Journal of Project Management 9 (2024) 109–130

Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Optimization of transport constraints and quality of service for joint resolution of uncertain scheduling and the job-shop problem with routing (JSSPR) as opposed to the job-shop problem with transport (JSSPT)

Khadija Assafra[a], Bechir Alaya[b*], Salah Zidi[c] and Mounir Zrigui[a]

[a]*Faculty of Sciences of Monastir, University of Monastir, Monastir, Tunisia*
[b]*Department of Management Information Systems and Production Management, College of Business and Economics, Qassim University, 6633, Buraidah, 51452, Saudi Arabia*
[c]*IResCoMath-Lab, University of Gabes, Tunisia*

C H R O N I C L E

Article history:
Received: November 20, 2023
Received in revised format: December 22, 2023
Accepted: January 23, 2024
Available online:
January 23, 2024

Keywords:
Optimization
Scheduling
Job Shop
Transportation
QoS
Modeling
JSSPR

A B S T R A C T

To better meet the qualitative and quantitative requirements of customers or relevant sector managers, workshop environments are implementing increasingly complex task management systems. The job shop scheduling problem (JSSP) involves assigning each task to a single machine while scheduling many tasks on different machines. Finding the best scheduling for machines is one of the challenging optimizations of difficult non-deterministic polynomial (NP) time problems. The fundamental goal of optimization is to shorten the makespan (total execution time of all tasks). This paper is interested in the joint resolution of scheduling and transport problems and more particularly the Job-shop problem with Routing (JSSPR) as opposed to the Job-shop problem with Transport (JSSPT). These two problems are modeled in the form of a disjunctive graph. For the JSSPT, the solution to the transport problem is not linked to any quality of service (QoS) criterion and the solution is therefore often semi-active. The Job-shop with Routing explicitly considers transport operations and uses algorithms from the transport community to solve the transport problem. It is shown that the routing part of the JSSPR is a problem of the vehicle routing family and of the Pickup and Delivery Problem family. QoS in the JSSPR is defined by the duration of tours, the duration of transport of parts and the waiting time for them. A new evaluation function – named Time-Lag Insertion Heuristic (TLH) – is proposed to evaluate a disjunctive graph by simultaneously minimizing the makespan and maximizing the quality of service. Thus, the solution obtained is not semi-active, but a compromise between the different criteria. This evaluation function is included in a metaheuristic. Our numerical evaluations demonstrate that, on the one hand, the TLH evaluation can find almost optimal solutions regarding the QoS criterion; and on the other hand, the TLH evaluation is not very sensitive to the order of insertion of the maximum time-lags during the different minimization steps.

1. Introduction

Most of the problems in planning and organizing tasks are, or boil down to, optimization problems. For the resolution of these problems, it is advisable to appeal to the construction of models. This is a very relevant solution, as long as it is carried out within a methodological framework. In most of the research dedicated to this type of problem, the authors assume that there is no downtime between two consecutive tasks, (Yang et al., 2021). Production planning and scheduling is usually a topic interesting for companies that seek to offer a quality service. The minimization delay in the manufacture and delivery of products may result in significant gains, especially in sectors where competition is difficult. Decision-making at the

* Corresponding author.
E-mail address: b.alaya@qu.edu.sa (B. Alaya)

doi: 10.5267/j.jpm.2024.1.002

operational level, related to minimizing costs and increasing production are two notable aspects that commonly attract the attention of complex industrial sites, Ojstersek et al. (2020). The difficulty of solving these scheduling problems comes from a large number of entities to manage and the need to construct a schedule comprising a large number of stains together with the exigence of many constraints such as constraints of precedence, transport, date, deadlines, and/or resource availability, Bueno et al. (2020). Research work on scheduling mobilizes a large number of researchers. This strong emulation is mainly due to the wide panorama of scheduling issues. Among them, we can mention the multi-path workshop problem, commonly called “Job Shop”, which holds a particularly prominent place so this problem is encountered in the industrial environment, Haifei, et al. (2021). Numerous research subjects have emerged from this problem. As the scheduling problems are diversified, we have dealt with a particular case of a workshop which is the Job-Shop or the walk-through workshop multiples which relate to the discontinuous production is us when dealing with relatively small quantities of varied products. The sequence of operations or jobs on resources in the operating plan may differ from one product to another (all products do not necessarily pass on all resources) (Hegen et al., 2022).

Among the most difficult scheduling problems are those relating to classic job-shop-type workshops. Their optimal resolution turns out, in most cases, to be very difficult because of their strongly combinatorial character, Lee and Loong (2019). Complexity theory classifies problems into two classes: P and NP, Dean (2015). Class P gathers the problems solvable by polynomial algorithms. An algorithm is said to be polynomial when its running time is bounded by O (p(x)), where p is a polynomial and x is the input length of an instance of the problem. Algorithms whose complexity cannot be polynomially bounded are called exponential and correspond to the class NP. NP-complete or NP-hard problems represent a large class among the NP class. The job shop problem belongs to this class. For a given computation time, it is possible to optimally solve small problems, while it becomes difficult to find a good admissible solution for a large problem, Pappas et al. (2017). The JSSP can be solved by addressing two sub-problems: the machine allocation problem which implies choosing an appropriate machine from among the alternate machines defined to handle each operation, and the issue by designating a machine for every operation, or the operation sequence problem and calculating its start and end time, Alvarez et al. (2021).The JSSP is known in 14 versions according to Abdolrazzagh-Nezhad and Abdullah (2017)and which respectively: flexible JSSP, Deterministic JSSP, static JSSP, dynamic JSSP, cyclic JSSP, periodic JSSP, no-wait JSSP, just-in-time JSSP, pre-emptive JSSP, re-entrant JSSP, large-scale JSSP, stochastic JSSP, assembly JSSP, and fuzzy JSSP. The use of exact methods requires a set of calculations, the number of which evolves exponentially with the size of the problem considered. It is therefore preferable to use approximate methods, such as those based on the principle of local search or evolutionary methods, Frihat et al. (2022). Today, the problem of JSSP is a required domain. A lot of literature searches focus on minimizing the makespan and increasing usage, Gao et al. (2020). Nowadays, most studies focus on the use of technical heuristics and meta-heuristics such as Fuzzy Logic (FL), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) etc. Most of the techniques used are the Ant Colonies Optimization (ACO) and Genetic Algorithm (GA), Türkyılmaz et al. (2020), Job-shop Scheduling Problem with Routing (JSSPR) which is a generalization of Job-shop Scheduling Problem with Transport (JSSPT). One of the "classic" challenges for modeling production systems with transport is the JSSPT. An extension of the Job-shop Scheduling Problem (JSSP) is the Job-shop Scheduling Problem with Transport. The JSSPT takes into account the management of a group of vehicles that must transport the parts between the machines. The JSSPR is defined as a JSSPT-type problem in which a quality of service (QoS) criterion concerning transport is added. Unlike the JSSPT, which has the sole objective of minimizing the makespan (end date of the last operation), the JSSPR takes into account the maximization of the QoS when minimizing the makespan. This article proposes, among other things, the disjunctive graph of the JSSPT has a new evaluation function, to obtain a solution of the JSSPR. The JSSPT disjunctive graph is introduced by Hurink and Knust (2005), Lacomme et al. (2013), Zeng et al. (2015), Zhang et al. (2014). The approach presented for the JSSPR differs from the JSSPT in the following points:

The definition of the QoS for the JSSPR is comparable to the QoS defined for the Dial-A-Ride Problem (DARP). A new disjunctive graph evaluation function that minimizes makespan and maximizes the QoS. An extension of the JSSPT benchmarks to encompass the JSSPR and non-unit capacity vehicle fleet cases. The new evaluation function is included in a metaheuristic-type resolution scheme. An Integer Linear Program (ILP) model for JSSPR is also introduced in this paper.

The rest of this paper is structured as follows: The JSSP study and its primary versions are presented and described in Section 2. Section 3 represents the graphical model of JSSPR with a QOS consideration. Finally, an analysis of the experience study is presented in section 4. Section 5 concludes the paper.

2. Related Work

The Job Shop Scheduling issue (JSSP), which Graham first proposed in 1966, Abukhader and Kadoore (2021), is a well-known intractable combinatorial optimization issue. It is one of the challenging NP optimizations that has been researched for decades and seeks to allocate numerous operations over several machines. The major goal of optimization has been to reduce the Makespan of full processes, (Mihoubi et al., 2021). JSSP management is complete by considering operations' accessibility and people's availability and equipment required to carry out an operation. Here, it's important to reduce the amount of time products spend in the workshop between receiving a customer order and the end of that workshop's product processing, (Zhang et al., 2021). On the other hand, with JSSP, processes are divided into jobs. Each job has a specific set

of products for which additional limitations are added and machines are given specific tasks, Cebi et al. (2020). The JSSP variant that is the simplest is: We display the input for the n jobs $J_1, J_2. \ldots, J_n$that must be scheduled on m machines with variable processing power using circles, and the processing in the machines using rectangles.

However, most of the research focuses on developing specific optimization characteristics for static or predictable settings. In the literature, many theories discuss different classes of manufacturing systems that are susceptible to unanticipated and unpredictable occurrences, such as job cancellation, machine breakdowns, urgent order modifications, changes to the due date (advance or postponement), delays in the arrival of raw materials, and changes in employment priority, Cunha et al. (2020). The following elements should be taken into account while discussing the Job Shop problem: the number of machines (workstations), the work order, the performance evaluation criterion, and the arrival model. Two distinct arrival patterns exist, Saidat et al. (2022), Jain and Meeran (1999): *(i)* Static: An idle machine receives n jobs that ask to be scheduled for work. *(ii)* Dynamic: sporadic arrival. There are two different sorts of work orders: fixed and repetitive order, which is an issue in flow shops, and random order, which is conceivable in all models. The following performance evaluation criteria are listed in Kalshetty et al. (2020), Baykasoglu et al. (2014): makespan, average warehouse job duration, delay, average job volume, and machine usage.

Remember that an operation can visit the same device. This phenomenon is called "recirculation" or range looping, Nouri et al. (2016). However, it is common to find more restrictive formulations. For example, if looping routes are not allowed, each operation is then composed of $m$operations and each machine must perform n operations. the total number of operations is then $m * n$. Moreover, if $m = n$ then the problem is said to square. Another possible restriction is that i^{th} operation must be performed by the j^{th} machine which amounts to dealing with a flow shop problem, Parveen and Ullah (2010), Gaham et al. (2018).

2.1 Versions of Job Shop

We detail the main versions of a Job shop, which most research works take into consideration: The classic Job Shop problem, Flexible Job Shop, Just In Time Job Shop, the cyclic problem of Job Shop, and Dynamic Job Shop Scheduling.

a) The classic JSSP

JSSP is a classic problem that has been widely researched in the literature, and goes as follows: In a job shop, there are a number of n well-determined jobs scheduled in several machines m. The JSSP aims to find an optimal operation sequence schedule for each of these jobs where the assignment of machines and the temporal relationships between the different jobs are defined, Vital et al. (2020). JSSP has been studied through the development of mathematical programming formulations, Sun and Noble (1999), Jamili (2016), and an extensive variety of algorithms Kechadi et al. (2013), Vancheeswaran and Townsend (1993).

b) Flexible JSSP

Flexible Job Shop Scheduling Problem (FJSSP) is a refinement of the classic JSSP. One of the well-known combinatorial optimization problems concerns a significant number of applications in industrial fields such as production management, Transport systems, The supply chain, and Manufacturing systems. The advantages of FJSSP are cited in Xie et al. (2019) as follows: reduction of execution time, reduction of bottleneck resources, reduction of delivery times, and reduction of idle time. The restrictions of FJSSP are also cited in Xie et al. (2019) as follows: each operation has certain operations to process, each operation can only be assigned to one machine at a time, in each machine, the processing time for each operation is determined, each machine can manage at most one operation, at time zero, all jobs are available, and all operations must be carried out on certain machines.

c) Just in Time JSSP

Just in Time-Job Shop Scheduling (JIT-JSSP) is different from the classic JSSP in that the operation has a due date. Jobs in JIT-JSSP have operations that must be scheduled on machines in a determined order. Each operation has a due date, and early and late penalty coefficients Hussein and Zayed (2021). Any difference between the end time of the operation and its due date results in both cases an advance penalty or a late penalty. Specifically, the completion of the operation earlier than its due date incurs a penalty. Likewise, the completion of the transaction after its due date incurs a penalty Zhang at al. (2022). The main objective of the JIT-JSSP is to minimize the weighted sum of the early and late penalties. JIT-JSSP manufacturing can be presented into two categories, Ahmadian et al. (2021): with due dates at the employment level, and with due dates at the exploitation level.

d) The cyclic problem of JSSP

Cyclic Job Shop Scheduling Problem (CJSSP), each operation $O_i = \{1, \dots, n\}$ is assigned to a machine $r_i, R = 1, \dots, R$, with $R < n$ and operations linked by precedence constraints constitute jobs, Quinton et al. (2021). For example, in a manufacturing or production context, one can represent the manufacturing process of a product, while an operation represents only one step in the work process. Taking into account the constraints on the machines in the CJSSP because of an insufficient number of resources, the operations compete for the working time of the machines. This problem is represented by the disjunction constraints, which consider that for a pair of operations$(i, j) \in O_2$, which must be executed on the same machine, i.e.$r_i = r_j$, two occurrences i and j, cannot be executed simultaneously, (Smutnicki & Pempera, 2022).

e) Dynamic JSSP

Dynamic Job Shop Scheduling Problem (DJSSP), assumes that a scheduling plan assigning several jobs n to several machines m to be executed is known at the initial time, but leads to the problem of machine availability and real-time disruptions, the initial plan must be changed dynamically to adapt to real conditions, thus generating a rescheduling (Alaya, 2017). Machinery downtime is mainly caused by machine breakdowns and wear of cutting tools, which is supposed to be predictable (Mohan et al., 2019). During the period of maintenance, the machine will be available again, but the machine prohibits the processing of any operation (Kardos et al., 2021). The dynamic scheduling problem studied is subjected to the following assumptions (Zhang et al., 2021):

From the start, all machines and jobs are available;
Each machine can only process one job at a time;
Each operation is assigned to at least one machine for processing;
An operation cannot be processed until the previous operations are completed;
An operation cannot be stopped once it has begun, except when the machine is unavailable;
Although the work time for each operation is known in advance, it may change while processing;
The time corresponding to maintenance is known in the beginning.
The last operation is assumed to be integer variables and not negative, (Zhou & Liao, 2020).

