A multi-agent system simulation based approach for collision avoidance in integrated Job-Shop Scheduling Problem with transportation tasks

Kader Sanogo, Abdelkader Mekhalef Benhafssa, M'hammed Sahnoun, Belgacem Bettayeb, Moussa Abderrahim, Abdelghani Bekrar

A multi-agent system simulation based approach for collision avoidance in integrated Job-Shop Scheduling Problem with transportation tasks

Kader SANOGO[a,*], Abdelkader MEKHALEF BENHAFSSA[a], M'hammed SAHNOUN[b], Belgacem BETTAYEB[c], Moussa ABDERRAHIM[d], Abdelghani BEKRAR[e]

[a]*CESI LINEACT, EA 7527, Angouleme Campus, La Couronne, 16400, France*
[b]*CESI LINEACT, EA 7527, Rouen Campus, S. E du Rouvray, Rouen, 76800, France*
[c]*CESI LINEACT, EA 7527, Lille Campus, Lille, 59800, France*
[d]*University of Relizane, , Relizane, 48000, Algeria*
[e]*LAMIH, UMR CNRS 8201, UPHF - Campus Mont Houy, Valenciennes, 59313, France*

Abstract

At the operational level of Flexible Manufacturing Systems (FMS), several research works have developed solutions not fully realizable considering the operational constraints, which can be difficult to model. One of the well-known problems in FMS is the Job-Shop Scheduling Problem (JSSP) with transportation tasks. Indeed, this problem is addressed by many researchers, but most of their studies do not consider operational constraints such as collision avoidance between transporters. Moreover, optimized scheduling may contain deadlock situations due to often neglected geometric constraints. Therefore, in this paper, we propose a simulation approach based on a multi-agent system to test the optimized results known in the literature in more realistic conditions, where collision avoidance between transporters is considered. We propose as well a sim-optimization approach to explore suitable solutions considering collision avoidance constraints. In addition, an algorithm is proposed to solve deadlocks. The obtained results highlight the impact of collision avoidance on the performance of the literature solutions while being more realistic. Furthermore, these results demonstrate the efficiency of the proposed algorithms, since all identified collisions are avoided and all deadlocks are also solved. The proposed approach is expected to be more attractive to the industries as it ensures a small gap between the theoretical and actual results.

Keywords: JSSP, Transportation tasks, AGV, Simulation, Multi-agent system, Collision avoidance

1. Introduction

For several decades, the fierce competitiveness of industrial companies leaves no room for inefficiency and wasting of time and resources. Companies are constantly looking for new processes, new technologies, and new paradigms to optimize their industrial productions and increase customer satisfaction. This involves solving several operational optimization problems within the production system of the company.

The planning and scheduling of production tasks represent one of the most difficult and challenging optimization problems usually encountered by operations managers in all industrial sectors. Thus, several research works have studied these types of problems for different configurations and environments of production systems using different approaches. The proposed solutions have found many real-world applications, although sometimes assumptions might be more or less realistic, which creates a gap between the actual and the theoretical performances of the solutions. This is why some classic problems in the literature are perpetually revisited to include new assumptions, to take into account new technologies, new organizational paradigms, new external and internal constraints, or to use new methods and tools of resolution.

*Corresponding author
Email addresses: ksanogo@cesi.fr (Kader SANOGO), amekhalef@cesi.fr (Abdelkader MEKHALEF BENHAFSSA), msahnoun@cesi.fr (M'hammed SAHNOUN), bbettayeb@cesi.fr (Belgacem BETTAYEB), abderrahim.moussa@univ-relizane.dz (Moussa ABDERRAHIM), abdelghani.bekrar@uphf.fr (Abdelghani BEKRAR)

In Flexible Manufacturing Systems (FMS) environment, a great interest is noted on several variants of the well-known Job-Shop Scheduling Problem (JSSP) [9]. The JSSP is a problem in which products processing must be sequenced on several workstations [2]. Each product must undergo several operations one after the other, and each of these operations must be carried out by a particular machine defined in a predetermined order. Each machine can only perform a specific operation per product (job) and cannot be substituted by any other one. The main objective of the JSSP is to optimize the scheduling resulting from the sequencing of the operations of the jobs (products) to be processed on each machine, while respecting the precedence constraints between the operations constituting each job.

The laborious and repetitive tasks of transporting products between the different machines and stocks on the shop floor are at the heart of the job-shop activities, as they have a direct impact on the efficiency of industrial activities. Thanks to the progress of industrial automation, these transportation tasks are now performed by mobile robotic systems called Autonomous Guided Vehicles (AGV). It is therefore becoming essential to optimize the scheduling of the transportation tasks as well as the production tasks.

Although several researchers have tried to solve the JSSP with transportation tasks, most of these works did not consider operational constraints such as shared pathways and collision avoidance between transporters. Consequently, the optimized scheduling can, in practice, generate some deadlocks due to the often neglected spacial and geometric constraints.

Thus, we propose in this paper, a simulation approach based on a multi-agent system in order to simulate some JSSP optimized results from the literature ([1]), and to put in evidence the possible collisions and deadlocks. As a consequence, algorithms for implementing collision avoidance and for fixing deadlocks (or blocking situations) are suggested. The simulations are carried out on the well-known benchmark instances proposed by Bilge and Ulusoy [7] while using the scheduling extracted from the Variable Neighborhood Search (VNS) algorithm developed by Abderrahim *et al.* [1].

This paper is organized as follows: Section 2 presents a literature review related to JSSP with transportation tasks, and collisions and deadlocks avoidance issues. Section 3 is dedicated to the problem description, where we describe the model of the considered JSSP, the transportation tasks, and the problem of collision avoidance and deadlocks. Section 4 contains the main contribution of this work, which are the proposed approach and the suggested algorithmic solutions. In Section 5, experiments are conducted, and the results are highlighted and discussed. Finally, a conclusion ends this paper.

2. Related work

The issue of job-shop scheduling is at the crossroads with algorithmics, mathematics, operations research, and industrial engineering [45, 24]. Thereby, researchers have handled it by focusing on one or many aspects according to their specialties. Several researchers have considered transportation tasks in their works, such as [27, 25, 16]. The resulting problem is considered as a simultaneous scheduling of production and transportation tasks. Their main objective is the minimization of both costs of production and completion time of the whole scheduling (Cmax, a.k.a. makespan).

Many studies dealing with the problem of transportation in job-shop problems are based on Bilge and Ulusoy [7] benchmark instances. They propose four different layouts, which are well-known in the literature, composed of four machines and one load/unload station (stock) while transportation tasks are done by two AGVs. They solve the problem as a nonlinear mixed integer programming model with the minimization of the makespan as the objective.

Taking it from there, Abderrahim *et al.* [1] address the JSSP in a vehicle-based manufacturing facility that is mainly related to the job assignment and resources. They consider the processing resources that accomplish fabrication tasks for specific products and the transportation resources as the two types of resources belonging to the job-shop production cell. They use a Variable Neighborhood Search (VNS) algorithm to schedule both product manufacturing and transportation tasks in the aim to minimize the makespan. They also improve the lower bounds found by Bilge and Ulusoy [7] with a new calculation method.

Fontes and Homayouni [17] consider integrated production and transportation scheduling problem as two inter-related problems that need to be addressed simultaneously. They formulate the joint production and transportation scheduling as a novel mixed integer linear programming model and show the efficiency of their model in finding optimal solutions by using a commercial software (Gurobi) for computational experiments on benchmark problem instances.

Ham [18] on his side consider machines and autonomous mobile robots (transporters) both as constraining resources. Then, he proposes a novel application of constraint programming for the JSSP with AGVs. This application proves its efficiency on well-known benchmark instances. He also proposes a medium-scale benchmark instance.

Saad *et al.* [33] studies the transportation in the job-shop problem with blocking and no waiting constraints, which is an extension of the classical job-shop problem that takes into account transport operations and the absence of storage space between machines. They formulate the problem by means of a new disjunctive graph, model blocking situations, and describe properties of partial schedule that lead to deadlock situations. They solve the problem with both a greedy heuristic approach based on priority and avoiding blocking cycle rules and an exact method based on a Mixed Integer Linear Program (MILP).

Simulation has been used to solve several problems faced in FMS, such as workshop energy consumption [6] and AGVs fleet size management [34]. However, few works deal with the validation of optimized scheduling by simulation. Zaidi *et al.* [41] used simulation to demonstrate the difference between the simulated and the theoretical schedule in a simple example. This difference should be more pronounced with cases using defined paths of the robots.

Xu *et al.* [40] are among the first ones to introduce a simulation approach to tackle the Flexible Job-Shop Scheduling Problem (FJSSP). Like Jianfeng *et al.* [19], they develop a multi-objective scheduling algorithm that considers the following objectives: makespan, robot travel distance, time difference with due date, and critical waiting time. To do so, they develop a multi-agent-based simulator to test the proposed solution and implement a sim-optimization (simulation-optimization) based genetic multi-objective algorithm. The simulation model is featured to be interactive and allows getting all the real-time information such as remaining jobs, robot positions and machine conditions.

In this regard, sim-optimization has been an active field of research for the last decade to deal with FMS issues with operational constraints' consideration [20, 31]. This approach consists in combining optimization and simulation to solve complex problems that could not (or hardly) be solved by optimization alone [44, 4]. For example, A. Allal *et al.*[3] used a sim-optimization approach to deal with the routing and the scheduling of maintenance for offshore wind farms, which is actually a complex problem.

The implementation of concurrent processes in discrete event systems can lead to deadlocks, which can be defined as resource conflicts that prevent concurrent processes from completing their tasks [43]. A deadlock occurs when the four following conditions are met: (i) mutual exclusion, (ii) hold and wait, (iii) no pre-emption, and (iv) circular wait [10]. The first three conditions are always present in transportation systems based on AGVs [43]. Thus, avoiding the fourth condition to arise is an issue that is addressed in the literature. Researchers have proposed solutions that are suitable for centralized control systems [8], distributed control systems [47], or for both systems [42].

AGVs handling transportation tasks in a workshop is a fairly well documented issue in the literature. The research works can be divided into two categories according to the assumptions made. The first category limits the role of transporters to the simple execution of transportation tasks along a predefined path. This category neglects collision risks [40, 7], fleet management and/or disruptions due to breakdowns [17, 18], battery capacity [1, 39], etc. It should also be noted that this category mainly concerns works relating to JSSP with transportation tasks. The second category takes into account collision risks [36, 15], traffic routing [29, 28], path planning [30, 46], deadlocks resolution [43], etc. The objective is to find a collision-free and time-optimized path [14, 32] for the transporters.

Through this literature review, we can see that the JSSP with transportation tasks is addressed in several studies, but these works mainly focus on the development of solutions based on optimization methods. Additionally, in optimization-based approaches, real operational constraints are usually neglected for simplification purposes. Therefore, the real implementation of these solutions might be difficult, as they do not match the real industrial environment, especially the spatial constraints. Furthermore, simulation, which can be considered as an intermediate step before the real implementation, allows considering more temporal and spatial constraints (e.g. space and shape constraints). Hence, in order to reduce the gap between theoretical study and real implementation, the following contributions are presented in this paper: (i) Verify the feasibility of optimized scheduling by simulation since, to the best of our knowledge, no study focuses on verification by simulation of the proposed optimized solutions for well-known benchmark instances, such as those of Bilge and Ulusoy [7]; (ii) Consider collision avoidance as an operational constraint during simulation, since collision avoidance in JSSPs with transport tasks is not a well-covered issue. To this end, algorithms for collision avoidance and pathways negotiation are proposed. We also introduce a sim-optimization approach to find optimized and more suitable schedules.; (iii) Show the impacts of collision avoidance on optimized schedule results and propose a heuristic, based on collaboration between transporters, to solve deadlocks during simulations; (iv) Test

the sensitivity of our approach by varying the processing times and verify its ability to adapt the optimized scheduling.

3. Problem description

This section will be devoted to the description of JSSP, transportation tasks within the workshop, collision avoidance, and deadlocks.

3.1. JSSP

The JSSP is the problem of sequencing a set of jobs $\mathcal{J} = \{J_1, J_2, ..., J_I\}$ to be processed on a set of machines $\mathcal{M} = \{M_1, M_2, ..., M_M\}$ in a job-shop organization. Each job J_i is composed of a sequence $(O_{i1}, O_{i2}, ..., O_{in})$ of operations to be performed consecutively. The operation O_{ij}, which means operation j of job i, can be performed only on the machine $M_k \in \mathcal{M}$ with the processing time τ_k. Furthermore, a machine can only perform one operation at a time, and preemption of operations is not allowed.

3.2. Transportation task

During the manufacturing process, a job J_i may move from one machine to another to perform its next operation. This transportation task is carried out by one of the single-load AGVs available $V_p \in \mathcal{V}$. A transportation task is denoted by $T_{i,j}$, which means the transportation of the job J_i to the machine selected to perform the operation $O_{i,j}$.

In this paper, our objective is not to find a collision-free path but to avoid collisions and deadlocks while transporters follow a static predetermined path (also called motion planning [21]). In fact, to the best of our knowledge, this work is the first to consider collision avoidance between transporters when they are carrying out transportation tasks while following a predefined path within JSSP. Besides, this work proposes algorithms to deal with deadlocks or blocking situations that could happen in this context.

3.3. Collision avoidance & Deadlocks

Collisions and deadlocks need to be prevented for the safety of the workshop [26]. Collision avoidance is defined as the simple act of preventing AGVs from colliding. Besides, a deadlock (or blocking situation) is a problem that prevents the execution from proceeding well to completion. It is a situation in which an AGV is blocked and cannot perform any operations [12, 37].

Two methods are mentioned in [12] to avoid collisions and deadlocks in a static predetermined path: (i) Centralized motion planning, which is currently used in most industrial AGV systems. It aims to plan a collision-free and deadlock-free path for the entire fleet of AGVs by a central controller, (ii) Decentralized motion planning, which allows AGVs to react individually to avoid collisions and/or deadlocks according to their awareness of the local environment [11, 13]. Recently, hybrid approaches were developed to take advantage of the centralized and decentralized methods to control the robot fleet and define a strategy of collision avoidance [27]. For this study, we decided to adopt hybrid motion planning based on optimization and simulation methods. As it is based on the local reaction of AGVs, this method is, therefore, more appropriate in a multi-agent system rather than the heavy computational demands of a central controller.

