Swarm Intelligence-Driven Dispatching Rules for Semiconductor Production

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

Abstract. The immense complexity of semiconductor production demands for advanced dispatching strategies that transcend traditional rule-based systems. This paper introduces a novel approach to dispatching rules derived from swarm intelligence techniques, specifically designed to tackle the intricate dynamics of large-scale semiconductor manufacturing processes. Our approach integrates simulation and optimization methods to investigate and enhance operational efficiency, addressing both the scalability of schedules and their practical implementation. We employ a customizable simulation framework to model a semiconductor manufacturing environment, wherein various dispatching rules as well as our proposed swarm intelligence-driven method are assessed. The effectiveness of these dispatching rules is quantitatively evaluated through a series of simulations that measure key performance indicators such as work-in-progress levels, throughput, and operational variability across different production scenarios. This study not only elucidates the potential of swarm intelligence techniques in refining production dispatching strategies, but also provides a simulation-based evaluation framework 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 strategies are critical in semiconductor manufacturing because they directly impact the production throughput and utilization of resources. Dispatching refers to the process of assigning production operations to specific machines in real-time, while scheduling involves pre-planning the sequencing and timing of operations to optimize certain key performance indicators, such as minimizing the completion time of scheduled operations or balancing the load across different machines [4].

For smaller volumes at specific workcenters in a semiconductor fab, which can be modeled as flexible job shops, optimal solutions can be achieved using mathematical

optimization [12]. However, in larger-scale and more dynamic environments, the high complexity and computational constraints limit the feasibility of applying mathematical optimization. Consequently, exact optimization is typically implemented at a local scale and isolated to individual workcenters. In complex scenarios, this localized approach to production scheduling cannot guarantee the global performance of a semiconductor fab.

Dispatching involves prioritizing operations and determining the machines to perform them. These decisions are based on the availability of resources and the urgency of jobs [2]. However, as production scenarios become complex, particularly in high-tech industries such as semiconductor manufacturing, the static nature of traditional dispatching rules often turns out to be insufficient. Static dispatching rules do not account for rapid changes in production conditions or intricate dependencies between different production processes. This inadequacy can lead to sub-optimal dispatching decisions that compromise operational efficiency [3].

The complexity of scheduling in large-scale semiconductor production is significantly influenced by the unpredictable nature of processing times and machine availabilities [5]. Variability in processing times can arise from differences in equipment conditions, material handling durations, or the unique characteristics of production lots, while frequent machine breakdowns can cause substantial disruptions, underscoring the need for strong and adaptable scheduling strategies [13]. The complexity is further compounded by the need to accommodate a wide mix of product types, each with distinct processing needs and priorities, and to integrate new processes seamlessly into the production schedule without disrupting ongoing operations [6]. Scheduling involves meticulously planning and organizing production operations to achieve optimal resource utilization and synchronized product flows across various manufacturing stages, which is crucial for minimizing the makespan of schedules and maintaining high throughput [4].

In this paper, we explore the capacities of scheduling for large-scale Semiconductor Manufacturing Scheduling Problems (SMSPs), representing the operations of semiconductor fabs. Modern semiconductor fabs manage the processing of tens of thousands of operations daily across thousands of 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 necessary capabilities. The scheduling challenge in semiconductor manufacturing thus involves assigning specific operations to appropriate machines and determining an optimal operation sequence on each machine. Additionally, schedules must be flexible and capable of rapid readjustments in response to machine failures or dynamic process deviations.

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 (CP) approach [10] on large-scale SMSP instances. Our approach synergistically combines probabilistic operation sequencing with a greedy machine assignment strategy, targeting the objective of minimizing makespan. Building on this foundational work, our paper extends the previous method by proposing an enhanced approach, GSACO-O, which is specifically designed to optimize the operational throughput of semiconductor fabs. This develop-

ment marks a significant advancement in our methodology, aiming to derive dispatching rules that commute with simulation methods.

The paper is organized as follows. Section 2 provides a literature review. In Section 3, we formulate SMSP in terms of the well-known Flexible Job Shop Scheduling Problem (FJSSP). Our GSACO-O algorithm is presented in Section 4. In Section 5, we describe a customized simulation framework extending [11]. Section 6 provides and discusses experimental results. Finally, Section 7 concludes the paper.

2 Literature Review

First, we survey the relevant literature on the challenges inherent to SMSP, a problem with complex operational dynamics and the critical need for efficient dispatching strategies. Second, we address the literature on solving methods for FJSSP, which we use as general scheduling model to represent semiconductor manufacturing processes. Third, we turn to swarm intelligence techniques, with particular focus on the meta-heuristic ACO algorithm and its application to FJSSP. Finally, we review traditional dispatching methods, detailing their advantages and limitations within the context of semiconductor manufacturing.