2.2 For the JSSP with transportation

Taking into account the material manipulation leads to extending the notationα, β, λ,from Grahamet al. (1979). The notation is extended to$\alpha(k), \beta(k), \lambda(k)$, where$k$ denotes the number of transportmeans in the system (Zhang et al., 2012). The field α tells us about the machining environment, the type of production workshop, and the number of machines present in the system. The field β relates to the characteristics of the problems scheduling and groups the constraints imposed by the production environment and by the resources. The field λ concerns the performance criteria that will represent the objectives to achieve for the problem in question. End date of machine operations: These constraints guarantee that the end date C_i of the machine operation i is greater than or equal to its start datet_i plus its operating time PT_i.

$$C_i = t_i + PT_i \quad \forall i \in \mathrm{O} \tag{1}$$

Precedence between transport operations and machine operations. This group of constraints ensures that the start date of the machine operation can only be done once completed, the transport operation immediately preceding it. The duration of the latter depends on the robot k assign*ed to it.*

$$t_i \geq t'_{\lambda(i-1)} + \sum_{k=1}^{K} \left(f_{\lambda(i-1)} * t^k_{\mu_{i-1},\mu_i}\right) \forall i \in \mathrm{O} - \mathrm{PT} \tag{2}$$

With O represents the set of machine operations.

Precedence constraints between machine operations and transport operations. This group of constraints ensures that a transport operation only begins once the operation's previous machine is completed.

$$t'_{\lambda i} \geq C_i \quad \forall i \in \mathrm{O} - \mathrm{U} \tag{3}$$

WithUrepresents the set of machine operations that have no successors

Disjunction constraints on machines. This constraint requires that each machine not only process one transaction at a time. Given two operations$i \in \mathrm{O}$ and $j \in \mathrm{O}$processed consecutively on the same machine $\mu_i = \mu_j = \mu$, the constraints are given by the Eqs. (4-6).

The constraints (4) and (5) are arbitrated by the binary variable $b_{ij} \in \{1,0\}$ which activates a single constraint at a time corresponding to a single order of execution. When $b_{ij} = 0$ the constraint (5)is still valid because H is sufficientlylarge and the constraint (4) becomes active. In this case, constraint (4),can be rewritten in $t_i \geq C_i$, which means that machine operation i can onlystart processing if the operation machine j is finished. On the other hand, if $b_{ij} = 1$, it is the constraint (5) which is active implying that the operation machine j begins only once the operation machine i finished. Constraint (6) guarantees that the machine operation $i \in O$ takes place in first letthe machine operation $j \in 0$ take place first.

$$t_i \geq C_i * H * b_{ij} \forall (i,j) \in O \& \mu_i = \mu_j \quad (4)$$
$$t_i \geq C_j * H * (1 - b_{ij}) \forall (i,j) \in O \& \mu_i = \mu_j \quad (5)$$
$$b_{ij} + b_{ji} = 1 \forall (i,j) \in O \& \mu_i = \mu_j \quad (6)$$

Constraints for assigning robots to transport operations. This set of constraints ensures that for each transport operation (∀ i ∈ T) a robot is assigned to it.

$$\sum_{k=1}^{K} f_{ik} = 1 \quad \forall \text{ i} \in \text{ T} \quad (7)$$

Constraints on disjunction for transport operations. These restrictions state that a single robot cannot do two Transport operations at once. The constraints (8), (9), and(10)assigns 1 $to\ e_{ij}$ if and only if the two operations transport $i\ and\ j$ are assigned to the same robot k .In the case where two transport operations are assigned to the same robot, we have $d_{ij} = 1$ if transport operation i is performed before transport operation$j$ and $d_{ij} = 1$ otherwise. Constraint (14) therefore expresses thatonly one of these two situations product.

$$g_{ij}^k \geq 1 - (1 - f_{ik})H - 1 - (1 - f_{ij})H \quad \forall (i,j) \in T^2 \& k \in K \quad (8)$$
$$g_{ij}^k \leq f_{ik} \forall (i,j) \in T^2 \& k \in K \quad (9)$$
$$g_{ij}^k \leq f_{jk} \forall (i,j) \in T^2 \& k \in K \quad (10)$$
$$e_{ij} = \sum_{k=1}^{K} g_{ij}^k \forall (i,j) \in T^2 \quad (11)$$
$$t_i' \geq t_j' + \sum_{k=1}^{K} \left(t_{\mu_j,\mu_{j+1}}^K + v_{\mu_{j+1},\mu_i}^K\right)\left(e_{ij} - 1\right)H + \left(d_{ij} - 1\right)H \forall (i,j) \in T^2 \quad (12)$$
$$t_i' \geq t_i' + \sum_{k=1}^{K} \left(t_{\mu_i,\mu_{i+1}}^K + v_{\mu_{i+1},\mu_j}^K\right)\left(e_{ij} - 1\right)H + d_{ij} * H \quad (13)$$
$$\forall (i,j) \in T^2 \quad (14)$$
$$d_{ij} + d_{ji} = 1 \qquad \forall (i,j) \in T^2 \quad (15)$$

The criterion to be minimized is the end date of the last “Makespan” machine operation given by the variable $C_{max} = max\ (C_i), \forall\ i \in U$.

$$C_{max} \geq C_i (15)$$

2.3 QoS for the DARP

DARP is provided by Stein (1978) , and it is formalized by Cordeau and Laporte (2003), which proposes the following definition: each customer has a transport request between a collection point (loading node or pickup node) and a transportation point. Each node i is connected to a service period d_i, which corresponds to the time required to depositor load the client p_i; and at a time window $[E_i\ ; L_i]$during which the loading or delivery operation must be carried out. A vehicle arriving before the time window for an operation must wait for the opening of the operation before starting the service. Customer transportis delivered by a uniform and constrained vehicle fleet with capacity Q. All tours start and end at the depot.

The originality of the DARP proposed by Cordeau and Laporte (2003) lies in the type of solution sought: the dates of the visits are not necessarily adjusted to the left (at the earliest) to maximize the QoS. Starting a visit later can limit, for example, the time a customer spends in the vehicle. Generally, for “classic” problems, a calculation of the dates at the earliest is sufficient to schedule the transport operations. An algorithm solving a DARP maximizes the QoS from the customer's point of view, which means that it is necessary to define an algorithm to determine the date of arrival of the vehicle on a node,

the start date of service (the date that can be different from the arrival date), the end of service date and the departure date of the vehicle. Four variables are needed to explain a route (Fig. 1):

A_i is the date of arrival of the vehicle at node i.
$st_i$is the start date of the service at node i.
$C_i$is the end date of the service at node i, $C_i = st_i + D_i$.
$D_i$is the departure date of the vehicle from node i.
Cordeau and Laporte (2003), introduces three criteria (Figure 1) to meet DARP service quality considerations:
$R_p$is the transport time (Riding Time) of the customer p, it is defined between the date of departure from the node $i$where the customer is loaded and its start date on its delivery node i'. The node immediately after the loading node is not always the delivery node; several clients can be loaded and/or unloaded. $R_p = st_{i'} - D_i$.
DT_r is the trip's duration (Duration Time) of the vehicle r, described as the variation from the arrival date $E_n^r$of the vehicle at the depot to its date of departure B_0^r of the repository, $DT_r = E_n^r - B_0^r$.
WT_p is the Waiting Time of the customer p between (1) its date of departure D_i from the node$i$ and the end dateC_i of its service at the node i, $WT_p^1 = B_i - C_i$; (2) its start date of service st_i at the node $i$and its arrival date A_i to the node i, $WT_p^2 = st_i - A_i$. The waiting time is given by: $WT_p = WT_p^1 + WT_p^2$.

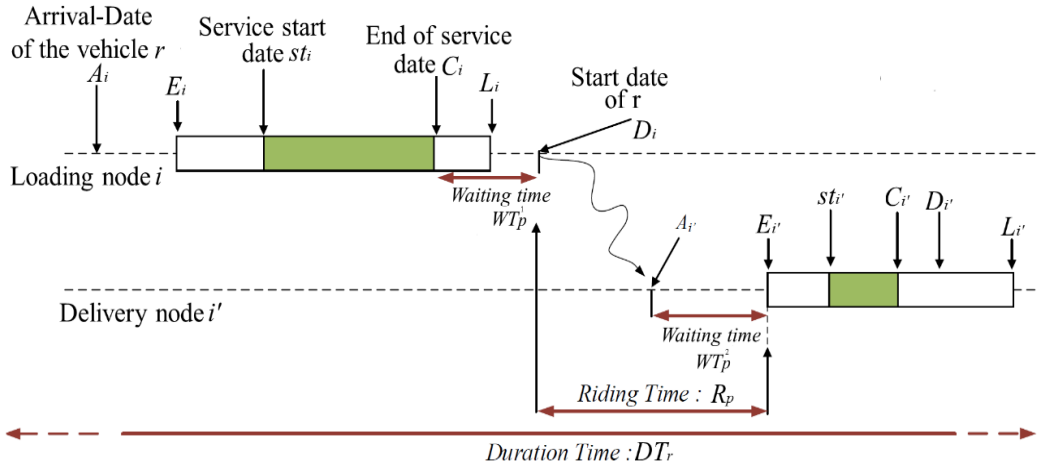

Fig. 1. Description of a production tour.

A set of routes that satisfies all demands while taking into account the time windows on the one hand and the vehicle maximum capacity on the other is known as a DARP solution. In addition, measurements of a solution's QoS take into account the following factors:

The Overall Riding Time: $ORT(x) = \sum_{p=1}^{k} R_p$.
The Overall Duration: $OD(x) = \sum_{i=1}^{m} D_i$, where m is the round number.
The Overall Waiting Time: :$OWT(x) = \sum_{j=1}^{2k} WT_j$.

Given the three criteria mentioned above, a DARP solution is not semi-active, i.e. the start dates of the various operations (departure, start of service, etc.) are not necessarily planned earlier. For given rounds, the start dates minimizing the various criteria can be calculated by the algorithm on$O(k^2)$ of Cordeau and Laporte (2003), or on$O(k)$ by the algorithm of Firat and Woeginger (2011). These two algorithms do not provide the same dates, but a different compromise between these criteria. Iteratively minimizing the criteria preserves the previously minimized criteria in the algorithm of Cordeau and Laporte (2003), which is frequently employed in heuristic approaches.

3. Graphical Modeling of JSSPR with QoS Consideration

JSSPR involves resolving the JSSP with an explicit modeling of the transport and a QoS criterion associated with the transport. To solve this problem, an original approach based on approaches from vehicle routing problems and approaches from scheduling problems is proposed. This approach includes the consideration of a QoS requirement in the objective function and explicit transport modeling.

3.1 Graphical Modeling

The authors of Hurink and Knust (2005) and Lacomme et al. (2013) added nodes to the connective-disjunctive graph of the JSSP to generalize it and describe the loading and transportation operations of the vehicles. Thus, in the disjunctive graph there are three types of vertices that model:*(i)* processing operations ($PO_{i,j}$);*(ii)* loading (pickup)operations $LO_{i,j}$; *(iii)* delivery operations $DO_{i,j}$, Hurink and Knust (2005).

The triplet $(DO_{i,j}, PO_{i,j}, LO_{i,j})$<represented by Fig. 2, defines a set of three operations successively representing a delivery (or unloading) operation, a machine operation, and a pick-up operation. loading. In general, the start date of a machine-operation $PO_{i,j}$depends on the end date of the delivery operation $DO_{i,j}$(the delivery/unloading time is assumed to be zero or included in the delivery time). The start date of a loading operation $PO_{i,j}$ mustbe greater than the end date of the operation $PO_{i,j}$.