To implement collision avoidance, a safety area is defined around each AGV. When two AGVs are too close to entering each other's safety area, the one behind must stop to avoid hitting the one ahead. If the AGV ahead is waiting for a job before handling a transportation task, the AGV behind has to stay motionless until the AGV ahead moves. Taking from there, the following deadlock situation is identified and could happen: V_p is waiting for the job J_i before handling $T_{i,j}$. But, before completing the operation $O_{i,j}$, J_i has to be transported by V_q for operation $O_{i,j-1}$. Thus, while V_p is still waiting for $T_{i,j}$, V_q is coming behind V_p for $T_{O_{i,j-1}}$. Therefore, due to collision avoidance, V_q will stay motionless behind V_p until it moves. Consequently, $O_{i,j}$ will never happen because $O_{i,j-1}$ can not be performed either.

4. Methods

This section will address the description of the experimental method and the proposed solutions, the benchmark instances and the simulation framework used for this work.

4.1. Experimental protocol

In order to achieve the goals of this study and to experiment the proposed solutions, we first define our experimental procedure, which is illustrated in Figure 1.

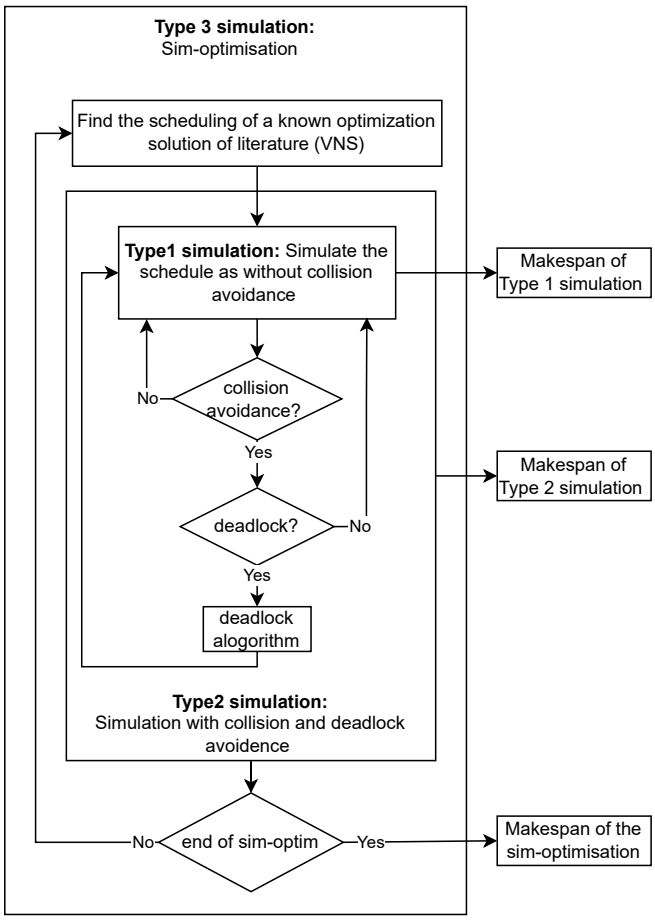


Figure 1: Diagram of the experimental protocol

The experimental method aims to: check the feasibility of known-optimized scheduling proposed in the literature, consider collision avoidance between AGVs and exhibit its effects on the makespan, and then implement a heuristic to solve deadlocks. The method is composed of the following steps:

1. **Find a JSSP benchmark instance with a known optimized solution of the literature**: The instance should take into account transportation tasks performed by AGVs as well as processing tasks performed by production machines.
2. **Simulate the schedule as it is**: In this step, the simulation of both schedules of AGVs and machines is executed to detect potential collision situations without considering their effect. The simulation must remain faithful to the information collected in the previous step.
3. **Simulate the schedules with collision avoidance**: It consists of considering collision avoidance between AGVs during the simulation. It means that all collision situations observed in the previous step are prevented.
4. **Fix deadlocks**: at this stage, it is a matter of applying the proposed algorithm to fix deadlocks.
5. **Keep the makespan**: once the execution is completed, we retain the makespan value (also called Cmax).
6. **Sim-optimization process:** If a gap is observed between the makespans of Type 1 and Type 2 simulations, we run the whole process (VNS optimization and type 2 simulation) until an equivalent makespan is reached or the sim-optimization process is completed.

4.2. *Proposed solutions*

In this work, two heuristics are proposed: an algorithm for collision avoidance and another for fixing deadlocks. These algorithms are inspired by traffic regulations and bird flocking.

4.2.1. *Collision avoidance*

The collision avoidance algorithm is inspired by bird flight. Birds fly in tight rows when they are in colonies while keeping a minimum distance to avoid colliding. Based on the same principle, the collision avoidance rule allows AGVs to be far enough separated to avoid colliding. To do so, a safety area is defined around each AGV. If another AGV is about to penetrate this safety area, it has to stop. This is presented on Figure 2. It is also assumed that each AGV has a field of view centered on its steering.

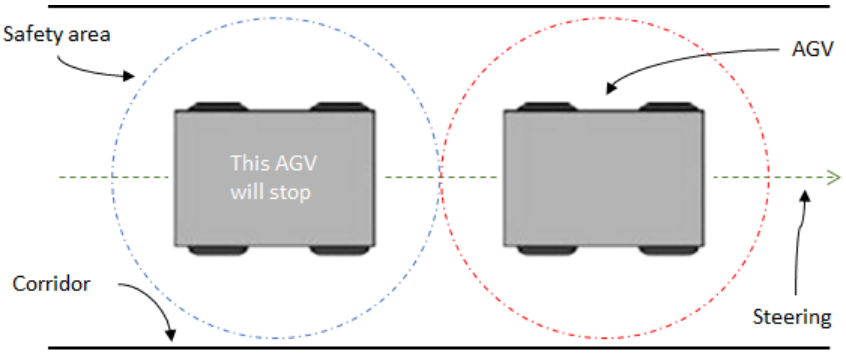


Figure 2: Collision avoidance

The pseudocode of the proposed algorithm for implementing the collision avoidance is shown on Algorithm 1, which ensures that frontal, rear, and side impacts are all prevented. It is a scalable and low-computational algorithm that is executed by each AGV within the fleet. Hence, collisions are avoided in a decentralized way, whatever the fleet size. In [35], a study with a fleet size ranging from 2 to 10 AGVs was conducted.

Algorithm 1 Pseudocode of collision avoidance algorithm

Input: any AGV within the fleet
Output: the AGV motion planning

```
1: procedure AVOID_COLLISIONS(current_agv)
2:     if any other transporter in the safety area then
3:         if any other transporter the field of view then
4:             STOP_MOTION
5:         else
6:             CONTINUE                          ▷ it continues his way
7:         end if
8:     end if
9: end procedure
```

In a more complex topology of the environment, the local perceptions of each AGV are not enough to avoid blocking situations. For that, more global perceptions, based on information exchange between AGVs and external sensors installed in the workshop, are necessary to avoid such situations. When an intersection or a turn is present on the AGV path, it can create blocking situations if the corridors are not wide enough for two AGVs. In this case, we assume that each robot reserves a part of the path to follow from its current position until the next intersection, as shown in Figure 4. Checkpoints are defined as the threshold before the intersection (see Figure 3), where the AGV (called checker) verifies the state of the next portion of the path that will be used for its travel. Algorithm 2 is proposed for implementing this path verification. The following situations can be distinguished:

1. Another AGV moves towards the checker's position along the path. Then, the checker must wait until the path is free because it is already reserved by the other one.
2. Another AGV moves in the opposite direction to the checker along the path. Then, the checker can reserve a portion of the path and go on. Frontal impacts are prevented.
3. The path is free. Then, the AGV reserves a path portion and continues.

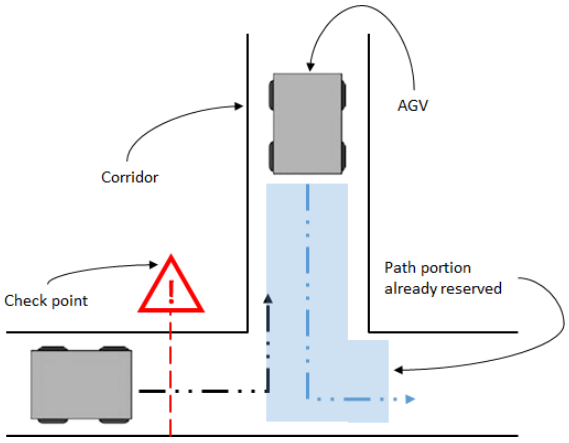


Figure 3: Check point illustration

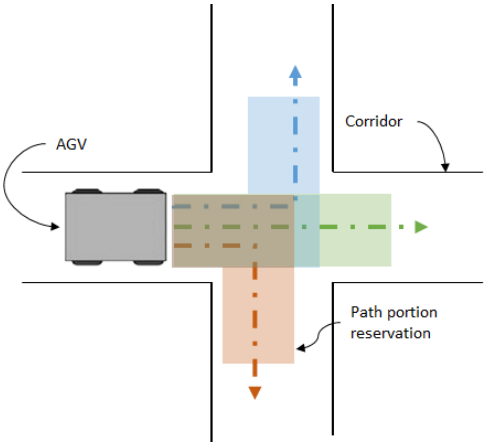


Figure 4: Path portion reservation

Algorithm 2 Pseudocode of checkpoints and path reservation algorithm

Input: any AGV within the fleet
Output: Check path availability and make path reservation

```
1:  procedure CHECK_AND_RESERVE_PATH(current_agv)
2:      let mypath: the bi-directional path the checker is about to take
3:      if [location] of current_agv = checkpoint then
4:          if another AGV on mypath and moving towards current_agv then
5:              STOP_MOTION                          ▷ wait until the path is free
6:          else
7:              RESERVE_PATH(current_agv)            ▷ procedure for path reservation
8:              CONTINUE
9:          end if
10:     end if
11: end procedure
```

4.2.2. Fixing deadlocks

The proposed solution for fixing deadlocks (or blocking situations) is based on collaboration between AGVs. In fact, when an AGV meets another AGV which is waiting for a job to perform an operation that should be preceded by another operation, both AGVs exchange their remaining transportation tasks. The relevance of this collaboration lies in the fact that the number of operations of each job is known by AGVs. Therefore, they know whether a job J_i is ready for a particular operation $O_{i,j}$ according to the σ ($\sigma < j$) previous performed operations ($O_{i,1}, ..., O_{i,\sigma-2}, O_{i,\sigma-1}, O_{i,\sigma}$) of J_i and the rank of $O_{i,j}$ in J_i operation-list. This collaboration prevents any deadlocks and enables the successful completion of the execution. Algorithm 3 shows the pseudocode of the algorithm for implementing this collaboration. It should be noted that Algorithm 3 is appropriate for decentralized deadlocks resolution since it does not involve a

central controller to redefine the pathways and/or reschedule the AGVs. Moreover, in the current study, deadlocks occur when AGVs share pathways as a common resource. Algorithm 3 ensures that no robot can monopolize the pathway for its own use. It also represents a fast algorithm that can be used in the real-time decision process for a fleet of two or more transporters.

Algorithm 3 Pseudocode of Deadlocks Resolution Algorithm (DRA)

Input: any AGV within the fleet
Output: exchange the remaining transportation tasks with another AGV

```
1:  procedure DEADLOCKS_RESOLUTION(current_agv)
2:      if another_agv is idle in front of current_agv then
3:          get another_agv current transportation task
4:          if another_agv is waiting for J_i to handle T_{i,j} and J_i is not ready for O_{i,j} then
5:              EXCHANGE_REMAINING_TRANSPORTATION_TASKS(current_agv, another_agv)
6:          else
7:              AVOID_COLLISIONS(current_agv)
8:          end if
9:      end if
10: end procedure
```

It may happen that AGV2 waits for the job it has just delivered to take it over during the next transportation task. In that case, AGV2 will wait until the completion of the job processing before moving on with it. If AGV1 comes while it is waiting, AGV1 must wait to avoid collisions. This is what the fifth line of Algorithm 3 means. In addition, it may also happen that exchanging the remaining transportation tasks between the two AGVs leads to another deadlock similar to the previous one. Thus, AGV2 will make an empty trip (also called "deadheading" [7]) to allow AGV1 to complete its current transportation task. However, this scenario is quite rare.

4.3. Variable Neighborhood Search (VNS) algorithm

The VNS algorithm was first introduced by N. Mladenovic and P. Hansen in 1997 [22]. The VNS implementation proposed by Abderrahim *et al.* [1] is used during our study. It is based on neighborhood switching and works to find the local optimum through the vertical local search as well as horizontal search to avoid the current valley. To do so, schedules are represented by two strings: the first one represents machines' production tasks and the second one represents vehicles' transportation tasks (see figure 5.(a)). From this two-string-based representation, A local search is initiated to explore the solution set for a better makespan. The algorithm uses two asynchronous local search stages: a local search is done to find a transportation string within the current transportation neighborhood, and then, a second local search is used on the production neighborhood to find the production string that gives the best Cmax with the current transportation string. This is repeated with all the elements from the current transportation neighborhood until the local search stop condition is met or a better production-transportation combination is found. Thereafter, the whole process is repeated with a new transportation neighborhood until reaching the stop condition. A detailed flowchart is sketched in Figure 5.(b). For more details, please refer to [1].

4.4. Sim-optimization approach

As highlighted by Barton [5], when a model has a large number of parameters or when one or more parameters have too many values, it becomes very difficult to evaluate all possible alternatives, requiring the use of optimization via simulation (OvS) techniques [38], which is a specific type of sim-optimization approaches. Indeed, integrating collision and deadlock avoidance constraints in the modeling of advanced schedules is not easy. We must consider and forecast any collision on each common portion of the pathway within the working environment. To alleviate this problem, the sim-optimization approach schematized in Figure 6 is used to find optimized and suitable schedules by considering collision avoidance. In fact, after running the VNS algorithm, the obtained schedule is simulated considering the collision avoidance constraint. If the simulated makespan is delayed, we look for another optimized schedule from VNS. To limit the computational effort, the process is stopped after 5 attempts since the VNS algorithm is configured to return a schedule after 50 runs. This approach aims to find feasible advanced schedules in simulation or, at least, find advanced schedules with the smallest delay.

