Swarm Intelligence-Driven Dispatching Rules for Large-Scale Semiconductor Production: Integrating Simulation and Optimization to Enhance Operational Efficiency

Ramsha Ali[1 [0000−0002−4794−6560]], Peyman Eftekhari[1 [0009−0007−4198−392X]], Martin Gebser[1 [0000−0002−8010−4752]], Stephan Leitner[2 [0000−0001−6790−4651]], and Gerhard Friedrich[1 [0000−0002−1992−4049]]

[1] Department of Artificial Intelligence and Cybersecurity,
[2] Department of Management Control and Strategic Management,
University of Klagenfurt, Austria
{ramsha.ali,peyman.eftekhari,martin.gebser,
stephan.leitner,gerhard.friedrich}@aau.at
https://www.aau.at

Abstract. The escalating complexity of semiconductor production, characterized by high-volume, low-mix and low-volume, high-mix environments, demands advanced dispatching strategies that transcend traditional rule-based systems. This paper introduces a novel set of dispatching rules derived from Swarm Intelligence techniques, specifically designed to tackle the intricate dynamics of large-scale semiconductor manufacturing. The research integrates simulation and optimization methodologies to investigate and enhance operational efficiency, addressing both the scalability of solutions and their practical implementation. We employ a customizable simulation framework to model a semiconductor manufacturing environment, wherein various dispatching rules, including FIFO, CR, RANDOM, and our proposed SI-driven method, are assessed. The core of our approach lies in its ability to adaptively reconfigure scheduling priorities based on real-time production dynamics, thus significantly improving the responsiveness and throughput of manufacturing processes. The effectiveness of these dispatching rules is quantitatively evaluated through a series of simulations that measure key performance indicators such as Work in Progress (WIP) levels, throughput, and operational variability across different production scenarios. This study not only elucidates the potential of swarm intelligence in refining production dispatching but also provides a framework that incorporates scalable simulation tool that can assist in the further development of intelligent dispatching systems.

Keywords: Swarm intelligence · Ant colony optimization · Greedy search · Semiconductor production · Scheduling · Dispatching · Simulation

1 Introduction

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 [4].

For smaller volumes at workcenters in the semiconductor fab, which can be modeled as flexible job shops, optimal solutions can be achieved using mathematical optimization [12]. 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 scenarios, this localized approach to production scheduling may lead to solutions that are globally sub-optimal.

Dispatching involves determining which job should be performed at which time, by which resource. This decision is based on the availability of resources and the urgency of job orders [2]. However, as production scenarios become more complex, particularly in high-tech industries such as semiconductor manufacturing, the static nature of traditional dispatching rules often proves insufficient. These rules typically do not account for the rapid changes in production conditions or the intricate dependencies between different production processes. Traditional dispatching solutions often fall short in such an environment, as they cannot adequately adapt to the dynamic and complex nature of semiconductor production. This inadequacy can lead to sub-optimal decisions that compromise operational efficiency [3].

The complexity of scheduling in large-scale semiconductor production, a cornerstone of operational management, is significantly influenced by the unpredictable nature of processing times and machine availabilities. Variability in processing times can arise from differences in equipment performance, material handling durations, or the unique characteristics of each set of wafers, while frequent machine breakdowns can cause substantial disruptions, underscoring the need for strong and adaptable scheduling strategies [13]. Scheduling involves meticulously planning and organizing production activities to ensure optimal resource utilization and synchronized product flows across various manufacturing stages, which is crucial not only for maintaining high throughput but also for minimizing makespan [4]. The task is particularly daunting in semiconductor manufacturing due to the high variability in production processes, the sensitive nature of the materials, and stringent quality requirements [5]. Each semiconductor product may pass through hundreds of processing steps, requiring precise timing and coordination. The complexity is further compounded by the need to accommodate a high mix of product types, each with distinct processing needs and priorities, and to integrate new product introductions seamlessly into the production schedule without disrupting ongoing operations [6].

In this paper, we explore the complexities of scheduling within large-scale Semiconductor Manufacturing Scheduling Problems (SMSP), commonly referred to as fabs. Fabs are intricate production settings distinguished by their specialized job flows and advanced machinery. Modern semiconductor fabs manage the processing of tens of thousands of operations daily across over 1000 different machines [1]. These machines are grouped by functionality—such as diffusion, etching, or metrology—and each step in the manufacturing process can be assigned to any machine that offers the neces-

sary capabilities. The scheduling challenge in semiconductor production thus involves assigning specific operations to appropriate machines and determining the optimal sequence of these operations on each machine. Additionally, schedules must be flexible and capable of rapid adjustments in response to machine failures or deviations in the process.

In our previous work [7], we proposed a Greedy Search based Ant Colony Optimization (GSACO) algorithm for (re-)scheduling semiconductor production operations. Our algorithm harnesses Ant Colony Optimization (ACO) [8] for exploration, while Greedy Search (GS) [9] enables responses in short time. In this way, GSACO overcomes limitations of a state-of-the-art Constraint Programming approach [10] on large-scale SMSP instances. Our approach synergistically combines probabilistic operation sequencing with a greedy machine assignment strategy, targeting up to five operations per lot with the primary objective of minimizing makespan. Building on this foundational work, the current paper extends these initial concepts by proposing an enhanced algorithm, GSACO-I, specifically designed to optimize operational throughput in addition to minimizing makespan. This development marks a significant advancement in our methodology, aiming to refine the dispatching rules further though simulation.

The paper is organized as follows. Section 2 provides literature review. In Section 3, we formulate SMSP in terms of the well-known flexible job shop scheduling problem (FJSSP). Our GSACO-I algorithm is presented in Section 4. In Section 5 we present the customized simulation adopted from [11]. Section 6 provides and discusses experimental results. Finally, Section 7 concludes the paper.

2 Literature Review

We briefly survey the relevant literature on the challenges inherent in SMSP, a sector marked by its complex operational dynamics and the critical need for highly efficient dispatching systems. Semiconductor production processes are characterized by their high variability, sophisticated product mixes, and stringent quality demands, which collectively necessitate innovative approaches to production scheduling and dispatching. Later, we survey the relevant literature on methods for FJSSP, which we use as general scheduling model to represent semiconductor production processes and SMSP. Then we explore the traditional dispatching methods detailing their advantages and limitations within the context of semiconductor fabrication facilities. These conventional methods form the baseline against which the efficacy of advanced dispatching techniques, particularly those driven by Swarm Intelligence (SI), are evaluated. Furthermore, the review will cover essential performance metrics traditionally used to assess the effectiveness of dispatching rules.