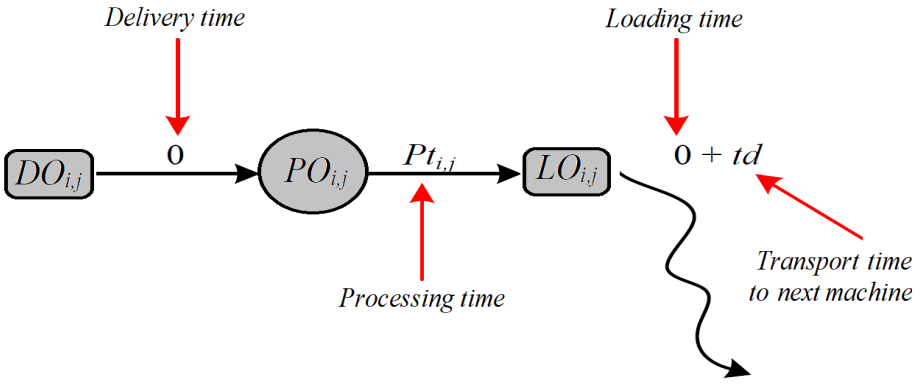

Fig. 2. Connective arcs defining the passage of a job on a machine.

Therefore, each job $j$with $n_j$its sequence contains machine-operations $3 \times n_j - 2$nodes: n_j machine-operations, $n_j - 1$ delivery operations. The job order must include the depot if jobs are to be moved from the starting depot to the first machine in their order.

Each pair of machine-operations $(PO_{i,j}, PO_{i,j+1})$ is a request for a transfer $TR_{(PO_{i,j},PO_{i,j+1})}$of the job $i$which needs to be moved away from the machine $\mu_{i,j}$ at the machine$\mu_{i,j+1}$. Each transfer request $TR_{(PO_{i,j},PO_{i,j+1})} = (LO_{i,j}, DO_{i,j+1})$ to machine $\mu_{i,j}$ at the machine $\mu_{i,j+1}$ is defined as a loading operation $LO_{i,j}$ and by a transportation operation $DO_{i,j+1}$. This loading operation $LO_{i,j}$and this delivery operation$DO_{i,j+1}$ are characterized by the vehicle ϑ which is assigned to them and by their respective start dates. Atransfer request is fully determined by an ordered sequence of operations realizing a path from $\mu_{i,j}$ to $\mu_{i,j+1}$,in which all the operations (loading and delivery) belonging to the path are assigned to the same vehicleϑ. The period of the transfer is determined by the time between the start date of the loading operation $LO_{i,j}$and the start date of the transportation operation $DO_{i,j+1}$. A disjunction between two operations of the loading and/or delivery type corresponds to a transport operation and is modeled by an arc from the vertex which models the loading operation to the machine $\mu_{i,j+1}$, at the vertex which models the delivery to the machine $\mu_{k,p}$, having the value $td^{r,c}_{\mu_{i,j},\mu_{k,p}}$. This remark is valid for all setups $(LO, DO), (LO, LO), (DO, DO), (DO, LO)$, where LO is a loading operation and DO a delivery operation. A vehicle's route is made up of an organized list of loading operations and transportation operations according to the mentioned restrictions: 1) A delivery operation $DO_{i,j+1}$ must be sequenced after the loading operation $LO_{i,j}$; 2) at all times, the number of jobs in the vehicle must be less than the capacity of the vehicle. As in Fig. 3, the route is$tr = (LO_{i,j}, LO_{k,p}, DO_{i,j+1}, DO_{k,p+1})$, for a capacity vehicle equals 2. Both constraints are checked.

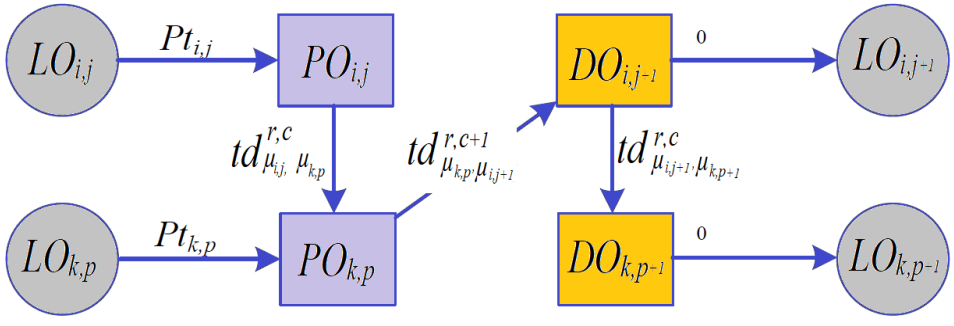

Fig. 3. Loading and delivery operations with a non-unit capacity vehicle.

3.2 Solution and evaluation

A directed disjunctive graph is obtained: *1)* by assigning a vehicle to each loading and delivery operation; *2)* by defining the disjunctions between machine-operations sharing the same machine; and *3)* by settling conflicts between operations that are given the same vehicle. Considering that only semi-active solutions are typically sought, an acyclic network can be evaluated to determine the earliest start dates of operations. This evaluation is commonly carried out by the longest path algorithm. Fig. 4 presents an evaluated and directed disjunctive graph. The graph in Fig. 4 illustrates a schedule by defining the start dates of the machine operations and by defining the rounds of the two vehicles (of capacity two each), these are made up of an organized sequence of loading and transportation operations. For example, the machine-operation $PO_{1,2}$is carried out from date 30 to date 35 (Fig. 4). This last date is not explicitly modeled and does not appear in the graph, it results from the sum of the start date which is 30, and of the duration of the operation which is 5. The date (at the earliest) of the loading operation $LO_{1,2}$is55, this date defines the moment when the vehicle R_2 loads the job J_1 to be transported from its current position (on the machine (M_2) to the next machine in the range of J_1, that is, the machine M_3. Between dates 35 and55, the coin occupies the machine's output bufferM_2, which is assumed to be of unlimited capacity.

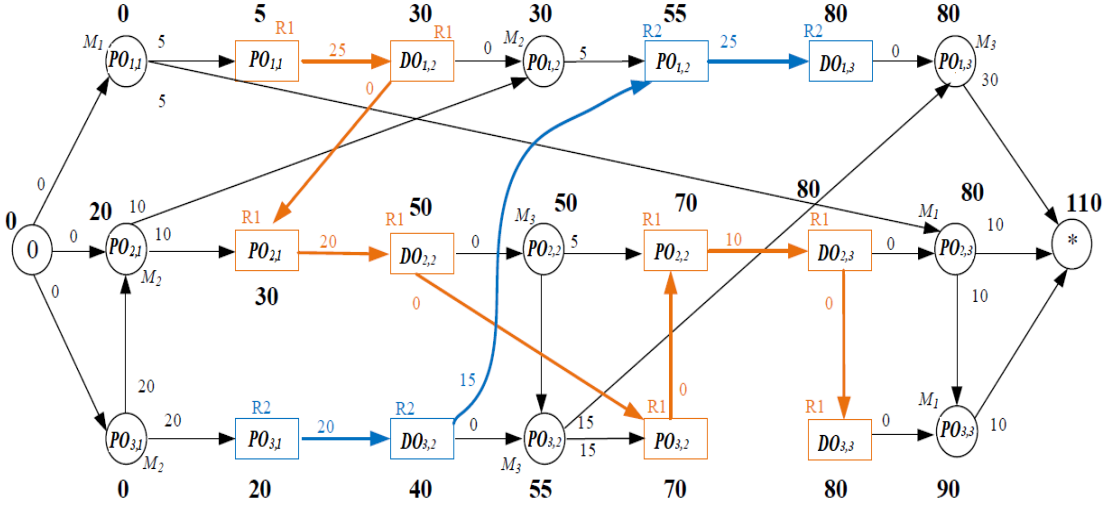

Fig. 4. Evaluated solution of a problem with a directed disjunctive graph.

The vehicle $R_1$is assigned to the transport of the job J_1 from the machine $M_1$to the machine M_2 (the transport time is 25 units of time) between the machine-operations $PO_{1.1}$and $PO_{1.2}$. The vehicle begins the loading operation on date 5 and the transportation operation on date 30. The following operation is allocated to the vehicleR_1comprised in transporting job $J_2$from machine M_2 to machine M_3. Between the delivery operation$DO_{1,2}$ (unloading of the job J_1 on the machine M_2 on the date 30) and the operation of loading $LO_{2,1}$of the job J_2 on M_2(on the date 30), the vehicle R_1 performs an empty transport between the machines M_2 and M_2 with a travel time of 0, this transport is modeled by an arc of zero value and in reality corresponds to waiting for the vehicle on the same machine. Between the delivery operation $DO_{3,2}$and the loading operation $LO_{1,2}$of the job J_1, the vehicle R_1performs an empty transport from M_3 to M_2it is represented by a 15-valued arc (15 is the transport time from M_3 to M_2). The vehicle R_2 is assigned to the transport of J_3 from M_2 to M_3 and then to the transport of $J_1$from M_2 to M_3 (cf. Fig. 4). Fig. 5 explains the Gantt graph of the directed and rated disjunctive graph shown in Fig. 4.

Fig. 5 highlights vehicle and job wait times. For example, the vehicle R_1 performs the delivery operation $DO_{2,2}$ of the job J_2 on the machine M_3 on date 50 then, it waits until date 50 to perform the loading operation of the job J_3on the same machine M_3, therefore the vehicle R_1 waits 20 units of time.

Similarly, the job J_3 is delivered on date 40 on the machine M_3 by the vehicle R_2, and it waits until date 55 in the input buffer of the machine M_3 before starting its machine-operation $PO_{1,2}$. Such a situation is induced by the order of passage of the jobs on the machine M_3 (this order is J_2,J_3,J_1), which is itself the result of the choice of arbitration of the machine disjunctions.

Some jobs may have no waiting time on some machines, in this case: *i)* The machine-operation is set to begin afterwards the delivery date; *ii)* the end date of the machine-operation corresponds to the start date of the loading operation. For example, the job J_2 finishes its first processing on date 30 on the machine M_2, it is charged by the loading operation ($LO_{2,1}$) at

the date 30, then transported from M_2 to $M_3$from the date 30 to the date 50 (thanks to the transport operation$T(DO_{2,1}, DO_{2,2}) = (PO_{2,1}, DO_{2,2}, R_1)$. This same job $J_2$is deposited on date 50 (delivery operation $DO_{2,2}$) on the machine M_3.

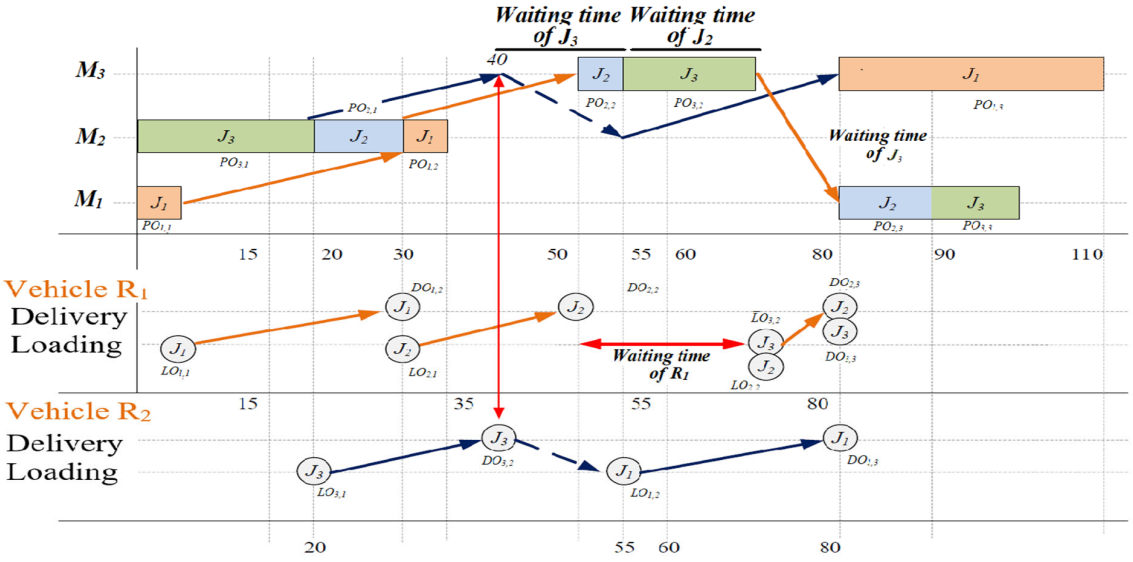

Fig. 5. Gantt graph of a semi-active solution.

3.3 QoS in JSSPR

Each transport operation $T(PO_{i,j}, PO_{y,x}) = (X_{i,j}, Y_{y,x}, r)$of a machine $\mu_{i,j}$ (who performs the machine-operation $PO_{i,j}$)up to another machine $\mu_{y,x}$ (which performs the machine-operation $PO_{y,x}$) is composed of an initial operation $X_{i,j}$ (which is equal to either $DO_{i,j} or LO_{i,j}$), of a final operation $Y_{y,x}$(which is equal to $DO_{y,x} or LO_{y,x}$) and a vehicle r assigned to both operations. Therefore: $T(PO_{i,j}, PO_{y,x}) = (DO_{i,j}, DO_{x,y}, r)|(DO_{i,j}, LO_{x,y}, r)|(LO_{i,j}, LO_{x,y}, r)|(LO_{i,j}, DO_{x,y}, r)$. In addition, a transport operation$T(PO_{i,j}, PO_{y,x})$ is defined by:

the date of departure $B_{i,j}$of the vehicle r of the machine $\mu_{i,j}$;
the date of arrival $A_{y,x}$of the vehicle r on the machine $\mu_{y,x}$;
The vehicle $r$assigned to the transport.