2.1 Semiconductor Manufacturing Scheduling Problem (SMSP)

Semiconductor manufacturing is characterized by complex multi-step production processes, involving highly specific and sensitive 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 of extremely clean environments to avoid contamination of the microscopically small circuits, making operational efficiency a challenging goal [5].

Unlike other manufacturing industries that focus on a limited range of products, semiconductor manufacturers typically deal with a wide mix of products on the same production line. This product mix, coupled with a 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 the throughput or quality of production.

The SMSP is a complex and critical challenge, aiming to optimize the scheduling of operations during the semiconductor manufacturing process. Different local methods have been adopted for several types of machines [15]. Their core aspects include machine assignment, batch processing, priority constraints, setup times, machine availability, and reentrant flow. The key challenges concern the handling of fab complexity, dynamic job arrivals, and machine disruptions while maximizing throughput [18].

2.2 Flexible Job Shop Scheduling

The FJSSP model allows for solving small-scale scheduling instances by mathematical optimization [19]. A schedule usually assigns the production operations to available

machines, and additionally determines a sequence for performing the assigned operations on each machine. High complexity results from the objective to optimize key performance indicators such as the completion time of scheduled operations or balanced load across different machines [4]. The solving approaches for NP-hard combinatorial optimization problems can be classified into exact and approximate methods [20]. While exact optimization methods can be successfully applied to small-scale FJSSP instances, large real-world scenarios like SMSP call for approximate techniques based on (meta-)heuristics, machine learning, and more.

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]. 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 independently, 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. 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 semiconductor production scenarios [1]. Arguably, ACO is a well-suited meta-heuristic algorithm for this domain due to its dynamic nature and multi-objective optimization capacities [21], enabling decision-making within comparably short computing time [22]. Beyond the FJSSP and SMSP domains 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 Rules

In semiconductor manufacturing, the efficient management of production flow is crucial for maintaining high throughput and minimizing delays. The First In First Out (FIFO) strategy is one of the simplest and most widely used dispatching rules in various manufacturing sectors, including semiconductors. It prioritizes operations in the order of release, regardless of their complexity or processing time. While FIFO is straightforward to implement and ensures a fair processing, it may be sub-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 (CR) strategy is a more sophisticated dispatching rule that prioritizes operations based on the ratio of a job's due date to the remaining processing time. This method aims to minimize tardiness and is particularly useful in semiconductor manufacturing, where delivery schedules are tight and delays can be costly. However, the effectiveness of CR is highly dependent on accurate estimates of processing times and can become intricate in heterogeneous production environments [43].

Random dispatching of operations, irrespectively of their characteristics or due dates, is often used as baseline in simulation studies to demonstrate the effectiveness of more sophisticated strategies. While it does not optimize 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].

Traditional dispatching rules provide static strategies for managing production processes at low overhead, 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 significant potential for advanced dispatching approaches that can adapt to the dynamic conditions of semiconductor production lines. The integration of real-time data analytics and dynamic strategies, such as those building on swarm intelligence algorithms, could close the existing gap by providing more flexible and responsive dispatching methods.

3 SMSP 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

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

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-O 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 an 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 planning horizon, 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-O for instance in Table 3 whose operations and their average processing times are given in Table 3a is displayed in Figure 3 and Figure 4.

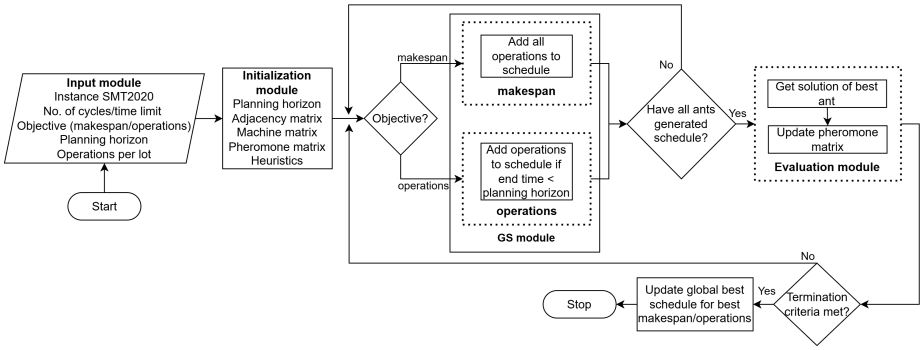


Fig. 1: GSACO-O framework

Table 2: GSACO-O parameters

Parameter	Description
o	Objective
l	Cycles/time limit
n	Operations per lot
h	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

4 GSACO-O Algorithm

The framework of GSACO-O algorithm is illustrated in Figure 1 and the input and internal parameters, is summarized in Table 2. The four main submodules, highlighted in bold, are discussed in detail in the following subsections.

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 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.

Input: dataset, instance, l
Output: best schedule found by ants
Parameters: o, n, h, k, τ_y, τ_z, ρ, c
Initialize adjacency, pheromone, and machine matrix;
if o == *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 o == *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 variant (GSACO-O) for scheduling

Algorithm 1 provides a pseudo-code representation of GSACO-O for minimizing makespan and optimizing operations within the planning horizon.