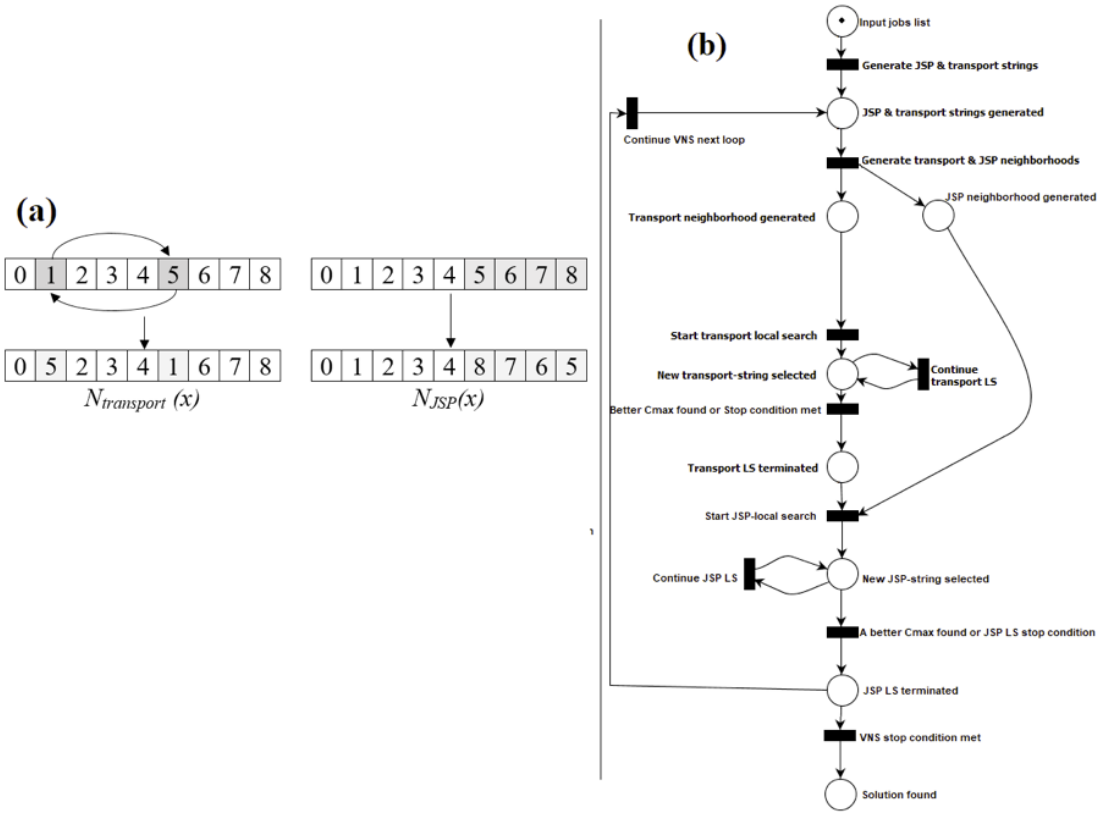


Figure 5: VNS neighborhoods (a) and its implementation (b) proposed by [1]

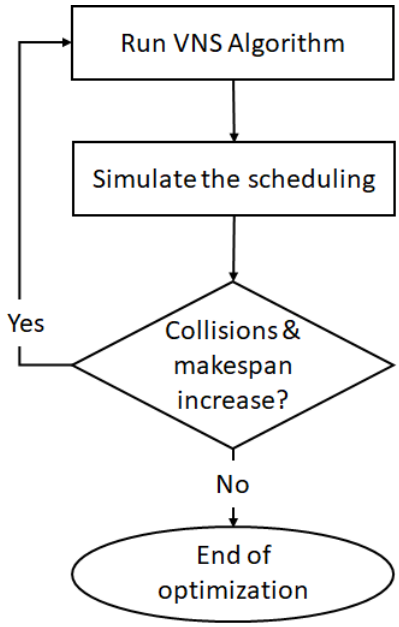


Figure 6: Sim-optimization approach

4.5. Layouts & Instances

Experiments are carried out through simulations using the well-known benchmark instances proposed by Bilge and Ulusoy [7]. They proposed 4 different layouts of the job-shop, composed each of an L/U (load/unload) station and 4 machines. The L/U station is used as a storage place for all jobs before the beginning of processing (row material) and after the end of operations completion (finished products). Transportation tasks are carried out by two identical uni-charge AGVs. All transportation tasks start and end at the L/U station. Each layout has a particular trip orientation, travel times, and L/U station and machine location. They also proposed 10 different job sets, where each job set is composed of 5 to 8 jobs. Jobs are made up of several operations with the corresponding machines and the associated processing times. The proposed test instances are noted "EX$\alpha\beta$" where α and β represent respectively the job set and the layout. Notice that travel times and processing times are in seconds.

Table 1 shows the travel times between all elements of the workshop for each layout. In order to simulate the system, travel times are turned into travel distances. It is assumed that the distance unit is a meter and the speed of AGVs is 1 m/s (meter/second). When two elements can be connected by several paths, the shortest one is privileged. Therefore, the presented distances in Table 1 are the shortest. Figure 7 shows the 4 layouts used for the experiments with the travel distances.

	L/U	M1	M2	M3	M4
L/U	0	6	8	10	12
M1	12	0	6	8	10
M2	10	6	0	6	8
M3	8	8	6	0	6
M4	6	10	8	6	0

(a) Layout 1

	L/U	M1	M2	M3	M4
L/U	0	4	6	8	6
M1	6	0	2	4	2
M2	8	12	0	2	4
M3	6	10	12	0	2
M4	4	8	10	12	0

(b) Layout 2

	L/U	M1	M2	M3	M4
L/U	0	2	4	10	12
M1	12	0	2	8	10
M2	10	12	0	6	8
M3	4	6	8	0	2
M4	2	4	6	12	0

(c) Layout 3

	L/U	M1	M2	M3	M4
L/U	0	4	8	10	14
M1	18	0	4	6	10
M2	20	14	0	8	6
M3	12	8	6	0	6
M4	14	14	12	6	0

(d) Layout 4

Table 1: Travel times

4.6. Simulation model

The simulation model is based on multi-agent system that allows an ease representation of complex systems. The proposed model involves 4 main agents that are: AGVs, machines, jobs, and stocks. Each machine is equipped with two buffers: an input-buffer for storing jobs before processing and an output-buffer for storing processed jobs before transportation. These buffers and the L/U station are considered as stocks. The interactions between agents are described as follows:

- AGVs pick up jobs from output-buffers (resp. the load station), and deliver them to input-buffers (resp. the unload station).
- Stocks store jobs before (resp. after) their processing by machines.
- Machines retrieve jobs from input-buffers, process them and store them in output-buffers.
- Jobs are transported by AGVs to the stocks.

For each agent, we consider the following assumptions:

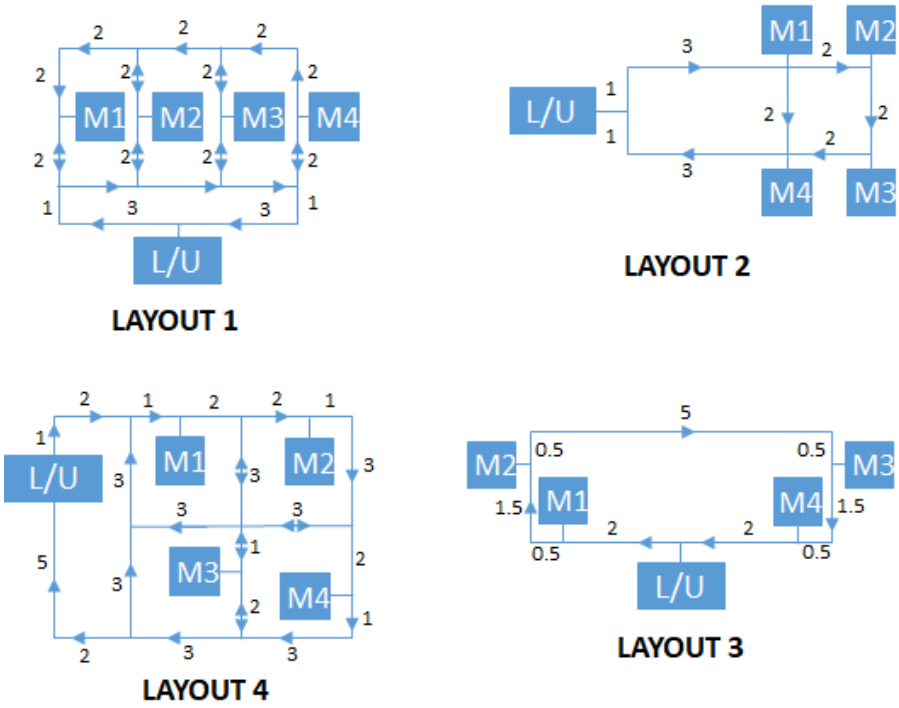


Figure 7: The 4 layouts proposed by Bilge and Ulusoy

- AGV:
  - AGVs are independent of each other.
  - Each AGV can transfer one and only one job at a time.
  - Time required to load and unload jobs from buffers is included in transportation time.
  - AGVs have an orders list in which all the transportation tasks to be carried out are recorded with the jobs involved, the operations to be performed and the associated machines according to the scheduling.
  - It is possible to know the current transportation task of AGVs thanks to the current-order list, which is the i^{th} element of the orders-list.
  - AGVs have to move through the direction imposed by the layout.
  - AGVs have to respect exactly the defined scheduling, i.e. take the right job, follow the right path and go to the right machine as it is described in the scheduling.
  - AGVs may collaborate, but should avoid collisions.
  - Issues such as: failure, battery recharging, fleet management, etc. are not considered and left for future works.
- Machine:
  - Machines completely independent.
  - Each machine can perform one and only one operation at a time.
  - Machine setup times and breakdowns are ignored.
  - Machines process jobs according to the scheduling.
  - All the machines are available at the beginning of the simulation.

- Job:
  - Jobs are independent.
  - Each job can be processed on one and only one machine at a time.
  - Each job can be transported by one and only one AGV at a time.
  - Job operations must be performed according to the define sequence.
  - Preemption of operation is not allowed.
- Stock:
  - Machine buffers have an identical limited capacity.
  - L/U station capacity is unlimited.
  - Stocks can store products as long as possible.

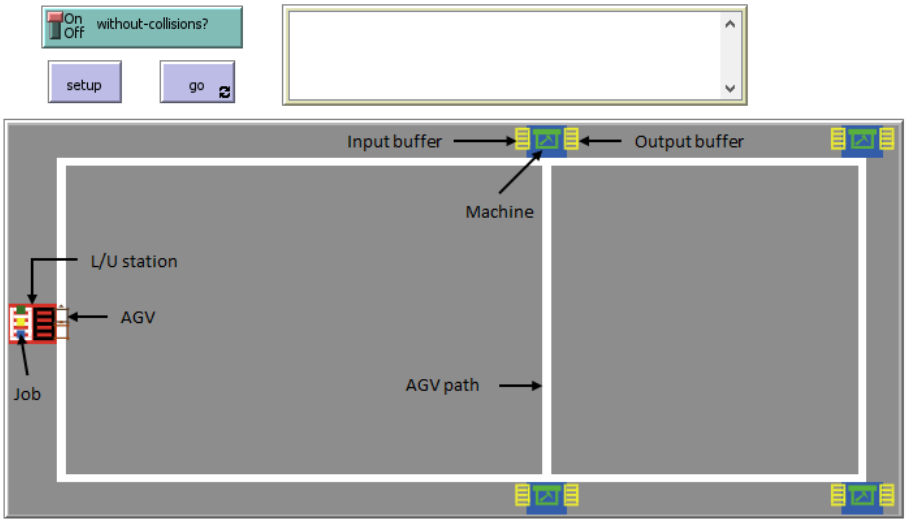


Figure 8: Layout 2 simulation interface on Netlogo

4.7. Framework

The whole simulation model, composed of the MAS, the layout and the instances, is implemented on NetLogo 6.2. NetLogo is both a programming language and a simulator that is used to model and simulate systems making several agent's interactions. It works as a desktop application and makes the modeled system able to be used directly and easily by any other user. Each agent is called ***turtle*** and each turtle belongs to a ***breed***. It is possible to define global parameters for the whole model, and parameters can be restricted to a breed. Characteristics or parameters defined for a particular breed can not be called by another breed. In our case for example, AGVs, machines, jobs and stocks are considered as 4 different breeds. Figure 8 shows a screenshot of layout 2 simulation interface.

Simulation begins with the ***setup*** button. It initializes (or reinitializes) the model and the simulation environment. Once the environment is set up, the simulation is launched with the button ***go***. The latter can be used either in ***one time*** or ***forever*** mode. In the first mode, it triggers an action and then stops. The second mode, on its side, ensures a continuous execution of the simulation. It will stop only if a stop condition is specified in the code or if the go button is pressed again.

The simulation time is counted in ***ticks***, which is a simulation step. In order to be able to simulate the production and transportation tasks, we assume that 20 ticks represent 1 second. Therefore, if an operation should be performed in 8 seconds in the benchmark instances, this operation will be performed in $8 \times 20 = 160$ ticks during the simulation.

Likewise, an AGV moves a patch forward in one tick. Thus, travel that should be completed in 2 seconds, for example, will be completed in $2 \times 20 = 40$ ticks.