Let $st_{i,j}$be the start date of the machine operation $PO_{i,j}$and let $C_{i,j} = st_{i,j} + Pt_{i,j}$its end date. Since a transport operation $T(PO_{i,j}, PO_{y,x})$ is a pair of loading and delivery operations, $B_{i,j}$ is the departure date of the vehicle (start date of the transportation operation) (see Fig. 6); and$A_{y,x}$ is the arrival date of the vehicle. The earliest start date of a machine-operation depends both on the arrival date of the vehicle transporting the job and on the end date of the machine-operation formerly scheduled on this machine.

The distinction between $st_{y,x}$ and $A_{y,x}$describes the Waiting Time of the job y in the input buffer of the machine $\mu_{y,x}$(Fig. 6). Similarly, the difference between $B_{i,j}$and $C_{i,j}$defines the waiting time of the job i in the output buffer of the machine $\mu_{i,j}$. The waiting time of the vehicle is defined by the difference between the date of arrival $A_{i,j}$ of the vehicle on the machine and the date of departure of the vehicle $B_{i,j}$. There is no constraint which requires that the departure date$B_{i,j}$ of the vehicle be scheduled immediately after the job loading operation, this means that a job can be loaded and wait in the vehicle before it is loaded. it does not begin its movement. Similar remarks concern the delivery operations $DO_{y,x}$ which do not need to be started immediately on the date of the vehicle's arrival $A_{y,x}$, this means that the job might not be allowed to get off the vehicle immediately upon His arrival. These considerations take on their full meaning when the problems of vehicle routing are subject to safety regulations, which are frequent – in particular – in the problems of transporting children or transporting dangerous or precious products. For example, in the case of school pick-up, it may be forbidden to drop off a child at his place of destination before the scheduled date or before the school opens. Similarly, it is forbidden for a deliverer to deposit his goods in front of the depot if it is not yet open. To reduce waiting times (both for vehicles and jobs), a coordinated strategy must be favored concurrently addressing the issues of scheduling machine operations, assignment of vehicles to transport operations, and rounds. of vehicles. For scheduling problems, the start dates of the different operations are very frequently at the earliest ("left-shifted"), because generally the solutions sought are semi-active and the objective function only takes into account the makespan. The latter, also referred to as C_{max}, is the start date of the final fictitious operation,

τ whose start date is equal to the end date of the operation ending at the latest. The makespan therefore corresponds to the time required to perform all the operations: $C_{max} = st_{\tau} - st_0$.

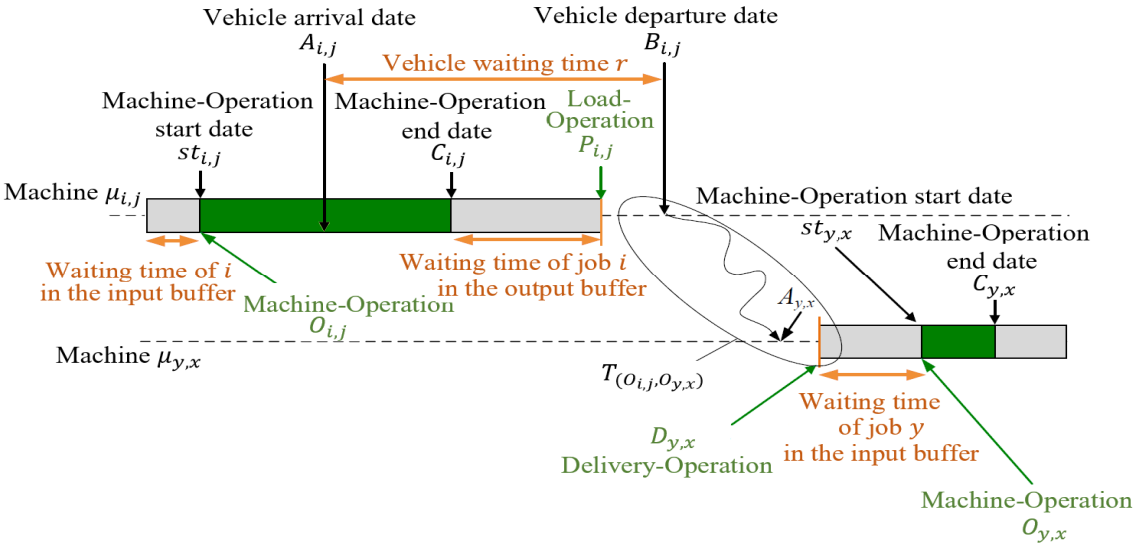

Fig. 6. Synchronization of a transport operation and two machine operations.

The advantage of an explicit modeling of transport operations is to be able to propose start dates that are not at the earliest. Depending on the dates of arrival $A_{i,j}$, depart $B_{i,j}$ and start $st_{i,j}$ of the machine operations, the waiting times for vehicles and jobs are different in terms of their length and their location. It can be beneficial to delay the start date of an operation (transport or machine) to reduce unnecessary waiting times. By analogy with Cordeau and Laporte (2003), it is possible to define the QoS of the Job-Shop Scheduling Problem with Routing by considering the time required to complete a job (Total Duration – TD), the time spent by the part in transport (Total Riding Time – TRT) and the waiting time of the parts in the input and output buffers of the machines (Total Waiting Time – TWT). Fig. 7 is a visualization of these three criteria, machine operations are in grey, loading operations are in orange, and delivery operations are in green. Thus, the Total Duration (TD) corresponds to the classic notion in industrial engineering of "flow time" and it's determined by the distinction between the end date of the last operation of the job and the start of processing of the job. This is the time required for the full range of one piece. The Total Riding Time represents the time spent by the product (or the customer) between the moment it is loaded and the moment it is delivered to the input buffer of the next machine. This duration is visually represented in Fig. 7 (TRT) by a set made up of the following order: a loading node (in orange), a delivery node (in green), and a machine-operation (in Grey). The Waiting Time of a job is the sum of the waiting times of the part in the input and output buffers of the machines. In Figure ٧, these times are between delivery operations (in orange) and machine operations (in gray) and between machine operations and loading operations (in green).

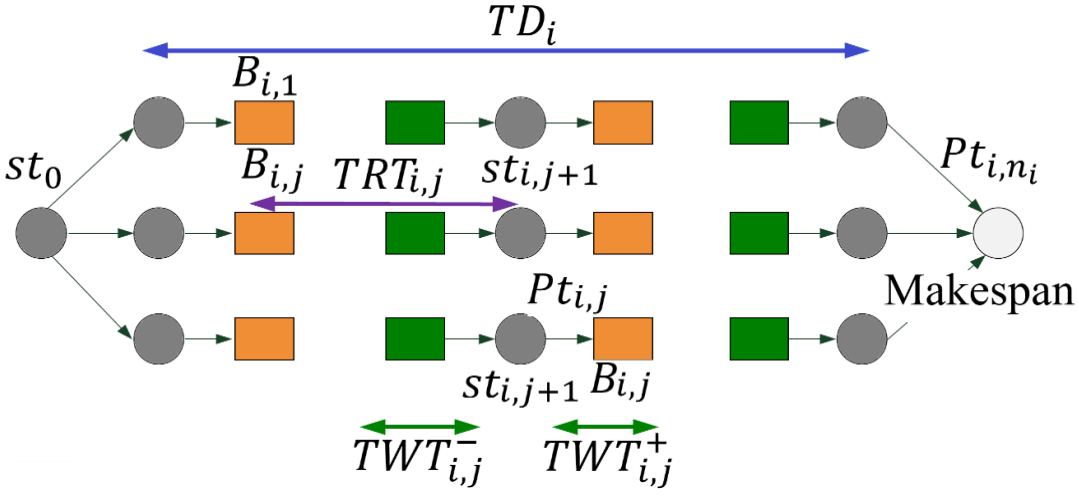

Fig. 7. The new criteria defining the QoS in the JSSPR.

Fig. 8, which is a Gantt diagram, presents the different QoS criteria. The Total Duration of the job J_2 is in blue, it starts at the first operation ofJ_2 on the machine M_2 (operation-machine$PO_{2,1}$) and ends at the end of the last operation of J_2 on the machine M_1 (machine-operation $PO_{2,3}$).

The Riding Time (RT) of the job J_1 (in purple on the Gantt diagram in Fig. 8) is made up of two parts, because there are two transport operations in the route. The first RT of J_1 is between the first operation (machine-operation $PO_{1,1}$) on the machine M_1, scheduled on date 0, and the start of the following machine-operation (machine-operation $PO_{1,2}$) starting on date 30. The second part of RT ofJ_1 begins when$J_1$is loaded into the vehicle, on date 55, and ends when J_1 goes on the machine M_3on date 80.

Two Waiting Times (WT) are illustrated in green, the first concerns the WT of J_3 in the input buffer of the machine M_3 on date 40, the piece waits until date 55 before being produced (operation- machine $PO_{3,2}$). The second illustrated $WT$is located in the output buffer of the machine M_3, the job J_2 ends its passage on M_3on date 55 (machine-operation $PO_{2,2}$) and waits for the loading operation (P) at the date 70. More formally, the following definitions are introduced to define service quality Hurink and Knust (2005), Lacomme et al. (2007).

Total Duration (TD). The Total Duration of a job i (TD_i) is determined by the distinction among the start and end dates of its first and last machine operations (this duration is also called "flow time"). In sequencing problems: $TD_i = st_{i,n_i} + Pt_{i,n_i} - st_{i,1}$. The Total Duration $TD_i$of a job i is shown in Fig. 7. The Total Duration (TD) is the sum of all the TD_i: $TD = \sum_{i=1}^{n}(TD_i) = \sum_{i=1}^{n}\left(st_{i,n_i} + Pt_{i,n_i} - st_{i,1}\right)$.

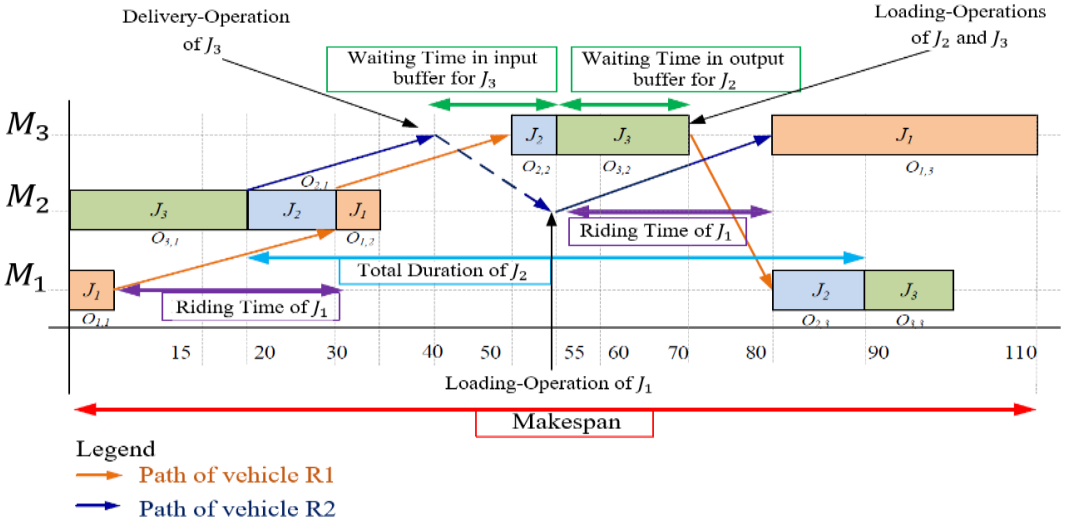

Fig. 9. QoS Criteria in Gantt chart.

Total Riding Time (TRT). A Riding Time is associated with a request $R_{(O_{i,j},O_{i,j+1})}$ and is calculated by $TRT_{i,j} = st_{i,j+1} - B_{i,j}$. It therefore relates to the time elapsing among the date of leaving the vehicle r from the machine $\mu_{i,j}$ where the loaded of job i and the start date of the service of the job i on the machine $\mu_{i,j+1}$ (the date of start of the machine-operation $O_{i,j+1}$), as Fig. 7 shows. The Total Riding Time (TRT) of job i is $TRT_i = \sum_{j\in L_i}\left(st_{i,j+1} - B_{i,j}\right)$, with L_i is the set of loading operations of the job i. The Total Riding Time is the totality of all the Riding Times of all the jobs: $TRT = \sum_{i=1}^{n}(TRT_i)$.