2.1 Semiconductor Manufacturing Scheduling Problem (SMSP)

Semiconductor manufacturing is characterized by its complex multi-step fabrication process, each involving highly precise and controlled conditions. The fabrication of semiconductor devices often requires hundreds of steps during the photolithography,

etching, and chemical deposition phases, among others. This complexity is compounded by the requirement for extremely clean environments to avoid contamination of the microscopically small circuits, making operational excellence a challenging goal [5].

Unlike other manufacturing industries that produce a limited range of products, semiconductor manufacturers typically deal with a high mix of products on the same production line. This high mix coupled with the rapid product evolution typical of the tech industry results in significant fluctuations in production volumes. Manufacturers must, therefore, be highly responsive to changes in demand and technology without compromising on the speed or quality of production .

The SMSP is a complex and critical challenge to optimize scheduling tasks during the manufacturing process of a semiconductor. Different methods have been adopted for a long time for varied types of machines [15]. The core aspects include job scheduling, batch processing, priority constraints, setup times of the machine, machine availability, and reentrant flow. The key challenges stay with complexity, dynamic job arrivals, and reducing downtime while maximizing throughput [18]. Many researchers and developers have been working on the topic and suggested their methods to solve the problem.

2.2 Flexible Job Shop Scheduling

Considering the manufacturing industry, FJSSP is a common problem, especially for small batch and custom productions. The mathematical models allow us to solve the optimality issue for small-scale instances. As [19] explains, it is important to assign the FJSSP on a machine in a particular sequence. As the optimization criteria need the start time of all the operations, it is important to optimize the completion time also. The most suitable approach to solve this time problem depends on the function and the mathematical properties associated with it. Researchers also explained that many studies are optimizing non-regular criteria that require equal efforts for timing decisions to get the best sequencing decisions in solution approaches.

The study [20] explained that the existing methods to solve NP-hard combinatorial optimization problems are either exact or approximate. The exact methods are challenging to solve FJSSP when problems need large-size scheduling. Considering their NP-hardness, it is difficult to allocate reasonable time. [20] have also explained that FJSSP instances intractability is in constant need of more approximate methods. So many solutions like machine learning techniques, heuristics, meta-heuristics methods, and more are continuously being developed to tackle real-world problems more effectively.

2.3 Ant Colony Optimization

ACO is a meta-heuristic algorithm that mimics the foraging behavior of ants [23]. It was originally devised to solve the traveling salesperson problem [24], and meanwhile ACO has been adopted to various optimization problems, including routing [25], scheduling [26], task allocation [27], project planning [28], and network optimization [29].

ACO is a well-suited metaheuristics algorithm to solve SMSP as it is highly appropriate to handle the dynamic and complex nature of semiconductor scheduling with its

multi-objective nature [21]. ACO has a history of application to be applied to SMSP for wafer scheduling to balance the load and minimize the makespan. The dynamic nature of the ACO algorithm continuously updates the "pheromone" based on the completed jobs, and guides others to follow the optimal scheduling decisions, ultimately making the system reduce their decision-making time [22].

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 [30]. 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 [31–36]. While these methods have been designed and evaluated on small to medium-scale FJSSP benchmarks [35], our work addresses the large-scale SMSP instances encountered in the domain of semiconductor production scheduling [1]. 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 [37–41].

2.4 Dispatching

In semiconductor manufacturing, the efficient management of production flow is crucial for maintaining high throughput and minimizing delays. This section explores dispatching methods that have been traditionally employed to orchestrate the flow of work-in-process (WIP) through the various stages of production.

FIFO is one of the simplest and most widely used dispatching rules in various manufacturing sectors, including semiconductors. It prioritizes jobs in the order they arrive, regardless of their complexity or processing time. While FIFO is straightforward to implement and ensures a fair processing order, it may not be optimal in environments where job priorities vary significantly, leading to potential inefficiencies in the handling of urgent or time-sensitive processes [42].

The Critical Ratio is a more sophisticated dispatching method that prioritizes jobs based on the ratio of the remaining processing time to the time remaining until the job's due date. This method aims to minimize tardiness and is particularly useful in semiconductor manufacturing, where delivery schedules are tight, and delays can be costly. However, its effectiveness is highly dependent on accurate estimates of processing times and can become complex to manage in high-mix production environments [43].

Random dispatching assigns priorities to jobs at random, irrespective of their characteristics or deadlines. This method is often used as a benchmark in simulation studies to demonstrate the effectiveness of more sophisticated rules. While it does not optimize

Table 1: Basic notations (adapted from [7])

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

any specific performance metric, random dispatching can sometimes yield surprisingly good average performance due to its stochastic nature, though it generally lacks consistency and predictability [44].

These conventional methods provide a baseline for managing operations but often fall short in environments characterized by high variability and complex product mixes, such as in semiconductor manufacturing. Given the limitations of these conventional methods, there is a significant opportunity for advanced dispatching approaches that can adapt to the dynamic conditions of semiconductor production lines. The integration of real-time data analytics and more sophisticated algorithms, such as those derived from Swarm Intelligence, could address these gaps by providing more flexible and responsive dispatching solutions.

3 Problem Formulation

We formulate SMSP in terms of the general FJSSP model, using the basic notations listed in Table 1 (see [7]). 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 $t_m \in \{1, \ldots, T\}$ denotes the tool group to which a machine $m \in \{1, \ldots, M\}$ belongs.
- There are J jobs, where each $j \in \{1, \ldots, J\}$ represents a sequence of operations $O_{1,j,t_1}, \ldots, O_{n_j,j,t_n}$, to be performed on a production lot. Note that $t_i \in \{1, \ldots, T\}$ 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 $N = \sum_{j \in \{1,\ldots,J\}} n_j$.
- 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 incorporates key features drawn from the semiconductor production scenarios outlined in the SMT2020 dataset [1]. In these scenarios, each job corresponds to a specific product, with operation sequences—referred to as production routes—remaining consistent across the same product type. Given that these production

routes can span several months and encompass hundreds of operations within a physical fab, it's common for different lots of the same product to be at various stages of their production routes during (re-)scheduling. Therefore, our model does not differentiate production routes by product; instead, we focus on the operation sequences of a specific length n_j related to each job j. This approach facilitates the management of operations for lots at different stages within the same production route.