4.1 Input Module

This module reads in an SMSP instance obtained from SMT2020 dataset [1]. Additionally, it also configures the instance from the simulator develop by [11] for the average processing times of the operations. 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 mod-

ule takes several inputs for the GSACO-O optimization process such as the objective o, 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 operations per lot n as upper bound on the length n_j of the operation sequence for a job j. The operations per lot 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 n 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 a small instance that follows the explanation of the proposed GSACO-O algorithm. In Table 3a, the operations belonging to two jobs can be obtained with the parameter $n = 5$. The average duration of the operations for given tool group is given in seconds to avoid float calculation. Later the duration is converted to the minutes for visualization purpose. The lots that belong to the products are shown in Table 3b. The lots are at different stages of the production identified by the step number.

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. Since the products are at different stages and the remaining operations to schedule are obtained from the step given in Table 3b up to the maximum step in Table 3a for corresponding lot and product. The resulting $(N+1) \times (N+1)$ adjacency matrix, where $N = 43$ is the total number of

Table 3: Example instance

(a) Operations

No.	Operation	Tool group name	Avg duration (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_84	54
10	$O_{5,2,0}$	Litho_FE_111	130

(b) Lots

Lot	Product	Step
1	1	1
2	2	2
3	1	1
4	2	3
5	1	1
6	2	2
7	1	1
8	2	3
9	1	1
10	2	2

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ 41 \\ 42 \\ 43 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & \dots & 41 & 42 & 43 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(a) Adjacency matrix

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ \vdots \\ 41 \\ 42 \\ 43 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & \dots & 10 & 11 & 12 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(b) Machine matrix

Fig. 2: Matrices for instance in Table 3

operations, 0 entries indicate the absence, and 1 entries the existence of edges, is given in Figure 2a.

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:

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

With $\tau_y = 1$, this reproduces the adjacency matrix in Figure 2a 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 $m \in \{1, \dots, 12\}$ to the tool groups $t \in \{1, \dots, 6\}$, responsible for processing the remaining operations in Table 3, we obtain the machine matrix shown in Figure 2b.

Input: horizon h
Output: schedule, selected edges of an ant, and operations finished in horizon h
$sequence \leftarrow []$;
$selected \leftarrow \emptyset$;
$next \leftarrow \{(0, O_{1,j,t}) \mid j \in \{1, \ldots, J\}\}$;
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** $a_m \leftarrow 0$;
$finished \leftarrow 0$;
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
  $s_{i,j,t,b} \leftarrow \max(\{a\}$
    $\cup \{s_{i-1,j,t',m} + d_{i-1,j,t'} \mid i > 1\})$;
  $a_b \leftarrow s_{i,j,t,b} + d_{i,j,t}$;
  if $a_b \leq h$ **then** $finished \leftarrow finished + 1$;
end
return $\langle \{s_{i,j,t,m} \mid (O', O_{i,j,t}) \in selected\}, selected, finished \rangle$;

Algorithm 2: Greedy Search (GS)

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-O, each ant applies greedy search to efficiently construct some feasible schedule for a given SMSP instance. The GS procedure based on the given objective o, outlined by the pseudo-code in Algorithm 2, includes two phases: operation sequencing and machine assignment.

The first phase constructs an ordered list of 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

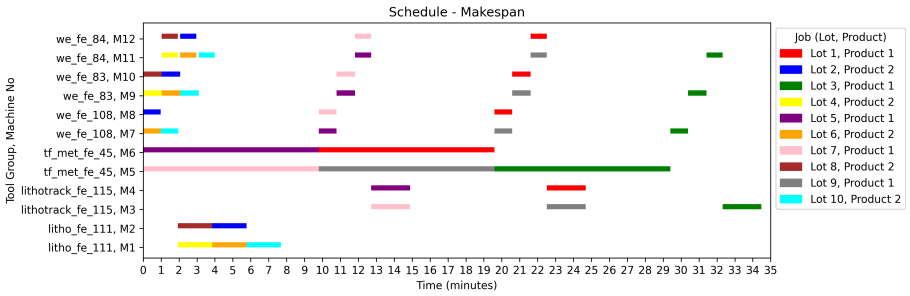


Fig. 3: Feasible schedule for the instance in Table 3

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.

For example, starting with an empty sequence of operations, as outlined in Table 3, the search begins at a start node 0 which symbolizes the initial state where no operations have been processed yet. The first step involves identifying all initially selectable edges, which correspond to the first operation in each lot.

To minimize the makespan, the algorithm selects a random edges and the selection process continues iteratively, with each step involving the evaluation of available operations that can be added to the existing sequence. As operations gets selected, their corresponding edges are directed, indicating a fixed order. This directionality prevents the reselection of these operations. The procedure is completed once all operations are integrated into the sequence. 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 and end times to all operations. The schedule is shown in Figure 3.