Total Waiting Time (TRT). The Waiting Time of a job i is stored in a machine's output buffer$\mu_{i,j}$ is named $TWT_{i,j}^{+}$, defined by $B_{i,j} - (st_{i,j} + Pt_{i,j})$. The Waiting Time of a job$i$ in the input buffer of a machine $\mu_{i,j}$ is named$TWT_{i,j}^{-}$ and is defined by$st_{i,j} - A_{i,j}$ (see Fig. 7). The Total Waiting Time (TWT) is the total of the Waiting Times in the output buffers $\sum_{i=1}^{n}\sum_{j\in L_i} TWT_{i,j}^{+}$more the Waiting Times during buffer input$\sum_{i=1}^{n}\sum_{j\in U_i} TWT_{i,j}^{-}$where $U_i$is the set of transportation (delivery) operations of job i and L_i is the set of loading operations of job i. The total Waiting Time is the totality of all the Waiting Times:$TWT = \sum_{i=1}^{n}\sum_{j\in L_i} TWT_{i,j}^{+} + \sum_{i=1}^{n}\sum_{j\in U_i} TWT_{i,j}^{-}$

As suggested by Cordeau and Laporte (2003) for the DARP, it is possible to define a strict hierarchical objective function$F$ for the JSSPR whose initial step is to reduce the makespan and then secondly decrease the TD, followed by the TRT and finally the TWT: $F = \alpha.C_{max} + \beta.TD + \gamma.TRT + \delta.TWT$, With $(\alpha,\beta,\gamma,\delta)$respects the following constraints:
$Min(\alpha.C_{max}) > Max(\beta.TD + \gamma.TRT + \delta.TWT)$;
$Min(\beta.TD)) > Max(\gamma.TRT)$;

$Min(\gamma.TRT) > Max(\delta.TWT)$.

3.4 Principle of the Time-Lag Heuristic assessment

It is impossible to directly apply the approach of Cordeau to the routes present in a disjunctive graph for the JSSPR because the routes are not independent of each other. The algorithm of Cordeau is dedicated to the evaluation of a unique and independent tour. For the JSSPR, it is necessary to design an evaluation procedure that simultaneously takes into account the whole of the graph which contains all the routes. The dependence between routes is linked to the fact that a vehicle performs transport operations belonging to different jobs, which adds temporal dependencies between job ranges. This particular point is illustrated in Fig. 9, where the vehicle R_1, whose route is modeled by the arcs in orange, performs, for example, a transport operation for the job J_1, then for the job J_2.

Algorithm 1: Principle of criterion minimization at step c

1. G: the graph with minimized criterion
2. st: the start dates of operations
3. c: the criterion to be minimized
4. φ_{TL}^{c}: maximum time-lags to insert during the c step
5. θ_{TL}^{c}: order of insertion of the maximum time-lags during the c step
6. **Start**
7. | **For**($\forall \theta \in \varphi_{TL}^{c}$according to the order of insertionθ_{TL}^{c}**do**
8. ||Insertion_time_lag(G, st, v_{θ}^{-}) // see algorithm 3
9. | **End for**
10. | return $TL_{j,i}$
11. **End**

The new evaluation function, proposed in this paper, TLH – Time-Lag Insertion Heuristic – is based on the same principle and is based on the heuristic insertion of time-lags in the disjunctive graph of the JSSPT. The evaluation function TLH minimizes the makespan – by calculating the earliest start dates of the operations – then TLH minimizes the Total Duration (TD), then the Total Riding Time (TRT) and finally the Total Waiting Time (TWT). The deterioration of a previously optimized criterion is prevented by the iterative addition of maximum time-lags in the graph. The TLH evaluation function is composed of four tasks:

Task 1: minimizing the makespan.
Task 2: minimizing the TD.
Task 3: minimizing the TRT.
Task4: minimizing the TWT.

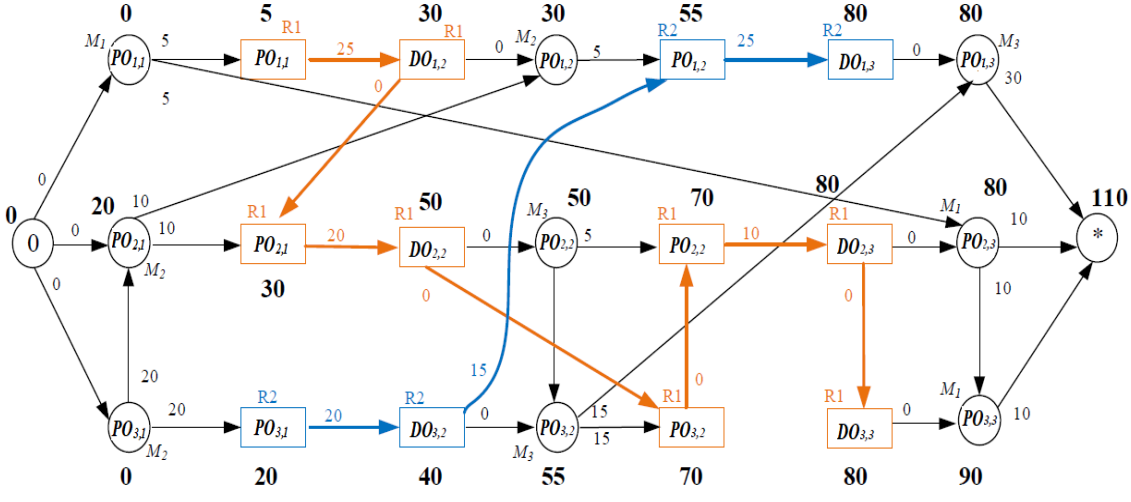

Fig. 9. Routes dependency.

The stages of the TLH evaluation range from the most encompassing time-lags to the "smallest" time-lags, because inserting the most encompassing time-lags makes it possible to constrain future time-lags. Moreover, inserting time-lags at the start of the procedure that includes few operations risks creates an increase in the other criteria, because these will not be constrained. For example, the reduction of the Riding Times can cause an increase in the Total Durations of the jobs, indeed, to minimize the Riding Times, the jobs will be loaded only when the following machine is free and will be carried out without waiting. In this case, transport becomes the most critical resource.

Selection of the maximum time-lag value is an important point of TLH evaluation and a challenging issue because the time-lag value should be minimum, but adding too small value time-lag in the disjunctive graph can lead to a cyclic graph. This situation is to be avoided.

The minimum value $SP_{i,j}$of a maximum time-lag $TL_{j,i}$ (the time-lag is represented by an arc with a negative value even though this value is positive), from the operation $j$to the operation $i$is the difference between the earliest start date LS_i of i and the earliest start date ES_j of j.

Let be a position of insertion of a maximum time-lag $TL_{j,i}$ from $j$to$i$, the minimum value (which is equal to $SPM_{i,j}$)of maximum time-lag, is $L_{j,i} = ES_j - LS_i$ where LS_i and ES_j verify that:

1) $LS_* = ES_* \; and \; LS_j = ES_j$
2) $\forall k \; LS_k = \min\limits_{u \in Succ(k)} (LS_u - l_{u,k})$

such that $Succ(k)$ is the set of successors of k, and $l_{u,k}$is the path length between u and k. With $l_{u,k} = p_{t,k} || \; l_{u,k} = TL_{k,u}$ From the definition of the earliest and latest dates of operations, the proposition can be inferred immediately. In general, among two operations i and j of the same route, whatever their types (loading, delivery, or machine operation), one or more paths are going from i to j. These paths are composed of a succession of arcs whose extremities do not necessarily belong to the same range as i and j.To avoid the creation of a cycle in the graph by inserting a maximum time-lag between i and j, the longest path must be found. The length of this path is the minimum value that the time lag between these two operations can take to avoid creating a cycle. Algorithm 2 is used to calculate the minimum value of a time-lag.

Algorithm 3 exemplifies the principle of inserting a maximum time-lag $TL_{i,j}$ in the graph$G$. The time-lag is initialized to its initial value $v_{j,i}$, then the graph is evaluated using a Bellman-Ford type least-distance algorithm. When a cycle is found, the cycle duration value δ is positive $(\delta > 0)$. The time-lag value is elevated byδ, until there are no more cycles detected.

Algorithm 2: Calculation of the minimum time-lag

1. $TL_{j,i}$: the value of time-lag from j to i
2. ES: the earliest start dates of operations
3. G: the graph
4. $i \; and \; j$: two operations linked by a maximum time-lag from j to i
5. **Start**
6. | $LS_* = ES_*$ // the final fictitious operation
7. |$LS_j = ES_j$
8. **For** $(\forall k \in W)$**do**
9. || $l_{u,k} = p_{t,k} || l_{u,k} = TL_{k,u}$
10. ||$LS_k = \min\limits_{u \in Succ(k)} (LS_u - l_{u,k})$
11. | **End for**
12. |$TL_{i,j} = ES_j - LS_i$
13. | return $TL_{j,i}$
14. **End**

The initial minimum value $v_{j,i}^-$ when minimizing the Total Duration, of the maximum time-lag of a job J_i is the total of all operations$O_{i,j} \; \forall j \in \{1, \dots, n_i\}$aboutit, plus the transport times between the machines where the operations $O_{i,j} \; \forall j \in \{1, \dots, n_i\}$ are carried out.

Algorithm 3: Time-lag insertion

1. G: the graph with inserted time-lag
2. $v_{j,i}$: the value of time-lag
3. $TL_{i,j}$: the time-lag to be inserted between $i \; and \; j$
4. **Start**
5. | $LS_* = ES_*$ // the final fictitious operation
6. |$LS_j = ES_j$
7. **For** $(\forall k \in W)$**do**
8. || $l_{u,k} = p_{t,k} || l_{u,k} = TL_{k,u}$
9. ||$LS_k = \min\limits_{u \in Succ(k)} (LS_u - l_{u,k})$
10. | **End for**
11. |$TL_{i,j} = ES_j - LS_i$
12. | return $TL_{j,i}$
13. **End**

4. **Experimentations and Results**

4.1 Presentations of instances and studies

To our information, there isn't a single instance accessible that is committed to the Job-shop Scheduling Issue with Routing. On the instances originally established for the simultaneous scheduling of machine operations and transport operations, Zeng et al. (2015), a new batch of instances is provided. The scientific community generally recognizes these examples as the JSSPT reference dataset for two-unit capacity vehicles.

The instances in our study are made of two sets (called datasets) $D_1 and D_2$: the first involves instances with a ratio of the transport time t_{ik} and the processing time p_k greater than $0.25 (t_{ik}/p_k > 0.25)$ which means that the processing times are much longer in duration than the transport time. These are predominantly scheduling instances. The second set (or dataset) contains the instances with $t_{ik}/p_k \leq 0.25$. The instances of the second dataset are therefore transport-dominant. These instances are all produced by integrating 10 jobsets and 4 layouts and defining the dataset D_1 with 40 instances and the dataset D_2 with 42 instances.

Instances of D_1 are denoted EX_{xy}, with x denotes the jobset currently used and y the layout taken into account. For the set D_2, the instances are named EX_{xyz}, where z is an additional digit with a value of 0 if the transport times are divided into two parts and the processing times doubled or a value of 1 if transport times are halved and processing times tripled.

For the occurrences of Zeng et al. (2015), the transit times for the two vehicles are the same regardless of the load and the vehicle. For numerical assessments including a group of non-unit capacity vehicles, these situations are extended. In this instance, two vehicles are taken into account, each of which has a capacity of two jobs, each of which corresponds to the same volume. The job operating ranges and processing times are the same as in the original instances. JSSPR's objective is to shorten the makespan and increase QoS by defining $h_{Qos}(y) = 1/h_{cost}(y)$. The two criteria are ordered by an aggregate sum with the coefficients $(\alpha; \beta)$ modeling the level of necessity of QoS and makespan. The objective function is described as follows:

$$F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$$

where:

y: a solution to the problem;
$h_{Cmax}(y) = Cmax$ of the solution y, i.e. the makespan of y;
$h_{cost}(y) = TD(y) + TRT(y) + TWT(y)$, which are respectively the Total Duration, the Total Riding Time, and the Total Waiting Time of the solution y;
$h_{Qos}(y) = 1/h_{cost}(y)$ is the QoS of y.
Giventhe metaheuristic used and the objective function $F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$, GRASP×ELS , Gondran et al. (2017) can solve the JSSPT and the JSSPR. It should be noted that GRASP×ELS is a combination of two metaheuristics, GRASP (Greedy Randomized Adaptive Search Procedure) (Prins, 2009) and ELS (Evolutionary Local Search), which has been cited as Gondran et al. (2017) for the Vehicle Routing Problem (VRP).

The objective function for the JSSPT resolution is specified with $\alpha = 1 and \beta = 0$ corresponding to an evaluation of the dates at the earliest. The coded algorithm is generic and in this specific case, it simply corresponds to the initial stage of the TLH evaluation and is therefore equivalent to a classical shortest path algorithm.

Two different approaches are experimented with to solve the JSSPR. The first approach is an integrated approach, while the second is a sequential approach: The objective function of the integrated approach (integrated JSSPR) is $F(y) = \alpha \times h_{Cmax}(y) + \beta \times h_{cost}(y)$ where with $\alpha = 10\,000 \; and \; \beta = 0$, the TLH evaluation is used at each iteration of the metaheuristic.