The simulator presented in Figure 8 allows real-time visualization of the evolution of agents, and it is able to perform a dynamic scheduling simulation optimization process [35] as well as the possibility to provide advanced schedules. In this latter case, the simulator takes the advanced scheduling matrix as input. Table 2 presents the matrix of the instance EX11 advanced scheduling (taken as an example), generated after optimization using VNS algorithm [1]. The first row of Table 2 reports jobs (J_i). The second line gives the locations where jobs must be sent. The location can be a machine (M_k) or the L/U station. The third line identifies the transporter (T_p) in charge of the transportation. And finally, the fourth line refers to the operations (O_{ij}) that jobs must undergo. "R" is the job return to the L/U station after being processed. The matrix is read by column. For example, the first column means that the transporter T_2 must transport the job J_3 to the machine M_3 to undergo the operation O_{31}. By doing so, all the transportation tasks can be extracted for each transporter, and put in their orders-list. Similarly, production tasks can be extracted for each machine.

Once production and transportation tasks are extracted, the model is simulated. At the end of each simulation, the simulator reports the makespan, the number of collisions encountered and the number of deadlocks solved. This process is schematized on Figure 9.

J_3	J_4	J_1	J_1	J_2	J_1	J_5	J_3	J_3	J_4	J_5	J_2	J_2	J_1	J_5	J_4	J_3	J_2
M_3	M_4	M_1	M_2	M_1	M_4	M_3	M_4	M_1	M_2	M_1	M_3	M_2	L/U	L/U	L/U	L/U	L/U
T_2	T_2	T_1	T_1	T_2	T_2	T_1	T_1	T_2	T_1	T_1	T_2	T_2	T_1	T_1	T_2	T_2	T_1
O_{31}	O_{41}	O_{11}	O_{12}	O_{21}	O_{13}	O_{51}	O_{32}	O_{33}	O_{42}	O_{52}	O_{22}	O_{23}	R	R	R	R	R

Table 2: Matrix of EX11 advanced scheduling

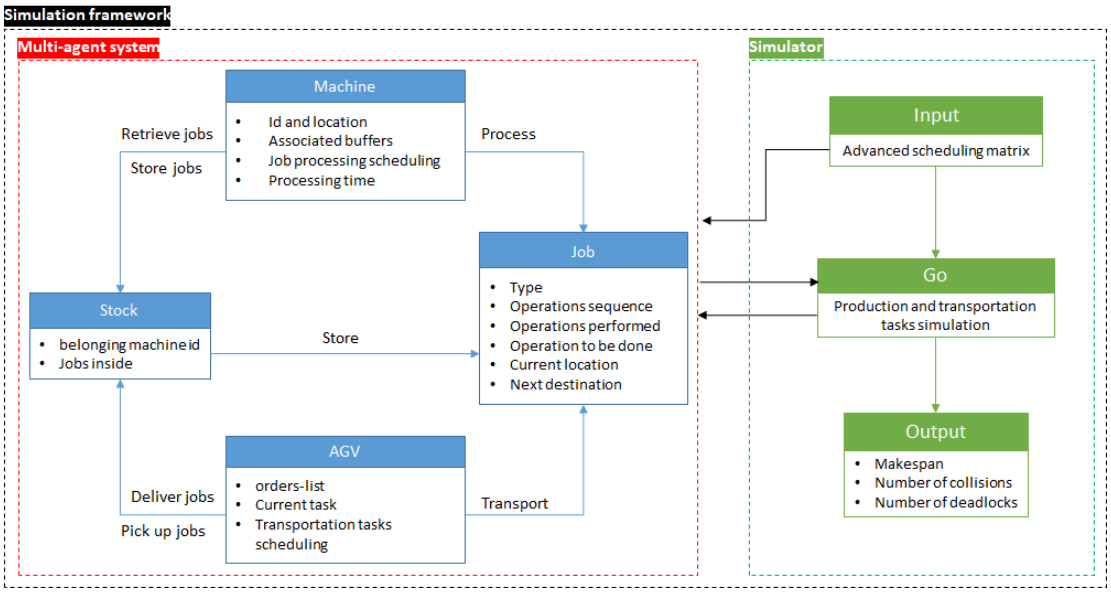


Figure 9: Simulation framework diagram

4.8. Sensitivity analysis

In order to go further in our study, simulations with a probabilistic consideration were conducted. It was assumed that machine processing times follow a normal distribution $\mathcal{N}(\mu, \sigma^2)$: μ, the mean value is the machine processing time as presented in the job set instance, and σ is the standard deviation that represents the margin of error from the mean. σ is written as $\sigma = \lambda \times \mu$, where λ is the relative standard deviation (RSD) divided by 100 ($RSD = \frac{\sigma}{\mu} \times 100$),

considered as the margin of error. For example, in job set 1 in Appendix A, the processing time of the first operation of job 1 is 8s. However, with probabilistic consideration, this processing time will be given by the normal distribution $\mathcal{N}(8, (8\lambda)^2)$. The same approach will be used for all other operations.

Three different cases were tested according to the value of λ: 0.1 (RSD = 10%), 0.2 (RSD = 20%), and 0.3 (RSD = 30%). For each value of λ, all the benchmark instances (10 job sets for 4 layouts) were simulated 50 times in both situations (with and without collision avoidance consideration). At the end of each run, the makespan, the number of collisions, and the number of deadlocks were reported. Once the 50 runs are completed, we calculate the average and standard deviation of the reported values. Finally, these results are compared with the optimization results.

5. Experiments

This section is dedicated to the presentation of the experiments conducted on the simulator developed using Netlogo 6.2. The hardware used is an *Intel(R) Core(TM) i7-8565U CPU* with 1.99 GHz and 16.0 Go RAM. As announced previously, the aims of these experiments are:

- check the feasibility of a known optimized schedule by simulation;
- consider collision avoidance and exhibit its effects;
- identify the deadlocks and blocking situations;
- implement the proposed algorithms to fix the troubles encountered.

5.1. Experimental protocol

The experiments consist of simulating the scheduling proposed by Abderrahim *et al.* [1], which are, to the best of our knowledge, among the best results in the literature that optimize the job-shop scheduling problem using the benchmark instances of Bilge and Ulusoy [7]. They proposed a solution based on the *Variable Neighborhood Search (VNS)* algorithm to schedule the production of a set of jobs with consideration of transportation tasks. The aim of their work is to minimize the maximum completion time of the job sets (Cmax) and to improve the lower bound proposed in the literature with a new calculation method. The results of Bilge and Ulusoy [7] do not consider products' return time to the L/U station because they focus only on the minimization of the maximum completion time of production tasks. In our case, transportation tasks are as important as production tasks. Therefore, we consider the makespan as the maximum completion time of both production and transportation tasks. Consequently, the results of Abderrahim *et al.* [1] are presented in both situations: with and without products' return time consideration. On one hand, the experiments prove the efficiency of their approach compared to the results of [7], and on the other hand, they are used to generate the schedules that will be simulated.

Our experimentation process is described as follows:

1. We run each instance with the algorithm proposed by Abderrahim *et al.* [1] 50 times. Each instance is launched with and without products' return times, and then, the best-optimized scheduling is selected for each case.
2. The best-optimized scheduling with products' return to the L/U station is simulated for each instance.
3. The best-optimized scheduling with products' return to the L/U station is simulated again with collision avoidance consideration between AGVs. If a deadlock is encountered, Algorithm 3 presented in section 4.2.2 is implemented.
4. If the simulation with collision avoidance consideration is delayed, the sim-optimization approach is applied to find a suitable scheduling. As mentioned in Section 4.4, this approach is limited to 5 attempts.

Instance	Cmax[7]	LB[1]	BFS[1]	BFS-L/U	Type 1 simulation	Type 2 simulation	Type 3 simulation	# Collisions	# Deadlocks
EX11	96.0	76.0	96.0	120.0	120.0	120.0	-	0	0
EX21*	105.0	86.0	100.0	122.0	122.0	125.65	125.65 (122.0)	1 → 3	0
EX31*	105.0	88.0	102.0	132.0	132.15	132.2	132.2 (132.0)	2 → 0	0
EX41	118.0	78.0	116.0	147.0	147.0	147.05	-	3	0
EX51	89.0	65.0	87.0	116.0	116.0	116.15	-	1	0
EX61*	120.0	108.0	120.0	138.0	138.0	143.2	142.9 (139.0)	2	0
EX71*	119.0	77.0	121.0	159.0	159.0	162.6	160.15 (160)	1 → 1	0
EX81*	161.0	161.0	161.0	167.0	167.1	171.45	171.45 (167.0)	5 → 3	0
EX91*	120.0	105.0	117.0	130.0	130.1	130.1	130.1 (130.0)	1 → 2	0
EX101*	153.0	133.0	152.0	169.0	169.0	173.45	173.30 (169.0)	4 → 3	0
EX12*	82.0	76.0	82.0	92.0	92.0	102.8	99.0 (99.0)	1 → 0	0
EX22*	80.0	76.0	76.0	82.0	82.0	82.2	82.2 (82.0)	1 → 0	0
EX32	88.0	80.0	85.0	95.0	95.0	95.0	-	1	0
EX42*	93.0	70.0	92.0	109.0	109.1	116.8	109.0 (109.0)	1 → 0	1 → 0
EX52*	69.0	64.0	69.0	84.0	84.0	85.05	84.0 (84.0)	1 → 0	0
EX62*	100.0	98.0	98.0	102.0	102.15	108.5	102.35 (102.0)	2 → 1	1 → 0
EX72*	90.0	74.0	84.0	101.0	101.0	101.0	101.0 (101.0)	0	0
EX82	151.0	151.0	151.0	155.0	155.15	155.3	-	3	5
EX92	104.0	98.0	102.0	106.0	106	106.1	-	1	0
EX102*	139.0	128.0	139.0	142.0	142.0	156.15	145.0 (145.0)	3 → 1	2 → 0
EX13	84.0	74.0	84.0	99.0	99.0	99.4	-	1	0
EX23	86.0	82.0	86.0	92.0	92.0	92.0	-	0	0
EX33*	86.0	82.0	86.0	101.0	101.0	107.0	104.1 (104.0)	1 → 0	1 → 0
EX43	95.0	71.0	92.0	112.0	112.0	112.0	-	0	0
EX53*	76.0	63.0	74.0	93.0	93.0	96.05	93.1 (93.0)	2 → 1	0
EX63*	104.0	100.0	104.0	110.0	110.05	117.75	117.75 (110.0)	2	2
EX73*	91.0	76.0	87.0	112.0	112.0	114.75	112.0 (112.0)	2 → 1	1 → 0
EX83*	153.0	153.0	153.0	155.0	155.0	161.9	155.15 (155.0)	2	2 → 3
EX93*	110.0	100.0	105.0	108.0	108.0	116.65	109.15 (109.0)	2 → 0	0
EX103*	143.0	133.0	143.0	148.0	148.0	181.15	148.05 (148.0)	4 → 2	1 → 0
EX14	108.0	76.0	103.0	144.0	144.0	144.0	-	0	0
EX24	116.0	84.0	113.0	156.0	156.0	156.15	-	0	0
EX34*	116.0	87.0	113.0	160.0	160.0	162.4	162.4 (160.0)	1	0
EX44	126.0	81.0	131.0	180.0	180.0	180.0	-	0	0
EX54	99.0	62.0	97.0	140.0	140.0	140.15	-	0	0
EX64	120.0	103.0	129.0	168.0	168.0	168.15	-	0	0
EX74	136.0	78.0	134.0	191.0	191.0	191.0	-	0	0
EX84	163.0	163.0	163.0	190.0	190.0	190.25	-	1	0
EX94*	125.0	102.0	123.0	156.0	156.0	160.2	158.0 (158.0)	1 → 0	0
EX104*	171.0	136.0	169.0	202.0	202.0	202.0	203.3 (202)	1	0

Table 3: Comparison of the obtained results: (**Cmax:** Results obtained by Bilge and Ulusoy [7]; **LB:** Lower Bound of the makespan computed by Abderrahim *et al.* [1]; **BFS:** Makespan of the Best Found Solution obtained by Abderrahim *et al.* [1] without return to the L/U station; **BFS-L/U:** Makespan of the Best Found Solution among all the experiments with products' return times consideration; **Type 1 simulation:** Results of simulation without collision avoidance to validate the simulator; **Type 2 simulation:** Results of simulation with collision avoidance and the algorithm for fixing deadlocks for the BFS-L/U scheduling; **Type 3 simulation:** Results of the sim-optimization process (values in brackets are the VNS results); **#Collisions:** The number of collisions during the Type 1 simulation → their new number after the sim-optimization; **#Deadlocks:** The number of deadlocks solved → their new number after the sim-optimization.)

5.2. Numerical results

Optimization and simulation results of the benchmark instances are presented in Table 3. The benchmark instances are indicated in *"Instance"* column. Instances with an asterisk are those concerned by the sim-optimization approach. *"Cmax"* column represents the optimized results of the original benchmark instances proposed by Bilge *et al.* [7]. *"LB"* column represents the lower bounds obtained by Abderrahim *et al.* [1] with their new calculation method. The

column *"BFS"* represents the Best Found Solution obtained with the VNS algorithm proposed by Abderrahim et al. [1]. This column, like *"Cmax"* column, does not take into account the return of products to the L/U station after processing. This is done in order to compare these two columns and to highlight the efficiency of the latter, which is giving better results in most cases.

"BFS-L/U" column represents the Best-Found Solution with products' return to the L/U station using the VNS algorithm proposed by Abderrahim et al. [1] among all the run of the algorithm. This will, firstly, allow including products' return time in the makespan and, secondly, to better highlight AGVs' collision risks. *"Type 1 simulation"* column represents the makespan obtained by the simulation of BFS-L/U scheduling, and *"Type 2 simulation"* column represents the makespan obtained by the simulation with collision avoidance consideration of the BFS-L/U scheduling. *"Type 3 simulation"* column relates the values of the sim-optimization approach. Values in brackets are the corresponding makespan obtained by the VNS algorithm during the sim-optimization process. The column *"#Collisions"* represents the number of collisions encountered during Type 1 simulation and finally, the column *"#Deadlocks"* represents the number of deadlocks fixed thanks to Algorithm 3 during Type 2 simulation. In these last two columns, the arrow ($\rightarrow$) indicates the new values found after sim-optimization process.

