European Journal of Operational Research 306 (2023) 1140–1157

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources

Dalila B. M. M. Fontes[a,b,*], S. Mahdi Homayouni[a], José F. Gonçalves[b]

[a] LIAAD, INESC TEC, Porto, Portugal
[b] Faculdade de Economia, Universidade do Porto, Porto, Portugal

ARTICLE INFO

Article history:
Received 6 July 2021
Accepted 7 September 2022
Available online 11 September 2022

Keywords:
Scheduling
Job shop scheduling with transport resources
Joint production and transport scheduling
Particle swarm optimization
Simulated annealing

ABSTRACT

This work addresses a variant of the job shop scheduling problem in which jobs need to be transported to the machines processing their operations by a limited number of vehicles. Given that vehicles must deliver the jobs to the machines for processing and that machines need to finish processing the jobs before they can be transported, machine scheduling and vehicle scheduling are intertwined. A coordinated approach that solves these interrelated problems simultaneously improves the overall performance of the manufacturing system. In the current competitive business environment, and integrated approach is imperative as it boosts cost savings and on-time deliveries. Hence, the job shop scheduling problem with transport resources (JSPT) requires scheduling production operations and transport tasks simultaneously. The JSPT is studied considering the minimization of two alternative performance metrics, namely: makespan and exit time. Optimal solutions are found by a mixed integer linear programming (MILP) model. However, since integrated production and transportation scheduling is very complex, the MILP model can only handle small-sized problem instances. To find good quality solutions in reasonable computation times, we propose a hybrid particle swarm optimization and simulated annealing algorithm (PSOSA). Furthermore, we derive a fast lower bounding procedure that can be used to evaluate the performance of the heuristic solutions for larger instances. Extensive computational experiments are conducted on 73 benchmark instances, for each of the two performance metrics, to assess the efficacy and efficiency of the proposed PSOSA algorithm. These experiments show that the PSOSA outperforms state-of-the-art solution approaches and is very robust.

1. Introduction

The job shop manufacturing system under consideration consist of a set of machines on which a set of jobs has to be processed and a set of vehicles that transport the jobs around the shop floor. This type of system can be found in modern smart manufacturing and warehousing systems in many industrial applications such as automotive (in chassis lines, engine and gear box assembly lines, final assembly and trim manufacturing processes, just-in-time parts, and pre-production staging), electronics, food and beverage, paper, healthcare, distribution and retail, among others (e.g. JBT, 2022b; MiR Solutions, 2022b; Transbotics - A Scott Company, 2022a).

Automated guided vehicles (AGVs) are often employed in a variety of industries to move (i) raw materials from a receiving dock to a warehouse or even directly to the production line; (ii) work-in-process (jobs) from one workstation to another or from a load/unload area to the workstations; (iii) finished products from a workstation to a load/unload area or to storage; or (iv) picking and handling (both inbound and outbound) for replenishment (AGV network, 2022; Fragapane, De Koster, Sgarbossa, & Strandhagen, 2021). There are several automation solution providers and all describe case studies. For example, a major manufacturer of Building Materials located in the US uses AGVs to move wallboards in and out of a curing room. Once the cement boards are produced and palletized, they are transported to the curing room for drying. Then, the pallets of dried cement boards need to be removed from the curing room and transferred to the Packaging/Wrapping Conveyor (MHI, 2022). Other examples are a gas-detection equipment

* Corresponding author.
E-mail address: fontes@fep.up.pt (D.B.M.M. Fontes).

https://doi.org/10.1016/j.ejor.2022.09.006

production plant in the UK, in which AGVs are used to transport raw materials and assemblies around the 91,000 m^2 plant; a Korean electronic component manufacturer that uses AGVs to transport materials and finished products at the production floor; a FORD plant that uses AGVs to deliver fresh industrial and welding materials to the different robot stations of the Body & Stamping plant, among others (see, e.g., JBT, 2022a; MiR Solutions, 2022a; Transbotics - A Scott Company, 2022b).

In such manufacturing systems, jobs are picked up at the load/unload area (LU) to be delivered to the first machine processing them. Jobs are then transported between the machines they need to be processed on until all operations have been executed and finally transported back to the LU. Therefore, a solution to this problem involves the scheduling of the (production) operations as well as the assignment and scheduling of the (transport) tasks. Although these problems are closely interrelated, most of the time, the production scheduling is solved first without considering transportation aspects. Then, the transport tasks are sequentially allocated to vehicles. The lack of coordination can lead to sub-optimal solutions with respect to intermediate stocks, operational costs, resource utilization, or customer service level and satisfaction. The benefits of integrating the production and transport scheduling regarding job completion time are clearly shown in Section 5.4 with an average improvement between 17 and 94%.

The job shop scheduling problem with transport resources (JSPT) involves determining the machine scheduling, the transport allocation, and the vehicle scheduling. Although solving these interrelated problems simultaneously improves the overall performance of the manufacturing system, their integration makes the scheduling problem much more challenging due to the addition of extra decisions and constraints to the problem. As a result, developing proper scheduling methods to obtain satisfactory results is crucial.

The JSPT is NP-hard as it extends two NP-hard problems: the job-shop scheduling problem (JSP) (Lenstra & Kan, 1979) and the vehicle scheduling problem, which is similar to a pickup and delivery problem (Lenstra & Kan, 1981). As is the case for the JSP, most research on the JSPT minimizes the makespan, that is, the maximum completion time over all jobs. However, in the literature, there are a few works that minimize the exit time, that is, the time required to process and return all jobs to the LU (initially proposed by Deroussi, Gourgand, & Tchernev, 2008). Although the two metrics are closely linked, as both aim at minimizing the completion time of the set of jobs, the exit time is more realistic and accurate since only after delivering the jobs to the LU they are ready to be sent to the final customer.

Since the JSPT is NP-hard, exact approaches can only solve small-sized instances. Thus, we propose a hybrid particle swarm optimization and simulated annealing algorithm (PSOSA), which can find optimal or near-optimal solutions, within reasonable computation times. To assess the solutions of the PSOSA for small-sized instances, we use the solutions obtained by solving the MILP model proposed in (Fontes & Homayouni, 2019) when minimizing the C_{max} and the ones obtained by solving a modified version of that model when minimizing the ET. Further, we also devise a very fast lower bounding procedure, which we use to assess the PSOSA solutions for medium and large-sized instances. Finally, extensive computational experiments are conducted to show the efficacy and efficiency of the proposed approaches and to provide some insights to the managers dealing with such a complex problem. We consider three sets of benchmark instances with a total of 73 instances.

The main contributions of this work are twofold: First, we propose a hybrid particle swarm optimization and simulated annealing algorithm (PSOSA) capable of finding good quality solutions and outperforming state-of-the-art algorithms. Second, we provide lower bounds on the solutions produced by the PSOSA through a very fast lower bounding procedure. Additionally, we adapt the MILP formulation by Fontes & Homayouni (2019) so that the exit time is minimized.

The remainder of this paper is organized as follows. Section 2 presents a brief literature review on the most recent works on the JSPT. Section 3 provides a detailed description of the problem, of the mathematical model that is used to obtain optimal solutions, and of the novel lower bounding procedure. In Section 4, we propose the PSOSA algorithm and, in Section 5, we report on the computational experiments. Finally, in Section 6, we draw some conclusions and point out future research directions.

2. Literature review

Several researchers have demonstrated the benefits of coordinating production (machine) scheduling with transport (vehicles) scheduling. From the literature, it is clear that neglecting transport tasks/times as well as not integrating their schedules can have severe consequences. It may lead to bottlenecks, additional intermediate inventories, lower resource (machines and vehicles) utilization, larger operational costs, and deterioration of customer service and satisfaction.

Bilge & Ulusoy (1995) seem to have been the first to address the JSPT. They formulated the problem as a mixed integer programming (MIP) model that minimizes the C_{max}; however, the model was not used to solve any problem instance. Additionally, they proposed a sliding time window (STW) heuristic and reported computational experiments involving 82 problem instances that they also proposed. These instances, due to being commonly used, have become defacto benchmark instances and are referred to, in this paper, as BU instances. Over the years several heuristic approaches have been proposed such as genetic algorithms (GAs), simulated annealing (SA), tabu search (TS), among others.

Abdelmaguid, Nassef, Kamal, & Hassan (2004) proposed a two steps hybrid GA (HGA): in the first step, an operations-based encoding GA determines the sequence of the operations on each machine and, in the second step, a greedy heuristic assigns to each task the vehicle that can accomplish it sooner. The HGA solutions improve on previously known ones for several of the BU instances, considering the C_{max} as the objective function. A similar approach was proposed by Reddy & Rao (2006); however, the heuristic was embedded into the GA. Hence, the problems are addressed simultaneously rather than sequentially. The computational results have shown its performance to be superior to that of Abdelmaguid et al. (2004).

Hurink & Knust (2002, 2005) tackled a slightly different version of the problem as they assume that there is only one vehicle and that the jobs are only moved between machines, since they are initially located at the machine processing their first operation. As there is only one vehicle no transport task assignment is required. The latter assumption implies that jobs do not need to be transported from the LU to the machine processing their first operation, which leads to a smaller problem instance as there is one less transport task for each job; the one taking the job from the LU to the machine processing its first operation. Additionally, such consideration implies a smaller optimal C_{max}. They proposed a TS algorithm to find heuristic solutions, which are evaluated through a lower bounding procedure they also proposed. The TS was implemented with three neighborhood structures originating three algorithm versions (one-stage, two-stages, and a combination of both). Neighbor solutions are generated by using a move operator that moves an operation/task to another position in the corresponding schedule. At each iteration, the one-stage TS moves either an operation or a task, but not both. The two-stages TS first moves an operation (outer stage) and then moves a task (inner stage).

Finally, the combined version runs the one-stage TS first and then the two-stages TS. Computational experiments were carried out on the instances they proposed, referred to, in this paper, as HK instances.

Deroussi et al. (2008) were the first to propose an alternative metric for the completion time of the set of jobs - the exit time. The exit time (ET) is the time required to process all operations of all jobs and to transport the jobs back to the LU, which implies an additional transport task for each job from the machine processing its last operation to the LU. Thus, minimizing the exit time, rather than the makespan, leads to a larger instance to be solved. Additionally, such consideration implies a larger optimal $C_{\max}$. They proposed a hybrid simulated annealing/local search (SALS) algorithm that uses a tasks-based encoding. The SA is used to guide the search and the local search to obtain a neighbor solution by using three consecutive random moves. A move can be either an exchange, in which two tasks are swapped, or an insert, in which a task is removed from its location and relocated elsewhere. Additionally, the moves can be intra or inter vehicles. The sequence of operations on each machine is given by the order in which they arrive at the machine, following the FIFO rule. Computational experiments were conducted using the BU instances, for both objective functions ($C_{\max}$ and ET), and the results obtained were in line with the best-known at the time.

Zhang, Manier, & Manier (2012) also proposed a hybrid GA considering the minimization of the $C_{\max}$ as the objective. The GA is used to assign operations to machines and tasks to vehicles and the TS is used to find a sequence of operations and tasks for each machine and vehicle, respectively. The solutions for the BU instances were compared with those of Abdelmaguid et al. (2004) and those of Deroussi et al. (2008) and found to be worse. Zhang et al. (2012) also solved the HK instances, but the results obtained were inferior to those of Hurink & Knust (2005).

Lacomme, Larabi, & Tchernev (2013) modeled the problem by resorting to disjunctive graphs and found solutions by using a memetic algorithm (MA) that embeds a local search procedure into a GA, as the mutation operator. The problem was solved considering both performance metrics, $C_{\max}$ and ET. The memetic algorithm was tested on the BU instances as well as on the HK instances and the solutions obtained are among the best-known ones.

Zheng, Xiao, & Seo (2014) proposed a TS algorithm that encodes solutions using a two-dimensional array: one representing the sequence of operations and the other the assignment of the vehicles. Neighbor solutions are found by alternating between changing the assignment of the vehicles while keeping the operations sequence fixed and changing the operations sequence while keeping the assignment of the vehicles fixed. The solutions they found are among the best-known solutions for the BU instances considering the $C_{\max}$.

Timed colored Petri net (TCPN) models were proposed for the JSPT by Baruwa & Piera (2016), which were then solved using an anytime layered search (ALS) algorithm that combines breadth-first iterative deepening with sub-optimal breadth-first branch-and-bound and backtracking. The solutions found for the BU instances, considering both the $C_{\max}$ and the ET, match those of Zheng et al. (2014), except for one problem instance.

Most recently, Ham (2020) proposed two constraint programming (CP) models. In the first model, job pickup and job drop-off (by a vehicle) are considered as two distinct tasks; while in the second model they are merged into a single task. Computational experiments were conducted on some of the BU instances, considering both metrics ($C_{\max}$ and ET), and on another set of larger instances derived from the ones proposed by Storer, Wu, & Vaccari (1992), considering ET only. The latter instances are referred to, in this paper, as SWV instances. The computational results reported show that the second model outperforms the first one. Moreover, the results reported match the best-known results for the BU instances solved, except for one instance when considering the ET.

A few works proposing mathematical programming models have also been reported. El Khayat, Langevin, & Riopel (2006) formulated the JSPT as a mixed integer linear programming (MILP) model and as a CP model; however, at the beginning of the planning horizon, all jobs are located at the machine processing their first operation rather than at the LU. As was the case for (Hurink & Knust, 2005), such an assumption amounts to have one less transport task for each job. The results reported show that the MILP and CP models can solve the BU instances efficiently. A similar MILP model was proposed by Zheng et al. (2014), also not taking into account the transport required to make jobs available at the machines processing their first operation. Additionally, they impose constraints that force the completion of the operations to occur only when jobs can be transported to the machine processing the subsequent operation, rather than allowing the model to decide on the precise timing. These additional constraints shorten the search space; however, they may prevent finding an optimal solution. The MILP model was used to solve only one small size test case. Lastly, Fontes & Homayouni (2019) proposed a MILP model based on the novel idea of using two sets of chained decisions, one for production scheduling and another for transportation scheduling. The decision chains are interconnected through the completion time constraints, both for operations and tasks. The model also contemplates the travel time required to make the job available at the machine performing its first operation. The MILP model was solved to optimality for all but two BU instances.

A summary of the works reviewed is provided in Table 1.

3. Problem definition and formulation

This section provides a detailed description of the problem being studied, of the proposed MILP model, and of the lower bounding procedure.