To optimize operations within the planning horizon h, the algorithm initiates by selecting a random edge from the list of candidate operations. The start time for the first operations is set to zero. Based on the duration of each operation and taking into account the earliest available machine, the algorithm calculates the operation's end time. If this end time falls within the planning horizon h, indicating that the operation can be accommodated within the schedule, it is then added to the sequence. The algorithm continues this process iteratively, adding one operation at a time. In each iteration, it reassesses the remaining operations to determine which can fit into the updated schedule without exceeding the planning horizon. This iterative process persists until no more operations can be added within the designated periods, thereby optimizing the use of

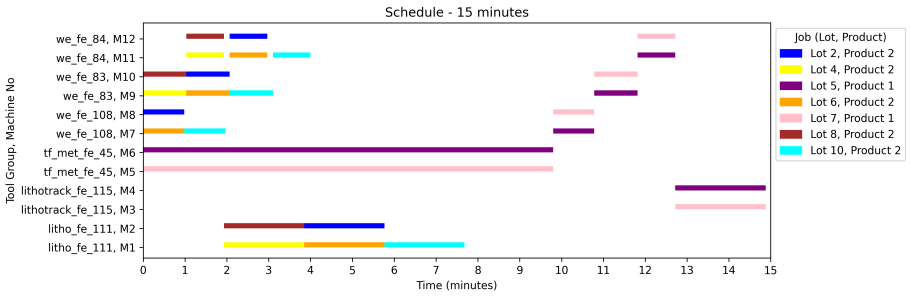


Fig. 4: Feasible schedule for the instance in Table 3

available time and resources within the planning horizon. The schedule for 15 minutes is shown in Figure 4.

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-O iteration. Among them, a schedule of shortest makespan or schedule of maximum operations 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 [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 Framework

To simulate a semiconductor fab, we adapted the simulator PySCFabSim developed by [11] designed for optimizing the scheduling of semiconductor fabrication plants. The simulator is intended for both academic research and practical application, facilitating the development, testing, and deployment of new scheduling algorithms. It supports various approaches including reinforcement learning, priority-based rules, and evolutionary algorithms.

Commercial simulators used for optimizing semiconductor fabs often face deployment challenges in research settings due to licensing and proprietary limitations. These constraints hinder reproducibility and adaptation to new research needs. To address

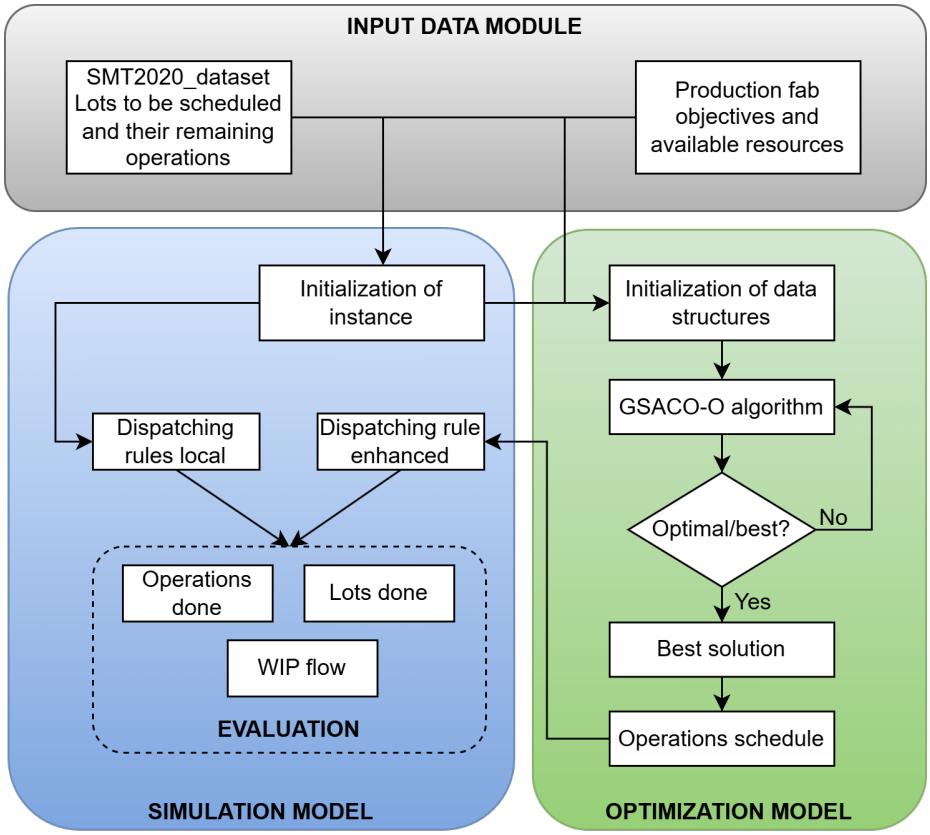


Fig. 5: Simulator-Scheduler framework

these issues, the authors have developed a customizable, open-source simulation tool that supports extensive experimentation and algorithm development.