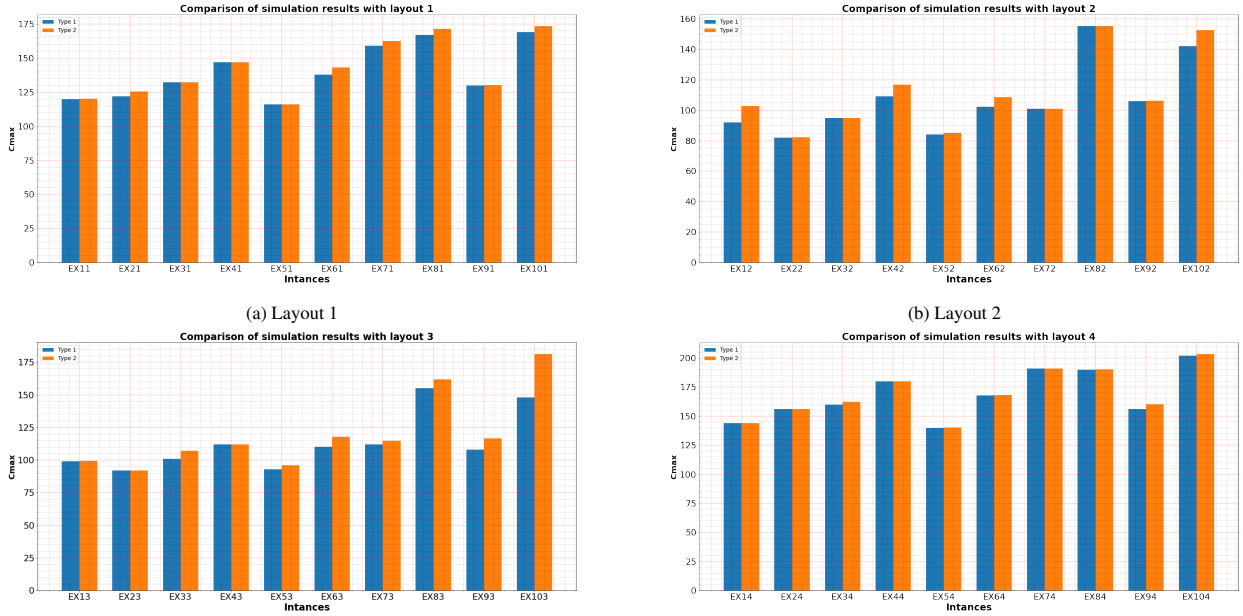
5.3. Discussion

"*BFS-L/U*", "*Type 1 simulation*" and "*Type 2 simulation*" columns of Table 3 are used to make a comparison between the three scenarios. "*BFS-L/U*" is considered as the reference of comparison, which is the theoretical solution. It can be observed that the values of "*BFS-L/U*" and "*Type 1 simulation*" columns are globally identical, with a very small difference of less than 0.15s in the worst case (for example EX31 and EX82). This difference is due to the computation approximation of C++ for VNS and NetLogo for simulation. This first result validates the simulator and demonstrates the feasibility by simulation of the schedules obtained from VNS algorithm without collision consideration. The simulation satisfies exactly all the characteristics of the generated scheduling: the sequence of transportation tasks of each AGV, the sequence of processing tasks of each machine, and the travel time of the layout were the same in both cases.

Regarding the different results presented in Table 3, we can make the following conclusions:

- **Collision avoidance consideration increases the makespan:** This phenomenon is more noticeable when looking at Figure 10. The different charts are used to compare the obtained Cmax for both types of simulation. There are 50% of significant makespan increases (more than 1s) for layouts 1 and 2, 70% for layout 3, and 30% for layout 4. This can be explained by the fact that AGVs delay each other due to collision avoidance. Indeed, when an AGV is waiting for a task to be processed, the one following it is obliged to wait until the former finishes its task, which creates a delay in its scheduling. This can be illustrated in Figure 11 which represents the Gantt diagrams of the instance EX12 using simulation types 1 and 2. The schedules resulting from type 2 simulations contain an asterisk ("*"). On AGV2 scheduling after type 2 simulation, it can be noted a longer delay between $T_{O_{4,2}}$ and $T_{O_{5,2}}$ compared to the scheduling of type 1 simulation. This is because AGV2 must stop behind AGV1 while the latter was waiting for J_2 before continuing. It should also be noted that this waiting time delayed AGV2 scheduling.
  However, there is an exception when the processing time of the job to be loaded is greater than the sum of travel time and motionless time. Indeed, if an AGV comes to load a job that is still being processed on the machine, it will have to wait until the operation is completed, regardless of what has happened on its way. Therefore, its scheduling will not be delayed. This phenomenon can be observed on EX32 Gantt diagrams, where the makespan is still 95s despite the recorded collision. In Figure 12, there is no difference between the two schedules, while the collision encountered took place at 77.3s. Figure 13 is a zoom on AGVs Gantt diagrams at the moment of collision. During type 1 simulation, AGV1 meets AGV2 at 77.3s, arrives to its destination at 81.4s, and waits until J_1 processing completion at 82.6s. In type 2 simulation, AGV1 waits behind AGV2 from 77.3 to 78.65s, and finally arrives at 82.6s to load J_1.

- **Known-optimized scheduling can be disturbed:** An AGV delaying another due to collision avoidance and/or job waiting can disturb optimized scheduling. For example, if an operation O_{xy} was initially scheduled to be processed before another operation $O_{x'y'}$ and if the operation O_{xy} is delayed, the initial sequence may no longer be respected. Sometimes, this can lead to resource conflicts, and therefore deadlocks. We observe that some



(a) Layout 1

(b) Layout 2

(c) Layout 3

(d) Layout 4

Figure 10: Comparison of the obtained values after simulations

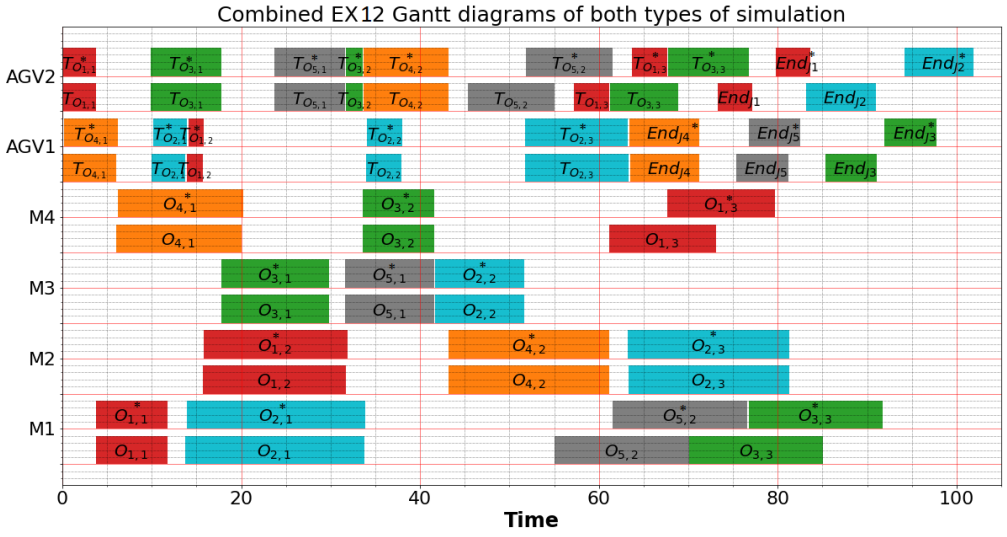


Figure 11: Gantt diagram of EX12 after simulations

collisions avoided lead to deadlocks. That is why, in Table 3, the number of deadlocks is always less than or equal to the number of collisions, except for EX82, which is presenting another phenomenon explained in the next paragraph.

- **A deadlock resolution can lead to new ones:** This phenomenon was observed with EX82. We counted 5 deadlocks and 3 collisions, but this only accounts for the situations where the AGVs encountered issues after exchanging the remaining transportation tasks. Subsequently, the AGVs encountered new deadlocks due to job waiting and/or collision avoidance. On Figure 14, the three collisions identified took place at the 45th, 60th, and 145th seconds. But, the transportation task exchanges took place at the 45th, 60th, 75th, 91st, and 124th

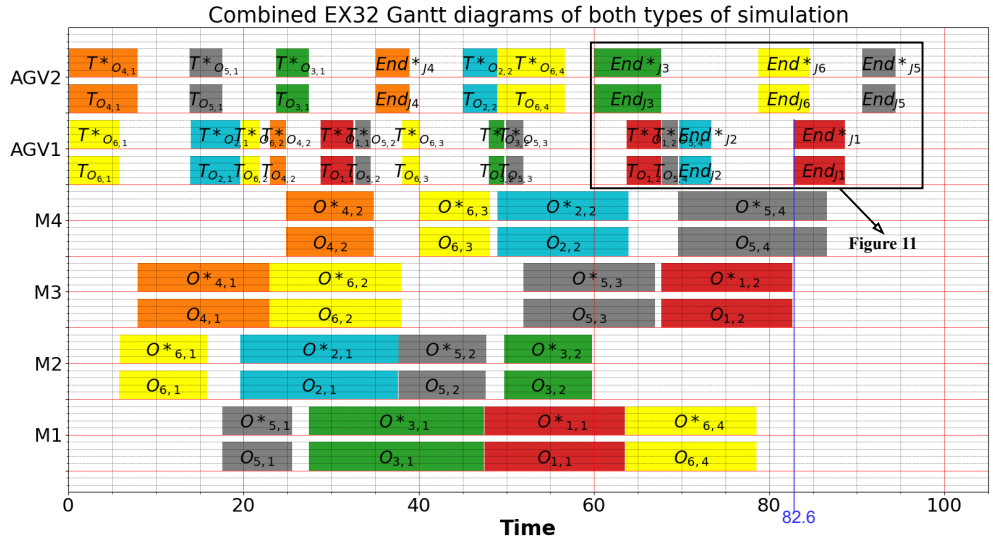


Figure 12: Gantt diagram of EX32 after simulations

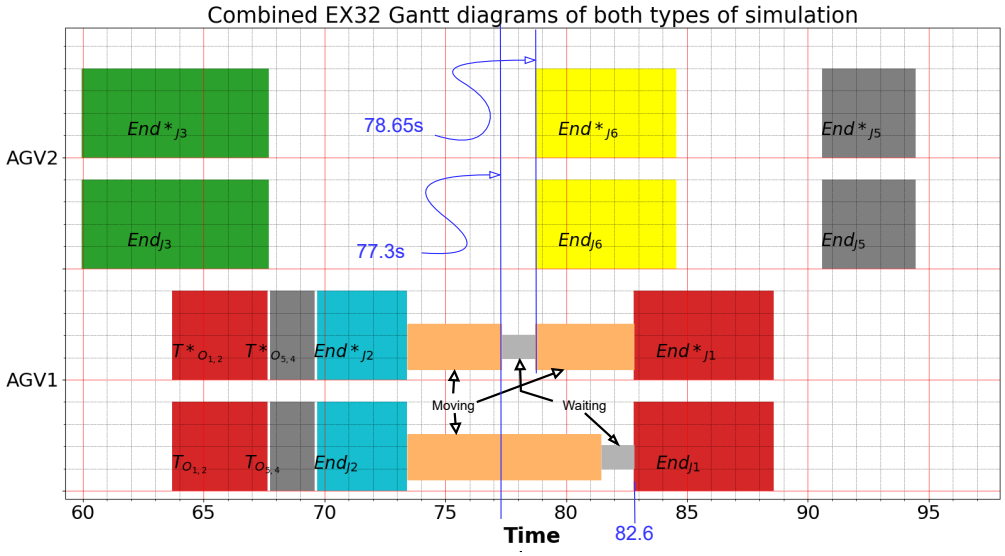


Figure 13: Zoom on AGVs schedules in EX32

seconds.

- **An AGV scheduling with long waiting times increases deadlock risks:** There are optimized schedules that force the AGV to wait a long time for a job. During this time, the AGV blocks the path and can interfere with the activities of other AGVs. This is the case, for example, for EX82, EX63, and EX83.
- **Routing flexibility avoids deadlocks:** The more flexible the layout is in terms of routing, the less risk of AGVs meeting in the corridors. Therefore, this avoids situations leading to deadlocks [23] and explains why no deadlocks were recorded for layouts 1 and 4, which offer more possibilities to reach a destination. However,

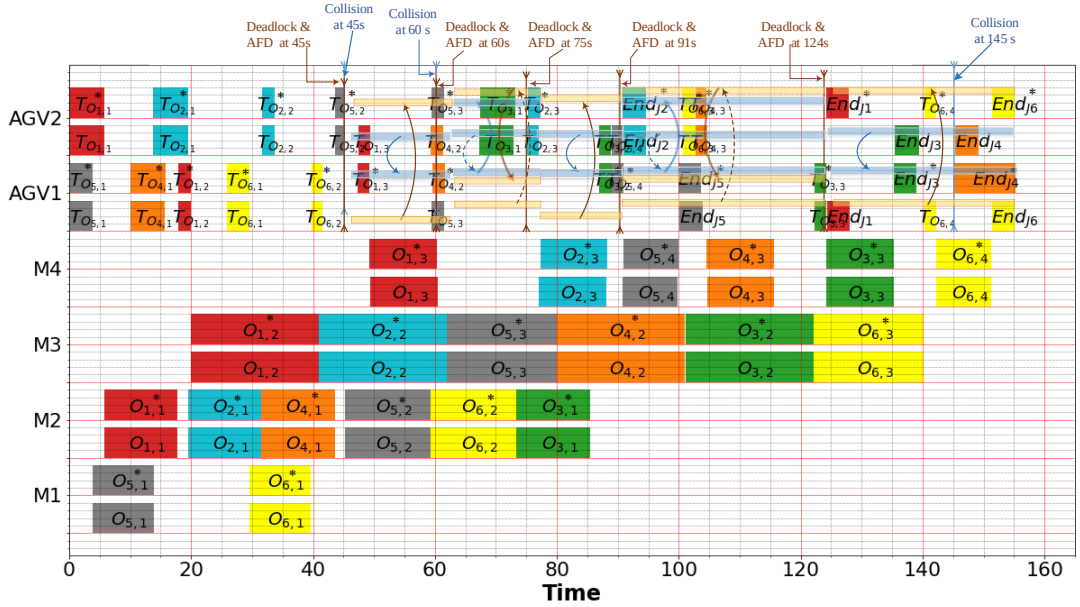


Figure 14: Gantt diagram of EX82 after both types of simulations with collisions and DRA execution

with layout 1, many collisions were recorded because AGVs meet much more frequently at the intersections.