For the sequential approach (sequential JSSPR), $\alpha = 1 and \beta = 0$ in the objective function during the GRASP×ELS (this step is equivalent to solving the JSSPT). Then, to obtain a solution from the JSSPR, the TLH assessment is applied to the best solution discovered during the resolution of the JSSPT.
Our numerical assessments are divided into two studies. The first study concerns the ability of GRASP×ELS to provide results comparable to the literature for solving the JSSPT with a fleet of unit-capacity vehicles. To our knowledge, there is no article in a journal proposing numerical results for the JSSPT with a fleet of non-unit capacity vehicles or proposing a study on QoS. The second study is composed of two numerical evaluations: the first deals with the order of insertion of the maximum time-lags during the TLH evaluation; and the second numerical evaluation looks at the TLH function's speed at finding high-quality answers in reasonable amounts of time Lacomme et al. (2007). All of the investigations are conducted under identical circumstances. A GRASP×ELS includes the TLH assessment. The ELS iterations total 60, the GRASP iterations total 200, and the number of neighbors during the ELS iterations total 30. To lessen the impact of chance on the result, each experiment is replicated five times using a different set of random numbers each time.

4.2 First study: Capacity of GRASP×ELS to solve the JSSPT with a fleet of unit capacity vehicles

The study focuses on the last three publications of the JSSPT with a fleet of vehicles (group of vehicles) of unit capacity, Gondran et al. (2017), Gondran et al. (2018), He and Hao (2023). These three methods offer many new best solutions and are tested on all instances of the two datasets.

Table. 1
MTSP, DSMG and Petri-EUF methods

Method		DatasetD_1		DatasetD_2	
		Best	Time	Best	Time
Gondran et al. (2017)	MTSP	✓	✓	×	×
Gondran et al. (2018)	DSMG	✓	✓	×	×
He and Hao (2023)	Petri-EUF	✓	×	✓	×

✓Information is provided
× No provided information

Recall that the MTSP method is introduced by Gondran et al. (2018) and consists of a Tabu Search type algorithm (MTSP). In Gondran et al. (2018), the authors describe an approach based on genetic algorithms: the DSMG to enable flexible processing and dynamic scheduling, additionally resolving a static version of the JSSPT. The Petri-EUF method is suggested by He and Hao (2023). For each method, the results in terms of solutions and computation time provided in the articles are summarized in Table 1.

JSSPT, which is determined by an objective function only based on the makespan lacking consideration of the QoS, can be solved using the proposed Time-Lag Insertion Heuristic (TLH) assessment because it is generic. In this case, the objective function is outlined by $\alpha = 1\ and \beta = 0$in the TLH evaluation. This is related to the longest path algorithm of the Dijkstra type.

Table 2 and Table 3 present the results acquired by GRASP×ELS solving the JSSPT with a fleet of unit capacity vehicles for the two datasets. The mean deviation$\overline{gap_k^{z,*}(y)}$

Attribute of the best performance is 26.8% for datasetD_1 (Table 2) and 6.5% for the dataset D_2 (Table 3).

For the dataset D_1 (Table 2), the GRASP×ELS participates with the most successful methods, with a deviation from the lower bound of $\overline{gap_k^{z,*}(y)} = 26.8\%$ which is more favorable than HRA and FMAS. The GATS+HM offers a deviation of around 20.6% which is slightly lower than the departure from GRASP×ELS with a calculation time of around 12.08 seconds, i.e. four times longer than the calculation time of GRASP×ELS, which is about $2.83\ seconds$. Likewise, the ALS offers a departure from about 25.5%, but a scale calculation time of $503.1\ seconds$ on average. The TS_SPMA obtains results of about25.5%, so a relative analysis of computing time is not sufficient.

Table 2
GRASP×ELS to solve the JSSPT for datasets $\boldsymbol{D_1}$

	Dataset $\boldsymbol{D_1}$: $\boldsymbol{t_{ik}/p_k > 0.25}$					
	$\overline{gap_k^{z,*}(y)}$	$\overline{st^{z,*}(y)}$	$\overline{stt^{z,*}(y)}$	$\overline{gap_k^{z,n}(y)}$	$\overline{st^{z,n}(y)}$	$\overline{stt^{z,n}(y)}$
MTSP	44.9 %	×	×	×	×	×
DSMG	25.5 %	503.1 s	3600	×	×	×
Petri-EUF	20.6 %	12.08 s	×	×	×	×
GRASP×ELS	26.8 %	2.83 s	10.50 s	27.4 %	2.10 s	10.51 s

× No provided information
$\overline{st^{z,*}(y)}$:The average normalized time to identify the ideal solution$y$, when it was the best replication $*$ among the n replications, for all instances of the dataset$z \in \{D_1\ ;\ D_2\}$
$\overline{stt^{z,*}(y)}$: The normalized total time to resolution for the best replication$*$among the n replications, for all instances of the dataset $z \in \{D_1\ ;\ D_2\}$
$\overline{st^{z,n}(y)}$: The total normalized time to identify the ideal solution during $n$replications, for all instances of the dataset$z \in \{D_1\ ;\ D_2\}$
$\overline{stt^{z,n}(y)}$: The total normalized resolution time for $n$replications, for all instances of the dataset$z \in \{D_1\ ;\ D_2\}$
$\overline{gap_k^{z,n}(y)}$: The average deviation for $n$replications for all instances of the dataset$z \in \{D_1\ ;\ D_2\}$

Mostly asked, is it possible to increase the number of iterations in the hopes of reducing the difference in solutions between GRASP×ELS and GATS+HM.

So many numerical evaluations were carried out to give GRASP×ELS a similar duration as the GATS+HM algorithm, but no significant improvement was obtained. There is therefore probably a very and too rapid convergence of GRASP×ELS and the diversification mechanisms have not allowed the exploration of another region of space. For the dataset D_2 (Table 3), GRASP×ELS offers high-quality results with an average deviation of 6.5%, which places it between the TS_SPMA method (6%) and the FMAS method (7.2%). But no comparative study is possible on the computation times, because none of the methods gives their execution times.

Table 3
GRASP×ELS to solve the JSSPT for datasets $\boldsymbol{D_2}$

	Dataset D_2: $t_{ik}/p_k > 0.25$					
	$\overline{gap_k^{z*}(y)}$	$\overline{st^{z*}(y)}$	$\overline{stt^{z*}(y)}$	$\overline{gap_k^{zn}(y)}$	$\overline{st^{zn}(y)}$	$\overline{stt^{zn}(y)}$
MTSP	6.0% %	×	×	×	×	×
DSMG	7.2 %	×	×	×	×	×
Petri-EUF	11.3 %	×	×	×	×	×
GRASP×ELS	6.5 %	0.15 s	8.01 s	7.0 %	1.60 s	8.02 s

This first numerical study leads us to consider that GRASP×ELS is adapted and well-parameterized to solve the JSSPT with two vehicles of unit capacity.

4.3 Second study: JSSPT and integrated JSSPR

The second study is a comparison of the makespan and QoS criteria between the JSSPT and the integrated JSSPR. The resolution of the JSSPT does not consider the QoS evaluation and corresponds to a *Dijkstra*-type evaluation. To compare the resolution methods, the QoS of the JSSPT is calculated on the final solution returned by the GRASP×ELS. It is important to note that in the case of the JSSPT, the QoS is only evaluated without trying to maximize it: the start dates of operations are constraints and cannot be modified. This study is divided into two sections: a section for the case of a fleet of unit capacity vehicles, and a section for the case of a fleet of vehicles of non-unit capacity.

4.3.1 Unit Capacity Vehicles

Table ٤ and Table ٥present the results obtained for the JSSPT and the JSSPR. The first column is the name of the instance, the second is the lower bound proposed by Ulusoy et al. (1997). Columns 3, 4, 5, and 6 relate to the resolution of the JSSPT by the GRASP×ELS with the objective function: $F = h_{\mathrm{Cmax}}(y)$ where $h_{\mathrm{Cmax}}(y)$is the makespan. Columns 7 to 12 relate to the resolution of the built-in JSSPR (the objective function is: $F(x) = 10000 \times h_{cmax}(x) + 1 \times h_{cost}(x)$.

Columns 3 and 7 give the value of the makespan of the best solution found, during the five replications, respectively by the JSSPT and the integrated JSSPR. Columns 4 and 9 are the values of the cost of the best solution found, during the five replications, respectively by the JSSPT and the integrated JSSPR.

A value of $h_{cmax}^{i,*}(y)$ (column 3) equal to $h_{cmax}^{i,*}(x)$(column 7) means that the best solution obtained for the JSSPT and the best solution obtained for the built-in JSSPR have an equivalent makespan.

A deviation of *0.16%* means that the average makespan found during the solving of the integrated JSSPR is better than the average makespan found during the solving of the JSSPT. This difference results from a better diversification during the GRASP×ELS. A deviation of 0% means that the two methods have solutions with an equivalent makespan.

$st^{i,*}$ is the time taken by GRASP×ELS to locate the ideal solution, while $stt^{i,*}$ is the total running time of GRASP×ELS. For example, for instance, EX21 (Table 4), the period to locate the ideal solution for the JSSPT is 1.94 seconds, and the total time to solve the JSSPT is 8.72 seconds. The time required by GRASP×ELS to provide the best solution is 11.40 seconds and the total time is 19.93 seconds.

The difference in execution times is explained by the fact that the number of evaluations of the disjunctive graph is much greater for the JSSPR. Moreover, for the JSSPT, the time $st^{i,*}$ corresponds to the first solution obtained with the minimal makespan, while for the JSSPR, the solution with the minimal makespan and the minimal QoS can require much more time.

The gap between the QoS of the ideal solution found during the JSSPT and the ideal response found during the integrated JSSPR is defined by$\Delta_{cost}^{i,*} = \frac{h_{cost}^{i,*}(y) - h_{cost}^{i,*}(x)}{h_{cost}^{i,*}(y)}$which therefore represents the improvement in the QoS of the JSSPR compared to that of the JSSPT. A $\Delta_{cost}^{i,*} > 0$ means that the JSSPR solution has a better QoS than the JSSPT.This is the case of instance EX11 (Table ٤) with $\Delta_{cost}^{i,*} = 25.33\%$: the JSSPT solution has the criterion $h_{cost}^{i,*}(y) = 683$ and the solution of the integrated JSSPR has the value of $h_{cost}^{i,*}(x) = 510$. $\overline{\Delta_{cost}^{i,*}}$ is the average deviation of $\Delta_{cost}^{i,*}$,$\overline{\Delta_{cost}^{i,*}} = 35.78\%$*f*or the dataset$D_1$(Table 4) and $\overline{\Delta_{cost}^{i,*}} = 46.37\%$ for dataset D_2(Table 5). These gaps highlight the importance of considering the QoS during the resolution scheme.

For the dataset D_1(Table 4), the average amount of time needed to locate the ideal solution of the JSSPT is 2.83 seconds and the total time is 10.50 seconds. The average duration of the JSSPR is 9.38 seconds to locate the ideal solution and 23.14 seconds for the complete resolution. The differences between these durations are explained by the number of evaluations of the graph which is much more important for the JSSPR. The same remarks remain true for the dataset D_2 (Table 5).

Table 4
Results for the JSSPT and the JSSPR with datasets $\boldsymbol{D_1}$ (For unit capacity vehicles)