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 $m \in \{1, \ldots, M\}$ 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 $j \in \{1, \ldots, J\}$ 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 $j \in \{1, \ldots, J\}$, the makespan to complete all jobs is given by $\max\{s_{n_j,j,t,m} + d_{n_j,j,t} \mid j \in \{1, \ldots, J\}\}$. We take makespan minimization as the primary optimization objective for scheduling, as it reflects efficient machine utilization and maximization of fab throughput. Additionally, we aim to optimize the number of operations completed within a specified period, enhancing overall productivity and operational efficiency. This approach ensures not only the effective use of available machinery but also strives to maximize the output within time constraints, thereby optimizing operational flow and throughput within the manufacturing process.

An example schedule (generated by GSACO-I) for instance in Table 4 is displayed in Figure 4 and Figure 5.

4 GSACO Algorithm

The structure of our enhanced GSACO-I algorithm, along with its input and internal parameters, is summarized in Table 2 and illustrated in Figure 1. The four main submodules, highlighted in bold, are discussed in detail in the following subsections.

Moreover, Algorithm 1 provides a pseudo-code representation of GSACO-I for minimizing makespan and optimizing operations.

For a configurable cycle number or time limit l, each of the k ants applies greedy search using the GS procedure with respect to the objective. 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

Table 2: GSACO parameters

Parameter	Description
o	Objective
s	State
l	Cycles/time limit
n	Operations per lot
h	Planning period
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

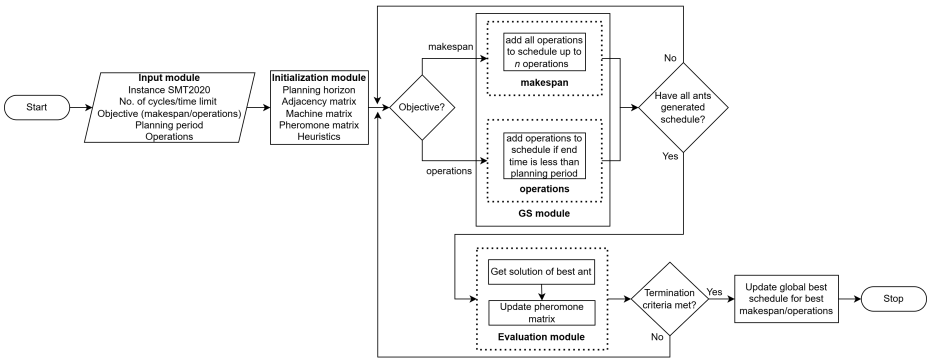

Fig. 1: GSACO-I framework

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 from the SMT2020 simulator develop by [11], and SMT2020 dataset [1]. The instance includes tool groups with their respective machines, jobs currently in progress, and production routes. In terms of dynamic state, the processing times for these routes are stochastic. Conversely, the simulator provides observed averages for processing times and job releases. Additionally, the module takes several inputs for the GSACO optimization process: the objective o, the state of the

Input: instance, l, o, s, n, h, objective
Output: best schedule found by ants
Parameters: k, τ_y, τ_z, ρ, c
Initialize adjacency, pheromone, and machine matrix;
if *objective* == *Makespan* **then**
  $makespan \leftarrow \infty$;
else
  $operations \leftarrow 0$;
end
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
  if *objective* == *Makespan* **then**
    $new \leftarrow$ shortest makespan of ants' schedules;
    if $new < makespan$ **then**
      $makespan \leftarrow new$;
      $best \leftarrow$ an ant's schedule of minimum $makespan$;
    end
  else
    $new \leftarrow$ maximum operations of ants';
    if $new > operations$ **then**
      $operations \leftarrow new$;
      $best \leftarrow$ an ant's schedule of maximum $operations$;
    end
  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) for Scheduling

fab s, planning period h, operation per lot n and a limit l on either the number of cycles or the time allocated for optimization.

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

$$\begin{aligned} E_c = &\{(0, O_{1,j,t_1}) \mid O_{1,j,t_1} \in V\} \\ &\cup \{(O_{i-1,j,t_{i-1}}, O_{i,j,t_i}) \mid O_{i,j,t_i} \in V, i > 1\} \end{aligned} \tag{1}$$

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

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

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 operations belonging to five jobs, as they can be obtained with the parameter $n = 5$ for the planning period. Conjunctive edges connect the dummy start node 0 to the operations which come first in their jobs and all operations to their successors. In addition, mutual disjunctive edges link operations to be processed on the same tool group. The resulting $(N + 1) \times (N + 1)$ adjacency matrix, where N is the total number of operations, 0 entries indicate the absence, and 1 entries the existence of edges, is given in Figure 2.

	0	1	2	…	41	42	43
0	0	1	0	…	0	0	1
1	0	0	1	…	0	0	0
2	0	0	0	…	1	0	0
⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
41	0	0	0	…	0	1	0
42	0	0	0	…	0	0	1
43	0	0	1	…	0	0	0

Fig. 2: Adjacency matrix for instance in Table 4

	1	2	3	…	10	11	12
0	0	1	0	…	0	0	0
1	0	0	1	…	0	0	0
2	0	0	0	…	1	0	0
⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
41	0	0	0	…	0	1	0
42	0	0	0	…	0	0	1
43	0	0	1	…	0	0	0

Fig. 3: Machine matrix for instance in Table 4

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:

Table 3: Example operations

No.	Operation	Tool group name	Avg process time (sec)
1	$O_{1,1,2}$	TF_Met_FE_45	588
2	$O_{2,1,3}$	WE_FE_108	59
3	$O_{3,1,4}$	WE_FE_83	62
4	$O_{4,1,5}$	WE_FE_84	54
5	$O_{5,1,1}$	LithoTrack_FE_115	130
6	$O_{1,2,2}$	TF_Met_FE_45	588
7	$O_{2,2,3}$	WE_FE_108	59
8	$O_{3,2,4}$	WE_FE_83	62
9	$O_{4,2,5}$	WE_FE_83	54
10	$O_{5,2,1}$	LithoTrack_FE_115	130
11	$O_{1,3,2}$	TF_Met_FE_45	588
12	$O_{2,3,3}$	WE_FE_108	59
13	$O_{3,3,4}$	WE_FE_83	62
14	$O_{4,3,5}$	WE_FE_84	54
15	$O_{5,3,1}$	LithoTrack_FE_115	130
16	$O_{1,4,2}$	TF_Met_FE_45	588
17	$O_{2,4,3}$	WE_FE_108	59
18	$O_{3,4,4}$	WE_FE_83	62
19	$O_{4,4,5}$	WE_FE_84	54
20	$O_{5,4,1}$	LithoTrack_FE_115	130
21	$O_{1,5,2}$	TF_Met_FE_45	588
22	$O_{2,5,3}$	WE_FE_108	59
23	$O_{3,5,4}$	WE_FE_83	62
24	$O_{4,5,5}$	WE_FE_84	54
25	$O_{5,5,0}$	LithoTrack_FE_115	115

