution(s) for the shortest planning horizon $n = 1$ on the LV/HM scenario. Within the same fraction of computing time, GSACO already converges to a makespan of 3725 and then proceeds with iterations that do not yield further improvements. That the best schedule found by GSACO is not optimal is witnessed by a marginally better solution obtained by OR-Tools after 7 minutes. However, for the other SMSP instances, OR-Tools is unable to improve over GSACO within 9 minutes, and it cannot even provide feasible schedules for planning horizons from $n = 2$ on the HV/LM scenario or $n = 3$ on LV/HM.

The quick convergence of our GSACO implementation is also outlined by the makespan improvements plotted in Figure ???. On the LV/HM scenario displayed in Figure ??, GSACO obtains its best schedules within 7 minutes for all planning horizons. Only for the longest planning horizon $n = 5$ on the HV/LM scenario in Figure ??, an improvement occurs after more than the 9 minutes of computing time listed in the rightmost column of Table 7. Comparing the SMSP instances for which OR-Tools manages to provide feasible schedules, the convergence to makespans in roughly the range of GSACO's results takes significantly more computing time. Hence, the time limit that would be necessary to break even with GSACO, as accomplished within 7 minutes for the shortest planning horizon $n = 1$ on the LV/HM scenario, cannot be predicted for the SMSP instances

Table 7. SMSP results obtained with CP and GSACO

n	Scenario	1 min		3 min		5 min		7 min		9 min	
		CP	GSACO	CP	GSACO	CP	GSACO	CP	GSACO	CP	GSACO
1	LV/HM	-	3735	18572	3725	3746	3725	3723	3725	3723	3725
	HV/LM	-	1405	-	1405	2242	1405	1609	1405	1600	1405
2	LV/HM	-	3773	-	3751	-	3750	-	3739	4398	3739
	HV/LM	-	1653	-	1644	-	1611	-	1611	-	1611
3	LV/HM	-	3880	-	3867	-	3836	-	3834	-	3834
	HV/LM	-	1902	-	1889	-	1889	-	1876	-	1876
4	LV/HM	-	4578	-	4540	-	4540	-	4540	-	4540
	HV/LM	-	2207	-	2113	-	2113	-	2093	-	2093
5	LV/HM	-	4680	-	4680	-	4553	-	4553	-	4553
	HV/LM	-	2667	-	2566	-	2566	-	2518	-	2518

with longer planning horizons. Moreover, we observe that the initial schedules obtained in the first GSACO iteration by some of the ants running in parallel are of relatively high quality, while OR-Tools sometimes finds outliers as its first solutions. This phenomenon occurs for the planning horizons $n = 1$ and $n = 2$ on the LV/HM or HV/LM scenario, respectively, where the latter schedule of makespan 17664 is found after more than 9 minutes and thus not listed in Table 7.

6 CONCLUSION

Modern semiconductor fabs are highly complex and dynamic production environments, whose efficient operation by means of automated scheduling and control systems constitutes a pressing research challenge. In this work, we model the production processes of large-scale semiconductor fabs in terms of the well-known FJSSP. In contrast to classical FJSSP benchmarks, this leads to large-scale scheduling problems, even if short planning horizons cover only a fraction of the long production routes with hundreds of operations. The resulting size and complexity of large-scale SMSP instances exceed the capabilities of common FJSSP solving methods. We thus propose the GSACO algorithm combining probabilistic operation sequencing with greedy machine assignment. Its efficient implementation utilizing tensor operations and parallel ant simulations enables GSACO’s convergence to high-quality solutions in short time. In this way, GSACO overcomes the scalability limits of exact optimization by means of a state-of-the-art CP approach and achieves the performance required for frequent re-scheduling in reaction to process deviations.

While simplified FJSSP models of semiconductor manufacturing processes provide insights regarding the scalability of scheduling methods, in future work, we aim at extending GSACO to incorporate the specific conditions and functionalities of tools performing the production operations. Such features include, e.g., machine setups to be equipped before performing operations, preventive maintenance procedures for which a machine needs to stay idle, and batch processing capacities to handle several production lots in one pass. In practice, these

specifics matter for machine assignment strategies and should also be reflected in the respective greedy phase of GSACO.

Our second target of future work concerns the utilization of schedules found by GSACO for decision making and control within simulation models of semiconductor fabs. This goes along with the extension of optimization objectives to practically relevant performance indicators, such as deadlines for the completion of jobs and minimizing the tardiness. In view of the dynamic nature of real-world production operations and unpredictable stochastic events, quantifying the improvements by optimized scheduling methods requires their integration and evaluation in simulation.

ACKNOWLEDGEMENTS

This work has been funded by the FFG project 894072 (SwarmIn). We are grateful to the anonymous reviewers for constructive comments that helped to improve the presentation of this paper.

7 Add-on

Dispatching and scheduling are critical in semiconductor manufacturing because they directly impact the throughput and utilization of resources. Dispatching refers to the process of assigning work to specific machines in real-time, while scheduling involves pre-planning the sequence and timing of operations to optimize certain objectives, such as minimizing the total completion time or balancing the load across different machines [1].