The main components of the job shop scheduling problem under consideration are the production system and the transport system. The production system is a traditional job shop scheduling problem; thus, it includes a set of independent jobs and a set of machines. Each job consists of a set of ordered operations and each operation must be processed uninterrupted on a dedicated machine during a predefined processing time. Hence, each job has its own independent machine order. Additionally, each machine can only process one operation at a time and each job can only be processed on one machine at a time.

The transport system comprises several identical vehicles that move at the same constant speed and can carry one job at a time. Thus, any vehicle can perform any transport task. Since the layout is known and the transport non-preemptive, travel times between any two locations are predetermined. Regarding the transport tasks, three different problem versions have been addressed before. In the “simplest” one, no LU is considered and thus, jobs are only transported between machines. The jobs are initially by the machine processing their first operation and considered completed whenever their last operation is processed. Hence, a job, say j, consisting of n_j (production) operations requires $n_j - 1$ (transport) tasks. The HK instances (Hurink & Knust, 2005) fall in this category. Additionally, the HK instances consider one vehicle only, which removes the need for vehicle assignment.

The other two problem versions also consider a load/unload area, from where the jobs enter the production system. Thus, the vehicles also transport the jobs from the LU to the machine processing their first operation. However, they differ in what regards job completion. If the performance metric is the makespan, then jobs are completed whenever the processing of their last operation is finished. If the performance metric is the exit time, then the

Table 1
Summary of the main characteristics of the works review.

Publication	Formulation	Heuristic	Instances & Objective(s)		
		Approach	BU	HK	SWV
Bilge & Ulusoy (1995)	MIP	STW	C_{max}		
Ulusoy, Sivrikaya, & Bilge (1997)	LB method	GA	C_{max}		
Abdelmaguid et al. (2004)	-	HGA	C_{max}		
Hurink & Knust (2005)	Disjunctive graph	TS		C_{max}	
El Khayat et al. (2006)	MILP & CP	-	C_{max}		
Reddy & Rao (2006)	-	moHGA	C_{max}		
Deroussi et al. (2008)	-	SALS	C_{max}&ET		
Zhang et al. (2012)	Disjunctive graph	GATS	C_{max}	C_{max}	
Lacomme et al. (2013)	Disjunctive graph	MA	C_{max}&ET	C_{max}	
Zheng et al. (2014)	MILP	TS	C_{max}		
Baruwa & Piera (2016)	TCPN	ALS	C_{max}&ET		
Fontes & Homayouni (2019)	MILP	-	C_{max}		
Ham (2020)	CP	-	C_{max}		C_{max}
This work	MILP & LB method	PSOSA	C_{max}&ET	C_{max}	C_{max}&ET

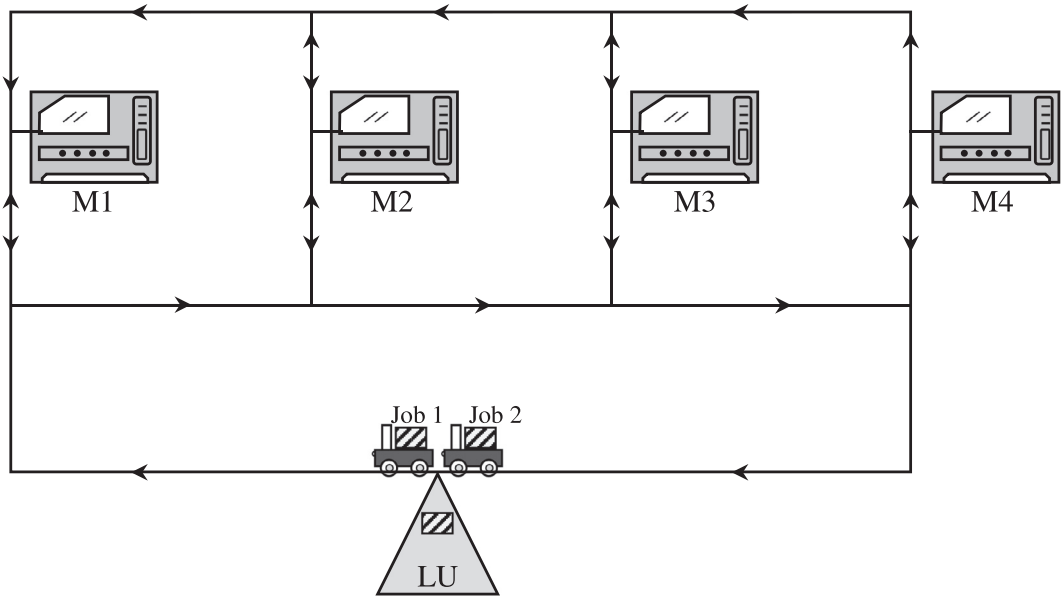

Fig. 1. Illustrative example: shop floor configuration based on layout 1 in (Bilge & Ulusoy, 1995).

jobs also have to leave the production system, that is, they must be transported back to the LU. Hence, job j, which consists of n_j (production) operations, requires n_j (transport) tasks if the makespan is being optimized and $n_j + 1$ if the exit time is being optimized. Both the BU instances and the SWV instances consider an LU and that all vehicles are initially parked there. An example showing the solution for each of the three problem versions is given below but first let us provide an illustrative example of a shop floor under consideration.

As shown in Fig. 1 all jobs and all vehicles are initially located at the LU. Then, each job is transported by a vehicle from the LU (where it enters the production system) to the machine processing its first operation. If the machine is idle, then processing starts immediately; otherwise, the job waits in the machine input buffer. In either case, as soon as the vehicle reaches the machine it drops off the job and can pursue its next assignment. Once an operation of a job is completed (say operation i of job j), a vehicle needs to pick up job j from the machine processing operation i and transport it to the machine processing operation $i+1$. Whenever an operation is completed the job either goes to the machine output buffer, where it waits for the arrival of a vehicle, or straight onto the vehicle if one is already available. Either way, the machine can start processing the next operation. Vehicles have to wait to pick up a job if they arrive at the machine processing the operation before its completion. Note that a transport requires a pickup and a drop-off. While the latter always involves traveling, the former only requires vehicle traveling (empty) if the vehicle is not at the pickup location. After all operations of a job have been processed, the job is returned to the LU (from where it leaves the production system).

Let us consider a problem instance with three jobs, each of which with three production operations, four machines (layout 1 in Bilge & Ulusoy, 1995), and two vehicles. This example will also be used to illustrate the lower bounding procedure, the encoding and decoding procedures, as well as the swarm transition procedure. The instance data are provided in Tables 2 and 3.

A solution to the JSPT defines a sequence of operations for each machine and a sequence of tasks for each vehicle. In Fig. 2, we show the Gantt charts of the optimal solutions when (a) no LU is considered and when an LU is considered and the performance metric is the minimization of the (b) makespan and of the (c) exit time.

Therefore, following the classification $\alpha|\beta|\gamma$ by Knust (2000), this problem can be classified as a $JA|\tau_{kl}, \tau'_{kl}|ET$, that is, a job-shop scheduling problem with a limited number (A) of identical vehicles, job-independent but machine-dependent travel times, and loaded and empty travel times, in which ET is minimized.

Table 2
Illustrative example: instance data.

Jobs	Job 1			Job 2			Job 3		
Operations	O_{11}	O_{21}	O_{31}	O_{12}	O_{22}	O_{32}	O_{13}	O_{23}	O_{33}
Machine	M1	M3	M2	M3	M1	M4	M2	M4	M1
Processing time	10	16	18	15	22	18	18	16	22

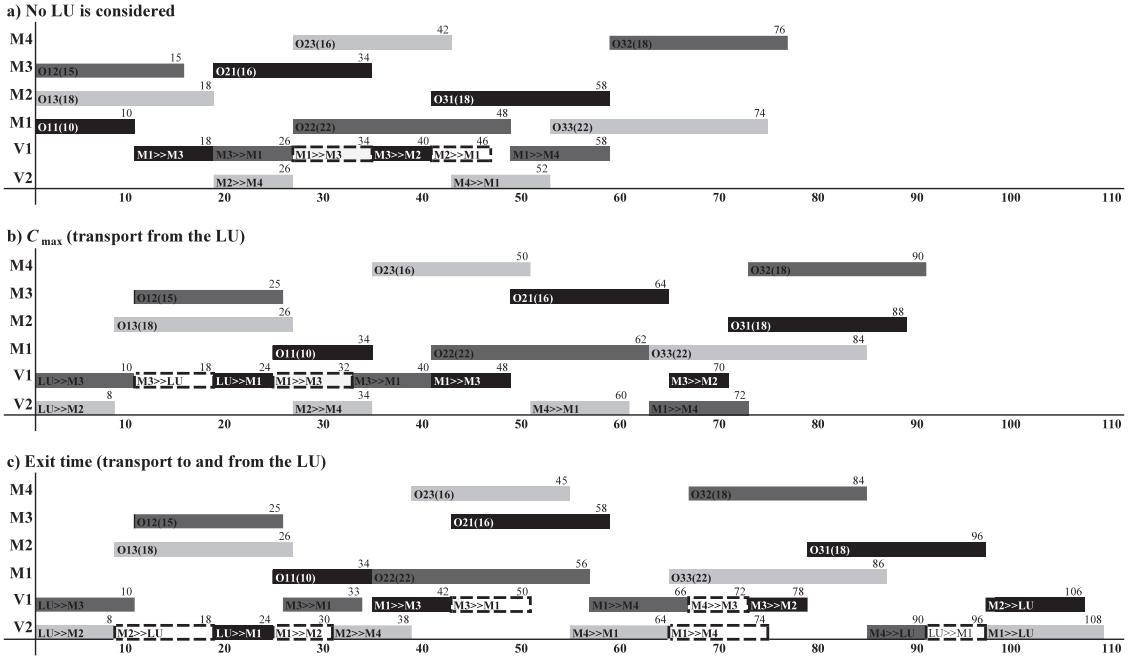

Fig. 2. Illustrative example: Gantt charts for the three problem versions. The production time is, respectively, 76, 90, and 108 time units. (Dashed line bars indicate the empty travels.).

Table 3
Illustrative example: transport times, layout 1 in (Bilge & Ulusoy, 1995).

	LU	M1	M2	M3	M4
LU	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

3.1. The MILP model

In this section, we present a modified version of the MILP model proposed in (Fontes & Homayouni, 2019), which (i) minimizes the exit time *ET* and (ii) takes into account the return of jobs to the LU. The processing of an operation on a given machine requires the transportation of the job to the corresponding machine. Hence, each transport task, except for the ones required to return the jobs to LU, is associated with a production operation. Note that this additional transport task is only required when the *ET* is being minimized as each job needs to be delivered to the LU, after all operations have been completed. Therefore, for each job we define a dummy operation that is to be processed at the LU with a 0 processing time. This way, we ensure that the job is returned to the LU as to process the last (dummy) operation the job needs to be transported to the corresponding machine, the LU by definition. Furthermore, the completion time of the last (dummy) operation of each job provides the job exit time (the time at which the job is returned to the LU). The notation used and the model proposed are as follows:

Sets and indices:

M: Set of machines, indexed by m;
J: Set of jobs, indexed by j and l;
f, t: First and last dummy jobs, each composed of as many operations as the number of machines that are, respectively, the first and last operations of the machine that processes them;
J_j: Set of n_j operations of job $j \in J \cup \{f, t\}$, indexed by i and k;
$n_j + 1$: Dummy last operation of job $j \in J$ processed at the LU with 0 processing time.
J'_j: Set of operations of job $j \in J$ including the dummy operation; i.e., $J'_j = J_j \cup \{n_j + 1\}$.

Parameters:

O_{ij}: Operation $i \in J'_j$ of job $j \in J$;
M_{ij}: Machine that processes operation $O_{ij}, i \in J'_j, j \in J$, where $M_{(n_j+1)j}$ is, by definition, the LU;
T_{ij}: Transport task to deliver job j to machine $M_{ij}, i \in J'_j, j \in J$;
P_{ij}: Processing time of operation $O_{ij}, i \in J'_j, j \in J$ with $P_{(n_j+1)j} = 0$;
τ_{ij}^{kl}: Travel time from machine M_{ij} to machine $M_{kl}, i \in J'_j,\ k \in J'_l,\ j, l \in J$;
A: Number of available vehicles;
LN: A sufficiently large positive integer.

Decision Variables:

w_{ij}^{kl}: Binary variable taking the value 1 if operation O_{kl} is processed immediately after operation O_{ij} on the same

machine and 0 otherwise, $k \in J_l, l \in J \cup \{t\}, i \in J_j, j \in J \cup \{f\}$;

x_{ij}^{kl}: Binary variable taking the value 1 if transport task T_{kl} is done immediately after transport task T_{ij} by the same vehicle and 0 otherwise, $k \in J'_l, l \in J, T_{ij}, i \in J'_j, j \in J$;

u_{ij}: Binary variable taking the value 1 if transport task T_{ij} is the first task of a vehicle and 0 otherwise, $i \in J'_j, j \in J$;

z_{ij}: Binary variable taking the value 1 if the transport task T_{ij} is the last task of a vehicle and 0 otherwise, $i \in J'_j, j \in J$.