Table 4: Example instance

Lot	Product	Step
1	1	1
2	2	2
3	3	1
4	4	3
5	5	1
1	1	2
2	2	1
3	3	3
4	4	1
5	5	2

$$\tau_e = \begin{cases} \tau_y & \text{if } e \in E_c \cup E_d \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

With $\tau_y = 1$, this reproduces the adjacency matrix in Figure 2 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 $m \in \{1, \ldots, 12\}$ to the tool groups $t \in \{1, \ldots, 6\}$, responsible for processing the remaining operations in Table 4, we obtain the machine matrix shown in Figure 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-I, 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 2 and Algorithm 3, includes two phases: operation sequencing and machine assignment.

GS Makespan To minimize makespan, the first phase constructs an ordered list of all operations within an SMSP instance, planned to schedule. This is achieved through a probabilistic decision-making rule derived from the pheromone matrix, which selects edges $(O', O_{i,j,t})$ and sequentially adds their target operations $O_{i,j,t}$ to the list. To ensure the feasibility of the resulting schedule, the prerequisite operation $O_{i-1,j,t'}$ for the same job j must already belong to the sequence in case $i > 1$. This requirement is met 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.

The selection of an edge leading to an unsequenced operation continues until every operation and its corresponding targeting edge has been processed. It is important to note that the chosen disjunctive edges connect operations based on their tool groups, meaning they do not dictate the machine assignments, which are determined in the subsequent phase.

With a complete sequence of operations available, the second phase involves the allocation of each operation to the earliest available machine. This machine assignment step follows the order established in the first phase to create a feasible schedule that assigns machines and start times to all operations.

GS Operations For optimizing operations, the first phase constructs a sequence comprising of all operations of an SMSP instance that can fit in planning period. Similarly, 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. The operations can only be added to the sequence if the ending time is within the period h. To ensure 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

Output: schedule and selected edges of an ant
$sequence \leftarrow []$;
$selected \leftarrow \emptyset$;
$next \leftarrow \{(0, O_{1,j,t}) \mid j \in \{1, \ldots, J\}\}$;
while $next \neq \emptyset$ **do**
  foreach $e \in next$ **do** $p_e \leftarrow \frac{\tau_e}{\sum_{e' \in next} \tau_{e'}}$;
  Randomly select an edge e based on p_e;
  for *selected edge* $e = (O', O_{i,j,t})$ **do**
    $sequence$.enqueue$(O_{i,j,t})$;
    $selected \leftarrow selected \cup \{e\}$;
    $next \leftarrow \{(O_1, O_2) \in next \mid O_2 \neq O_{i,j,t}\}$;
    $E \leftarrow \{(O_{i,j,t}, O_{i+1,j,t'}) \mid i < n_j\}$
      $\cup \{(O_{i,j,t}, O_{i',j',t}) \mid (O_1, O_{i',j',t}) \in next\}$; $next \leftarrow next \cup E$;
  end
end
foreach $m \in \{1, \ldots, M\}$ **do** $a_m \leftarrow 0$;
while $sequence \neq []$ **do**
  $O_{i,j,t} \leftarrow sequence$.dequeue();
  $a \leftarrow \infty$;
  foreach m **from** 1 *to* **to** M **do**
    if $t_m = t$ *and* $a_m < a$ **then**
      $a \leftarrow a_m$;
      $b \leftarrow m$;
    end
  end
  $s_{i,j,t,b} \leftarrow \max(\{a\}$
    $\cup \{s_{i-1,j,t',m} + d_{i-1,j,t'} \mid i > 1\})$; $a_b \leftarrow s_{i,j,t,b} + d_{i,j,t}$;
end
return $\langle \{s_{i,j,t,m} \mid (O', O_{i,j,t}) \in selected\}, selected \rangle$;

Algorithm 2: Greedy Search (GS) Makespan

the dummy start node 0 or already in sequence, while $O_{i,j,t}$ is the first yet unsequenced operation of its job j.

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. The 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.

The schedule for minimizing makespan is shown Figure 4, and a schedule for 15 minutes is shown in Figure 5.

Output: schedule and selected edges of an ant
$sequence \leftarrow []$;
$selected \leftarrow \emptyset$;
$period \leftarrow h$;
$next \leftarrow \{(0, O_{1,j,t}) \mid j \in \{1, \ldots, J\}\}$;
while $next \neq \emptyset$ **do**
  foreach $e \in next$ **do** $p_e \leftarrow \frac{\tau_e}{\sum_{e' \in next} \tau_{e'}}$;
  Randomly select an edge e based on p_e;
  $endtime \leftarrow e$;
  for *selected edge* $e = (O', O_{i,j,t})$ **do**
    $e \leftarrow starttime + processtime$;
    if $e < period$ **then**
      countinue
    end
    $sequence$.enqueue$(O_{i,j,t})$;
    $selected \leftarrow selected \cup \{e\}$;
    $next \leftarrow \{(O_1, O_2) \in next \mid O_2 \neq O_{i,j,t}\}$;
    $E \leftarrow \{(O_{i,j,t}, O_{i+1,j,t'}) \mid i < n_j\}$
      $\cup \{(O_{i,j,t}, O_{i',j',t}) \mid (O_1, O_{i',j',t}) \in next\}$; $next \leftarrow next \cup E$;
  end
end
foreach $m \in \{1, \ldots, M\}$ **do** $a_m \leftarrow 0$;
while $sequence \neq []$ **do**
  $O_{i,j,t} \leftarrow sequence$.dequeue();
  $a \leftarrow \infty$;
  foreach m **from** 1 *to* **to** M **do**
    if $t_m = t$ *and* $a_m < a$ **then**
      $a \leftarrow a_m$;
      $b \leftarrow m$;
    end
  end
  $s_{i,j,t,b} \leftarrow \max(\{a\}$
    $\cup \{s_{i-1,j,t',m} + d_{i-1,j,t'} \mid i > 1\})$; $a_b \leftarrow s_{i,j,t,b} + d_{i,j,t}$;
end
return $\langle \{s_{i,j,t,m} \mid (O', O_{i,j,t}) \in selected\}, selected \rangle$;

Algorithm 3: Greedy Search (GS) Operations

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 $\tau_e + c$.