It operates using a dispatcher that dynamically allocates lots to machines at each decision point, which could involve choosing a batch of lots waiting for machine availability. This functionality is crucial in simulating real-world manufacturing scenarios where dispatch decisions significantly impact operational efficiency.

An interface is provided within the simulator to facilitate the connection between the dispatching logic and the simulator's operational flow. This interface allows the simulator to integrate various dispatching methods, including those based on traditional rules or any enhanced rules.

The paper discusses the incorporation of priority-based dispatching rules, which are commonly used in industrial contexts. These rules prioritize lot dispatch based on various criteria such as:

1. FIFO: Lots are processed in the order they arrive or based on their waiting time.

2. CR: This method involves calculating a ratio of the remaining time to the due date divided by the expected processing time, prioritizing lots based on how critical their timing is relative to scheduled due dates.
3. RANDOM: Lots are processed in random order.

Additionally, we have integrated enhanced dispatching algorithms derived from the GSACO-O 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 5 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 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-O algorithm takes the stage to refine scheduling further. This component iteratively searches for the best 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, give is given to the simulator for evaluation over different processing times.

6 Experimental Evaluation

We implemented GSACO-O 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-O 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 adhere to the parameters specified in Table 4, setting a time constraint l of 10 minutes for objective makespan and 5 minutes for objective operations. Setting different time limits for these two aspects allows the algorithm to tailor its computational efforts to the specific demands of each task. While makespan optimization seeks the best possible sequence over all operations (requiring extensive computation), optimizing operational efficiency focuses on making quicker adjustments

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

Table 4: GSACO input parameter values

Parameter	Value
o	makespan/operations
l	$10/5$
n	1–5/15
h	1–6
k	10
τ_y	1
τ_z	0.00001
ρ	0.7
c	0.5

Table 5: 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

that enhance day-to-day operations without the need for extensive computation. For the SMSP instances derived from the SMT2020 dataset, we consider up to 5 operations per job for minimizing makespan and up to 15 operations per job for optimizing operational throughput. These parameters are practical, given the stochastic nature of the problem which often necessitates frequent rescheduling. Additionally, we define a short planning horizon h to accommodate near-term scheduling requirements. 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.

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 5, 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 focus on scheduling for operations n from 1 up to 5, to be performed per job. Hence, the operations O to schedule gradually increase from the number J of jobs, in case of the operations $n = 1$, to more than 10000 operations for the longest horizon $n = 5$.

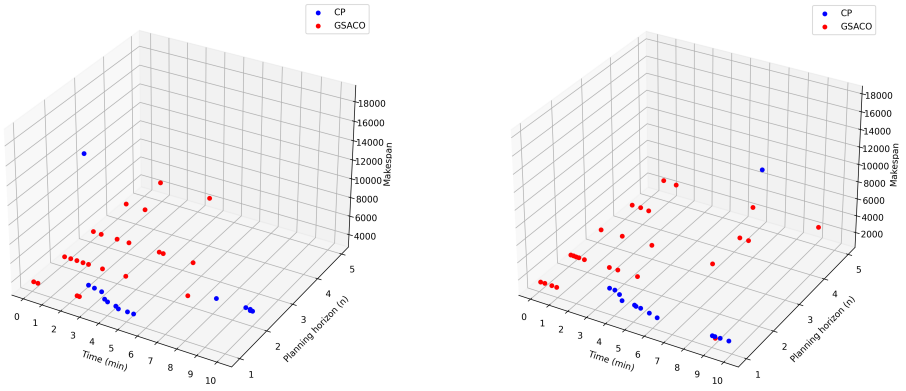
Table 6: SMSP results obtained with CP and GSACO [7]

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

While running with a time limit l of 10 minutes, Table 6 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-O 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. 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.

The quick convergence of our GSACO implementation is also outlined by the makespan improvements plotted in Figure 6. On the LV/HM scenario displayed in Figure 6a, 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 6b, an improvement occurs after more than the 9 minutes of computing time listed in the rightmost column of Table 6. 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 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 6.