Auxiliary Variables:

ET: Exit time of the last job;

c_{ij}: Completion time of operation $O_{ij}, i \in J'_j, j \in J$;

v_{ij}: Vehicle arrival time at machine $M_{ij}, i \in J'_j, j \in J$.

$$\text{Minimize} \quad ET \tag{1}$$

[0.1cm]Subject to:

$$ET \geq c_{(n_l+1)l}, \qquad \forall l \in J, \tag{2}$$

$$\sum_{i<k\in J_l} w_{il}^{kl} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j} w_{ij}^{kl} + \sum_{i\in J_f} w_{if}^{kl} = 1, \quad \forall l \in J \cup \{t\}, k \in J_l, \tag{3}$$

$$\sum_{i>k\in J_l} w_{kl}^{il} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J_j} w_{kl}^{ij} + \sum_{i\in J_t} w_{kl}^{it} = 1, \quad \forall l \in J \cup \{f\}, k \in J_l, \tag{4}$$

$$\sum_{j\in J} \sum_{i\in J'_j} z_{ij} = \sum_{j\in J} \sum_{i\in J'_j} u_{ij}, \tag{5}$$

$$\sum_{j\in J} \sum_{i\in J'_j} u_{ij} \leq A, \tag{6}$$

$$u_{kl} + \sum_{i<k\in J'_l} x_{il}^{kl} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J'_j} x_{ij}^{kl} = 1, \qquad \forall l \in J, k \in J'_l, \tag{7}$$

$$z_{kl} + \sum_{i>k\in J'_l,} x_{kl}^{il} + \sum_{j\in J\setminus\{l\}} \sum_{i\in J'_j} x_{kl}^{ij} = 1, \qquad \forall l \in J, k \in J'_l, \tag{8}$$

$$c_{kl} - v_{kl} - P_{kl} \geq 0, \qquad \forall l \in J, k \in J'_l, \tag{9}$$

$$c_{kl} - c_{ij} - P_{kl} \geq LN\left(w_{ij}^{kl} - 1\right), \quad \forall j,\ l \in J, i \in J_j, k \in J_l (if\ \ j = l : k > i), \tag{10}$$

$$v_{kl} - c_{(k-1)l} - \tau_{(k-1)l}^{kl} \geq 0, \qquad \forall l \in J, k \in J'_l \setminus \{1\}, \tag{11}$$

$$v_{1l} - \tau_{LU}^{1l} \geq 0, \qquad \forall l \in J, \tag{12}$$

$$v_{kl} - v_{ij} - \tau_{ij}^{(k-1)l} - \tau_{(k-1)l}^{ij} \geq LN\left(x_{ij}^{kl} - 1\right), \quad \forall j, l \in J, i \in J'_j, k \in J'_l \setminus \{1\}, (if\ \ j = l : k > i), \tag{13}$$

$$v_{1l} - v_{ij} - \tau_{ij}^{LU} - \tau_{LU}^{1l} \geq LN\left(x_{ij}^{1l} - 1\right), \qquad \forall j, l \in J :\ \ j \neq l, i \in J'_j, \tag{14}$$

$$w_{ij}^{kl}, x_{ij}^{kl} \in \{0, 1\}, \qquad \forall j, l \in J \cup \{f, t\}, i \in J'_j, k \in J'_l, \tag{15}$$

$$ET, c_{kl}, v_{kl} \geq 0, \qquad \forall l \in J, k \in J'_l, \tag{16}$$

$$0 \leq u_{kl}, z_{kl} \leq 1, \qquad \forall l \in J, k \in J'_l. \tag{17}$$

The objective is to minimize the exit time over all jobs, as in expression (1), and its value is determined as the largest completion time of the last (dummy) operation over all jobs, as in inequalities (2). Constraints (3) and (4) impose each operation to be immediately followed and immediately preceded by exactly one other operation on the same machine. Constraints (5) ensure the same number of first and last tasks for the vehicles while constraints (6) ensure it to be at most the number of available vehicles. Constraints (7) and (8) impose each task to be immediately followed and immediately preceded by exactly one other task, respectively. Since for each vehicle a set of chained tasks is determined through constraints (5) to (8), vehicles are handled implicitly. Thus, avoiding an additional index.

The completion time of each operation must satisfy two constraints. On the one hand, an operation can only be finished after the job arrives at the machine processing it and its processing time has elapsed, ensured by constraints (9). On the other hand, the completion of an operation requires the completion of the previous operation on the same machine in addition to its own processing time, ensured by constraints (10). Similarly, a job can only arrive at a machine to have a specific operation processed after its previous operation has been completed ($c_{(k-1)l}$) and the job has been transported from the machine where the previous operation was processed ($\tau_{(k-1)l}^{kl}$) as imposed by constraints (11); unless it is a job first operation ($k = 1$), in which case the job can arrive as soon as it has been transported from the LU (τ_{LU}^{1l}) as imposed by constraints (12). Additionally, if a vehicle has transported some other job ($j \in J : j \neq l$) immediately prior to the current one, then it needs to (i) deliver such job to the corresponding machine (v_{ij}); (ii) pick up the current job from where the previous operation was processed ($\tau_{ij}^{(k-1)l}$) or the LU if it is the first operation (τ_{ij}^{LU}), and (iii) deliver it to the corresponding machine ($\tau_{(k-1)l}^{kl}$ or τ_{LU}^{kl}), enforced by constraints (13) and (14). Finally, constraints (15) to (17) define the nature of the variables.

3.2. Lower bounding procedure

Based on some preliminary experiments, it was observed that the MILP model can only be solved to optimality for small-sized problem instances. For the medium-sized instances the MILP can provide feasible solutions and lower bounds. However, both are worse than the ones found by the PSOSA and the lower bounding procedure, respectively. Furthermore, for large-sized instances it cannot even find any feasible solution. Hence, a lower bound can be useful to assess the efficacy of the PSOSA in tackling large-sized instances. For this purpose, we propose a lower bounding procedure that considers a problem version in which enough resources (machines and vehicles) are available. Such an assumption implies that jobs never wait (to be processed or transported).

The problem being addressed has three elements, namely: jobs, machines, and vehicles. Therefore, a lower bound on the exit time must be at least as large as i) the largest exit time among all jobs ($LETJ$), ii) the largest completion time among all machines plus the corresponding travel time between the machine and the LU ($LCTMTT$), and iii) the minimum average vehicle travel time ($MATT$). Therefore, a lower bound on the exit time LB is obtained

as given by Eq. (18).

$$LB = \max\{LETJ, LCTMTT, MATT\}. \tag{18}$$

The exit time of a job ($ETJ_j, j \in J$) is defined to be at least the sum of (i) the travel time between the LU and the machine processing the job first operation; (ii) the summation of the travel times between the machines processing the remaining job operations; (iii) the travel time between the machine processing the job last operation and the LU, and iv) the summation of the processing time of all operations of the job (see Eq. (19)). The largest exit time among all jobs ($LETJ$) is given by Eq. (20). Recall that we are assuming that there are enough machines and vehicles and thus, jobs never wait.

$$ETJ_j = \tau_{LU}^{1j} + \sum_{i \in J_j \setminus \{1\}} \tau_{(i-1)j}^{ij} + \tau_{n_j j}^{LU} + \sum_{i \in J_j} P_{ij}, \qquad \forall j \in J, \tag{19}$$

$$LETJ = \max_{j \in J}\left\{ETJ_j\right\}. \tag{20}$$

Similarly, the completion time of a machine ($CTM_m, m \in M$) is defined to be the maximum over all operations it processes ($O_{ij} : M_{ij} = m, i \in J_j, j \in J$) of the sum of (i) the operation earliest starting time (EST_{ij}) and (ii) the total processing time required by all other operations processed on the same machine that have an earliest starting time not smaller than that of operation O_{ij}, as given in Eq. (21).

$$CTM_m = \max_{j \in J, i \in J_j : M_{ij} = m} \left\{ EST_{ij} + \sum_{\substack{l \in J \\ M_{kl} = m \\ EST_{ij} \le EST_{kl}}} \sum_{k \in J_l} P_{kl} \right\}, \qquad \forall m \in M. \tag{21}$$

The earliest starting time EST_{ij} of operation $O_{ij}, j \in J, i \in J_j$ is bounded from below by the travel time from the loading area to the machine processing it, if it is a job first operation, or by adding the earliest completion time of the previous operation $ECT_{(i-1)j}$ to the travel time between the machine processing it and that of the previous operation, as given by Eq. (22).

$$EST_{ij} = \begin{cases} \tau_{LU}^{1j}, & \forall j \in J, \\ ECT_{(i-1)j} + \tau_{(i-1)j}^{ij}, & \forall j \in J, i \in J_j \setminus \{1\}, \end{cases} \tag{22}$$

where the earliest completion time ECT_{ij} of operation $O_{ij}, j \in J, i \in J_j$ is given by the sum of its earliest starting time and its processing time, i.e., $ECT_{ij} = EST_{ij} + P_{ij}$. Again, no waiting times are considered as we are assuming that enough resources (machines and vehicle) are available.

Finally, the largest completion time among all machines plus the corresponding travel time between the machine and the LU ($LCTMTT$) is given by Eq. (23).

$$LCTMTT = CTM_{m^*} + \tau_{m^*}^{LU}, \tag{23}$$

where $m^* = \underset{m \in M}{\operatorname{argmax}}\{CTM_m\}$.

The minimum average vehicle travel time is obtained by adding the vehicle average loaded travel to the vehicle average minimum required empty travel. For the problem version considering the LU and using the exit time as the performance metric, empty travel may not be required since the jobs are initially available at the LU and must be returned to the LU. Therefore, in this case, the minimum average vehicle travel time is given by the average vehicle loaded travel and is calculated as in Eq. (24).

$$MATT = \left\lceil \frac{1}{A} \sum_{j \in J} \left(\tau_{LU}^{1j} + \sum_{i \in J_j \setminus \{1\}} \tau_{(i-1)j}^{ij} + \tau_{n_j j}^{LU} \right) \right\rceil. \tag{24}$$

Note that, in these calculations we consider that processing times are negligible since processing may be accomplished while other jobs are being transport; thus, only the transport time is taken into account.

To obtain a lower bound on the makespan $C_{\max}$ we use Eqs. (25)–(27) rather than Eqs. (19), (23), and (24) since upon completion the jobs are not transported back to the LU. Note that jobs are only transported between the LU and the machines processing the jobs first operation; thus, there will be some empty travel to pick up jobs from the LU. Therefore, the minimum average vehicle travel time is obtained by adding the average vehicle loaded travel and the minimum average vehicle empty travel. Furthermore, we can also add the smallest processing time of the last operation among all jobs, since only after processing the last operation the jobs are completed.

The minimum number of jobs requiring empty travel to pick them up from the LU is given by $|J| - A$, that is, the difference between the number of jobs and the number of vehicles. Each of these $|J| - A$ jobs requires an empty travel time at least as large as the smallest travel time between a machine and the LU.

$$ETJ_j = \tau_{LU}^{1j} + \sum_{i \in J_j \setminus \{1\}} \tau_{(i-1)j}^{ij} + \sum_{i \in J_j} P_{ij}, \qquad \forall j \in J, \tag{25}$$

$$LCTMTT = \max_{m \in M}\{CTM_m\}. \tag{26}$$

$$MATT = \left\lceil \frac{1}{A} \sum_{j \in J} \left(\tau_{LU}^{1j} + \sum_{i \in J_j \setminus \{1\}} \tau_{(i-1)j}^{ij} \right) \right\rceil + \left\lceil \frac{|J| - A}{A} \min_{m \in M} \left\{ \tau_m^{LU} \right\} \right\rceil + \min_{j \in J}\{P_{n_j j}\}. \tag{27}$$

The lower bounding procedure is computationally easy and provides good quality bounds. Let us illustrate the calculation of the lower bound on the exit time by considering the example previously described.

The largest job exit time is $LETJ = 94$ since $ETJ_1 = 74$, $ETJ_2 = 89$, and $ETJ_3 = 94$.

The completion times of the machines are $CTM_1 = 82$, $CTM_2 = 64$, $CTM_3 = 41$, and $CTM_4 = 83$ and the earliest starting times of the operations are $EST_{11} = 6$, $EST_{21} = 24$, $EST_{31} = 46$, $EST_{12} = 10$, $EST_{22} = 33$, $EST_{32} = 65$, $EST_{13} = 8$, $EST_{23} = 34$, and $EST_{33} = 60$.

For illustration purposes, below we provide the full calculation of ETJ_1, EST_{11} (which is simply the transport time between the LU and machine $M1$, since the first operation of job 1 is processed on machine $M1$), and CTM_1.

$$\begin{aligned} ETJ_1 &= \tau_{LU}^{11} + \tau_{11}^{12} + \tau_{12}^{13} + \tau_{13}^{LU} + P_{11} + P_{12} + P_{13} \\ &= 6 + (8 + 6) + 10 + (10 + 16 + 18) \\ &= 74. \end{aligned}$$

$$EST_{11} = \tau_{LU}^{11} = 6.$$

$$EST_{12} = ECT_{11} + \tau_{11}^{12} = 16 + 8 \text{ since } ECT_{11} = EST_{11} + P_{11} = 6 + 10 = 16.$$

$$CTM_{M1} = \max_{j \in J, i \in J_j : M_{ij} = M1} \left\{ EST_{ij} + \sum_{\substack{l \in J \\ M_{kl} = M1 \\ EST_{ij} \le EST_{kl}}} \sum_{k \in J_l} P_{kl} \right\}$$

$$= \max\{EST_{11} + P_{11} + P_{22} + P_{33}, EST_{22} + P_{22} + P_{33}, EST_{33} + P_{33}\}$$
$$= \max\{6 + 10 + 22 + 22, 33 + 22 + 22, 60 + 22\} = 82.$$

The largest completion time among all machines plus the corresponding travel time to the LU is $LCTMTT = CTM_4 + \tau^{LU}_{M4} = 83 + 6 = 89$.

The minimum average travel time is given by the average loaded travel time.

$$MATT = \left\lceil \frac{1}{2}\left(\tau^{11}_{LU} + \tau^{12}_{11} + \tau^{13}_{12} + \tau^{LU}_{13} + \cdots + \tau^{33}_{32} + \tau^{LU}_{33}\right) \right\rceil$$
$$= \left\lceil \frac{1}{2}(6 + 8 + 6 + 10 + \cdots + 10 + 12) \right\rceil = 51.$$

Finally, the *LB* on the exit time value is 94, which is the maximum between *LETJ*, *LCTMTT*, and *MATT*.

4. The proposed hybrid algorithm

Particle swarm optimization (PSO) is a population-based stochastic optimization technique inspired by the social group behavior of bird flocking or fish schooling, originally proposed by Eberhart & Kennedy (1995). Given its simple concepts and effectiveness, PSO has become popular and many variants have been developed over the years. Although PSO was originally developed for continuous optimization problems, it has been adapted and widely used to solve combinatorial optimization problems stemming from a variety of application areas, see, e.g., (Chih, 2022; Fathi, Rodríguez, Fontes, & Alvarez, 2016; Tang, Zhao, & Liu, 2014; Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2007; Zhang, Lu, & Yang, 2021).

In PSO, a population of solutions is called a swarm and each solution is called a particle. The basic idea is to move the particles around the search space with the aim of finding good solutions. Each particle represents a point in the *D*-dimensional search space (*D* being the number of decisions) and thus, it is represented by a position vector. A velocity vector is associated with each particle and it defines how the particle moves around the search space. These vectors are updated at every iteration by taking into account three components, namely: inertia, cognitive, and social components. The moves try to balance the tendency of moving towards a previous best position and of moving to the best swarm/neighborhood position. Initially, each particle is randomly located in the search space of the problem.