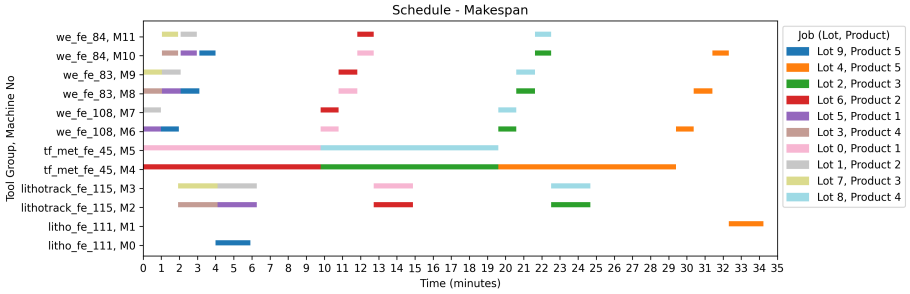

Fig. 4: Feasible schedule for the instance in Table 4

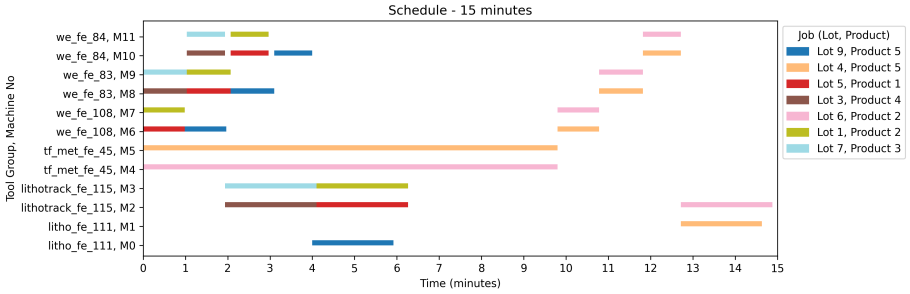

Fig. 5: Feasible 15-minutes schedule for the instance in Table 4

Note that our contribution parameter c is a constant, while approaches in the literature often take the inverse of an objective value [30], 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.

5 Simulation

To simulate a semiconductor fab, we adapted the simulator developed by [11], enhancing its capacity to handle the complexities and scalability challenges inherent in semiconductor manufacturing scheduling.

This simulator, tailored for the SMT2020 testbed, incorporates a comprehensive framework for the development and validation of innovative scheduling methods without the risk of overfitting. It operates using a dispatcher that dynamically allocates lots to machines based on predefined decision points, facilitating iterative simulation cycles that continue until designated performance benchmarks are achieved.

The dispatcher utilizes basic local rules such as FIFO, CR, and Random selection to manage lot assignments efficiently. These rules serve as foundational strategies, ensuring basic operational efficiency. Additionally, we have integrated enhanced dispatching

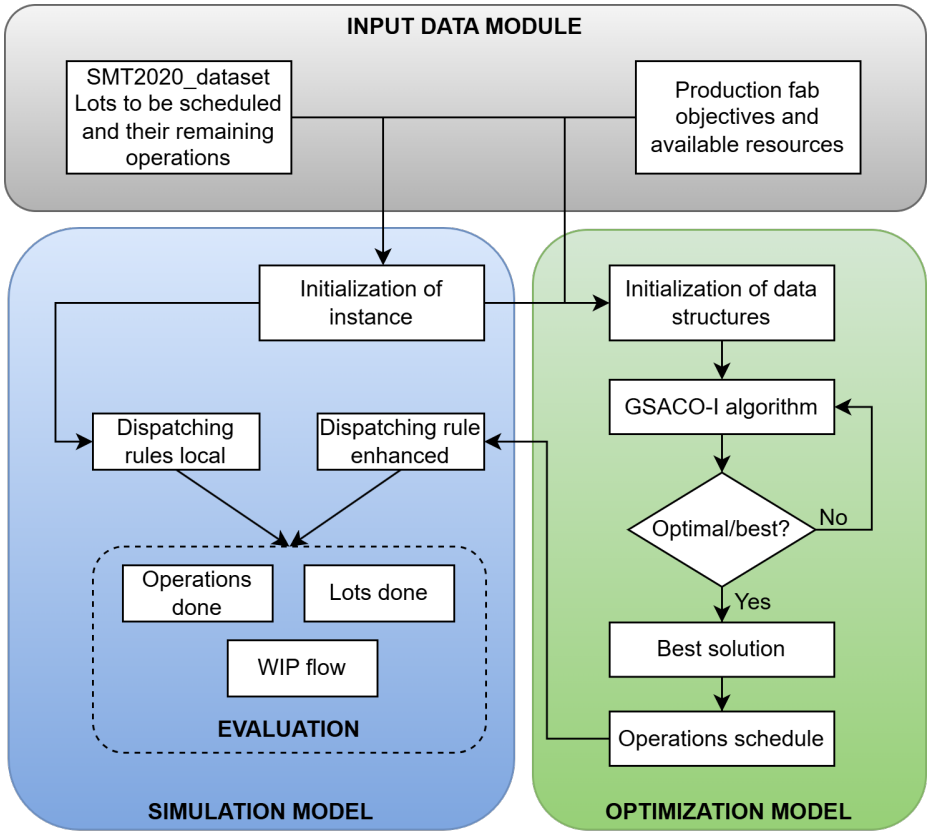

Fig. 6: Simulator-Scheduler framework

algorithms derived from the GSACO-I algorithm, which leverage more sophisticated decision-making processes to optimize scheduling tasks further. These enhancements not only improve the precision of the simulation but also significantly extend its applicability and effectiveness in complex manufacturing scenarios.

The framework depicted in Figure 6 outlines a sophisticated approach for scheduling operations within a production setting, incorporating both simulation and optimization models to enhance efficiency. At the core, the Input Data Module manages crucial scheduling data, including the SMT2020 dataset that lists lots to be scheduled with their remaining operations, alongside data concerning production objectives and available resources. This information feeds into the initialization processes for both the simulation and optimization modules.