Furthermore, this study investigates the efficiency of various dispatching strategies in a simulated manufacturing environment. Dispatchers play a critical role in production



(a) LV/HM scenario (b) HV/LM scenario

Fig. 6: Makespan improvements of schedules for SMSP instances obtained with CP and GSACO [7]

Table 7: Comparative performance of dispatching strategies (LV/HM)

Planning horizon	Dispatcher								Scheduler	
	FIFO		CR		RANDOM		GSACO-O		GSACO-O	
	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	diff
1	1419/0	-	1416/0	-0.21%	1403/0	-1.12%	1442//0	1.62%	1422/0	0.21%
2	2249/0	-	2328/0	3.51%	2275/0	1.15%	2409/0	7.11%	2368/0	5.29%
3	2384/0	-	3372/0	2.67%	3314/0	0.91%	3427/0	4.35%	3234/0	-1.52%
4	4018/1	-	4083/1	1.61%	4039/1	0.52%	4272/1	6.32%	4015/1	-0.07%
5	4901/1	-	5023/1	2.48%	4972/1	1.44%	5174/2	5.57%	4722/2	-3.65%
6	5665/4	-	5799/3	2.36%	5767/1	1.80%	6014/2	6.16%	5306/2	-6.33%

systems by determining the order in which jobs are processed. Effective dispatching can significantly enhance operational throughput and resource utilization. To this end, we compare traditional and enhanced dispatching based on a GSACO-O. The primary objective is to quantify and compare the performance changes of different dispatching strategies over multiple planning horizons.

We examine the efficacy of four distinct dispatching strategies using two different datasets, characterized as High Volume Low Mix (HV/LM) and Low Volume High Mix (LV/HM), to evaluate their effectiveness across six varied planning horizons. The dispatching strategies include FIFO, CR, RANDOM, and GSACO-O, which is also assessed as a scheduler. The performance metrics were recorded in terms of operations and lots completed, with FIFO serving as the baseline for performance comparison.

The results from the LV/HM dataset shown in Table 7 indicate variable performance across the strategies and horizons. FIFO maintained consistent throughput but was generally outperformed by our GSACO-O dispatcher, which showed substantial increases in operational efficiency. Similarly, the results from HV/LM dataset shown in Table 8

Table 8: Comparative performance of dispatching strategies (HV/LM)

Planning horizon	Dispatcher								Scheduler	
	FIFO		CR		RANDOM		GSACO-O		GSACO-O	
	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	Diff	Ops/Lots	Diff
1	1821/0	-	1798/0	-1.26%	1810/0	-0.60%	1831/0	0.54%	1848/0	1.48%
2	2831/0	-	2811/0	-0.71%	2852/0	0.74%	2957/0	4.45%	2960/0	4.55%
3	4020/1	-	4021/1	0.02%	4065/1	1.12%	4162/1	3.53%	4093/1	1.81%
4	4914/4	-	4934/3	0.40%	4975/3	1.24%	5118/3	4.15%	4970/3	1.07%
5	5960/6	-	6003/5	0.72%	6026/5	1.10%	6263/5	5.08%	5975/5	0.25%
6	6841/8	-	6946/8	1.53%	6973/9	1.92%	7162/7	4.69%	6716/7	-1.82%

further substantiate the superior performance of the GSACO-O strategy, confirming its effectiveness across different planning horizons.

The poor performance of GSACO-O over long planning horizon, as a scheduler compared to its role as a dispatcher can be attributed to the operational complexities. While dispatching focuses on immediate, localized decision-making related to the sequence and priority of jobs within the manufacturing process, scheduling involves long-term planning, requiring the management of more complex variables over extended periods. The GSACO-O algorithm, which combines greedy search and swarm intelligence, excels in environments where data is abundant and immediate adaptability is crucial, making it highly effective for dispatching. However, as a scheduler, the algorithm must contend with uncertainties such as fluctuating future orders and resource availability, which may not be as predictable as the current operational statuses utilized in dispatching. Moreover, the effectiveness of a scheduler is often gauged on long-term outcomes such as overall resource utilization and job load balancing, metrics that require a different approach or algorithmic tuning than those used for dispatching tasks.

In Figure 7 and Figure 8, the performance of dispatching rules; FIFO, CR, RANDOM, GSACO-O and GSACO-O as scheduler across different planning horizons ranging from 1 to 6 hours is presented. 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.

The plots in Figure 9 illustrate the Work-in-Progress (WIP) flow for the LV/HM scenario across various dispatching rules FIFO, CR, RANDOM, GSACO-O, and GSACO-O as scheduler 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. Across all planning horizons, the range of WIP counts is quite consistent across different dispatchers, as represented by FIFO, CR, RANDOM, GSACO-O, and GSACO-O (Scheduler). This suggests that no single dispatcher has a significantly larger WIP range, indicating a balanced approach to managing WIP levels over time. In case of dispatcher specific observation, 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