- **Schedules with same Cmax could be different:** This phenomenon, highlighted thanks to simulation, has been observed during the sim-optimization phase. We have indeed found schedules with the same Cmax after optimization but with different behaviors on simulation. For example, the former BFS-L/U of EX62 is 102s, its type 2 simulation presents two collisions and one deadlock and provides a Cmax of 108.5s. However, after sim-optimization, the new schedule gives a Cmax of 102s with only one collision and zero deadlocks for a Cmax of optimisation of 102. Figure 15 and Figure 16 show the before and after sim-optimisation schedules with their type 1 and Type 2 simulations. This phenomenon shows simulation relevance in exhibiting theoretical schedules' behaviors before their real potential implementation.

- **Best optimized schedules are not always the most suitable:** This phenomenon is highlighted by Type 3 simulation column in Table 3. Actually, it can happen that schedules with degraded Cmax give much better results in type 2 simulation than the BFS-L/U. For instance, in EX12, the BFS-L/U is 92s and its type 2 simulation reports 102.8s. However, after the sim-optimization process, the VNS algorithm gave a scheduling with 99s as Cmax, which is worse than the BFS-L/U, but whose type 2 simulation result is also 99s, so better than 102.8s. Several instances such as EX12, EX94, EX61, and EX71 are also affected by this phenomenon.

Furthermore, it can be noted that the algorithm for fixing deadlocks (Algorithm 3) avoids getting stuck in deadlocks. Its efficiency is proven given that all the deadlocks identified during the simulations have been overcome. However, as it is based on the local perception of AGVs, this algorithm can generate other deadlocks in the future, as is the case of EX82.

In addition, the makespan increase observed with type 2 simulations is the consequence of real operational constraints' consideration, particularly collision avoidance between AGVs. This makes the results more realistic and questions the optimized solutions proposed in the literature. Indeed, many of the proposed schedules would not be feasible as it stands without the deadlocks fixing algorithm. Hence, these literature results will be hardly exploitable by companies if they are not close to the reality in industries. The process of sim-optimization is a safe way to explore the space of optimized solutions and test their feasibility. Through these experiments, it was found that an instance can admit several equivalent solutions (i.e. same Cmax) but whose results in the simulation are different, so a priori,

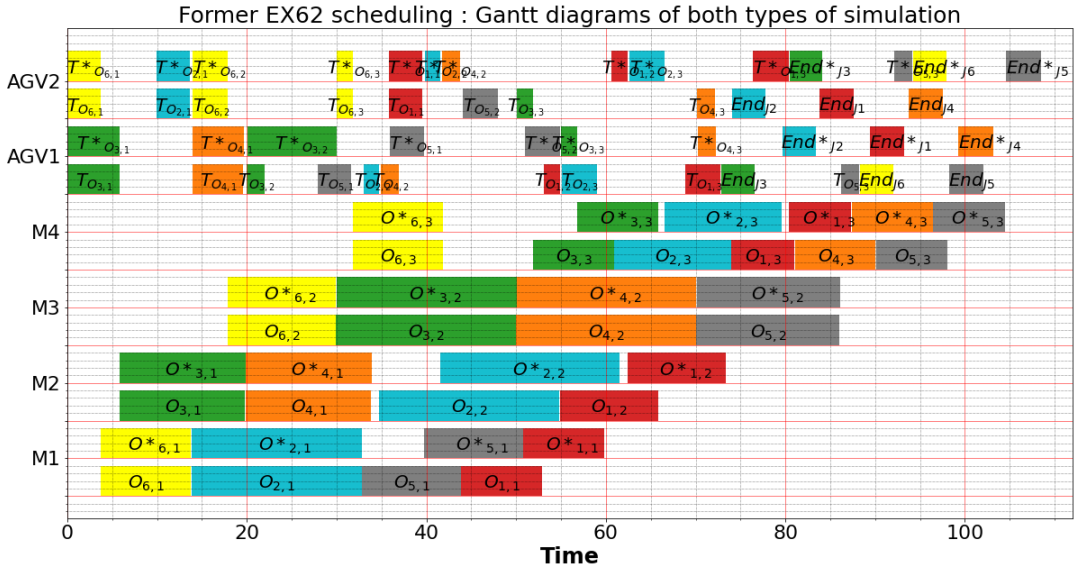


Figure 15: Former EX62 scheduling combined Gantt diagrams

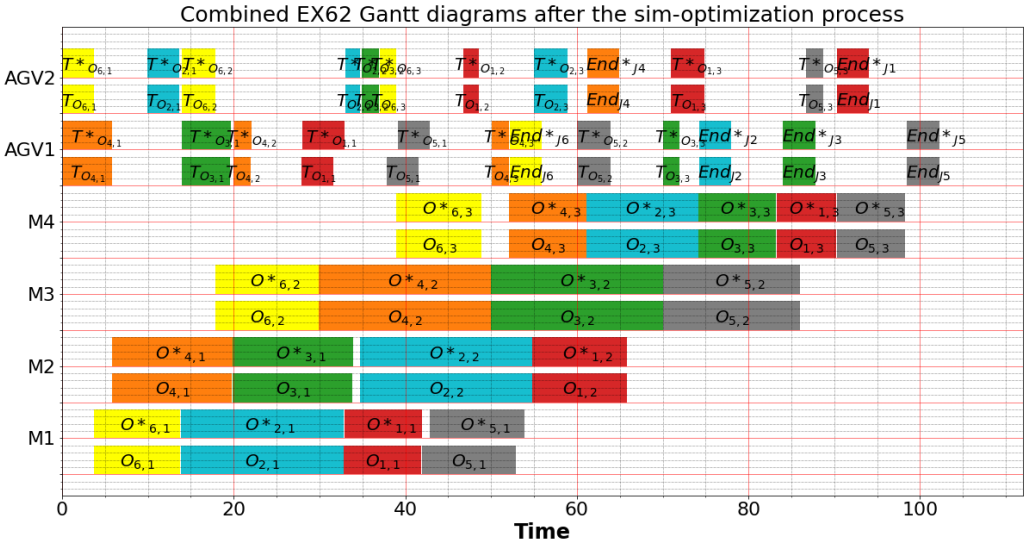


Figure 16: Combined EX62 Gantt diagrams after the sim-optimization process

they will be different in the real implementation. Moreover, notice that loading and unloading times of jobs [35] are not considered in this study to be able to compare with the literature results. The consideration of these aspects will also confirm our hypothesis, which states that realistic considerations degrade theoretical solutions, but reduce the gap between theoretical studies and real production systems. The obtained results and the proposed methodology can play an important role in resolving other problems faced by companies, such as the definition of the AGVs fleet size, AGVs' battery charging management, and FJSSP with transportation tasks.

		EX11	EX21	EX31	EX41	EX51	EX61	EX71	EX81	EX91	EX101
BFS-L/U		120	122	132	147	116	138	159	167	130	169
0% of RSD ($\lambda = 0$)	Type 1 simulation	120	122	132.1	147	116	138	159	167	130	169
	Type 2 simulation	120.1	125.7	132.2	147.0	116.1	143.2	162.6	171.4	130.1	173.4
	#Collisions	0	3	0	3	1	2	1	3	2	4
	#Deadlocks	0	0	0	0	0	0	0	0	0	0
10% of RSD ($\lambda = 0.1$)	Type 1 simulation (avg, sd)	(120.6, 1.2)	(122.7, 0.8)	(132.9, 1.3)	(147.1, 0.9)	(116.0, 0.1)	(139.9, 1.7)	(159.3, 0.8)	(169.3, 2.8)	(131.4, 1.2)	(171.2, 3.4)
	Type 2 simulation (avg, sd)	(120.3, 0.8)	(125.9, 1.5)	(133.3, 1.7)	(147.2, 0.9)	(116.2, 0.2)	(144.9, 1.7)	(162.6, 0.1)	(174.6, 3.1)	(131.9, 1.3)	(174.6, 2.7)
	#Collisions (min, avg, max)	(0, 0.0, 1)	(3, 3.0, 4)	(0, 0.0, 1)	(3, 3, 3)	(1, 1, 1)	(1, 1.9, 2)	(1, 1, 1)	(0, 2.8, 6)	(1, 2.3, 3)	(0, 3.2, 5)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
20% of RSD ($\lambda = 0.2$)	Type 1 simulation (avg, sd)	(121.2, 2)	(124.0, 2.2)	(134.5, 2.8)	(148.3, 1.92)	(115.9, 1.4)	(142.8, 3.9)	(159.7, 1.5)	(173.0, 5.8)	(132.7, 2.8)	(173.6, 5.6)
	Type 2 simulation (avg, sd)	(122.2, 2.6)	(126.0, 2.0)	(135.1, 2.9)	(148.2, 1.7)	(116.1, 1.2)	(147.6, 3.4)	(162.7, 0.4)	(177.9, 7.0)	(131.8, 2.5)	(176.9, 4.8)
	#Collisions (min, avg, max)	(0, 0.0, 1)	(3, 3.0, 4)	(0, 0.1, 1)	(3, 3.9, 5)	(0, 0.9, 1)	(0, 2, 4)	(1, 1, 1)	(1, 2.9, 6)	(0, 1.9, 2)	(0, 2.7, 5)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
30% of RSD ($\lambda = 0.3$)	Type 1 simulation (avg, sd)	(123, 3.7)	(125.5, 3.3)	(135.7, 3.7)	(149.9, 2.7)	(116.2, 1.5)	(147.1, 6.4)	(160.4, 2.1)	(175.5, 8.8)	(132.6, 3.6)	(177, 7.7)
	Type 2 simulation (avg, sd)	(124.4, 3.6)	(127.6, 3.6)	(136.3, 4.0)	(149, 2.4)	(116.2, 1.8)	(149.7, 5.0)	(163, 0.9)	(180.3, 9.9)	(132.9, 3.0)	(182.08, 11.53)
	#Collisions (min, avg, max)	(0, 0.1, 1)	(3, 0.5, 4)	(0, 0.7, 2)	(3, 4, 5)	(0, 1, 1)	(0, 1.8, 3)	(0, 1, 1)	(1, 3.1, 7)	(0, 2.1, 3)	(0, 1.5, 5)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Table 4: Sensitivity analysis of layout 1

5.4. Results of Sensitivity Analysis

The simulation results of the sensitivity analysis carried out on layout 1 of benchmark instances are presented in Table 4. The other results of layouts 2, 3, and 4 are presented in Appendix B.

Table 4 presents the obtained results in 4 main blocks of RSD value. Each block has 4 rows, which are: Type 1 simulation, Type 2 simulation, #Collisions, and #Deadlocks. For each type of simulation, the average (avg) and the standard deviation (sd) of the results of 50 runs are shown. Likewise, the minimum (min), the average (avg), and the maximum (max) number of collisions and deadlocks are shown. Moreover, the " 0% of RSD" block represents the results of simulations with deterministic processing times. The "BFS-L/U" row reports the best optimization results. The results of layouts 2, 3, and 4 are reported similarly in Appendix B.

The results show that:

- the more processing times are disturbed, the more the resulting makespan is also disturbed. Indeed, when the standard deviation of processing times increases, the standard deviation of the makespan also increases, with a slight increase in the average makespan.
- including uncertainties in the processing times leads to variability in the number of collisions. Whether a job processing is longer or shorter, this could lead to collision situations that didn't appear in the deterministic case. For example, both EX11 (see Table 4) and EX23 (see Appendix B2) instances have zero collision in the deterministic case. But uncertainties in processing times make their collisions number go up to 1 for EX11, and to 3 for EX23.
- the number of deadlocks can also vary due to uncertainties. In EX83, for example, the number of deadlocks increases to 6 whereas it was 2 in the deterministic case.
- layouts play an important role in the avoidance of deadlock situations, both in deterministic and probabilistic cases. Indeed, layouts 1 and 4 don't present any deadlocks and remain flexible even though the processing time is variable. Moreover, Algorithm 2 also helps because the path reservation avoids having two robots face-to-face on the same path section.

Overall, these results show the importance of Algorithm 1 and Algorithm 3, respectively, for collision avoidance and deadlock resolution. Indeed, even advanced schedules without any collisions or deadlocks can be disturbed due to uncertainties. Therefore, these schedules could be dead on arrival if there are no solutions to deal with the collisions and deadlocks that occur due to uncertainties. Furthermore, the proposed algorithms have proved their efficiency in a more realistic simulation environment. Even though we notice a makespan increase during type 2 simulations compared to type 1 simulations or optimization results, they ensure a reliable workflow. It should be underlined that these algorithms are robust and work well whatever the fleet size and the environmental behavior (deterministic or not).

6. Conclusion

The JSSP with transportation tasks has been widely addressed in the literature. However, almost all of this work ignores operational, temporal, and spatial, constraints such as collision avoidance between Autonomous Guided Vehicles (AGVs). Thus, in this paper, a multi-agent system-based simulation is proposed to evaluate the efficiency of a known-optimized result of the literature with the consideration of collision avoidance. The contributions of this paper are as follows: (1) verify the feasibility by simulation of the theoretical schedules proposed in the literature with realistic constraints' consideration such as collisions between AGVs and uncertainties in processing times, (2) highlight and avoid collisions between AGVs, (3) propose a heuristic to solve deadlocks, and (4) test the sensitivity of the proposed approach according to processing times variation.

To achieve these objectives, the simulations have been carried out on the well-known benchmark instances proposed by Bilge and Ulusoy [7] using the results extracted from the VNS algorithm proposed by Abderrahim *et al.*[1]. Each schedule has been simulated twice: "Type 1 Simulation" to validate the theoretical scheduling generated by the VNS algorithm and "Type 2 Simulation" to introduce the collision avoidance behavior. Regarding the latter, some avoided collisions led to deadlocks. Thus, a heuristic has been proposed to solve these problems. In addition, a

"Type 3 simulation" based on a sim-optimization approach has been presented to find more suitable schedules with collision avoidance consideration.