The Simulation Model kicks off with an initialization of instances based on the dataset, where it sets up the environment for running simulations with different dispatching rules—both local and enhanced. Moreover, it generates the average processing time of operations for the scheduling purpose. The dispatch rules are tested under

Table 5: GSACO input parameter values

Parameter	Value
o	$makespan/operations$
s	$deterministic/dynamic$
l	5
n	1–5/15
k	10
τ_y	1
τ_z	0.00001
ρ	0.7
c	0.5

different processing times to observe their effectiveness in managing operations and lot completions, as well as the flow of work-in-process.

On the optimization side, the GSACO-I algorithm—takes the stage to refine scheduling further. This component iteratively searches for the best or optimal scheduling solutions, assessing each iteration against a criterion to determine if a preferable solution has been achieved. Upon finding the best solution, it finalizes the operations schedule, which details the best sequence and allocation of operations, ready to be implemented to maximize production efficiency.

6 EXPERIMENTAL EVALUATION

We implemented our GSACO-I 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.[3] 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 5, where a time limit l of 5 minutes and n up to 5 operations per job for makespan and 15 operations per job for operation throughput are plausible for SMSP instances based on the SMT2020 dataset, whose stochastic events necessitate frequent re-scheduling in practice. For operations, we introduced a planning period h to schedule. 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.

[3] The source code is publicly available in our GitHub repository: https://github.com/prosysscience/GSACO

Table 6: 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

To compare GSACO-I with the state of the art in FJSSP solving, we also run the CP solver OR-Tools (version 9.5) [45], 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 [46], where the free and open-source solver OR-Tools excels as serial winner of the MiniZinc Challenge.[4]

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 6 illustrates the strengths of CP models on a benchmark set of small to medium-scale FJSSP instances [35], where the instances MK1–MK4 are due to [48] and the remaining eight instances have been introduced in [47]. 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 [46], 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 7, 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

[4] https://www.minizinc.org/challenge

Table 7: 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

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

Table 8: SMSP results obtained with GSACO

Instance	Planning period in hours					
	1	2	3	4	5	6
	Ops/Lots	Ops/Lots	Ops/Lots	Ops/Lots	Ops/Lots	Ops/Lots
LV/HM	1422/0	2368/0	3234/0	4015/1	4722/2	5306/2
HV/LM	1848/0	2960/0	4093/1	4970/3	5975/5	6716/7

While running with a time limit l of 10 minutes, Table 9 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-I 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 9. 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 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 so-

M: The figure seems somehow missing here.

Table 9: 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

Table 10: Percentage Change in Dispatcher Performance Over Planning Hours (HV/LM)

Dispatcher	Planning period in hours											
	1		2		3		4		5		6	
	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% char
FIFO	1821/0	-	2831/0	-	4020/1	-	4914/4	-	5960/6	-	6841/8	-
CR	1798/0	-1.26%	2811/0	-0.71%	4021/1	0.02%	4934/3	0.40%	6003/5	0.72%	6946/ 8	1.53%
RANDOM	1810/0	-0.60%	2852/0	0.74%	4065/1	1.12%	4975/3	1.24%	6026/5	1.10%	6973/9	1.92%
GSACO	1831/0	0.54%	2957/0	4.45%	4162/1	3.53%	5118/3	4.15%	6263/5	5.08%	7162/7	4.69%

lutions. 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 9.

The plots in Figure 7 presents the performance of various dispatching rules—FIFO, CR, RANDOM, and GSACO—across different planning horizons ranging from 1 to 6 hours. Each plot illustrates the total number of operations dispatched by each rule within these time frames, offering a clear comparison of their efficiency and effectiveness in handling operational tasks within a semiconductor manufacturing.

M: Please unify the range of the y-scales in subfigures of each of the Figures 7–10.

Table 11: Percentage Change in Dispatcher Performance Over Planning Hours (LV/HM)

Dispatcher	Planning period in hours											
	1		2		3		4		5		6	
	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% change	Ops/Lots	% char
FIFO	1419/0	-	2249/0	-	3284/0	-	4018/1	-	4901/1	-	5665/4	-
CR	1416/0	-0.21%	2328/0	-3.51%	3372/0	2.67%	4083/1	1.61%	5023/1	2.48%	5799/3	2.36%
RANDOM	1403/0	-1.12%	2275/0	1.15%	3314/0	0.91%	4039/1	0.52%	4972/1	1.44%	5767/1	1.80%
GSACO	1442/0	1.62%	2409/0	7.11%	3427/0	4.35%	4272/1	6.32%	5174/2	5.57%	6014/2	6.16%