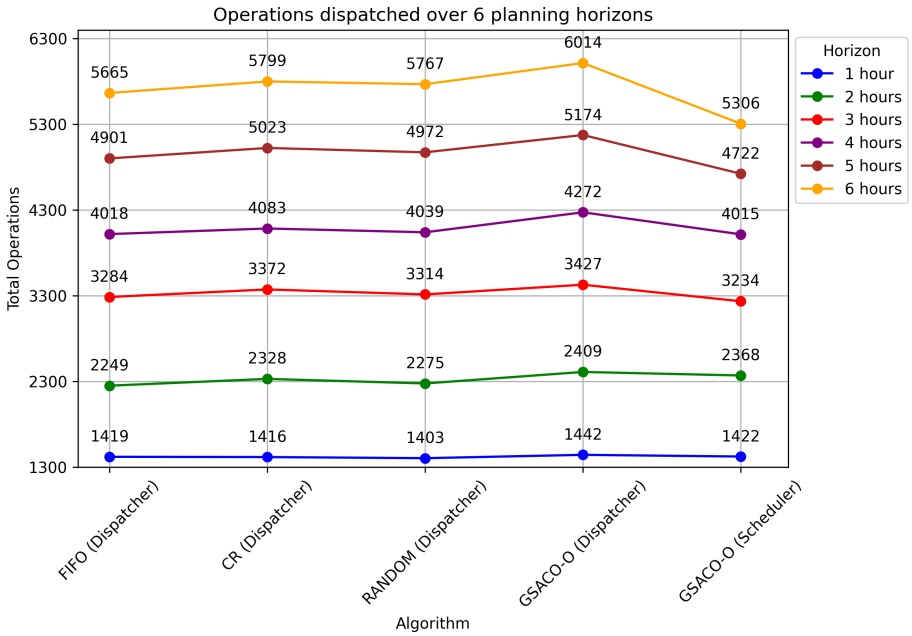


Fig. 7: Operations throughput (LV/HM)

WIP flow. Across the longer planning horizons such as 5 and 6 hours, there are more outliers in WIP counts across all dispatchers, which could indicate sporadic spikes in WIP. This trend shows increase in job complexity as the period extends. GSACO-O, in both dispatcher and scheduler forms, demonstrates a consistent WIP distribution that does not deviate significantly across time horizons. This indicate that the GSACO-O method is robust in maintaining predictable WIP flows, regardless of operational duration.

Similarly, the plots in Figure 10 illustrate the WIP flow for the HV/LM scenario. The FIFO dispatcher maintains a narrow interquartile range (IQR) and a relatively low WIP count in comparison to other dispatchers. Even at 6 hours, FIFO's WIP values are less dispersed than other algorithms, indicating a conservative and stable approach in managing WIP. The CR dispatcher shows a slightly wider IQR than FIFO, especially noticeable in the 5th and 6th hour plots. The RANDOM dispatcher has a consistent distribution similar to FIFO. Its WIP values increase consistently across longer planning hours, suggesting a flexible approach that does not tightly control WIP levels and makes it less predictable. GSACO-O, when used as a dispatcher, shows a stable and controlled WIP range similar to FIFO and CR but with some incremental flexibility. Across the time periods, GSACO-O maintains a low median and narrow IQR, suggesting an efficient handling of WIP without significant spikes. GSACO-O as a scheduler demonstrates similar behavior to its dispatcher, with low variability in WIP values.

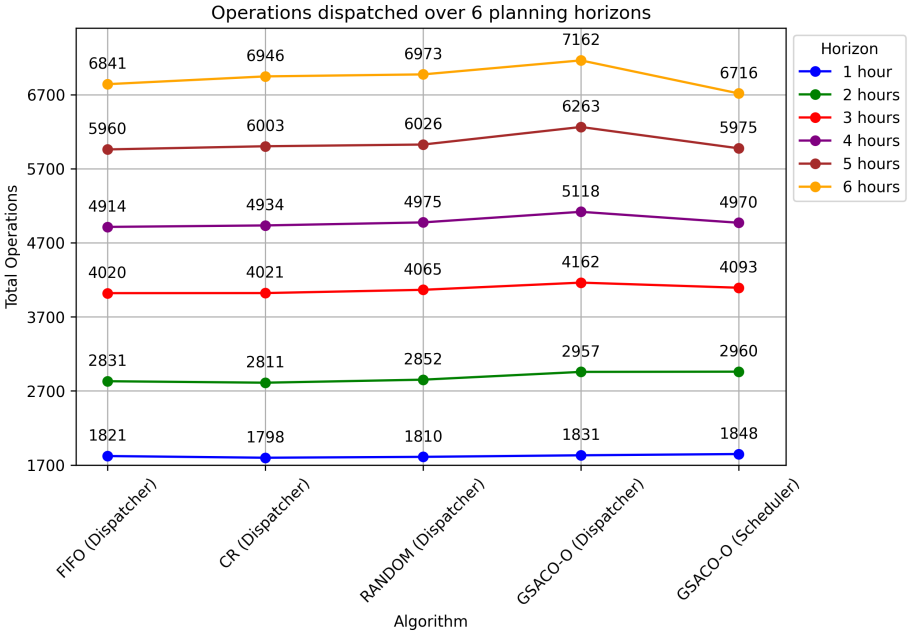


Fig. 8: Operations throughput (HV/LM)