Since the performance of metaheuristics can be improved by hybridizing them with a local search procedure, in this work, we combine a PSO algorithm with an SA algorithm. This way, taking advantage of the exploration capabilities of the former and of the local search capabilities of the latter. Local search methods, typically, suffer from excessive convergence and quite often get trapped into a local optimum. However, SA is able to obviate this drawback by occasionally accepting non-improving neighbor solutions. Moreover, the SA will not disrupt (significantly) the PSO evolution and learning processes, since the worse a solution is the less likely it is to be accepted. Also, as the algorithm progresses the probability of accepting worse solutions decreases. Lastly, hybrid metaheuristics combining PSO and SA have been shown to perform well for a variety of combinatorial optimization problems, see, e.g., Tang et al. (2019) for the flexible job shop scheduling problem, Javidrad, Nazari, & Javidrad (2018) for the stacking sequence design problem, and Dong, Zhang, & Xiao (2018) for the line balancing problem.

4.1. Framework of the PSOSA algorithm

Algorithm 1 illustrates the proposed hybrid particle swarm optimization and simulated annealing algorithm (PSOSA). The algorithm has an outer loop that handles the swarm of particles and controls the temperature and two inner loops. The first inner loop searches the neighborhood of the incumbent solution while the second one decides on whether to accept or not a worse neighbor solution.

Algorithm 1: The proposed PSOSA algorithm for the JSPT.

Initialize parameters: S^{max}, ω, c_1, c_2, T_I, T_F, α, R;
Generate a random initial *Swarm* (position and velocity vectors for each particle):$X^0_s = rand$, $V^0_s = \frac{rand - X^0_s}{2}$, $\forall s \in S^{max}$;
Calculate objective function for each particle: $F\left(X^0_s\right), \forall s \in S^{max}$;
Initialize the personal best for each particle: $P_s = X^0_s, F(P_s) = F\left(X^0_s\right), \forall s \in S^{max}$;
Initialize the incumbent and current solutions: $a = \arg\min_{s \in S^{max}}\left\{F\left(X^0_s\right)\right\}$;
$P^* = X^0_a, F(P^*) = F(X^0_a), X^{SA} = P^*, F(X^{SA}) = F(P^*)$;
Set the initial temperature and generation counter: $T = T_I, g = 0$;
while $(T \geq T_F) \parallel (P^* > Optimal/LB)$ **do**
  for $r = 1 : R$ **do**
    Generate a neighbor solution $Y$and calculate $F(Y)$;
    Calculate $\Delta F = F(X^{SA}) - F(Y)$;
    if $\Delta F \geq 0$ **then**
      Replace the current solution: $X^{SA} = Y$, $F(X^{SA}) = F(Y)$**if** $F\left(X^{SA}\right) < F(P^*)$ **then**
        Replace the incumbent solution: $P^* = X^{SA}$, $F(P^*) = F\left(X^{SA}\right)$
      end
    else
      Calculate the acceptance probability: $Pr = e^{\frac{\Delta F}{T}}$ **if** $Pr \geq rand$ **then**
        Replace the current solution: $X^{SA} = Y$, $F\left(X^{SA}\right) = F(Y)$
      end
    end
  end
  for $s = 1 :$ S^{max} **do**
    Update the velocity of particle s: V^{g+1}_s;
    Update the position of particle s: X^{g+1}_s;
    Calculate $F(X^{g+1}_s)$;
    if $F\left(X^{g+1}_s\right) < F(P_s)$ **then**
      Update particle's personal best: $P_s = X^{g+1}_s, F(P_s) = F\left(X^{g+1}_s\right)$
    end
  end
  $a = \arg\min_{s \in S^{max}}\{F(P_s)\}$;
  if $F(P_a) < F(P^*)$ **then**
    Update the incumbent solution: $P^* = P_a$, $F(P^*) = F(P_a)$
  end
  Update the current solution: $X^{SA} = P^*$, $F(X^{SA}) = F(P^*)$;
  Update the temperature and generation counter: $T = \alpha T, g = g + 1$,
end
Return the incumbent solution: P^* and $F(P^*)$.

The algorithm starts by initializing the parameters (swarm size S^{max}, inertia weight ω, cognitive c_1 and social c_2 acceleration coefficients, initial T_I and final T_F temperatures, cooling rate α, and number of iterations in the inner loop R) and generating a random swarm or particles. Each particle ($s \in S^{max}$) is initialized (at generation $g = 0$) with two random vectors that represent its position (X^0_s) and its velocity (V^0_s). A feasible schedule is then constructed (see Section 4.3) and evaluated according to the performance metric under consideration, which is either *ET* or $C_{\max}$ and is denoted by $F(X^0_s)$. The best position and corresponding metric value of each particle are initialized as the particle's initial position and metric value, i.e. $P_s = X^0_s$ and $F(P_s) = F(X^0_s)$. Finally, the position and objective function value of the incumbent (P^* and $F(P^*)$) and of the current solution (X^{SA} and $F(X^{SA})$ are set to those of the particle with the best objective function value.

The outer loop checks whether the temperature value remains higher than the final temperature: if not, then the algorithm terminates; otherwise, the temperature is decreased. The first inner loop, which is repeated R times, searches for a neighbor solution (see Section 4.4). If the neighbor solution is better than the current solution, then it replaces the current solution and if it is also better than the incumbent solution, then it also replaces the

incumbent solution. However, a worse neighbor solution may also be accepted as a replacement of the current solution; this way allowing for a more extensive search. The probability of acceptance depends on how worse the neighbor solution is and it decreases as the solution space exploration progresses. In the second inner loop, by following the PSO movement mechanism, the particles move to different positions and a new swarm of particles is obtained, see Section 4.2 for details. The incumbent and the current solutions are updated, if required, and the outer loop is restarted.

4.2. Swarm transition

The PSO algorithm we propose adapts the standard PSO-2011 (Clerc, 2012), which has been shown to be rotation, scale, and translation invariant (Bonyadi & Michalewicz, 2014; Zambrano-Bigiarini, Clerc, & Rojas, 2013).

A new swarm of particles is obtained by updating the particles' position and velocity. The velocity is updated first and then the new velocity, in fact a displacement, is used to update the position. Although several expressions have been proposed to update the velocity; typically, they use, in addition to the particle's current velocity, the particle's current position, the particle's best position, and the incumbent's position. It has been observed that the trajectories of the particles tend to remain parallel with the coordinate axes (Spears, Green, & Spears, 2010) and that the distribution of all possible next positions is not uniform; being denser near to the center of a D-dimensional rectangular around the particle's current position (Clerc, 2012).

Since the observed bias is introduced when updating the velocity, the PSO-2011 standard proposes to update it in a geometrical way (positioning all particles within a D-dimensional sphere), as follows: At each generation g and for each particle s, a center of gravity G_s^g is defined around three points: X_s^g - the current position of the particle, p_s^g - a point a little ¤beyond¤ the current position of the particle in the direction of the position of its personal best (P_s), and l_s^g - a point a little ¤beyond¤ the current position of the particle in the direction of the position of the incumbent solution (P^*), see Eqs. (28) to (30).

$$G_s^g = \frac{\left(X_s^g \oplus p_s^g \oplus l^g\right)}{3}, \text{ where} \tag{28}$$

$$p_s^g = X_s^g \oplus c_1 U_1^g \otimes \left(P_s \ominus X_s^g\right), \tag{29}$$

$$l_s^g = X_s^g \oplus c_2 U_2^g \otimes \left(P^* \ominus X_s^g\right). \tag{30}$$

In these equations, c_1 and c_2 are the cognitive and social coefficients, respectively, U_1^g and U_2^g are two independent random vectors uniformly distributed in $[0, 1]$, and $\otimes$, $\oplus$, and $\ominus$ denote element-wise vector arithmetic operators.

A random point $X_s^{'g}$ is chosen inside the hypersphere with center G_s^g and radius $\left\|G_s^g \ominus X_s^g\right\|$, where $\|\cdot\|$ represents the norm of a vector (see Eq. (31)). The velocity and position of particle s are then updated as in Eqs. (32) and (33), where ω is the inertia weight. Since the velocity value may become extremely large (positive or negative), a phenomenon known as ¤velocity explosion¤, we impose it to be in the range $[-V_{\max}, V_{\max}]$.

$$X_s^{'g} \sim \mathcal{H}\left(G_s^g, \left\|G_s^g \ominus X_s^g\right\|\right), \tag{31}$$

$$V_s^{g+1} = \begin{cases} V_{\max}, & \text{if } \omega V_s^g \oplus X_s^{'g} \ominus X_s^g > V_{\max}, \\ -V_{\max}, & \text{if } \omega V_s^g \oplus X_s^{'g} \ominus X_s^g < -V_{\max}, \\ \omega V_s^g \oplus X_s^{'g} \ominus X_s^g, & \text{otherwise}, \end{cases} \tag{32}$$

$$X_s^{g+1} = X_s^g \oplus V_s^{g+1}. \tag{33}$$

4.3. Solution representation and decoding

A solution to the JSPT is represented by a real two-part vector (particle) with N elements each, where N is the total number of production operations, including the dummy operations, i.e., $N = \sum_{j \in J} n'_j$ with $n'_j = n_j + 1$. Elements 1 to N are associated with the (production) operations, which are considered in their natural order; thus, elements 1 to n'_1 refer to the operations of job 1, elements $n'_1 + 1$ to $n'_1 + n'_2$ refer to the operations of job 2 and so on. Elements in the second part of the vector refer to the assignment of vehicles to the (transport) tasks required by the operations, which, as before, are considered in their natural order. Initially, all particles are randomly generated following a uniform distribution; the first N elements in the interval $[0, 1]$ and the second N elements in the interval $[0, A]$.

The decoding procedure has two distinct parts, one to obtain a sequence of production operations and another to obtain an assignment of vehicles. To decode the first part of the solution vector, we use the smallest position value (SPV) rule, originally devised by Bean (1994), to convert the real numbers into a permutation of operations.[1] In order to ensure feasibility, the permutation of operations is converted into a permutation of corresponding jobs and then the job numbers are converted into the ordered operation numbers, i.e., the first time a job number appears it is decoded to its first operation, the second time to its second operation, and so on.

Regarding the second part of the solution vector, the real numbers are converted into vehicle numbers as follows: numbers smaller than or equal to 1 are translated into vehicle 1, numbers larger than 1 but smaller than or equal to 2 are translated into vehicle 2, and so on until vehicle A, for which any real number larger than $(A-1)$ is translated into vehicle A. Each element of the second part of the solution vector refers to the unique task associated with each operation. Therefore, element $N+i$ assigns a vehicle to the task required to have operation i processed.

Let us illustrate the encoding and decoding procedures resorting to the example used before. The solution vector has $2 \times N = 24$ elements. An initial random particle for this problem can be seen in Fig. 3a and the corresponding permutation of operations, obtained using the SPV rule, is shown in Fig. 3b. For example, since 0.01 in the fifth position is the smallest value (in the first part of the particle), the first operation in the schedule is operation 5, followed by operation 7 which is the position for 0.05, the second smallest value, and so on. This permutation is then translated into the sequence of jobs depicted in Fig. 3c; operations 1 to 4 are represented by 1, the job they refer to, operations 5 to 8 by 2, and operations 9 to 12 by 3. The job numbers are then converted into the ordered operations by replacing the ith appearance of job j by O_{ij}. Regarding the second part of the vector, values less than or equal to 1 are replaced by 1 and the ones larger than 1 were replaced by 2. Thus, a vehicle (element value) is assigned to the task required by each operation (element position). A feasible schedule is depicted in Fig. 3d.

Using the sequence of operations obtained and the input data regarding machine-operation allocation (see Table 2), we determine the sequence of operations for each machine: $M1 - O_{22}, O_{11}, O_{33}$; $M2 - O_{13}, O_{31}$; $M3 - O_{12}, O_{21}$; $M4 - O_{32}, O_{23}$.

Regarding the assignment of the vehicles, vehicle 1 is assigned to tasks 1, 3, 4, 7, 8, 11, which are associated with

[1] Here and hereafter, when collectively referring to the (production) operations the dummy operation is included.

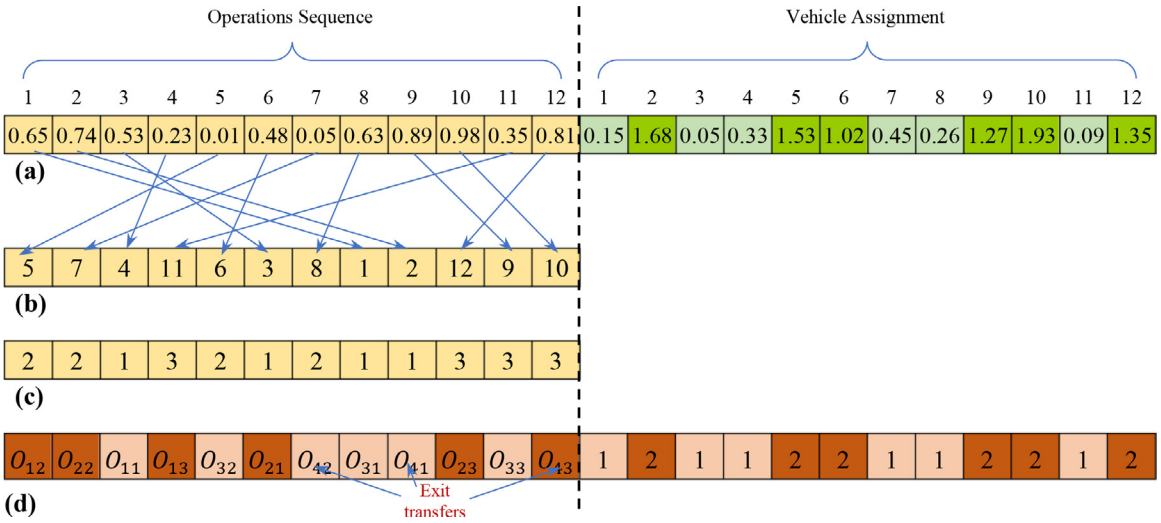

Fig. 3. Illustrative example: an initial solution and corresponding decoded feasible schedule.