Experiments have shown that, in general, the makespan increases when collision avoidance is considered. They have shown as well that solving one deadlock can lead to new ones, and AGVs delaying cannot impact the makespan if the job processing time is longer than the AGV travel time. We observed also that the layout plays an important role in the behavior of AGV, where layouts 1 and 4 provide more flexibility and avoid situations leading to deadlocks. The sim-optimization step has revealed that the optimal solution obtained through optimization alone is not always the best one to implement. Moreover, a scheduling problem can admit several equivalent optimized solutions that can have different outcomes after real implementation. Furthermore, results obtained under uncertainties in processing times indicate that collisions can result from delays/advances in processing times, and increasing processing times variability increases makespan variability.

These observations bring us closer to the reality in industries and prove that the consideration of real operational constraints will degrade theoretical results. Finally, the efficiency of the proposed algorithms is proved as all identified collisions have been avoided, and all deadlocks have been fixed.

From a short-term perspective, future work will be carried out assuming that transporters move freely in the workshop rather than on dedicated paths. It will consider more intelligent robots called AIVs (Autonomous Intelligent Vehicles), which are able to make more complex decisions such as optimal path selection, dynamic obstacle avoidance, and collaborative tasks. It will also take into account issues such as energy consumption, charging of transporter batteries, unforeseen obstacle avoidance, etc. These issues will be addressed in a multi-objective optimization approach. From a long and medium-term perspective, we will investigate the presence of human operators in the workshop to include the human aspect in the context of Industry 5.0. A stochastic model of the human operators in the workshop depending on the period of the day (morning, peak period, evening) will be established. This model will be integrated into a job-shop scheduling optimization algorithm with transportation tasks in order to propose optimized scheduling taking into account human activities. A sim-optimization method can be used to define a dynamic scheduling strategy in an uncertain environment.

References

[1] Abderrahim, M., Bekrar, A., Trentesaux, D., et al., 2020. Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints. Optimization Letters .

[2] Ahmadian, M.M., Salehipour, A., Cheng, T., 2021. A meta-heuristic to solve the just-in-time job-shop scheduling problem. European Journal of Operational Research 288, 14–29. doi:https://doi.org/10.1016/j.ejor.2020.04.017.

[3] Allal, A., Sahnoun, M., Adjoudj, R., Benslimane, S., Mazar, M., 2021. Multi-agent based simulation-optimization of maintenance routing in offshore wind farms. Computers & Industrial Engineering 157, 107342. URL: https://www.sciencedirect.com/science/article/pii/S0360835221002461, doi:https://doi.org/10.1016/j.cie.2021.107342.

[4] Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J., 2014. Simulation optimization: a review of algorithms and applications. 4OR-Q J Oper Res 12, 301–333. doi:https://doi.org/10.1007/s10288-014-0275-2.

[5] Barton, R.R., 2009. Simulation optimization using metamodels, in: Proceedings of the 2009 Winter Simulation Conference, WSC 2009, pp. 230–238. doi:10.1109/WSC.2009.5429328. 2009 Winter Simulation Conference, WSC 2009 ; Conference date: 13-12-2009 Through 16-12-2009.

[6] Benhafssa, A.M., Sahnoun, M., Bettayeb, B., Bekrar, A., 2021. Optimizing energy-conscious dynamic flexible job shop scheduling: Multi-agent simulation approach, in: 2021 1st International Conference On Cyber Management And Engineering (CyMaEn), pp. 1–6. doi:10.1109/CyMaEn50288.2021.9497301.

[7] Bilge, Ü., Ulusoy, G., 1995. A time window approach to simultaneous scheduling of machines and material handling system in an fms. Operations Research 43, 1058–1070.

[8] Chen, M., Lu, Y., Zhang, C., 2021. Deadlock-solving traffic control methods for automated guided vehicle systems, in: 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 51–57. doi:10.1109/IEEM50564.2021.9673048.

[9] Chen, R., Yang, B., Li, S., Wang, S., 2020. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Computers & Industrial Engineering 149, 106778. doi:https://doi.org/10.1016/j.cie.2020.106778.

[10] Coffman, E.G., Elphick, M., Shoshani, A., 1971. System deadlocks. ACM Computing Surveys (CSUR) 3, 67–78.

[11] De Ryck, M., Pissoort, D., Holvoet, T., Demeester, E., 2022. Decentral task allocation for industrial agv-systems with routing constraints. Journal of Manufacturing Systems 62, 135–144. doi:https://doi.org/10.1016/j.jmsy.2021.11.012.

[12] De Ryck, M., Versteyhe, M., Debrouwere, F., 2020a. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. Journal of Manufacturing Systems 54, 152–173. doi:https://doi.org/10.1016/j.jmsy.2019.12.002.

[13] De Ryck, M., Versteyhe, M., Shariatmadar, K., 2020b. Resource management in decentralized industrial automated guided vehicle systems. Journal of Manufacturing Systems 54, 204–214. doi:https://doi.org/10.1016/j.jmsy.2019.11.003.

[14] Drótos, M., Györgyi, P., Horváth, M., Kis, T., 2021. Suboptimal and conflict-free control of a fleet of agvs to serve online requests. Computers & Industrial Engineering 152, 106999.

[15] Elena, C., Valerio, D., Lorenzo, S., Cristian, S., Fantuzzi, C., 2017. Cooperative cloud robotics architecture for the coordination of multi-agv systems in industrial warehouses. Mechatronics 45. doi:http://dx.doi.org/10.1016/j.mechatronics.2017.04.005.
[16] Erik, A., Kuvvetli, Y., 2021. Integration of material handling devices assignment and facility layout problems. Journal of Manufacturing Systems 58, 59–74.
[17] Fontes, D.B.M., Homayouni, S.M., 2019. Joint production and transportation scheduling in flexible manufacturing systems. Journal of Global Optimization 74, 879–908.
[18] Ham, A., 2021. Transfer-robot task scheduling in job shop. International Journal of Production Research 59, 813–823.
[19] Jianfeng, R., Chunming, Y., Yan, L., 2020. A two-stage optimization algorithm for multi-objective job-shop scheduling problem considering job transport. Journal Européen des Systèmes Automatisés 53, 915–924. doi:https://doi.org/10.18280/jesa.530617.
[20] Klanke, C., Engell, S., 2022. Scheduling and batching with evolutionary algorithms in simulation–optimization of an industrial formulation plant. Computers & Industrial Engineering 174, 108760. URL: https://www.sciencedirect.com/science/article/pii/S0360835222007483, doi:https://doi.org/10.1016/j.cie.2022.108760.
[21] Kunchev, V., Jain, L., Ivancevic, V., Finn, A., 2006. Path planning and obstacle avoidance for autonomous mobile robots: A review, in: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Springer. pp. 537–544.
[22] Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & operations research 24, 1097–1100.
[23] Mohammadi, E.K., Shirazi, B., 2020. Toward high degree flexible routing in collision-free fmss through automated guided vehicles' dynamic strategy: A simulation metamodel. ISA Transactions 96, 228–244. URL: https://www.sciencedirect.com/science/article/pii/S001905781930285X, doi:https://doi.org/10.1016/j.isatra.2019.06.024.
[24] Mohan, J., Lanka, K., Rao, A.N., 2019. A review of dynamic job shop scheduling techniques. Procedia Manufacturing 30, 34–39. doi:https://doi.org/10.1016/j.promfg.2019.02.006. digital Manufacturing Transforming Industry Towards Sustainable Growth.
[25] Mokhtari, H., Hasani, A., 2017. A multi-objective model for cleaner production-transportation planning in manufacturing plants via fuzzy goal programming. Journal of Manufacturing Systems 44, 230–242.
[26] Moorthy, R., Hock-Guan, W., 2000. Deadlock prediction and avoidance in an agv system. Master of science, Sri Ramakrishna Engineering College, National University of Singapore .
[27] Moussa G, S., Bettayeb, B., Sahnoun, M., Duval, F., Bensrhair, A., 2021. Modular mobile manipulators coalition formation through distributed transportation tasks allocation. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 235, 2262–2272.
[28] Murakami, K., 2020. Time-space network model and milp formulation of the conflict-free routing problem of a capacitated agv system. Computers & Industrial Engineering 141, 106270. doi:https://doi.org/10.1016/j.cie.2020.106270.
[29] Parham, A., 2011. Alleviating the collision states and fleet optimization by introducing a new generation of automated guided vehicle systems. Modelling and Simulation in Engineering 2011, 8. doi:10.1155/2011/210628.
[30] Perez-Grau, F.J., Martinez-de Dios, J.R., Paneque, J.L., Acevedo, J.J., Torres-González, A., Viguria, A., Astorga, J.R., Ollero, A., 2021. Introducing autonomous aerial robots in industrial manufacturing. Journal of Manufacturing Systems 60, 312–324.
[31] Piana, S., Engell, S., 2010. Hybrid evolutionary optimization of the operation of pipeless plants. Journal of Heuristics 16, 311–336. doi:https://doi.org/10.1007/s10732-009-9105-7.
[32] Reith, K.B., Rank, S., Schmidt, T., 2021. Conflict-minimal routing for free-ranging transportation vehicles in in-house logistics based on an a-priori lane design. Journal of Manufacturing Systems 61, 97–111. doi:https://doi.org/10.1016/j.jmsy.2021.07.019.
[33] Saad, L., Oulaid, K., Adil, I., 2018. Scheduling for job shop problems with transportation and blocking no-wait constraints. Journal of Theoretical and Applied Information Technology 96.
[34] Sahnoun, M., Xu, Y., Abdelaziz, F.B., Baudry, D., 2019. Optimization of transportation collaborative robots fleet size in flexible manufacturing systems, in: 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), IEEE. pp. 1–5.
[35] Sanogo, K., Mekhalef Benhafssa, A., Sahnoun, M., Bettayeb, B., Bekrar, A., 2022. Multi-agent simulation for flexible job-shop scheduling problem with traffic-aware routing, in: Borangiu, T., Trentesaux, D., Leitão, P., Cardin, O., Joblot, L. (Eds.), Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future, Springer International Publishing, Cham. pp. 573–583.
[36] Sharad, C., Srivastava, A., Kumar, C., Surendra, K., Tiwari, M.K., 2008. Development of an intelligent agent-based agv controller for a flexible manufacturing system. Int J Adv Manuf Technol 36, 780–797. doi:10.1007/s00170-006-0892-9.
[37] Silitonga, P.D., Morina, I.S., Damanik, R., 2021. Application of simulation of dining philosophers problem in concurrent process. SMARTICS Journal 7, 1–6.
[38] Soares do Amaral, J.V., Montevechi, J.A.B., de Carvalho Miranda, R., de Sousa Junior, W.T., 2022. Metamodel-based simulation optimization: A systematic literature review. Simulation Modelling Practice and Theory 114, 102403. URL: https://www.sciencedirect.com/science/article/pii/S1569190X21001040, doi:https://doi.org/10.1016/j.simpat.2021.102403.
[39] XIAOHUI, L., XI, Y., YI, Z., YING, T., DONG, Y., 2020. Metaheuristic for solving multi-objective job shop scheduling problem in a robotic cell. IEEE Access 8. doi:10.1109/ACCESS.2020.3015796.
[40] Xu, Y., Sahnoun, M., Abdelaziz, F.B., Baudry, D., 2020. A simulated multi-objective model for flexible job shop transportation scheduling. Annals of Operations Research , 1–22.
[41] Zaidi, L., Bettayeb, B., Sahnoun, M., 2021. Optimisation and simulation of transportation tasks in flexible job shop with muti-robot systems, in: 2021 1st International Conference On Cyber Management And Engineering (CyMaEn), IEEE. pp. 1–6.
[42] Zajac, J., 2004. A deadlock handling method for automated manufacturing systems. CIRP Annals 53, 367–370. URL: https://www.sciencedirect.com/science/article/pii/S0007850607607185, doi:https://doi.org/10.1016/S0007-8506(07)60718-5.
[43] Zajac, J., Malopolski, W., 2021. Structural on-line control policy for collision and deadlock resolution in multi-agv systems. Journal of Manufacturing Systems 60, 80–92. doi:https://doi.org/10.1016/j.jmsy.2021.05.002.
[44] Zambrano Rey, G., Bonte, T., Prabhu, V., Trentesaux, D., 2014. Reducing myopic behavior in fms control: A semi-heterarchical simulation–optimization approach. Simulation Modelling Practice and Theory 46, 53–75. URL: https://www.sciencedirect.com/science/article/pii/S1569190X14000069, doi:https://doi.org/10.1016/j.simpat.2014.01.005.

simulation-Optimization of Complex Systems: Methods and Applications.
[45] Zhang, J., Ding, G., Zou, Y., *et al.*, 2019. Review of job shop scheduling research and its new perspectives under industry 4.0. J Intell Manuf 30. doi:https://doi.org/10.1007/s10845-017-1350-2.
[46] Zhong, M., Yang, Y., Dessouky, Y., Postolache, O., 2020. Multi-agv scheduling for conflict-free path planning in automated container terminals. Computers & Industrial Engineering 142, 106371.
[47] Zhou, Y., Hu, H., Liu, Y., Ding, Z., 2017. Collision and deadlock avoidance in multirobot systems: A distributed approach. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 1712–1726. doi:10.1109/TSMC.2017.2670643.