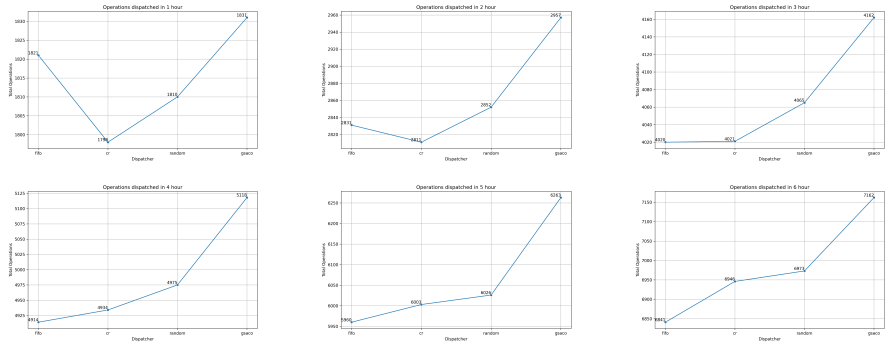

Fig. 7: Completed operations for HV/LM

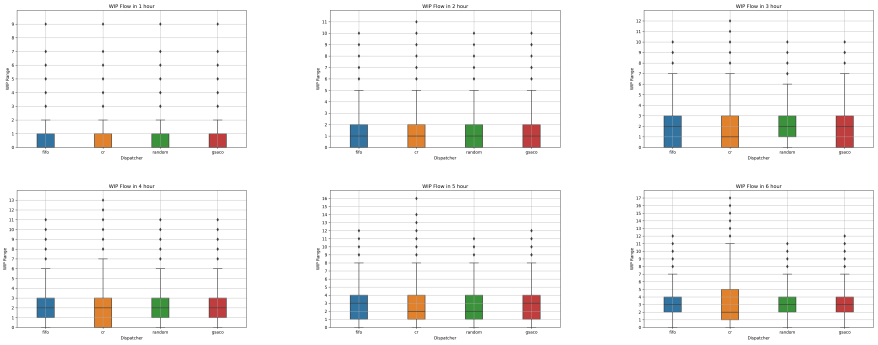

Fig. 8: WIP flow for HV/LM

The performance of dispatching rules based on specific operational times and goals are analyzed. While simpler rules like FIFO and CR are predictable and may perform adequately in stable environments, their performance can lag in more dynamic or complex scenarios where adaptive strategies like GSACO provide significant advantages. Random dispatching, surprisingly effective in some cases, suggests that stochastic elements might occasionally align with operational demands, although this method's reliability is less consistent. GSACO's consistently superior performance advocates for its use in environments where maximizing operational throughput is critical. This analysis not only aids in strategic decision-making but also highlights the potential for further refining these algorithms to enhance their applicability and effectiveness across various manufacturing contexts.

The plots in Figure 8 illustrate the Work in Progress (WIP) flow across various dispatching rules—FIFO, CR, RANDOM, and GSACO —over different planning horizons ranging from 1 to 6 hours. Each bar demonstrates the minimum and maximum WIP levels reached during the simulation period under each dispatching rule, with the height of the bar indicating the variability or stability of the WIP flow.

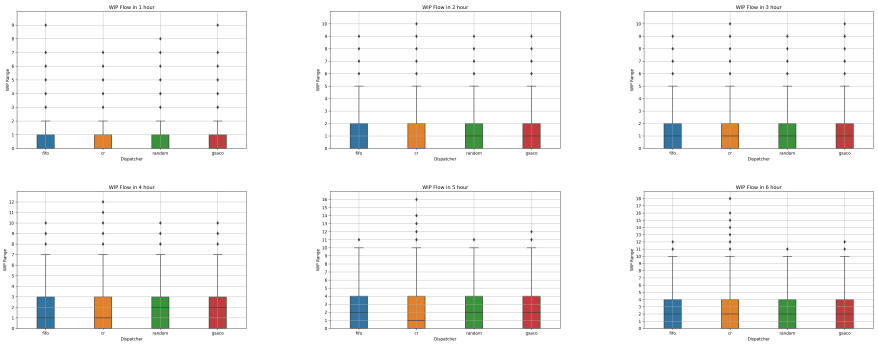

Fig. 9: WIP flow for LV/HM

FIFO shows the lowest range of WIP variability, suggesting consistent and predictable processing under this rule within the first hour. CR and Random display slightly higher variability, indicating a less stable WIP flow. GSACO exhibits the highest range, potentially due to more aggressive reordering and optimization strategies that initially increase variability before stabilizing.

As the planning horizon extends to 2, 3, 4, 5, and 6 hours, all methods show an increase in the maximum WIP range. FIFO consistently maintains a relatively lower range of WIP compared to other methods, indicating its less dynamic but stable approach. CR's performance exhibits moderate variability, reflecting its strategy of prioritizing jobs based on their deadlines, which might not always align with minimizing WIP. Random shows a moderate to high variability, which aligns with its inherent unpredictability. GSACO consistently demonstrates the highest variability. This could be attributed to its optimization processes, which might involve significant shifts in job prioritization and scheduling to achieve long-term efficiency, leading to larger fluctuations in WIP.

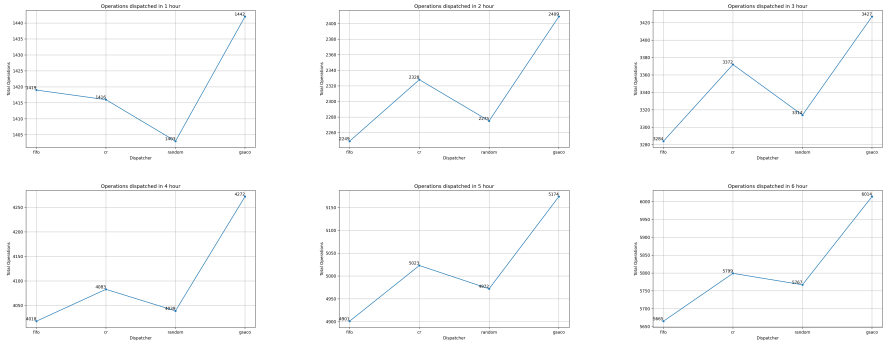

Fig. 10: Total operations completed LV/HM

7 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.

Acknowledgments

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.

References

1. Kopp, D., Hassoun, M., Kalir, A., Mönch, L.: SMT2020—a semiconductor manufacturing testbed. IEEE Transactions on Semiconductor Manufacturing 33(4):522–531 (2020)
2. Hopp, W. J., Spearman, M. L.: Factory physics. Waveland Press. (2011)
3. Uzsoy, R., Lee, C. Y., Martin-Vega, L. A.: A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning. IIE transactions, 24(4), 47-60 (1992)