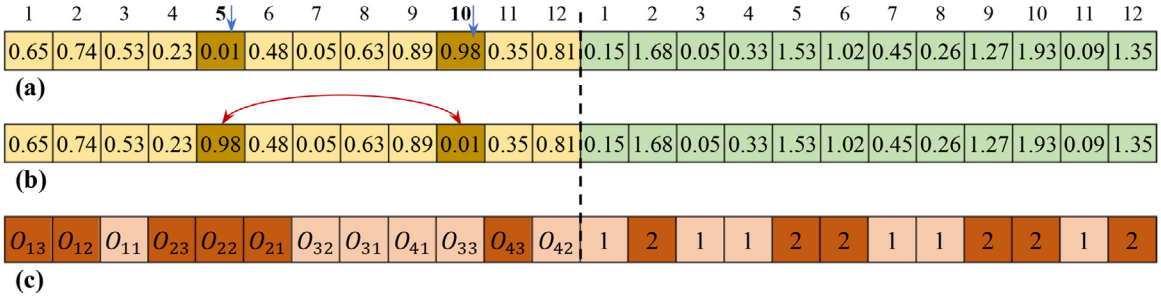

Fig. 4. Illustrative example: (a) original solution; (b) neighbor solution, (c) final feasible neighbor solution.

operations $O_{11}, O_{31}, O_{41}, O_{32}, O_{42}, O_{33}$. The sequence of these operations, which is given by the first part of the vector, is: $O_{11}, O_{32}, O_{42}, O_{31}, O_{41}, O_{33}$. Therefore, vehicle 1 transports job 1 from the LU to machine M_{11}, job 2 between machines M_{22} and M_{32} and then to the LU (job 2 exits), job 1 between machines M_{21} and M_{31} and then to the LU (job 1 exits), and job 3 between machines M_{23} and M_{33}. Similarly, vehicle 2 transports job 2 from the LU to machine M_{12} and then to machine M_{22}, job 3 from the LU to machine M_{13}, job 1 between machines M_{11} and M_{21}, and job 3 between machines M_{13} and M_{23} and then to the LU (job 3 exits).

4.4. Neighborhood search structure

To obtain a neighbor solution, we first generate a random number between 0 and 1: if it is less than 0.5, then we swap two elements of the first part of the solution vector; otherwise, we swap two elements of the second part. In both cases, we generate two random numbers (between 1 and N or between $N+1$ and $2N$, depending on whether we are swapping elements of the first or of the second part of the solution vector) to determine the elements to be swapped. The resulting neighbor solution may have a different sequence of operations on machines and vehicles or a different assignment of vehicles, depending on whether the swapped elements are in the first or the second part of the vector. Fig. 4 illustrates the application of this procedure by generating a neighbor solution to the one depicted in Fig. 3a, which, for convenience, is repeated in Fig. 4a. Assume the swap to be performed on the first part of the solution vector and that the two random numbers generated are 5 and 10. The encoded neighbor solution is shown in Fig. 4b and the corresponding feasible schedule in Fig. 4c.

5. Numerical results and discussion

The effectiveness and efficiency of the proposed MILP model, of the LB method, and of the PSOSA are assessed on three sets of benchmark problem instances,[2] namely: the BU instances proposed by Bilge & Ulusoy (1995), the HK instances proposed by Hurink & Knust (2005), and the SWV instances proposed by Ham (2020). The results obtained are compared with current state-of-the-art approaches.

For the BU instances, the results obtained with the LB method and the PSOSA are compared with the optimal solutions obtained by solving the MILP model for both the exit time (ET) and the makespan (C_{max}). Additionally, we also compare them with the memetic algorithm (MA) proposed by Lacomme et al. (2013) and the anytime layered search algorithm (ALS) proposed by Baruwa & Piera (2016) when optimizing the ET and with the tabu search algorithm (TS) proposed by Zheng et al. (2014) and the ALS (Baruwa & Piera, 2016) when optimizing the C_{max}.

The results obtained by the PSOSA for the HK instances are compared with those of Hurink & Knust (2005): heuristic solutions obtained by two tabu search algorithm versions (TS_{one} and TS_{long}) and lower bounds (LB_H) and with those of Lacomme et al. (2013) (MA).

Finally, for the SWV instances, the PSOSA and LB results are compared with the constraint programming (CP2) results by Ham (2020), while minimizing C_{max}.

The MILP model was solved using Gurobi® 9.0.3 and the proposed algorithms were coded and solved in Julia 1.2 (Bezanson,

[2] All data pertaining to these instances can be downloaded from https://fastmanufacturingproject.wordpress.com/2019/04/11/jspt-instances/.

Edelman, Karpinski, & Shah, 2017). We carried out all the computational experiments on a personal computer with a Core™ i7-8700 and 8 GB of RAM.

Regarding the state-of-the-art approaches, Ham (2020) implemented the CP models in IBM OPL CPLEX 12.8.0 and run the experiments on a personal computer with an Intel™ Core i7-4770 CPU with 16 GB of RAM. For the SWV instances he reports the best solution found within 1800 seconds as well as the CPLEX CP Optimizer lower bounds. No computation times are reported. The ALS (Baruwa & Piera, 2016) was implemented in C++ on a 2.60GHz AMD Opteron processor PC with 4GB of RAM. Several experiments were performed, with a 3600 seconds time limit, and the best solution obtained and the corresponding computation time are reported. Zheng et al. (2014) implemented the TS algorithm they propose in C and each instance was solved five times on a 2.60 gigahertz Pentium IV with 2 gigabyte of RAM. The best makespan value obtained is reported. No computation times are reported, except for a set of 23 small-sized instances they randomly generated. The MA (Lacomme et al., 2013) was implemented in Delphi 7.0 and the computational experiments were carried out on a 2.0 gigahertz Pentium Dual Core with 2 gigabyte of memory. Each instance was solved five times with a time limit of 120 seconds. They report the best solution found but no computation times. The tabu search algorithms of Hurink & Knust (2005) were implemented in C and tested on a Sun Ultra 2 work station (167 megahertz) with operating system Solaris 2.5 and 320 megabyte of general storage. While TS_{one} was run six times with a 600 seconds time limit, TS_{long} was run 12 times with a 3600 seconds time limit. For each instance, they report the best solution found but no computation times. Nevertheless, average computation times are reported for the two groups of problem instances. Further, they also report lower bounds obtained with constraint propagation and linear programming.

To select the parameters of the PSOSA, we follow Zambrano-Bigiarini et al. (2013) and set the inertia weight, the cognitive and social acceleration coefficients, the velocity limits, and the swarm size as follows: $\omega = \frac{1}{2*\ln(2)}$, $c_1 = 0.5 + \ln(2)$, $c_2 = 0.5 + \ln(2)$, $V_{\min} = -2$, $V_{\max} = 2$, and $S^{\max} = 40$. The inner loop parameters were empirically chosen. We performed a series of preliminary tests considering a subset of five problem instances: EX11 and EX71 from the BU instances, P2D1d1 from the HK instances, and SWV01 and SWV05 from the SWV instances. Each of these instances was solved 15 times for all combinations of the following values: $T_I = \{50, 100, 500, 1000, 2000\}$, $T_F = \{0.01, 0.05, 0.1, 0.5, 1\}$, $R = \{10, 20, 30, 40, 50\}$, and $\alpha = \{0.99, 0.995, 0.999, 0.9995, 0.9999\}$. After analyzing the average results, they were set to: $T_I = 100$, $T_F = 0.01$, $R = 50$, and $\alpha = 0.9999$.

5.1. Results for the BU instances

Bilge & Ulusoy (1995) proposed a set of instances that have become defacto benchmark instances. They proposed ten job sets, each with four to eight jobs and 13 to 21 operations, and four layouts with four machines each and one LU. The 40 instances obtained by combining the ten job sets and the four layouts have been named EX11, EX21, until EX104, in which the first figure (one or two digits) refers to the job set and the second one (last digit) refers to the layout. All instances consider two vehicles.

These instances were solved using the MILP model, the lower bounding procedure, and the PSOSA.

Figs. 5 and 6 summarize the results obtained. Details of the these results are provided in Tables A.10 and A.11 for the exit time *ET* and the makespan $C_{\max}$, respectively. The *GAP*s and the CPU Time Ratios are calculated as in Eqs. (34) and (35), respectively.

$$GAP_{LB} = \frac{Optimal - LB}{Optimal} \times 100 \quad GAP = \frac{Solution - Optimal}{Optimal} \times 100. \tag{34}$$

$$\text{CPU Time Ratio} = \frac{T(Solution)}{T(MILP) + T(ALS) + T(PSOSA)}. \tag{35}$$

Fig. 5 shows, in the upper graph, the optimality *GAP*s for the LB method and the PSOSA, ALS, and MA heuristics. Furthermore, in its lower graph, it shows the percentage CPU time ratios of the MILP model, the PSOSA, and the ALS. Note that the CPU time of each method, in seconds, is written on the area representing it.

The MILP model was able to find an optimal solution for all but two instances (EX71 and EX74), regardless of the objective function. For these two instances, we report the best solutions found and the time it took to find them within the 50,000 seconds time limit imposed. For all the other instances, we report the computation time that Gurobi required to find an optimal solution and verify its optimality.

The lower bounds are, on average, about 19.8% away from optimality, ranging from 2.4 to 36.6%.

PSOSA found an optimal/best-known solution for all instances, except instance EX71; outperforming both MA and ALS, which failed to do so for two and three instances, respectively. For the two instances for which no optimal solution is known, the PSOSA was able to improve on the best-known solutions by, on average, 0.32%, while the MA and the ALS improve on them by 0.29% and 0.24%, respectively. Furthermore,the PSOSA robustness can be inferred from the low values of the exit time percentage standard deviation that range from 0 to 2.53% with an average of 1.23%, see Table A.10. We chose to provide the percentage solution standard deviation since, on the one hand, it is a good metric regarding methodology robustness and, on the other hand, allows for a comparison across problem instances.

Regarding the computation times, for some instances the MILP is solved to optimally very quickly and in those cases the PSOSA requires a few extra seconds. However, the average computation time of the PSOSA is much smaller than that of the MILP. A direct comparison of the PSOSA computation times with those of the ALS is not possible since the algorithms have been implemented using different programming languages and applications and run on different computers. Nevertheless, it can easily be seen that PSOSA is faster for most instances. On average PSOSA took 9.6 seconds (ranging from 0.2 to 18.9 seconds) while ALS took 1117.9 seconds (ranging from 3.5 to 6996.6 seconds). Finally, since Lacomme et al. (2013) do not report computation times, no comparisons are possible.

Regarding the $C_{\max}$ objective function, Fig. 6 shows, in the upper graph, the optimality *GAP*s for the LB method and the PSOSA, the ALS, and the MA heuristics. Furthermore, in its lower graph, it shows the percentage CPU time ratios for the MILP model, the PSOSA, and the ALS. Note that the CPU time of each method, in seconds, is written on the area representing it.

As it can be seen, our lower bounding procedure improves on that of (Zheng et al., 2014) (denoted as LB_Z). We have found better values for 11 instances that are, on average, 17.32% higher (ranging from 1.39 to 41.07% higher) and match the best known lower bounds for all the others. The PSOSA can compete fairly with the TS algorithm as both found an optimal solution for all instances for which one is known and match the best-known solutions for the remaining two instances. The ALS algorithm was not able to find an optimal solution for instance EX103. Moreover, the PSOSA is very robust, since the makespan percentage variation is very small; it ranges from 0% to 2.16% with an average of 0.7%.

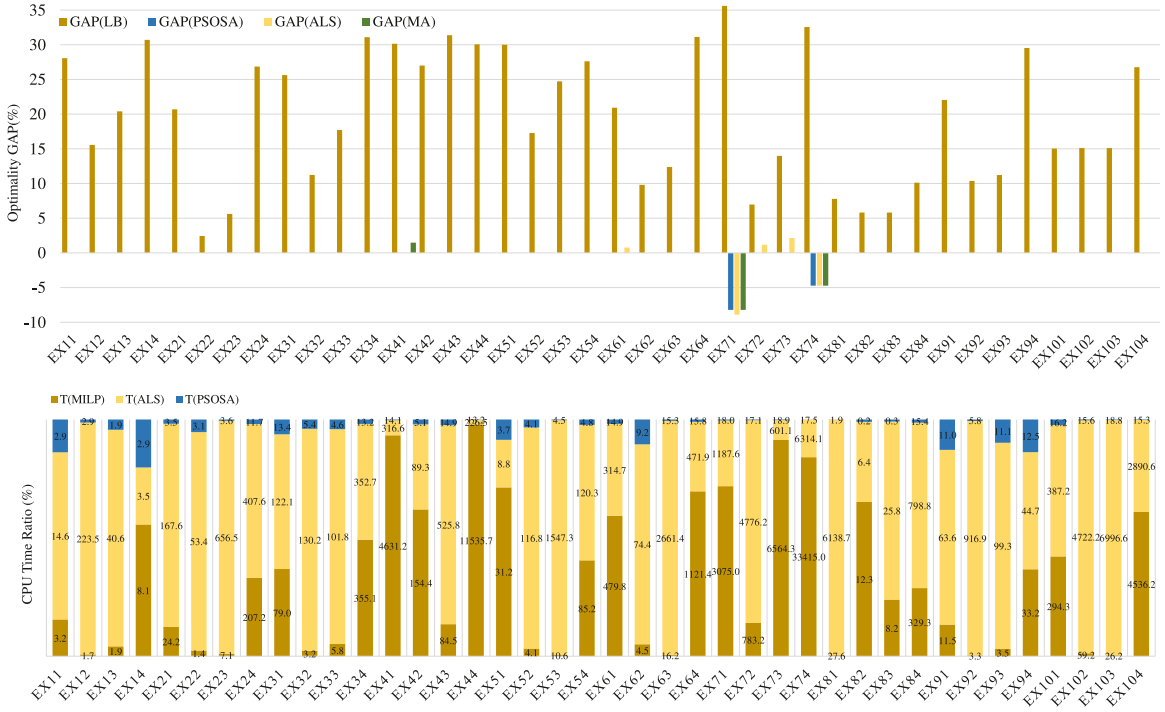

Fig. 5. Results for the BU instances with the objective of minimizing the exit time (*ET*). Upper graph: optimality GAP(%) for the LB procedure, the PSOSA, the ALS, and the MA. Lower graph: CPU time ratio (%) for the MILP, the PSOSA, and the ALS.