For smaller volumes at workcenters in the fab, which are modeled as flexible job shops, optimal solutions can be achieved using mathematical optimization [2]. However, in larger and more dynamic environments, the complexity and computational time constraints limit the feasibility of applying mathematical optimization. Consequently, optimization is typically implemented on a local scale, isolated to individual workcenters. In complex job shop scenarios, this localized approach to production scheduling may lead to solutions that are globally sub-optimal.

The complexity of scheduling is intensified by the unpredictable nature of processing times and machine availability. Variability in processing times may arise from differences in equipment performance, material handling durations, or the unique characteristics of each set of wafers. Furthermore, frequent machine breakdowns can lead to substantial disruptions, underscoring the need for strong and adaptable scheduling strategies [3].

The objectives of this paper are to:

1. Compare the dispatcher and scheduler over the planning horizon
2. Scheduling under different machine availabilities

7.1 A Subsection Sample

Please note that the first paragraph of a section or subsection is not indented. The first paragraph that follows a table, figure, equation etc. does not need an indent, either.

Subsequent paragraphs, however, are indented.

Sample Heading (Third Level) Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.

Sample Heading (Fourth Level) The contribution should contain no more than four levels of headings. Table 8 gives a summary of all heading levels.

Table 8. Table captions should be placed above the tables.

Heading level	Example	Font size and style
Title (centered)	Lecture Notes	14 point, bold
1st-level heading	1 Introduction	12 point, bold
2nd-level heading	2.1 Printing Area	10 point, bold
3rd-level heading	Run-in Heading in Bold. Text follows	10 point, bold
4th-level heading	<i>Lowest Level Heading.</i> Text follows	10 point, italic

Displayed equations are centered and set on a separate line.

$$x + y = z \tag{4}$$

Please try to avoid rasterized images for line-art diagrams and schemas. Whenever possible, use vector graphics instead (see Fig. 4).

Theorem 1. *This is a sample theorem. The run-in heading is set in bold, while the following text appears in italics. Definitions, lemmas, propositions, and corollaries are styled the same way.*

Proof. Proofs, examples, and remarks have the initial word in italics, while the following text appears in normal font.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [5], an LNCS chapter [6], a book [7], proceedings without editors [8], and a homepage [9]. Multiple citations are grouped [5–7], [5, 7–9].

Acknowledgments. A bold run-in heading in small font size at the end of the paper is used for general acknowledgments, for example: This study was funded by X (grant number Y).

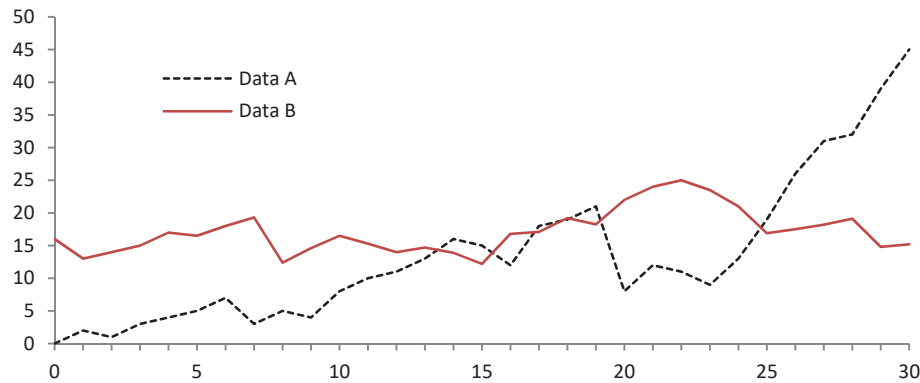


Fig. 4. A figure caption is always placed below the illustration. Please note that short captions are centered, while long ones are justified by the macro package automatically.

Disclosure of Interests. It is now necessary to declare any competing interests or to specifically state that the authors have no competing interests. Please place the statement with a bold run-in heading in small font size beneath the (optional) acknowledgments⁶, for example: The authors have no competing interests to declare that are relevant to the content of this article. Or: Author A has received research grants from Company W. Author B has received a speaker honorarium from Company X and owns stock in Company Y. Author C is a member of committee Z.

References

1. Pinedo, M. L.: Scheduling: theory, algorithms, and systems. 6th edn. Springer (2022)
2. Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A.: Deep reinforcement learning for semiconductor production scheduling. In 2018 29th annual SEMI advanced semiconductor manufacturing conference (ASMC) pp. 301-306. IEEE, (2018, April)
3. Leachman, R. C., Hodges, D. A.: Benchmarking semiconductor manufacturing. IEEE transactions on semiconductor manufacturing 9(2), 158-169 (1996)
4. McKay, K. N., Wiers, V. C.: Planning, scheduling and dispatching tasks in production control. Cognition, Technology & Work, 5, 82-93 (2003)
5. Author, F.: Article title. Journal 2(5), 99-110 (2016)
6. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1-13. Springer, Heidelberg (2016). <https://doi.org/10.1007/1234567890>
7. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
8. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1-2. Publisher, Location (2010)
9. LNCS Homepage, <http://www.springer.com/lncs>, last accessed 2023/10/25

⁶ If EquinOCS, our proceedings submission system, is used, then the disclaimer can be provided directly in the system.