Ulusoy et al. (1997)		**JSSPT resolution:** $\boldsymbol{F = h_{\text{cmax}}(y)}$				**Resolution of integrated JSSPR:** $F(x) = 10000 \times h_{\text{Cmax}}(x) + h_{\text{cost}}(x)$.					
Instance-name	*Lower-bound*	$h^{i*}_{cmax}(y)$	$h^{i*}_{cost}(y)$	st^{i*}	stt^{i*}	$h^{i*}_{cmax}(x)$	Δ^{i*}_{cmax}	$h^{i*}_{cost}(x)$	Δ^{i*}_{cost}	st^{i*}	stt^{i*}
EX11	96	683	0,01	6,8	96	0,00	510	25,33	5,05	15,9	96
EX21	100	799	1,94	8,7	100	0,00	529	33,79	11,40	19,9	100
EX31	103	858	5,21	9,0	99	3,88	543	36,71	4,28	20,3	103
EX41	112	831	3,52	11,37	112	0,00	673	19,01	4,80	23,7	112
EX51	87	641	0,02	6,5	87	0,00	441	31,20	4,09	16,1	87
EX61	118	901	1,73	12,73	118	0,00	582	35,41	0,08	24,4	118
EX71	116	1387	4,46	13,50	115	0,86	882	36,41	10,59	28,1	116
EX81	161	1338	0,03	13,77	161	0,00	597	55,38	22,35	30,2	161
EX91	116	807	1,43	10,52	116	0,00	589	27,01	3,87	21,9	116
EX101	148	1144	3,93	14,97	149	0,68	788	31,12	9,98	28,9	148
EX12	82	520	0,08	5,8	82	0,00	370	28,85	0,12	15,8	82
EX22	76	553	1,93	7,8	76	0,00	373	32,55	3,95	18,5	76
EX32	85	663	0,00	7,4	85	0,00	407	38,61	8,84	19,4	85
EX42	87	646	3,61	10,82	87	0,00	532	17,65	14,02	22,7	87
EX52	69	460	0,01	6,8	69	0,00	338	26,52	5,78	15,9	69
EX62	98	725	0,15	11,70	98	0,00	424	41,52	4,96	23,3	98
EX72	84	894	0,93	13,00	84	0,00	532	40,49	26,31	27,1	84
EX82	151	1206	0,16	12,75	151	0,00	408	66,17	5,72	38,9	151
EX92	102	657	1,81	10,21	102	0,00	496	24,51	16,89	21,0	102
EX102	135	1049	0,16	14,58	136	0,74	695	33,75	5,78	28,5	135
EX13	84	583	0,02	5,9	84	0,00	412	29,33	0,02	15,7	84
EX23	86	687	0,07	8,6	86	0,00	388	43,52	14,06	19,0	86
EX33	86	675	0,05	7,5	86	0,00	401	40,59	5,94	19,9	86
EX43	89	648	4,07	10,63	89	0,00	503	22,38	0,57	23,3	89
EX53	74	500	1,07	6,7	74	0,00	299	40,20	4,82	16,0	74
EX63	103	777	8,77	11,50	103	0,00	430	44,66	18,91	24,1	103
EX73	89	957	10,93	13,06	88	1,12	513	46,39	25,81	28,6	89
EX83	153	1232	0,05	12,65	153	0,00	422	65,75	20,80	36,5	153
EX93	105	677	1,09	10,09	105	0,00	528	22,01	5,06	21,3	105
EX103	140	1151	14,37	14,49	139	0,71	679	41,01	1,98	28,3	140
EX14	103	744	0,16	6,2	103	0,00	528	29,03	0,88	16,2	103
EX24	108	871	3,84	8,9	108	0,00	511	41,33	1,48	19,2	108
EX34	112	964	6,42	9,9	111	0,89	597	38,07	12,15	20,8	112
EX44	125	980	9,92	11,36	124	0,80	715	27,04	22,00	23,9	125
EX54	56	97	746	0,19	6,97	97	0,00	519	30,43	15,9	16.2
AVG		**109.4**	**873.78**	**2.83**	**10.50**	**109.3**	**0.16**	**546.08**	**35.78**	**9.38**	**23.14**

Table 5
Results for the JSSPT and the JSSPR with datasets $\boldsymbol{D_2}$ (For unit capacity vehicles)

Ulusoy et al. (1997)	**JSSPT resolution:** $\boldsymbol{F = h_{\text{cmax}}(y)}$				**Resolution of integrated JSSPR:** $\boldsymbol{F(x) = 10000 \times h_{\text{cmax}}(x) + h_{\text{cost}}(x)}$.					
Instance name	$h^{i*}_{cmax}(y)$	$h^{i*}_{cost}(y)$	st^{i*}	stt^{i*}	$h^{i*}_{cmax}(x)$	Δ^{i*}_{cmax}	$h^{i*}_{cost}(x)$	Δ^{i*}_{cost}	st^{i*}	stt^{i*}
EX101	83	548	1,56	13,40	80	0,01	501	8,39	1,31	24,04
EX201	88	669	3,54	16,81	85	0,01	539	19,06	2,65	28,52
EX301	91	720	3,42	18,43	88	0,01	600	16,22	25,82	30,41
EX401	96	730	7,85	27,31	93	0,01	659	9,26	9,86	40,78
EX501	72	532	1,30	14,34	69	0,01	456	14,14	9,70	24,32
EX601	109	820	2,36	22,22	106	0,01	652	19,95	23,89	35,13
EX701	87	1006	3,59	27,68	84	0,01	894	10,44	4,69	41,72
EX801	162	1261	1,17	21,43	159	0,01	595	52,03	38,05	46,67
EX901	107	684	2,83	20,40	104	0,01	615	9,68	13,12	32,03
EX102	139	966	16,21	27,32	136	0,01	808	15,69	20,96	42,14
EX202	77	466	1,17	11,16	74	0,01	387	16,98	19,71	22,93
EX302	77	593	1,24	13,29	74	0,01	380	35,69	1,56	26,31
EX402	81	578	1,18	13,86	78	0,01	406	29,55	3,18	27,63
EX502	75	490	4,80	24,30	72	0,01	454	7,29	5,47	36,99
EX702	65	433	1,63	12,24	62	0,01	328	24,39	13,45	23,34
EX802	99	742	1,97	19,08	96	0,01	432	41,35	20,52	32,93
EX702	80	827	3,31	22,84	77	0,01	483	41,07	11,94	38,30
EX802	152	1251	1,16	19,14	149	0,01	441	63,98	19,42	49,52
EX902	99	623	2,35	17,60	96	0,01	484	22,01	21,47	30,05
EX112	130	893	15,78	24,37	129	-1,54	684	22,80	25,63	40,24
EX113	75	433	1,17	11,34	72	0,01	385	11,20	5,07	22,50
EX213	83	682	1,22	14,49	80	0,01	382	43,63	23,70	27,18
EX313	83	568	1,23	13,98	80	0,01	414	26,92	1,43	27,49
EX413	75	525	5,67	23,67	72	0,01	436	16,83	28,66	36,27
EX513	64	406	1,23	12,75	61	0,01	318	21,91	12,33	23,26
EX613	101	760	1,66	19,69	98	0,01	475	37,04	29,00	33,06
EX713	85	572	1,42	13,87	82	0,01	493	13,59	2,68	23,85
EX813	89	663	14,38	17,06	86	0,01	538	18,49	27,77	28,86
EX913	93	756	2,29	18,49	90	0,01	578	23,07	20,23	30,41
EX123	99	725	26,71	28,03	96	0,01	697	3,40	4,60	40,39
EX124	74	503	1,17	14,72	71	0,01	428	14,84	7,00	24,78
EX224	107	812	3,64	23,34	102	1,90	615	23,73	29,14	35,67
EX324	91	1037	7,43	27,42	88	0,01	827	19,55	27,46	42,11
EX424	164	1337	1,19	22,59	161	0,01	654	50,27	27,55	44,31
EX524	106	679	6,10	21,05	103	0,01	585	13,45	7,01	32,38
AVG	**99**	**755**	**4,92**	**19,65**	**96**	**0,07**	**544**	**24,70**	**15,56**	**33,69**

4.3.2 Non-Unit Capacity Vehicles

Similar to the previous section, Table 6 and Table 7 introduce the best replications among the five GRASP×ELS replications for the JSSPT and the integrated JSSPR. In this section, the vehicles have non-unit capabilities, each of which can optionally carry two jobs simultaneously.

For the dataset D_1 (Table 6), the average deviation of the makespan, $\overline{\Delta_{cmax}^{l,*}} = 0.06\%$, means that the integrated JSSPR is on average very slightly better than the JSSPT concerning the makespan criterion.

Concerning the QoS criterion, $\overline{\Delta_{cost}^{l,*}} = 26.00\%$, which means that the integrated JSSPR finds solutions on average *26%* better than the JSSPT for the QoS criterion. The average computation times of the JSSPT are $\overline{st^{l,*}} = 3.61$ seconds and $\overline{stt^{l,*}} = 18.66 seconds$, and, $\overline{st^{l,*}} = 14.51$seconds and $\overline{stt^{l,*}} = 32.19$. The resolution of the JSSPR requires approximately twice as much time as the resolution of the JSSPT, but it makes it possible to obtain solutions of much higher quality in terms of QoS, without degrading the criterion of the makespan.

Table 6
Results for the JSSPT and the JSSPR with datasets $\boldsymbol{D_1}$ (For non-unit capacity vehicles)

Ulusoy et al. (1997)	JSSPT resolution: $F = h_{cmax}(y)$				Resolution of integrated JSSPR: $F(x) = 10000 \times h_{cmax}(x) + h_{cost}(x)$					
Instance name	$h_{cmax}^{i*}(y)$	$h_{cost}^{i*}(y)$	st^{i*}	stt^{i*}	$h_{cmax}^{i*}(x)$	Δ_{cmax}^{i*}	$h_{cost}^{i*}(x)$	Δ_{cost}^{i*}	st^{i*}	stt^{i*}
EX11	80	52	0,52	14,21	84,6	0,00	497	8,58	0,4	23,64
EX21	85	666	2,42	17,62	89,6	0,00	535	19,34	1,74	28,12
EX31	88	717	2,32	19,24	92,6	0,00	596	16,5	24,84	30,01
EX41	93	727	6,81	28,12	97,6	0,00	655	9,54	8,95	40,38
EX51	69	529	0,26	15,15	73,6	0,00	452	14,42	8,79	23,92
EX61	106	817	1,32	23,03	110,6	0,00	648	20,23	22,94	34,73
EX71	84	1003	2,55	28,49	88,6	0,00	890	10,72	3,78	41,32
EX81	159	1258	0,13	22,24	163,6	0,00	591	52,31	37,14	46,27
EX91	104	681	1,79	21,21	108,6	0,00	611	9,96	12,14	31,63
EX101	136	963	15,17	28,13	140,6	0,00	804	15,97	20,04	41,74
EX12	74	463	0,13	11,97	78,6	0,00	383	17,26	18,74	22,53
EX22	74	590	0,2	14,1	78,6	0,00	376	35,97	0,65	25,91
EX32	80	575	0,14	14,67	82,6	0,00	402	29,83	2,27	27,23
EX42	72	487	3,76	25,11	76,6	0,00	450	7,57	4,56	36,59
EX52	62	430	0,59	13,05	66,6	0,00	324	24,67	12,54	22,94
EX62	96	739	0,93	19,89	100,6	0,00	428	41,63	19,54	32,53
EX72	77	824	2,27	23,65	81,6	0,00	479	41,35	10,94	37,9
EX82	149	1248	0,12	19,95	153,6	0,00	437	64,26	18,44	49,12
EX92	96	620	1,31	18,41	100,6	0,00	480	22,29	20,54	29,65
EX102	126	890	14,74	25,18	133,6	1,61	680	23,08	24,64	39,84
EX13	72	430	0,13	12,15	76,6	0,00	381	11,48	4,16	22,1
EX23	80	679	0,18	15,3	84,6	0,00	378	43,91	22,74	26,78
EX33	80	565	0,19	14,79	84,6	0,00	410	27,2	0,52	27,09
EX43	72	522	4,63	24,48	76,6	0,00	432	17,11	27,74	35,87
EX53	61	403	0,19	13,56	65,6	0,00	314	22,19	11,34	22,86
EX63	98	757	0,62	20,5	102,6	0,00	471	37,32	28,04	32,66
EX73	79	909	1,72	23,74	83,6	0,00	528	41,35	25,34	38,07
EX83	151	1219	0,13	21,08	155,6	0,00	453	62,12	5,61	48,29
EX93	100	651	1,44	19,15	102,6	0,00	500	22,87	1,08	29,51
EX103	131	1000	4,32	25,56	135,6	0,00	719	27,52	32,44	39,92
EX14	82	569	0,38	14,68	86,6	0,00	489	13,87	1,77	23,45
EX24	86	660	13,34	17,87	90,6	0,00	534	18,77	26,84	28,46
EX34	90	753	1,25	19,3	94,6	0,00	574	23,35	19,24	30,01
EX44	96	722	25,67	28,84	100,6	0,00	693	3,68	3,69	39,99
EX54	71	500	0,13	15,53	75,6	0,00	424	15,12	6,09	24,38
AVG	**93.11**	**703,37**	**3,19**	**19,71**	**97,69**	**0.046**	**514,8**	**24,95**	**14,01**	**32,44**

Concerning dataset D_2 (Table ٧), $\overline{\Delta_{cmax}^{l,*}} = 0.00\%$ means that the JSSPT and the integrated JSSPR obtain solutions with an equivalent makespan. Remarks similar to the numerical results obtained with the dataset D_1 can be established for the results obtained from the dataset D_2. The average deviation of the QoS, for the dataset D_2, is $\overline{\Delta_{cost}^{l,*}} = 45.60\ \%$, and reflects the improvement brought by the integrated JSSPR for the criterion of the compared QoS to a JSSPT resolution. The computation times of the integrated JSSPR are nevertheless much greater than for the JSSPT.