Appendix A. Job sets proposed by Bilge and Ulusoy

Job set 1
Job 1: M1(8); M2(16); M4(12)
Job 2: M1(20); M3(10); M2(18)
Job 3: M3(12); M4(8); M1(15)
Job 4: M4(14); M2(18)
Job 5: M3(10); M1(15)

Job set 2
Job 1: M1(10); M4(18)
Job 2: M2(10); M4(18)
Job 3: M1(10); M3(20)
Job 4: M2(10); M3(15); M4(12)
Job 5: MI(10); M2(15); M4(12)
Job 6: M1(10); M2(15); M3(12)

Job set 3
Job 1: M1(16); M3(15)
Job 2: M2(18); M4(15)
Job 3: M1(20); M2(10)
Job 4: M3(15); M4(10)
Job 5: M1(8); M2(10); M3(15); M4(17)
Job 6: M2(10); M3(15); M4(8); M1(15)

Job set 4
Job 1: M4(11); M1(10); M2(7)
Job 2: M3(12); M2(10); M4(8)
Job 3: M2(7); M3(10); M1(9); M3(8)
Job 4: M2(7); M4(8); M1(12); M2(6)
Job 5: M1(9); M2(7); M4(8); M2(10); M3(8)

Job set 5
Job 1: M1(6); M2(12); M4(9)
Job 2: M1(18); M3(6); M2(15)
Job 3: M3(9); M4(3); M1(12)
Job 4: M4(6); M2(15)
Job 5: M3(3); M1(9)

Job set 6
Job 1: M1(9); M2(11); M4(7)
Job 2: M1(19); M2(20); M4(13)
Job 3: M2(14); M3(20); M4(9)
Job 4: M2(14); M3(20); M4(9)
Job 5: M1(11); M3(16); M4(8)
Job 6: M1(10); M3(12); M4(10)

Job set 7
Job 1: M1(6); M4(6)
Job 2: M2(11); M4(9)
Job 3: M2(9); M4(7)
Job 4: M3(16); M4(7)
Job 5: M1(9); M3(18)
Job 6: M2(13); M3(19); M4(6)
Job 7: M1(10); M2(9); M3(13)
Job 8: M1(11); M2(9); M4(8)

Job set 8
Job 1: M2(12); M3(21); M4(11)
Job 2: M2(12); M3(21); M4(11)
Job 3: M2(12); M3(21); M4(11)
Job 4: M2(12); M3(21); M4(11)
Job 5: M1(10); M2(14); M3(18); M4(9)
Job 6: MI(10); M2(14); M3(18); M4(9)

Job set 9
Job 1: M3(9); M1(12); M2(9); M4(6)
Job 2: M3(16); M2(11); M4(9)
Job 3: M1(21); M2(18); M4(7)
Job 4: M2(20); M3(22); M4(11)
Job 5: M3(14); M1(16); M2(13); M4(9)

Job set 10
Job 1: M1(11); M3(19); M2(16); M4(13)
Job 2: M2(21); M3(16); M4(14)
Job 3: M3(8); M2(10); M1(14); M4(9)
Job 4: M2(13); M3(20); M4(10)
Job 5: M1(9); M3(16); M4(18)
Job 6: M2(19); M1(21); M3(11); M4(15)

Appendix B. Sensitivity analysis

		EX12	EX22	EX32	EX42	EX52	EX62	EX72	EX82	EX92	EX102
	BFS-L/U	92.0	82.0	95.0	109.0	84.0	102.0	101.0	155.0	106.0	142.0
0% of RSD ($\lambda = 0$)	Type 1 simulation	92.0	82.0	95.0	109.1	84.0	102.2	101.0	155.2	106.0	142.0
	Type 2 simulation	102.8	82.2	95.0	116.8	85.1	108.5	101.0	155.3	106.1	156.2
	#Collisions	1	0	1	1	1	2	0	3	1	3
	#Deadlocks	0	0	0	1	0	1	0	5	0	2
10% of RSD ($\lambda = 0.1$)	Type 1 simulation (avg, sd)	(96, 3.5)	(81.8, 1.3)	(96.3, 3.1)	(109.7, 1)	(84.4, 1.1)	(105.3, 2.7)	(100.5, 0.8)	(158, 5.3)	(108.5, 3.4)	(144.6, 3)
	Type 2 simulation (avg, sd)	(102.9, 2)	(82.7, 1.7)	(95.2, 2.6)	(117, 1.3)	(85.3, 0.5)	109.7, 1.6)	(101.1, 0.6)	(157.9, 4.8)	(108, 2.4)	(156.1, 5.7)
	#Collisions (min, avg, max)	(0, 0.4, 3)	(0, 0.0, 1)	(0, 0.7, 1)	(1, 1, 1)	(2, 2.2, 4)	(2, 3.1, 6)	(0, 0, 0)	(1, 2.4, 5)	(0, 0.5, 2)	(1, 2.5, 7)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(5, 5, 5)	(0, 0.1, 1)	(2, 2, 2)
20% of RSD ($\lambda = 0.2$)	Type 1 simulation (avg, sd)	(97.4, 3.1)	(83.5, 1.9)	(99.1, 5.2)	(110.8, 2)	(85.8, 2.2)	(110.5, 6)	(101.7, 1.6)	(159.7, 8.6)	(110.2, 5.1)	(149.2, 5.1)
	Type 2 simulation (avg, sd)	(103.6, 3.3)	(84.2, 2.6)	(98.8, 4.8)	(116.7, 2.4)	(86.7, 1.9)	(113.7, 4.5)	(101.5, 1.3)	(159.8, 9.2)	(111.4, 4.4)	(159.3, 9.5)
	#Collisions (min, avg, max)	(0, 0.4, 3)	(0, 0.2, 1)	(0, 0.4, 1)	(1, 1.1, 3)	(0, 2.5, 4)	(0, 4.1, 8)	(0, 0.0, 1)	(1, 3.3, 7)	(0, 0.7, 3)	(1, 3.2, 6)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(5, 5, 5)	(0, 0.1, 1)	(2, 2, 2)
30% of RSD ($\lambda = 0.3$)	Type 1 simulation (avg, sd)	(99.6, 3.7)	(84.9, 3.5)	(101, 5.3)	(111.5, 3.7)	(88.2, 4.2)	(113.9, 8)	(102.7, 2.5)	(167.2, 10.5)	(114.4, 7.2)	(155.4, 10)
	Type 2 simulation (avg, sd)	(107.5, 5.7)	(84.4, 3.7)	(102.3, 6.8)	(118.2, 3.9)	(88.3, 3.5)	(117.2, 6.1)	(103.7, 3.1)	(161.5, 10.9)	(114.2, 6.2)	(163.4, 11.1)
	#Collisions (min, avg, max)	(0, 0.8, 3)	(0, 0.4, 2)	(0, 0.3, 3)	(1, 1.3, 3)	(0, 2.4, 4)	(0, 4.8, 8)	(0, 0.4, 2)	(1, 4.1, 8)	(0, 1.3, 5)	(1, 3.8, 8)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(5, 5, 5)	(0, 0.1, 1)	(2, 2, 2)

Table Appendix B1: Sensitivity analysis of layout 2

		EX13	EX23	EX33	EX43	EX53	EX63	EX73	EX83	EX93	EX103
	BFS-L/U	99	92	101	112	93	110	112	155	108	148
0% of RSD ($\lambda = 0$)	Type 1 simulation	99	92	101	112	93	110.05	112	155	108	148
	Type 2 simulation	99.4	92	107	112	96.05	117.75	114.75	161.9	116.65	181.15
	#Collisions	1	0	1	0	2	2	2	2	2	4
	#Deadlocks	0	0	1	0	0	2	1	2	0	1
10% of RSD ($\lambda = 0.1$)	Type 1 simulation (avg, sd)	(99.7, 1.7)	(93.8, 1.7)	(103.1, 2.4)	(112.4, 1.4)	(94, 1.5)	(112.6, 2.6)	(112.4, 0.7)	(155.9, 4.8)	(113.2, 4.3)	(150.8, 3.5)
	Type 2 simulation (avg, sd)	(100.3, 1.7)	(93.9, 2.3)	(107.6, 2.6)	(112.4, 1.7)	(96.1, 1.2)	(119.3, 3)	(114.9, 1.5)	(166.4, 9.6)	(117.8, 2.4)	(174, 11.1)
	#Collisions (min, avg, max)	(0, 1, 3)	(0, 0.7, 2)	(1, 1.4, 5)	(0, 0.2, 1)	(2, 2, 2)	(3, 3.4, 4)	(1, 1.6, 2)	(1, 3, 5)	(1, 2.4, 4)	(2, 3.5, 5)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(0, 0, 0)	(1, 2, 2)	(1, 1, 1)	(2, 5.8, 6)	(0, 0, 0)	(1, 1.6, 3)
20% of RSD ($\lambda = 0.2$)	Type 1 simulation (avg, sd)	(102.2, 3.3)	(96.9, 4.1)	(107.2, 4.2)	(113.4, 3.1)	(94.9, 2.7)	(114.9, 4.5)	(113, 1.8)	(159.4, 8.5)	(116.4, 4.7)	(153.8, 6)
	Type 2 simulation (avg, sd)	(102.4, 2.9)	(96, 3.4)	(110.6, 4.6)	(113.5, 2.5)	(97.4, 2.3)	(121, 4.7)	(115.5, 2.4)	(169.8, 13.5)	(120, 5.3)	(172.2, 12.7)
	#Collisions (min, avg, max)	(0, 1.1, 3)	(0, 1, 3)	(1, 1.8, 4)	(0, 0.1, 1)	(2, 2, 2)	(1, 3.4, 6)	(1, 1.5, 3)	(3, 4.3, 6)	(1, 2.1, 4)	(2, 3.6, 6)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(1, 1, 1)	(0, 0, 0)	(0, 0, 0)	(0, 1.7, 2)	(1, 1, 1)	(2, 5.3, 6)	(0, 0, 0)	(1, 1.8, 3)
30% of RSD ($\lambda = 0.3$)	Type 1 simulation (avg, sd)	(105.1, 5.5)	(100.6, 6.9)	(110.5, 6.1)	(116.6, 6.4)	(96.1, 4.6)	(121.2, 6.5)	(115.6, 3.9)	(163.9, 10.8)	(119, 6.1)	(158.2, 9.2)
	Type 2 simulation (avg, sd)	(105, 4.7)	(98.9, 5.3)	(113.6, 6.3)	(115.6, 3.9)	(97.7, 4)	(123.5, 7.1)	(117.1, 3.7)	(172.3, 11.5)	(122.8, 6.6)	(174.8, 11.1)
	#Collisions (min, avg, max)	(0, 1.1, 3)	(0, 1.4, 3)	(1, 2.2, 5)	(0, 0.7, 3)	(1, 2, 3)	(1, 3.4, 6)	(1, 1.6, 3)	(3, 4.1, 5)	(0, 2, 5)	(2, 3.5, 7)
	#Deadlocks (min, avg, max)	(0, 0, 0)	(0, 0, 0)	(1, 1.1, 3)	(0, 0, 0)	(0, 0, 0)	(0, 1.4, 2)	(1, 1, 1)	(2, 4.7, 6)	(0, 0, 0)	(1, 1.6, 3)

Table Appendix B2: Sensitivity analysis of layout 3

		EX14	EX24	EX34	EX44	EX54	EX64	EX74	EX84	EX94	EX104
	BFS-L/U	144.0	156.0	160.0	180.0	140.0	168.0	191.0	190.0	156.0	202
0% of RSD ($\lambda = 0$)	Type 1 simulation	144.0	156.0	160.0	180.0	140.0	168.0	191.0	190.0	156.0	202.0
	Type 2 simulation	144.0	156.2	162.4	180.0	140.2	168.2	191.0	190.3	160.2	203.3
	#Collisions	0	0	1	0	0	0	0	1	1	0
	#Deadlocks	0	0	0	0	0	0	0	0	0	0
10% of RSD ($\lambda = 0.1$)	Type 1 simulation (avg, sd)	(145, 1.7)	(156, 1.9)	(160.3, 1.0)	(180.7, 1.2)	(140, 0)	(169, 2.2)	(191.6, 0.8)	(191.5, 3.1)	(157.2, 1.8)	(201.9, 0.8)
	Type 2 simulation (avg, sd)	(144.9, 1.4)	(156.6, 2.3)	(162.8, 0.5)	(180.6, 0.8)	(140.2, 0)	(168.1, 1.7)	(191.3, 0.6)	(191.7, 2.4)	(160.3, 2.5)	(204.8, 2.1)
	#Collisions (min, avg, max)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.9, 1)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.1, 1)	(0, 0.0, 1)	(0, 0.7, 2)	(0, 1.0, 2)	(0, 0.0, 0)
	#Deadlocks (min, avg, max)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)
20% of RSD ($\lambda = 0.2$)	Type 1 simulation (avg, sd)	(146.4, 2.7)	(156.4, 4.2)	(161.7, 2.3)	(181.8, 1.9)	(140.3, 0.7)	(170.8, 4)	(191.7, 1.3)	(193, 3.6)	(159.5, 2.9)	(203.3, 2.2)
	Type 2 simulation (avg, sd)	(146.3, 3.1)	(156.1, 4.2)	(164.5, 2.9)	(181.6, 1.6)	(140.2, 0)	(169.2, 3.2)	(191.7, 1.3)	(193.6, 3.6)	(162.9, 4.8)	(205.1, 3.4)
	#Collisions (min, avg, max)	(0, 0.0, 0)	(0, 0.0, 0)	(1, 1.4, 3)	(0, 0.1, 1)	(0, 0.1, 1)	(0, 0.4, 2)	(0, 0.0, 1)	(0, 0.9, 2)	(0, 1.0, 2)	(0, 0.3, 1)
	#Deadlocks (min, avg, max)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)
30% of RSD ($\lambda = 0.3$)	Type 1 simulation (avg, sd)	(148.2, 4.6)	(158.9, 5.8)	(162.3, 4.5)	(183.1, 3)	(141.2, 1.9)	(173.4, 5.2)	(192.8, 2)	(196.3, 7.4)	(163.8, 6)	(207.5, 6.8)
	Type 2 simulation (avg, sd)	(147.5, 4.6)	(158.1, 5.8)	(167.3, 6.5)	(182.7, 2.6)	(140.6, 1.2)	(169.7, 3.6)	(193.3, 2.3)	(195.6, 5.4)	(165, 5.1)	(211.5, 7.3)
	#Collisions (min, avg, max)	(0, 0.0, 1)	(0, 0.0, 1)	(1, 1.5, 3)	(0, 0.1, 1)	(0, 0.2, 1)	(0, 0.5, 3)	(0, 0.2, 1)	(0, 1.0, 3)	(0, 1.2, 4)	(0, 0.6, 2)
	#Deadlocks (min, avg, max)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)	(0, 0.0, 0)

Table Appendix B3: Sensitivity analysis of layout 4