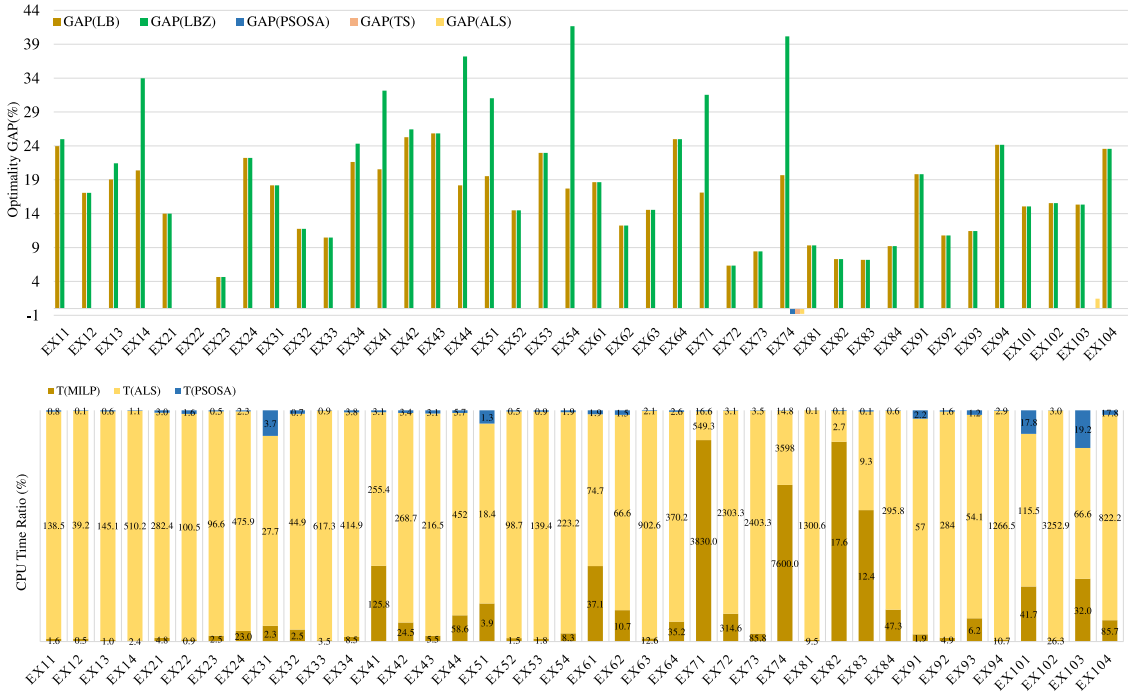

Fig. 6. Results for the BU instances with the objective of minimizing the makespan (C_{max}). Upper graph: optimality GAP(%) for the our lower bounding procedure *LB* and that of Zheng et al. (2014) LB_Z, the PSOSA, the TS, and the ALS. Lower graph: CPU time ratio (%) for the MILP, the PSOSA and the ALS.

Regarding the computation times the conclusions are similar to the ones drawn when optimizing the exit time. Although the PSOSA computation time is larger than that of the MILP for two instances (1.6 and 3.7 seconds instead of 0.94 and 2.3 seconds), for the remaining instances it was much faster. The average computation times are 3.8 and 312.6 seconds for the PSOSA and the MILP, respectively. As before a direct comparison of PSOSA computation times with those of ALS is not possible. However, PSOSA is much faster for all instances. On average it took 3.8 seconds (ranging from 0.1 to 19.2 seconds) while ALS took 559 seconds (ranging from 2.7 to 3598 seconds). Finally, since Zheng et al. (2014) do not report computation times, no comparisons are possible.

5.2. Results the for HK instances

Hurink & Knust (2005) proposed a set of 23 instances by modifying the well-known P1 and P2 instances from Muth (1963). There are seven P1 instances, each with six jobs and 36 operations, and

Table 4
Results for the HK instances with C_{max} as the objective function (as no LU is considered, all transport tasks to and from the LU are disregarded).

			PSOSA				MA		TS_{one}		TS_{long}	
Name	$\|M\| - \|J\| - N$	LB_H	C_{max}	*GAP*	σ%	*T*	C_{max}	*GAP*	C_{max}	*GAP*	C_{max}	*GAP*
P1D1d1	6-6-66	82	87	6.10	1.48	38.3	87	6.10	87	6.10		
P1D1t1	6-6-66	77	81	5.19	0.57	35.8	81	5.19	81	5.19		
P1D2d1	6-6-66	147	149	1.36	1.06	35.4	148	0.68	148	0.68		
P1D3d1	6-6-66	213	213	0.00	0.72	21.8	213	0.00	217	1.88		
P1T2t1	6-6-66	71	74	4.23	0.86	36.8	74	4.23	74	4.23		
P1T3t0	6-6-66	92	92	0.00	0.00	11.0	92	0.00	92	0.00		
P1tkl.1	6-6-66	136	136	0.00	0.95	18.0	136	0.00	137	0.74		
P2D1d1	10-10-190	880	978	11.14	2.48	120.4	1012	15.00	1044	18.64	990	12.50
P2D1t0	10-10-190	880	967	9.89	1.89	119.7	1017	15.57	1042	18.41	989	12.39
P2D1t1	10-10-190	880	955	8.52	2.31	159.5	983	11.70	1016	15.45	989	12.39
P2D2d1	10-10-190	892	989	10.87	2.00	116.7	1045	17.15	1070	19.96	993	11.32
P2D3d1	10-10-190	906	1054	16.34	1.53	124.8	1100	21.41	1070	18.10	1072	18.32
P2D5t2	10-10-190	1167	1292	10.71	0.40	124.8	1361	16.62	1325	13.54	1371	17.48
P2T1t1	10-10-190	874	956	9.38	0.85	124.6	978	11.90	1006	15.10	1018	16.48
P2T2t1	10-10-190	880	949	7.84	2.78	121.5	993	12.84	1015	15.34	1030	17.05
P2T5t2	10-10-190	898	991	10.36	2.28	121.1	1022	13.81	1102	22.72	1020	13.59
P2tkl.1	10-10-190	888	986	11.04	2.04	113.2	1009	13.63	1086	22.30	1018	14.64
P2tkl.2	10-10-190	896	987	10.16	2.09	120.5	1002	11.83	1028	14.73	1014	13.17
P2*0.5D1d1	10-10-190	482	551	14.32	2.32	122.2	581	20.54	555	15.15	558	15.77
P2*0.5D1t1	10-10-190	482	528	9.54	3.06	118.0	546	13.28	544	12.86	542	12.45
P2*0.5D2d1	10-10-190	497	634	27.57	0.71	113.9	673	35.41	633	27.36	666	34.00
P2*0.5D2t0	10-10-190	497	566	13.88	1.83	114.5	584	17.51	578	16.30	595	19.72
P2*0.5D2t1	10-10-190	497	597	20.12	1.06	116.2	620	24.75	613	23.34	620	24.75
Mean				9.50	1.53	93.4		12.57		13.40		16.63

sixteen P2 instances, each with 10 jobs and 100 operations. All instances consider a single vehicle. In these instances no LU area is considered and thus, jobs are only transported between machines. The jobs are initially by the machine processing their first operation and are considered completed whenever their last operation is processed.

Table 4 reports the results for these 23 instances. In addition to instance characteristics, it also reports the lower bounds obtained by Hurink & Knust (2005) (LB_H). Regarding the heuristic solutions it reports the best value, corresponding gap and computation time, and percentage makespan standard deviation for the PSOSA and the best value and corresponding gap for the MA, the TS_{one}, and the TS_{long}. The gaps are computed as in Eq. (36).

$$GAP = \frac{Solution - LB_H}{LB_H} \times 100. \quad (36)$$

As it can be seen from the results in Table 4, the PSOSA clearly outperforms the other three algorithms since, except for two instances, it always finds a solution as good or better. Additionally, the average percentage makespan standard deviation is very small (1.53%).

Regarding individual comparisons, the PSOSA matches the best MA solution for six instances, improves on the best MA solution for 16 instances, and finds a worse solution for the remaining one; the PSOSA average gap is 3% below that of the MA. In comparison with the best solution found by the TS_{one}, the PSOSA finds a better solution for 17 instances, matches it for four instances, and finds a worse solution for the remaining two; the average gap of the PSOSA is about 4% below that of the TS_{one}. Finally, for the 16 instances for which results are known for the TS_{long}, the PSOSA always finds better solutions with an average gap over 4.5% lower.

The PSOSA computation times range from 11.02 to 159.47 seconds with an average value of 93.43 seconds over all instances and of 28.2 and 122 seconds over instances in P1 and in P2, respectively. Hurink & Knust (2005) report only the average computation time for each group of problem instances. In the case of the TS_{one}, they report 77 and 161 seconds for P1 and P2 instances, respectively (about 173% and 32% over those of the PSOSA). Additionally, they also report no significant improvements if the time limit of 600 seconds is relaxed. The combined TS, which consists of applying the one-stage TS and then the two-stages TS, produces worse results when run using the same time limit (600 seconds). However, when run for 3600 seconds the average gap, over all instances, comes down from 34 to 16.1% (TS_{long}). Nevertheless, no computation times are given in this case. As before, no computation times are available for the MA.

5.3. Results for the SWV instances

Recently, Ham (2020) adapted 10 large-sized problem instances, originally proposed by Storer et al. (1992) for the job shop scheduling problem, by considering two arbitrarily generated travel time matrices for layouts with 10 and 15 machines and two fleet sizes with 3 and 5 vehicles. These instances involve 20 jobs and either 200 or 300 production operations.

Thus far, these instances have only been considered by Ham (2020) that proposed two constraint programming approaches (CP1 and CP2); since CP2 always outperforms CP1, here we only consider CP2 results. Although Ham (2020) only considers the makespan as the objective function, in here, we consider and report solutions for both the makespan and the exit time, see Tables 5 and 6, respectively. These tables report instances' characteristics, LB values, PSOSA solutions (best solution, objective function percentage standard deviation, gap, and computation time), and the best solution reported in (Ham, 2020) for fleet sizes of three and five vehicles.

From the results reported, it can be seen that PSOSA outperforms CP2 since, out of the 10 instances considered, it found better solutions for nine and seven instances for fleet sizes of three and five vehicles, respectively. Furthermore, the PSOSA average gaps are lower than those of CP2, about 3% and 1% lower for fleet sizes of three and five vehicles, respectively. Regarding computation times, no comparisons are possible as Ham (2020) does not report computation times.

Results for the SWV instances considering the exit time are here reported for the first time, see Table 6. The gaps are similar to those of the makespan, see Table 5.

The PSOSA remains robust in the presence of larger problem instances. The average percentage standard deviations are 1.18% and 0.98% for the makespan and the exit time, respectively, when three

Table 5
Results for the SWV instances with C_{max} as the objective function (the last transport task of each job is disregarded).

		A=3							A=5						
			PSOSA				CP2			PSOSA				CP2	
Name	$\|M\| - \|J\| - N$	*LB*	C_{max}	*GAP*	$\sigma\%$	*T*	C_{max}	*GAP*	*LB*	C_{max}	*GAP*	$\sigma\%$	*T*	C_{max}	*GAP*
SWV01	10-20-400	1518	1959	29.05	1.45	943.4	2021	33.14	1518	1626	7.11	1.62	976.1	1631	7.44
SWV02	10-20-400	1617	1925	19.05	1.88	962.2	1939	19.91	1617	1665	2.97	1.63	995.2	1667	3.09
SWV03	10-20-400	1526	1920	25.82	1.71	938.1	1930	26.47	1452	1641	13.02	1.65	966.6	1674	15.29
SWV04	10-20-400	1528	1953	27.81	1.25	995.9	1944	27.23	1525	1683	10.36	1.71	979.4	1711	12.20
SWV05	10-20-400	1538	1930	25.49	0.95	1009.6	1952	26.92	1538	1675	8.91	1.31	1005.2	1654	7.54
SWV06	15-20-600	2192	2686	22.54	1.04	1944.8	2771	26.41	1711	2123	24.08	1.64	2012.3	2109	23.26
SWV07	15-20-600	2224	2679	20.46	0.80	2011.1	2796	25.72	1605	2051	27.79	1.21	2072.7	2022	25.98
SWV08	15-20-600	2160	2715	25.69	0.83	2083.2	2796	29.44	1765	2188	23.97	1.37	2057.7	2230	26.35
SWV09	15-20-600	2212	2738	23.78	0.99	2009.5	2851	28.89	1798	2099	16.74	1.92	2001.0	2147	19.41
SWV10	15-20-600	2235	2730	22.15	0.89	2028.7	2750	23.04	1807	2197	21.58	1.38	2019.7	2219	22.80
Mean				24.18	1.18	1492.6		26.72			15.65	1.55	1508.6		16.34

Table 6
Results for the SWV instances with *ET* as the objective function.

		A=3					A=5				
Name	$\|M\| - \|J\| - N$	*LB*	*ET*	*GAP*	$\sigma\%$	*T*	*LB*	*ET*	*GAP*	$\sigma\%$	*T*
SWV01	10-20-440	1581	2026	28.15	1.64	984.4	1545	1674	8.35	1.74	979.0
SWV02	10-20-440	1642	2044	24.48	1.08	918.3	1642	1701	3.59	1.35	919.4
SWV03	10-20-440	1620	2066	27.53	0.90	917.8	1479	1669	12.85	1.80	963.5
SWV04	10-20-440	1621	2072	27.82	1.17	956.5	1552	1753	12.95	1.49	938.0
SWV05	10-20-440	1630	2053	25.95	0.92	949.2	1569	1712	9.11	1.74	989.7
SWV06	15-20-640	2279	2837	24.48	0.88	2080.4	1729	2162	25.04	1.51	2087.7
SWV07	15-20-640	2346	2888	23.10	0.76	2021.6	1609	2098	30.39	1.30	2041.6
SWV08	15-20-640	2259	2833	25.41	0.97	2078.9	1783	2205	23.67	1.17	2028.9
SWV09	15-20-640	2296	2865	24.78	0.85	1968.9	1827	2131	16.64	1.68	2043.8
SWV10	15-20-640	2361	2924	23.85	0.68	2017.9	1811	2245	23.96	1.11	2063.2
Mean				25.56	0.98	1489.4			16.66	1.49	1505.5

Table 7
Impact of the transport time on the makespan of the job shop scheduling problem for the BU instances.

Name	JSP	JSP+T	JSPT	$\delta\%$	$\rho\%$	Name	JSP	JSP+T	JSPT	$\delta\%$	$\rho\%$
EX11	60	116	96	60.00	20.83	EX61	88	134	118	34.09	13.56
EX12	60	96	82	36.67	17.07	EX62	88	105	98	11.36	7.14
EX13	60	94	84	40.00	11.90	EX63	88	115	103	17.05	11.65
EX14	60	121	103	71.67	17.48	EX64	88	148	120	36.36	23.33
EX21	70	117	100	42.86	17.00	EX71	66	138	111	68.18	24.32
EX22	70	89	76	8.57	17.11	EX72	66	104	79	19.70	31.65
EX23	70	94	86	22.86	9.30	EX73	66	111	83	25.76	33.73
EX24	70	124	108	54.29	14.81	EX74	66	157	126	90.91	24.60
EX31	70	123	99	41.43	24.24	EX81	141	162	161	14.18	0.62
EX32	70	92	85	21.43	8.24	EX82	141	151	151	7.09	0.00
EX33	70	100	86	22.86	16.28	EX83	141	153	153	8.51	0.00
EX34	70	124	111	58.57	11.71	EX84	141	179	163	15.60	9.82
EX41	54	126	112	107.41	12.50	EX91	81	136	116	43.21	17.24
EX42	54	106	87	61.11	21.84	EX92	81	121	102	25.93	18.63
EX43	54	110	89	64.81	23.60	EX93	81	120	105	29.63	14.29
EX44	54	150	121	124.07	23.97	EX94	81	142	120	48.15	18.33
EX51	48	97	87	81.25	11.49	EX101	112	179	146	30.36	22.60
EX52	48	83	69	43.75	20.29	EX102	112	172	135	20.54	27.41
EX53	48	85	74	54.17	14.86	EX103	112	173	137	22.32	26.28
EX54	48	112	96	100.00	16.67	EX104	112	213	157	40.18	35.67
Mean										43.17	17.30

vehicles are considered and 1.55 and 1.49%, when five vehicles are considered.