4. Pinedo, M. L.: Scheduling: theory, algorithms, and systems. 6th edn. Springer (2022)
5. May, G. S., Spanos, C. J.: Fundamentals of semiconductor manufacturing and process control. John Wiley & Sons. (2006)
6. Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., Rose, O.: A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. Journal of scheduling, 14, 583-599. (2011)
7. Ali, R., Qaiser, S., El-Kholany, M. M., Eftekhari, P., Gebser, M., Leitner, S., Friedrich, G. A Greedy Search Based Ant Colony Optimization Algorithm for Large-Scale Semiconductor Production. In Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2024), pages 138-149. (2024)
8. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances. In Handbook of Metaheuristics, pages 311–351. Springer (2019)
9. Papadimitriou, C. H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall. (1982)
10. Perron, L., Didier, F., Gay, S.: The CP-SATLP solver (invited talk). In Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming, pages 3:1–3:2. Leibniz International Proceedings in Informatics, (2023)
11. Kovács, B., Tassel, P., Ali, R., El-Kholany, M., Gebser, M., Seidel, G.: A customizable simulator for artificial intelligence research to schedule semiconductor fabs. In 2022 33rd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC) (pp. 1-6). IEEE, (2022, May)
12. 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)
13. Leachman, R. C., Hodges, D. A.: Benchmarking semiconductor manufacturing. IEEE transactions on semiconductor manufacturing 9(2), 158-169 (1996)
14. McKay, K. N., Wiers, V. C.: Planning, scheduling and dispatching tasks in production control. Cognition, Technology & Work, 5, 82-93 (2003)
15. Chan, C. W.: Situation aware dispatching system for semiconductor manufacturing, (2024)
16. May, G. S., Spanos, C. J.: Fundamentals of semiconductor manufacturing and process control. John Wiley & Sons, (2006)
17. Mönch, L., Fowler, J. W., Mason, S. J.: Production planning and control for semiconductor wafer fabrication facilities: modeling, analysis, and systems (Vol. 52). Springer Science & Business Media, (2012)
18. El-Kholany, M., Ali, R., Gebser, M.: Hybrid ASP-based multi-objective scheduling of semiconductor manufacturing processes (Extended version). arXiv preprint arXiv:2307.14799, (2023)
19. Shen, L., Dauzère-Pérès, S., Neufeld, J. S.: Solving the flexible job shop scheduling problem with sequence-dependent setup times. European journal of operational research, 265(2), 503-516, (2018)
20. Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., Tang, L.: A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. Expert Systems with Applications, 205, 117796, (2022)
21. Nayar, N., Gautam, S., Singh, P., Mehta, G.: Ant colony optimization: A review of literature and application in feature selection. Inventive Computation and Information Technologies: Proceedings of ICICIT 2020, 285-297, (2021)
22. Zhou, X., Ma, H., Gu, J., Chen, H., Deng, W.: Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. Engineering Applications of Artificial Intelligence, 114, 105139, (2022)

23. Dorigo, M., Stützle, T.: Ant colony optimization: overview and recent advances (pp. 311-351). Springer International Publishing, (2019)
24. Stützle, T., Dorigo, M.: ACO algorithms for the traveling salesman problem. Evolutionary algorithms in engineering and computer science, 4, 163-183, (1999)
25. Rizzoli, A. E., Montemanni, R., Lucibello, E., Gambardella, L. M.: Ant colony optimization for real-world vehicle routing problems: from theory to applications. Swarm Intelligence, 1, 135-151, (2007)
26. Luo, D. L., Wu, S. X., Li, M. Q., Yang, Z.: Ant colony optimization with local search applied to the flexible job shop scheduling problems. In 2008 International Conference on Communications, Circuits and Systems (pp. 1015-1020). IEEE, (2008)
27. Rugwiro, U., Gu, C., Ding, W.: Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning. Journal of Internet Technology, 20(5), 1463-1475, (2019)
28. Khelifa, B., Laouar, M. R.: A holonic intelligent decision support system for urban project planning by ant colony optimization algorithm. Applied Soft Computing, 96, 106621, (2020)
29. Wang, J., Osagie, E., Thulasiraman, P., Thulasiram, R. K.: HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Networks, 7(4), 690-705, (2009)
30. Türkyılmaz, A., Şenvar, Ö., Ünal, I., Bulkan, S.: A research survey: heuristic approaches for solving multi objective flexible job shop problems. Journal of Intelligent Manufacturing, 31(8), 1949-1983, (2020)
31. Leung, C. W., Wong, T. N., Mak, K. L., Fung, R. Y.: Integrated process planning and scheduling by an agent-based ant colony optimization. Computers & Industrial Engineering, 59(1), 166-180, (2010)
32. Li, L., Keqi, W., Chunnan, Z.: An improved ant colony algorithm combined with particle swarm optimization algorithm for multi-objective flexible job shop scheduling problem. In 2010 International Conference on Machine Vision and Human-machine Interface (pp. 88-91). IEEE, (2010)
33. Xing, L. N., Chen, Y. W., Wang, P., Zhao, Q. S., Xiong, J.: A knowledge-based ant colony optimization for flexible job shop scheduling problems. Applied Soft Computing, 10(3), 888-896, (2010)
34. Thammano, A., Phu-ang, A.: A hybrid artificial bee colony algorithm with local search for flexible job-shop scheduling problem. Procedia computer science, 20, 96-101, (2013)
35. Arnaout, J. P., Musa, R., Rabadi, G.: A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations. Journal of Intelligent Manufacturing, 25, 43-53, (2014)
36. El Khoukhi, F., Boukachour, J., Alaoui, A. E. H.: The "Dual-Ants Colony": A novel hybrid approach for the flexible job shop scheduling problem with preventive maintenance. Computers & Industrial Engineering, 106, 236-255, (2017)
37. Abdel-Basset, M., Ding, W., El-Shahat, D.: A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. Artificial Intelligence Review, 54(1), 593-637, (2021)
38. Fontes, D. B., Homayouni, S. M., Gonçalves, J. F.: A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. European Journal of Operational Research, 306(3), 1140-1157, (2023)
39. Li, Q., Ning, H., Gong, J., Li, X., Dai, B.: A hybrid greedy sine cosine algorithm with differential evolution for global optimization and cylindricity error evaluation. Applied Artificial Intelligence, 35(2), 171-191, (2021)
40. Mohd Tumari, M. Z., Ahmad, M. A., Suid, M. H., Hao, M. R.: An improved marine predators algorithm-tuned fractional-order PID controller for automatic voltage regulator system. Fractal and Fractional, 7(7), 561, (2023)

41. Suid, M. H., Ahmad, M. A.: A novel hybrid of Nonlinear Sine Cosine Algorithm and Safe Experimentation Dynamics for model order reduction. Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, 64(1), 34-50, (2023)
42. Kumar, P. R.: Re-entrant lines. Queueing systems, 13(1), 87-110, (1993)
43. Baker, K. R.: Introduction to sequencing and scheduling. John Wiley and Sons google schola, 2, 560-562, (1974)
44. Blackstone, J. H., Phillips, D. T., Hogg, G. L.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. The International Journal of Production Research, 20(1), 27-45, (1982)
45. Perron, L., Didier, F., Gay, S.: The CP-SAT-LP Solver (Invited Talk). In 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, (2023)
46. Ku, W. Y., Beck, J. C.: Mixed integer programming models for job shop scheduling: A computational analysis. Computers & Operations Research, 73, 165-173, (2016)
47. Fattahi, P., Saidi Mehrabad, M., Jolai, F.: Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of intelligent manufacturing, 18, 331-342, (2007)
48. Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Annals of Operations research, 41(3), 157-183, (1993)
49. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
50. 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/10.10007/1234567890
51. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
52. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
53. LNCS Homepage, http://www.springer.com/lncs, last accessed 2023/10/25