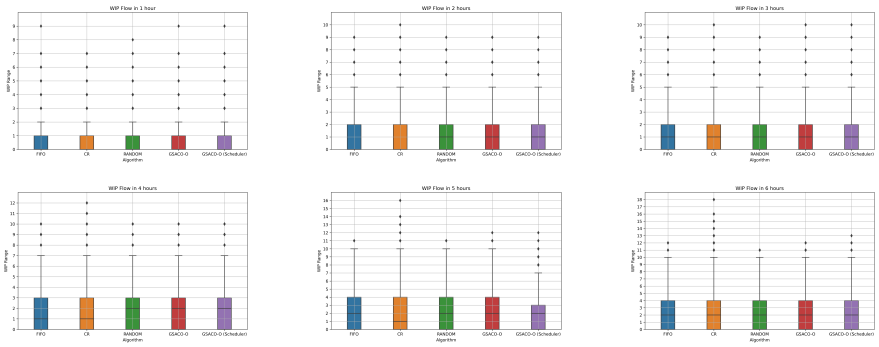


Fig. 9: WIP flow for LV/HM

In conclusion, the comparative analysis across different dispatching algorithms (FIFO, CR, RANDOM, GSACO-O as dispatcher, and GSACO-O as scheduler) reveals distinct characteristics in WIP handling as planning horizons extend from 1 to 6 hours. GSACO-O, whether used as a dispatcher or scheduler, consistently demonstrates robust performance with stable WIP ranges and limited outliers, suggesting a reliable choice for applications requiring predictability in WIP levels. FIFO and CR also show controlled WIP distributions but with slightly more variability in extended periods, in-

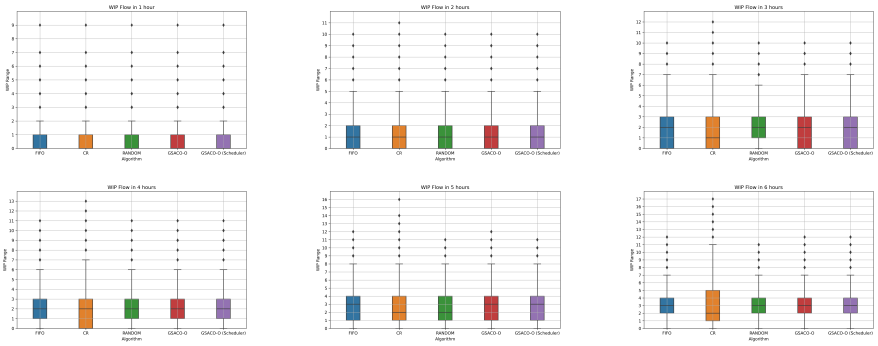


Fig. 10: WIP flow for HV/LM

dicating that while they maintain order, they might face occasional increases in WIP as time progresses. RANDOM, by contrast, allows more variability, with a slightly wider IQR and an increasing number of outliers in longer horizons, which reflects a more flexible but less controlled approach to WIP management. As planning horizons extend, all algorithms experience a rise in WIP counts and outliers, highlighting the impact of prolonged operational periods on workload variability. For scenarios prioritizing stability and predictability, GSACO-O and FIFO emerge as preferred choices.

Additionally, if WIP levels remain consistent across dispatching strategies, using a scheduler like GSACO-O could be computationally expensive without significant benefit in WIP performance. In such cases, opting for a dispatcher approach rather than a scheduler could reduce computational overhead while achieving similar WIP outcomes. These insights can guide the selection of dispatching and scheduling algorithms in settings where managing WIP levels is crucial to operational efficiency and workflow stability, with computational cost considerations also taken into account.

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. In our previous work [7], we showed the scalability of our proposed approach. We thus proposed the enhance variant of our previous GSACO algorithm, GSACO-O for optimizing operational troughput over different planning horizons.

We evaluated our scheduler through simulation as a dispatcher and compared the performance with traditional dispatching rules. In future work, we aim to evaluate our

enhanced dispatching rule under different machine breakdown scenarios. Our second target of future work concerns hyperparameter tuning of GSACO-O parameetrs using machine learning.

We evaluated our scheduler by simulating it as a dispatcher and comparing its performance with traditional dispatching rules. The results indicate that our approach can potentially streamline operations. For future work, we plan to assess the effectiveness of our enhanced dispatching rule under various machine breakdown scenarios, which will provide insights into its resilience and adaptability in dynamic environments. Additionally, we aim to explore hyperparameter tuning of GSACO-O parameters through machine learning techniques to further optimize performance. Leveraging machine learning could help us identify optimal configurations more efficiently, enhancing the scheduler's responsiveness and adaptability to different manufacturing setups.

Acknowledgments

This work has been funded by the FFG project 894072 (SwarmIn) and the Austrian Science Fund (FWF) cluster of excellence 10.55776/COE12.

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