5.4. Impacts of the joint scheduling method

A common practice in the job shop scheduling community is to assume that travel times are negligible and/or that there are unlimited transport resources available. Once an optimal JSP schedule is on hand, the required transport tasks are added. However, it is unlikely that this approach will generate optimal solutions. Having this in mind, we show that ignoring the transport resources is misleading. To illustrate it, in Tables 7–9 we compare the makespan obtained for, respectively, the BU, HK, and SWV instances by the following approaches:

- **JSP:** Schedule of the production operations, ignoring transport tasks.
- **JSP+T:** Schedule of the production operations, ignoring transport tasks, with the required travel and waiting times added afterwards.
- **JSPT:** Joint schedule of the production operations and transport tasks.

To find solutions to the JSP version of the instances, we adapted the PSOSA proposed. Solutions are encoded by a vector with $\sum_{j\in J}|J_j|$ elements. Hence, the second part of the original vector is

Table 8
Impact of the transport time on the makespan‡ of the job shop scheduling problem for the HK instances.

Name	JSP	JSP+T	JSPT	δ%	ρ%	Name	JSP	JSP+T	JSPT	δ%	ρ%
P1D1d1	55	134	87	58.18	54.02	P2D5t2	930	2181	1292	38.92	68.81
P1D1t1	55	108	81	47.27	33.33	P2T1t1	930	1529	956	2.80	59.94
P1D2d1	55	192	149	170.91	28.86	P2T2t1	930	1686	949	2.04	77.66
P1D3d1	55	266	213	287.27	24.88	P2T5t2	930	1558	991	6.56	57.21
P1T2t1	55	102	74	34.55	37.84	P2tkl.1	930	1499	986	6.02	52.03
P1T3t0	55	117	92	67.27	27.17	P2tkl.2	930	1903	987	6.13	92.81
P1tkl.1	55	173	136	147.27	27.21	P2*0.5D1d1	498	877	551	10.64	59.17
P2D1d1	930	1677	978	5.16	71.47	P2*0.5D1t1	498	874	528	6.02	65.53
P2D1t0	930	1410	967	3.98	45.81	P2*0.5D2d1	498	1112	634	27.31	75.39
P2D1t1	930	1665	955	2.69	74.35	P2*0.5D2t0	498	979	566	13.65	72.97
P2D2d1	930	1791	989	6.34	81.09	P2*0.5D2t1	498	1069	597	19.88	79.06
P2D3d1	930	1974	1054	13.33	87.29						
Mean										47.90	93.87

‡ all transport tasks to and from the LU are disregarded.

Table 9
Impact of the transport time on the makespan of the job shop scheduling problem for the SWV instances.

		A=3				A=5			
Name	JSP	JSP+T	JSPT	δ%	ρ%	JSP+T	JSPT	δ%	ρ%
SWV01	1444	3201	1959	35.66	63.40	2295	1626	12.60	41.14
SWV02	1496	3049	1925	28.68	58.39	2343	1665	11.30	40.72
SWV03	1443	3137	1920	33.06	63.39	2273	1641	13.72	38.51
SWV04	1505	3235	1953	29.77	65.64	2263	1683	11.83	34.46
SWV05	1482	3346	1930	30.23	73.37	2264	1675	13.02	35.16
SWV06	1711	4376	2686	56.98	62.92	3210	2123	24.08	51.20
SWV07	1660	4704	2679	61.39	75.59	3198	2051	23.55	55.92
SWV08	1803	4754	2715	50.58	75.10	3397	2188	21.35	55.26
SWV09	1695	4686	2738	61.53	71.15	3091	2099	23.83	47.26
SWV10	1797	4585	2730	51.92	67.95	3450	2197	22.26	57.03
Mean				43.98	67.69			17.76	45.67

ignored. The neighborhood operator swaps two randomly selected operations.

To find the makespan resulting from adding the required travel and waiting times (JSP+T schedule), we resort to a greedy heuristic to assign the vehicles to the transport tasks as follows: production operations are considered one at the time (from left to right) and to the corresponding transport task we assign the vehicle that can pickup the job the earliest. Ties are broken by assigning a random vehicle.

In Tables 7 to 9 we also report the percentage difference (δ%) between JSP and JSPT makespans and the percentage increase (ρ%) of the JSP+T makespan over the JSPT makespan, which are computed as:

$$\delta\% = \frac{C_{\max}^{\text{JSPT}} - C_{\max}^{\text{JSP}}}{C_{\max}^{\text{JSP}}} \times 100 \text{ and } \rho\% = \frac{C_{\max}^{\text{JSP+T}} - C_{\max}^{\text{JSPT}}}{C_{\max}^{\text{JSPT}}} \times 100.$$

The former clearly shows that transport tasks cannot be ignored as the required travel times may impose a much larger makespan, up to almost three times larger in our experiments; while the latter clearly shows that production and transport need to be optimized simultaneously as otherwise the makespan can be significantly larger, up to two times larger.

From the results reported in Tables 7 to 9 it can be seen that the JSPT makespan is larger than that of the JSP; on average about 43 and 48% larger for the BU and HK instances, respectively, and about 44 and 18% larger for the SWV instances with 3 and with 5 vehicles, respectively. These results clearly show that transport (tasks and resources) cannot be ignored.

Additionally, the results also show that production and transport should be optimized simultaneously; since the JSP+T makespan is considerably larger than that of the JSPT; on average about 17 and 94% larger for the BU and HK instances, respectively, and about 68%, and 46% larger for the SWV instances with 3 and with 5 vehicles, respectively.

Finally, as expected, the average δ% and the average ρ% considerably decrease when the number of vehicles increases from 3 to 5 (see Table 9). It is clear that having a larger fleet of vehicles decreases the impact of the transport as it significantly reduces waiting times.

6. Conclusion

This paper addresses the scheduling problem in job shop environments considering transport tasks and vehicles. This problem is harder than most scheduling problems, since it involves not only scheduling production operations but also assigning vehicles to transport tasks and scheduling the transport tasks. These two scheduling problems, when considered in isolation, are NP-hard (Lenstra & Kan, 1979; 1981). In this study, they have been considered jointly to ensure feasibility of the production plan. The computational experiments clearly show that production and transport should be jointly optimized, since otherwise the makespan is considerably larger, it increases by up to 94%. Furthermore, they also show that having a larger fleet of vehicles decreases the impact of the transport by significantly reducing waiting times.

We propose a hybridized particle swarm optimization and simulated annealing algorithm that is capable of finding good quality solutions to this NP-hard problem regardless of optimizing the exit time or the makespan. We also propose a lower bounding procedure that almost instantaneously provides good quality lower bound values, which can be used to assess the efficacy of the PSOSA in tackling large-sized instances. Although, no accurate computation time comparisons are possible, the PSOSA computation times are much lower than the ones reported in the literature. Moreover, the robustness of the PSOSA is clearly demonstrated as the percentage standard deviation of the 15 solutions found for each objective function and each of the 73 instances considered

is always below 3.06% and 2.53% for the makespan and the exit, respectively. This is a very important and valued characteristic in practice.

The computational experiments with the BU instances have shown that an optimal solution (both for the exit time and for the makespan) was found for all but two problem instances. Regarding the latter two instances, we found solutions as good as the best-known ones.

Regarding the HK medium-size problem instances, all considering a single vehicle, we found new best-known solutions for 15 instances (between 0.4 and 4.4% better) and matched the best-known solution for six instances; for the remaining two instances our solutions are 0.68 and 0.16% way from the best-known ones (considering the best reported results on different works).

Finally, regarding the SWV problem instances (all large-sized) recently proposed, the PSOSA finds nine and seven new best-known solutions, when three and five vehicles are considered, respectively (between 0.12 and 4.18% better). For the remaining instances our solutions are between 0.46 and 1.43% away from the best-known ones. Overall, our solutions are, on average, 3% and 1% better that the best-known ones, when three and five vehicles are considered, respectively.

The problem considered in this work is an extension of the well-known job shop scheduling problem since jobs need to be transported around the shop floor by a limited number of vehicles. Nevertheless, several other issues can also be incorporated; for example, relaxing common and standard assumptions such as collision avoidance, deadlock resolution, and congestion management. None of these issues has been discussed within the job shop scheduling problem with transport resources. Other interesting and meaningful issues that may be considered are sustainability issues, such as energy consumption and product deadlines.

Acknowledgments

This work is supported by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within projects POCI-01-0145-FEDER-031821- PTDC/EGE-OGE/31821/2017 and POCI-01-0145-FEDER-031447- PTDC/ EEI-AUT/ 31447/2017.

Appendix A. Detailed results for BU instances

Tables A.10 and A.11 report detailed results for the BU instances when optimizing the exit time ET and the makespan $C_{\max}$, respectively.

Table A10
Results for the BU instances with the objective of minimizing the exit time (ET).

		MILP		LB		PSOSA				ALS			MA	
Name	$\|M\| - \|J\| - N$	ET^*	T	ET	GAP_{LB}	ET	GAP	$\sigma\%$	T	ET	GAP	T	ET	GAP
EX11	4-5-31	114	3.2	82	28.07	114	0.00	1.46	2.9	114	0.00	14.6	114	0.00
EX12	4-5-31	90	1.7	76	15.56	90	0.00	0.89	2.9	90	0.00	223.5	90	0.00
EX13	4-5-31	98	1.9	78	20.41	98	0.00	0.93	1.9	98	0.00	40.6	98	0.00
EX14	4-5-31	140	8.1	97	30.71	140	0.00	1.29	2.9	140	0.00	3.5	140	0.00
EX21	4-6-36	116	24.2	92	20.69	116	0.00	1.42	3.5	116	0.00	167.6	116	0.00
EX22	4-6-36	82	1.4	80	2.44	82	0.00	1.83	3.1	82	0.00	53.4	82	0.00
EX23	4-6-36	89	7.1	84	5.62	89	0.00	1.08	3.6	89	0.00	656.5	89	0.00
EX24	4-6-36	134	207.2	98	26.87	134	0.00	1.76	11.7	134	0.00	407.6	134	0.00
EX31	4-6-38	121	79.0	90	25.62	121	0.00	2.04	13.4	121	0.00	122.1	121	0.00
EX32	4-6-38	89	3.2	79	11.24	89	0.00	2.28	5.4	89	0.00	130.2	89	0.00
EX33	4-6-38	96	5.8	79	17.71	96	0.00	2.19	4.6	96	0.00	101.8	96	0.00
EX34	4-6-38	148	355.1	102	31.08	148	0.00	1.50	13.2	148	0.00	352.7	148	0.00
EX41	4-5-43	136	4631.2	95	30.15	136	0.00	0.77	14.1	136	0.00	316.6	138	1.47
EX42	4-5-43	100	154.4	73	27.00	100	0.00	2.53	5.1	100	0.00	89.3	100	0.00
EX43	4-5-43	102	84.5	70	31.37	102	0.00	1.06	14.9	102	0.00	525.8	102	0.00
EX44	4-5-43	163	11535.7	114	30.06	163	0.00	1.13	13.2	163	0.00	226.5	163	0.00
EX51	4-5-31	110	31.2	77	30.00	110	0.00	1.53	3.7	110	0.00	8.8	110	0.00
EX52	4-5-31	81	4.1	67	17.28	81	0.00	1.49	4.1	81	0.00	116.8	81	0.00
EX53	4-5-31	89	10.6	67	24.72	89	0.00	1.46	4.5	89	0.00	1547.3	89	0.00
EX54	4-5-31	134	85.2	97	27.61	134	0.00	1.22	4.8	134	0.00	120.3	134	0.00
EX61	4-6-42	129	479.8	102	20.93	129	0.00	1.25	14.9	130	0.78	314.7	129	0.00
EX62	4-6-42	102	4.5	92	9.80	102	0.00	1.01	9.2	102	0.00	74.4	102	0.00
EX63	4-6-42	105	16.2	92	12.38	105	0.00	0.72	15.3	105	0.00	2661.4	105	0.00
EX64	4-6-42	151	1121.4	104	31.13	151	0.00	1.76	15.8	151	0.00	471.9	151	0.00
EX71†	4-8-46	146	3075.0	94	35.62	134	-8.22	1.56	18.0	133	-8.90	1187.6	134	-8.22
EX72	4-8-46	86	783.2	80	6.98	86	0.00	2.52	17.1	87	1.16	4776.2	86	0.00
EX73	4-8-46	93	6564.3	80	13.98	93	0.00	2.17	18.9	95	2.15	601.1	93	0.00
EX74†	4-8-46	169	33415.0	114	32.54	161	-4.73	1.32	17.5	161	-4.73	6314.1	161	-4.73
EX81	4-6-46	167	27.6	154	7.78	167	0.00	0.00	1.9	167	0.00	6138.7	167	0.00
EX82	4-6-46	155	12.3	146	5.81	155	0.00	0.00	0.2	155	0.00	6.4	155	0.00
EX83	4-6-46	155	8.2	146	5.81	155	0.00	0.00	0.3	155	0.00	25.8	155	0.00
EX84	4-6-46	178	329.3	160	10.11	178	0.00	0.50	15.4	178	0.00	798.8	178	0.00
EX91	4-5-39	127	11.5	99	22.05	127	0.00	0.52	11.0	127	0.00	63.6	127	0.00
EX92	4-5-39	106	3.3	95	10.38	106	0.00	0.58	5.8	106	0.00	916.9	106	0.00
EX93	4-5-39	107	3.5	95	11.21	107	0.00	0.00	11.1	107	0.00	99.3	107	0.00
EX94	4-5-39	149	33.2	105	29.53	149	0.00	0.77	12.5	149	0.00	44.7	149	0.00
EX101	4-6-48	153	294.3	130	15.03	153	0.00	1.74	16.2	153	0.00	387.2	153	0.00
EX102	4-6-48	139	59.2	118	15.11	139	0.00	0.46	15.6	139	0.00	4722.2	139	0.00
EX103	4-6-48	139	26.2	118	15.11	139	0.00	0.85	18.8	139	0.00	6996.6	139	0.00
EX104	4-6-48	183	4536.2	134	26.78	183	0.00	1.77	15.3	183	0.00	2890.6	183	0.00
Mean			1701.0		19.81		-0.32	1.23	9.6		-0.24	1117.9		-0.29

Table A11
Results for the BU instances with the objective of minimizing the makespan (C_{max}) - last transport task of each job is disregarded.

		MILP		LB		LB_Z		PSOSA				TS		ALS		
Name	$\|M\| - \|J\| - N$	C^*_{max}	T	C_{max}	GAP_{LB}	C_{max}	GAP_{LB}	C_{max}	GAP	$\sigma\%$	T	C_{max}	GAP	C_{max}	GAP	T
EX11	4-5-26	96	1.6	73	23.96	72	25.00	96	0.00	0.00	0.8	96	0.00	96	0.00	138.5
EX12	4-5-26	82	0.5	68	17.07	68	17.07	82	0.00	0.00	0.1	82	0.00	82	0.00	39.2
EX13	4-5-26	84	0.98	68	19.05	66	21.43	84	0.00	0.00	0.6	84	0.00	84	0.00	145.1
EX14	4-5-26	103	2.4	82	20.39	68	33.98	103	0.00	0.25	1.1	103	0.00	103	0.00	510.2
EX21	4-6-30	100	4.8	86	14.00	86	14.00	100	0.00	1.62	3.0	100	0.00	100	0.00	282.4
EX22	4-6-30	76	0.94	76	0.00	76	0.00	76	0.00	0.00	1.6	76	0.00	76	0.00	100.5
EX23	4-6-30	86	2.5	82	4.65	82	4.65	86	0.00	0.00	0.5	86	0.00	86	0.00	96.6
EX24	4-6-30	108	23	84	22.22	84	22.22	108	0.00	1.83	2.3	108	0.00	108	0.00	475.9
EX31	4-6-32	99	2.3	81	18.18	81	18.18	99	0.00	2.16	3.7	99	0.00	99	0.00	27.7
EX32	4-6-32	85	2.5	75	11.76	75	11.76	85	0.00	0.00	0.7	85	0.00	85	0.00	44.9
EX33	4-6-32	86	3.5	77	10.47	77	10.47	86	0.00	0.00	0.9	86	0.00	86	0.00	617.3
EX34	4-6-32	111	8.5	87	21.62	84	24.32	111	0.00	0.67	3.8	111	0.00	111	0.00	414.9
EX41	4-5-38	112	125.8	89	20.54	76	32.14	112	0.00	1.71	3.1	112	0.00	112	0.00	255.4
EX42	4-5-38	87	24.5	65	25.29	64	26.44	87	0.00	1.15	3.4	87	0.00	87	0.00	268.7
EX43	4-5-38	89	5.5	66	25.84	66	25.84	89	0.00	1.98	3.1	89	0.00	89	0.00	216.5
EX44	4-5-38	121	58.6	99	18.18	76	37.19	121	0.00	1.63	5.7	121	0.00	121	0.00	452
EX51	4-5-26	87	3.9	70	19.54	60	31.03	87	0.00	0.53	1.3	87	0.00	87	0.00	18.4
EX52	4-5-26	69	1.5	59	14.49	59	14.49	69	0.00	0.00	0.5	69	0.00	69	0.00	98.7
EX53	4-5-26	74	1.8	57	22.97	57	22.97	74	0.00	0.35	0.9	74	0.00	74	0.00	139.4
EX54	4-5-26	96	8.3	79	17.71	56	41.67	96	0.00	0.51	1.9	96	0.00	96	0.00	223.2
EX61	4-6-36	118	37.1	96	18.64	96	18.64	118	0.00	0.69	1.9	118	0.00	118	0.00	74.7
EX62	4-6-36	98	10.7	86	12.24	86	12.24	98	0.00	0.53	1.5	98	0.00	98	0.00	66.6
EX63	4-6-36	103	12.6	88	14.56	88	14.56	103	0.00	0.49	2.1	103	0.00	103	0.00	902.6
EX64	4-6-36	120	35.2	90	25.00	90	25.00	120	0.00	1.78	2.6	120	0.00	120	0.00	370.2
EX71†	4-8-38	111	3830.0	92	17.12	76	31.53	111	0.00	1.17	16.6	111	0.00	111	0.00	549.3
EX72	4-8-38	79	314.6	74	6.33	74	6.33	79	0.00	1.87	3.1	79	0.00	79	0.00	2303.3
EX73	4-8-38	83	85.8	76	8.43	76	8.43	83	0.00	1.97	3.5	83	0.00	83	0.00	2403.3
EX74†	4-8-38	127	7600.0	102	19.69	76	40.16	126	-0.79	0.63	14.8	126	-0.79	126	-0.79	3598.0
EX81	4-6-40	161	9.5	146	9.32	146	9.32	161	0.00	0.00	0.1	161	0.00	161	0.00	1300.6
EX82	4-6-40	151	17.6	140	7.28	140	7.28	151	0.00	0.00	0.1	151	0.00	151	0.00	2.7
EX83	4-6-40	153	12.4	142	7.19	142	7.19	153	0.00	0.00	0.1	153	0.00	153	0.00	9.3
EX84	4-6-40	163	47.3	148	9.20	148	9.20	163	0.00	0.00	0.6	163	0.00	163	0.00	295.8
EX91	4-5-34	116	1.9	93	19.83	93	19.83	116	0.00	0.55	2.2	116	0.00	116	0.00	57.0
EX92	4-5-34	102	4.9	91	10.78	91	10.78	102	0.00	0.58	1.6	102	0.00	102	0.00	284.0
EX93	4-5-34	105	6.2	93	11.43	93	11.43	105	0.00	0.00	1.2	105	0.00	105	0.00	54.1
EX94	4-5-34	120	10.7	91	24.17	91	24.17	120	0.00	0.82	2.9	120	0.00	120	0.00	1266.5
EX101	4-6-42	146	41.7	124	15.07	124	15.07	146	0.00	0.48	17.8	146	0.00	146	0.00	115.5
EX102	4-6-42	135	26.3	114	15.56	114	15.56	135	0.00	0.52	3.0	135	0.00	135	0.00	3252.9
EX103	4-6-42	137	32.0	116	15.33	116	15.33	137	0.00	0.66	19.2	137	0.00	139	1.46	66.6
EX104	4-6-42	157	85.7	120	23.57	120	23.57	157	0.00	0.71	17.8	157	0.00	157	0.00	822.2
Mean			312.6		15.97		19.01		-0.02	0.70	3.8		-0.02		0.02	559.0

In both tables and for each problem instance we report: instance characteristics ($|M|$-number of machines, $|J|$-number of jobs, and N-number of operations and tasks), optimal objective function value (ET^* or C^*_{max}) and required computation time (T in seconds), and the lower bound value and corresponding optimality gap. Heuristic solutions (best value and corresponding optimality gap and computation time) are reported for the PSOSA and the state-of-the-art approaches.

Regarding computation time, for the MILP model we report the time that Gurobi required to find an optimal solution and verify its optimality, except for instances EX71 and EX74. For these two instances no optimal solution was found within the 50,000 seconds time limit imposed, regardless of the objective function. Hence, we report the time it took to find the best solutions found within the time limit.

Since we solved each instance 15 times, we also report the percentage exit time and makespan standard deviations ($\sigma\%$). We chose to provide the percentage solution standard deviation since, on the one hand, it is a good metric regarding methodology robustness and, on the other hand, allows for a comparison across problem instances.

The optimality gaps GAP_{LB} and GAP are computed as:

$$GAP_{LB} = \frac{Optimal - LB}{Optimal} \times 100 \text{ and } GAP = \frac{Solution - Optimal}{Optimal} \times 100.$$

References

Abdelmaguid, T., Nassef, A. O., Kamal, B. A., & Hassan, M. F. (2004). A hybrid Ga/Heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research, 42*(2), 267–281.

AGV network (2022). AGV applications - where are automated guided vehicles used? https://www.agvnetwork.com/agv-applications, last checked on 2022-05-30.

Baruwa, O., & Piera, M. (2016). A coloured petri net-based hybrid heuristic search approach to simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research, 54*(16), 4773–4792.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing, 6*(2), 154–160.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review, 59*(1), 65–98.

Bilge, Ü., & Ulusoy, G. (1995). A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research, 43*(6), 1058–1070.

Bonyadi, M. R., & Michalewicz, Z. (2014). Spso 2011: analysis of stability; local convergence; and rotation sensitivity. In Proceedings of the 2014 annual conference on genetic and evolutionary computation (pp. 9–16). ACM.

Chih, M. (2022). Stochastic stability analysis of particle swarm optimization with pseudo random number assignment strategy. *European Journal of Operational Research*. In press.

Clerc, M. (2012). Standard particle swarm optimisation. Technical Report, HAL Id: hal-00764996. https://hal.archives-ouvertes.fr/hal-00764996.

Deroussi, L., Gourgand, M., & Tchernev, N. (2008). A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research, 46*(8), 2143–2164.

Dong, J., Zhang, L., & Xiao, T. (2018). A hybrid PSO/SA algorithm for bi-criteria

stochastic line balancing with flexible task times and zoning constraints. *Journal of Intelligent Manufacturing, 29*(4), 737–751.
Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Mhs'95. proceedings of the sixth international symposium on micro machine and human science (pp. 39–43). Ieee.
El Khayat, G., Langevin, A., & Riopel, D. (2006). Integrated production and material handling scheduling using mathematical programming and constraint programming. *European Journal of Operational Research, 175*(3), 1818–1832.
Fathi, M., Rodríguez, V., Fontes, D. B. M. M., & Alvarez, M. J. (2016). A modified particle swarm optimisation algorithm to solve the part feeding problem at assembly lines. *International Journal of Production Research, 54*(3), 878–893.
Fontes, D. B. M. M., & Homayouni, S. M. (2019). Joint production and transportation scheduling in flexible manufacturing systems. *Journal of Global Optimization, 74*(4), 879–908.
Fragapane, G., De Koster, R., Sgarbossa, F., & Strandhagen, J. O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research, 294*(2), 405–426.
Ham, A. (2020). Transfer-robot task scheduling in job shop. *International Journal of Production Research, 59*(3), 813–823.
Hurink, J., & Knust, S. (2002). A tabu search algorithm for scheduling a single robot in a job-shop environment. *Discrete Applied Mathematics, 119*(1–2), 181–203.
Hurink, J., & Knust, S. (2005). Tabu search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research, 162*(1), 99–111.
Javidrad, F., Nazari, M., & Javidrad, H. (2018). Optimum stacking sequence design of laminates using a hybrid PSO-SA method. *Composite Structures, 185*, 607–618.
JBT (2022a). Jbt automated systems - applications. https://www.jbtc.com/automated-systems/products-and-applications/applications/, last checked on 2022-06-05.
JBT (2022b). Jbt automated systems - industries. https://www.jbtc.com/automated-systems/products-and-applications/industries/, last checked on 2022-06-05.
Knust, S. (2000). *Shop-scheduling problems with transportation*. Ph.D. thesis
Lacomme, P., Larabi, M., & Tchernev, N. (2013). Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics, 143*(1), 24–34.
Lenstra, J. K., & Kan, A. R. (1979). Computational complexity of discrete optimization problems. *Annals of discrete mathematics* (vol. 4, pp. 121–140). Elsevier.
Lenstra, J. K., & Kan, A. R. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11*(2), 221–227.
MHI (2022). AGVs transport palletized cement board. https://www.mhi.org/downloads/industrygroups/agvs/casestudies/agvs-transport-palletized-cement-board-b.pdf, last checked on 2022-05-30.
MiR Solutions (2022a). Autonomous and collaborative mobile robots from MiR - case studies. https://www.mobile-industrial-robots.com/case-studies/, last checked on 2022-06-20.
MiR Solutions (2022b). Autonomous and collaborative mobile robots from MiR - industries. https://www.mobile-industrial-robots.com/industries/, last checked on 2022-06-20.
Muth, J. (1963). Probabilistic learning combinations of local job-shop scheduling rules. *Industrial Scheduling*.
Reddy, B., & Rao, C. (2006). A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. *The International Journal of Advanced Manufacturing Technology, 31*(5–6), 602–613.
Spears, W. M., Green, D. T., & Spears, D. F. (2010). Biases in particle swarm optimization. *International Journal of Swarm Intelligence Research, 1*(2), 34–57.
Storer, R. H., Wu, S. D., & Vaccari, R. (1992). A new search spaces for sequencing problems with application to job shop scheduling. *Management Science, 38*, 1495–1509.
Tang, H., Chen, R., Li, Y., Peng, Z., Guo, S., & Du, Y. (2019). Flexible job-shop scheduling with tolerated time interval and limited starting time interval based on hybrid discrete pso-sa: an application from a casting workshop. *Applied Soft Computing, 78*, 176–194.
Tang, L., Zhao, J., & Liu, J. (2014). Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research, 236*(3), 978–990.
Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. *European Journal of Operational Research, 177*(3), 1930–1947.
Transbotics - A Scott Company (2022a). Scott automation - industries. https://scottautomation.com/en/industry, last checked on 2022-06-05.
Transbotics - A Scott Company (2022b). Scott automation - insights. https://scottautomation.com/en/insights, last checked on 2022-06-20.
Ulusoy, G., Sivrikaya, F., & Bilge, Ü. (1997). A genetic algorithm approach TC the simultaneous scheduling of machines and automated guided vehicles. *Computers & Operations Research, 24*(4), 335–351.
Zambrano-Bigiarini, M., Clerc, M., & Rojas, R. (2013). Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In Proceedings of the IEEE congress on evolutionary computation (pp. 2337–2344). IEEE.
Zhang, L., Lu, J., & Yang, Z. (2021). Optimal scheduling of emergency resources for major maritime oil spills considering time-varying demand and transportation networks. *European Journal of Operational Research, 293*(2), 529–546.
Zhang, Q., Manier, H., & Manier, M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research, 39*(7), 1713–1723.
Zheng, Y., Xiao, Y., & Seo, Y. (2014). A tabu search algorithm for simultaneous machine/AGV scheduling problem. *International Journal of Production Research, 52*(19), 5748–5763.