Table 7
Results for the JSSPT and the JSSPR with datasets $\boldsymbol{D_2}$ (For non-unit capacity vehicles)

Ulusoy et al. (1997)	**JSSPT resolution:** $F = h_{cmax}(y)$				**Resolution of integrated JSSPR:** $F(x) = 10000 \times h_{cmax}(x) + h_{cost}(x)$					
Instance name	$h^{i*}_{cmax}(y)$	$h^{i*}_{cost}(y)$	st^{i*}	stt^{i*}	$h^{i*}_{cmax}(x)$	Δ^{i*}_{cmax}	$h^{i*}_{cost}(x)$	Δ^{i*}_{cost}	st^{i*}	stt^{i*}
EX110	123	739	0.01	6,79	138	0,00	523	35.86	1.39	25.74
EX210	145	1069	0,00	9,17	160	0,00	545	52.51	7.06	29.55
EX310	147	1082	0,02	9,67	162	0,00	649	43.53	18.23	33.91
EX410	113	723	0,01	14.43	128	0,00	661	12.62	1.26	31.27
EX510	99	622	0,00	6,55	114	0,00	367	45.1	1.65	24.95
EX610	183	1484	0,03	13.12	198	0,00	683	57.24	4.39	33.81
EX710	143	1480	0,04	13,64	158	0,00	555	65.75	18.63	45.73
EX810	289	2537	0,00	15,22	304	0,00	691	75.77	6.39	54.52
EX910	171	1052	1.49	14.36	186	0,00	717	35.4	1.17	29.04
EX1010	235	1661	1.11	17.45	250	0,00	949	46.09	9.18	40.15
EX120	120	661	0,00	5.22	135	0,00	507	27.4	1.36	23.14
EX220	140	1027	0.02	7.26	155	0,00	529	52.02	9.69	27.09
EX320	142	1065	0,00	9.78	157	0,00	613	45.96	5.89	30.99
EX420	111	717	0.05	14.35	126	0,00	611	18.81	1.93	30.71
EX520	97	617	0,00	4.22	112	0,00	345	48.18	1.98	25.51
EX620	178	1316	0.04	15.2	193	0,00	631	55.39	23.23	30.98
EX720	140	1630	0.01	13.45	155	0,00	508	72.02	27.63	42.63
EX820	284	2370	0.02	15.55	299	0,00	623	76.74	25.33	52.86
EX920	170	1067	0.00	12.11	185	0,00	657	41.95	4.92	27.39
EX1020	233	1621	1.01	18.54	248	0,00	903	47.53	3.24	37.28
EX130	119	703	0.01	8.6	134	0,00	505	32.16	1.31	21.99
EX230	143	1079	0,00	7.15	158	0,00	497	57.41	22.43	26.68
EX330	143	1059	0.06	8.12	158	0,00	617	45.26	6.37	29.94
EX430	110	770	0.04	11.62	125	0,00	613	24.3	1.44	29.14
EX530	96	640	0.04	7.45	111	0,00	343	50.44	2	23.44
EX630	179	1396	0.12	11.2	194	0,00	643	57.24	9.02	30.46
EX730	141	1497	0.05	18.6	156	0,00	523	68.3	36.53	42.31
EX830	285	2461	0.03	13.2	300	0,00	635	77.21	23.03	52.09
EX930	171	941	0.15	11.3	186	0,00	667	32.78	3.79	26.89
EX1030	234	1660	1.85	14.5	249	0,00	915	48.1	3.37	36.33
EX140	121	676	0.01	8.3	136	0,00	531	25.53	1.48	24.81
EX241	214	1483	0.03	10.3	229	0,00	746	52.97	15.73	28.89
EX340	145	1039	0.01	11.89	160	0,00	671	38.98	4.07	30.47
EX341	215	1655	0.14	5.23	230	0,00	970	44.62	1.69	32.91
EX441	166	1043	0.02	15.2	181	0,00	902	17.14	1.83	33.1
AVG	**164.1**	**1218**	**0.18**	**9.64**	**179.1**	**0,00**	**629.8**	**46.52**	**8.81**	**32.76**

The second study highlights the importance of the QoS during the optimization scheme and the impact of the QoS on a solution. This QoS can be greatly improved by solving an integrated JSSPR without losing quality in terms of makespan. However, the integrated JSSPR requires more computation time. These remarks remain true whatever the dataset considered and whatever the capacity of the vehicles (unitary or non-unitary).

4.3.3 Capacity Vehicles

To further emphasize the importance of QoS in both JSSPT and JSSPR, Figure 10 showcases the contrast among four averages (AVG1, AVG2, AVG3, and AVG4), which correspond to the findings from the preceding section's tables. This comparison pertains to the utilization of two datasets, D1 and D2, across two vehicle capacity scenarios: unitary and non-unitary.

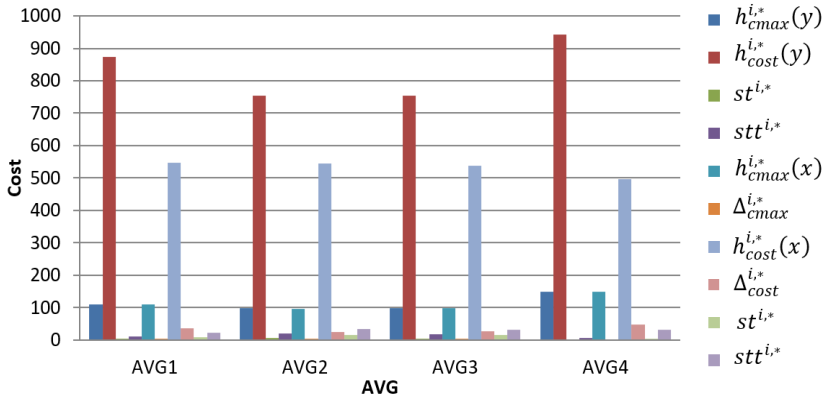

Fig. 10. Comparative chart for the two capacities of vehicles (unitary or non-unitary).

After this comparison, we conclude that the combined JSSPR necessitates additional processing time. These observations hold regardless of the dataset under consideration or the vehicle capacity being unitary or non-unitary.

5. Conclusion and Future Work

This article has proposed to take into account a quality criterion in a scheduling problem with routing. The vast majority of problems combining scheduling and transport do not consider such criteria. The transport problem is often seen as a "sub-problem" or "a constraint" of the problem scheduling and then a semi-active solution is sufficient. Consideration of the QoS criterion requires obtaining solutions that are no longer semi-active.

This research presents an evaluation function of a disjunctive graph, named TLH, for the JSSPR, which can be used in the case of a group of vehicles of unit capacity and the case of a non-unit capacity fleet. TLH evaluation is included in a GRASP×ELS type metaheuristic and is tested numerically. These instances are extended to take into account the case of a group of vehicles of non-unit capacity. Two algorithmic schemes using TLH evaluation are tested: the first scheme, named sequential JSSPR, minimizes – during GRASP×ELS – only the makespan, then the TLH evaluation is applied to the best solution to maximize its QoS. The second scheme, called integrated JSSPR, uses the TLH function for each evaluation of a graph, and GRASP×ELS has the objective function of minimization of makespan and maximization of QoS.

The work presented in this paper offers several perspectives. It should be noted that additional constraints could be added to the JSSPR. These constraints would concern the transport part: for example, a maximum distance that vehicles can travel which can be justified by the use of electric vehicles and therefore by the need to recharge the vehicle's batteries. The JSSPR could take into account constraints on the loading and unloading orders of the parts in the vehicles: thus, the first part to be delivered would be the one loaded last in the vehicle.

A Constraint Programming (CPP) model could be proposed for the JSSPR, and hybrid approaches could be considered. It should also be noted that additional constraints could be added in the JSSPR. These constraints would concern the transport part: for example, a maximum distance that vehicles can travel and which can be justified by the use of electric vehicles and therefore by the need to recharge the vehicle's batteries. The JSSPR could take into account constraints on loading and delivery orders. Unloading of parts into vehicles: thus, the first part to be delivered would be the one loaded last into the vehicle. Another area of research would be to allow vehicles to exchange the products transported. In this case, the transport of the JSSPR would approach the problems of Vehicle Routing Problem with Trailers and Transshipments and would impose coordination constraints between the tours.

References

Abdolrazzagh-Nezhad, M .,& Abdullah, S. (2017). Job shop scheduling: Classification, constraints and objective functions. *International Journal of Computer and Information Engineering*, *11*(4), 429-434.

Abukhader, R., & Kakoore, S. (2021). Artificial Intelligence for Vertical Farming–Controlling the Food Production.

Ahmadian, M. M., Salehipour, A., & Cheng, T. C. E. (2021). A meta-heuristic to solve the just-in-time job-shop scheduling problem. *European Journal of Operational Research*, *288*(1), 14-29.

Alaya, B. (2017, April). EE-(m, k)-Firm: A Method to Dynamic Service Level Management in Enterprise Environment. In *International Conference on Enterprise Information Systems*, *2*, 114-122.

Álvarez-Gil, N., Rosillo, R., de la Fuente, D., & Pino, R. (2021). A discrete firefly algorithm for solving the flexible job-shop scheduling problem in a make-to-order manufacturing system. *Central European Journal of Operations Research*, *29*, 1353-1374.

Baykasoğlu, A., Hamzadayi, A., & Köse, S. Y. (2014). Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. *Information Sciences*, 276, 204-218.

Bueno, A., Godinho F., Moacir, F., & Alejandro, G. (2020). Smart production planning and control in the Industry 4.0 context. *Computers industrial engineering*, *149*, 106774.

Cebi, C., Atac, E., & Sahingoz, O. K. (2020, July). Job shop scheduling problem and solution algorithms: a review. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*,1-7.

Cordeau, J. F., & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, *37*(6), 579-594.

Cunha, B., Madureira, A. M., Fonseca, B., & Coelho, D. (2020). Deep reinforcement learning as a job shop scheduling solver. In *Hybrid Intelligent Systems: 18th International Conference on Hybrid Intelligent Systems (HIS 2018),* 350-359.

Dean, W. (2015). Computational Complexity Theory, *The Stanford Encyclopedia of Philosophy* , Edward N. Zalta (ed.), https://plato.stanford.edu/archives/fall2021/entries/computational-complexity/>

Firat, M., & Woeginger, G. J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, *39*(1), 32-35.

Frihat, M., Hadj-Alouane, A. & Sadfi, C. (2022). Optimization of the integrated problem of employee timetabling and job shop scheduling. *Computers & Operations Research*, *137*, 105332.

Gaham, M., Bouzouia, B., & Achour, N. (2018). An effective operations permutation-based discrete harmony search approach for the flexible job shop scheduling problem with makespan criterion. *Applied Intelligence*, *48*, 1423-1441.

Gao, D., Wang, G. G., & Pedrycz, W. (2020). Solving fuzzy job-shop scheduling problem using DE algorithm improved by a selection mechanism. *IEEE Transactions on Fuzzy Systems*, *28*(12), 3265-3275.

Gondran, M., Huguet, M. J., Lacomme, P., Quilliot, A., & Tchernev, N. (2018). A Dial-a-Ride evaluation for solving the job-shop with routing considerations. *Engineering Applications of Artificial Intelligence*, *74*, 70-89.

Gondran, M., Lacomme, P., & Tchernev, N. (2017). Resolution of a job-shop with routing considering a quality of service for customer: JSSP with routing. In *7th International Conference on Industrial Engineering and Systems Management (IESM)*.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* , 5, 287-326.

Haifei, Y., Songjian, H., Dongsheng ,Y., Zhiyong,W., Wei ,F.,&Atila B.(2021). Job Shop Scheduling Based on Digital Twin Technology. *Complexity*, (2021),1-12 https://doi.org/10.1155/2021/8823273.

He, P., & Hao, J. K. (2023). Memetic search for the minmax multiple traveling salesman problem with single and multiple depots. *European Journal of Operational Research*, *307*(3), 1055-1070.

Hegen, X., Shuangyuan, S., Danni, R., & Jinjin, H. (2022). A survey of job shop scheduling problem: the types and models. *Computers Operations Research*, *142,*105731, doi.org/10.1016/j.cor.2022.105731.

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.

Hussein, M., & Zayed, T. (2021). Critical factors for successful implementation of just-in-time concept in modular integrated construction: A systematic review and meta-analysis. *Journal of Cleaner Production*, *284*, 124716.

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European journal of operational research*, *113*(2), 390-434.

Jamili, A. (2016). Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms. *Expert systems with Applications*, *55*, 341-350.

Kalshetty, Y. R., Adamuthe, A. C., & Kumar, S. P. (2020). Genetic algorithms with feasible operators for solving job shop scheduling problem. *Journal of Science Resources*, *64*, 310-321.

Kardos, C., Laflamme, C., Gallina, V., & Sihn, W. (2021). Dynamic scheduling in a job-shop production system with reinforcement learning. *Procedia CIRP*, *97*, 104-109.

Kechadi, M. T., Low, K. S., & Goncalves, G. (2013). Recurrent neural network approach for cyclic job shop scheduling problem. *Journal of Manufacturing Systems*, *32*(4), 689-699.

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.

Lee, T., & Loong, Y. (2019). A review of scheduling problem and resolution methods in exible ow shop. *International Journal of Industrial Engineering Computations*, *10*(1), 67-88.

Li, X., Xie, J., Ma, Q., Gao, L., & Li, P. (2022). Improved gray wolf optimizer for distributed flexible job shop scheduling problem. *Science China Technological Sciences*, *65*(9), 2105-2115.

Mihoubi, B., Bouzouia, B., & Gaham, M. (2021). Reactive scheduling approach for solving a realistic flexible job shop scheduling problem. *International journal of production research*, *59*(19), 5790-5808.

Mohan, J., Lanka, K., & Rao, A. N. (2019). A review of dynamic job shop scheduling techniques. *Procedia Manufacturing*, *30*, 34-39.

Nouri, H. E., Driss, O. B., & Ghédira, K. (2016). A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review. In *Artificial Intelligence Perspectives in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016),* 1*,* 1-11.

Ojstersek, R., Brezocnik, M.,&Buchmeister, B. (2020). Multi-objective optimization of production scheduling with evolutionary computation. *International Journal of Industrial Engineering Computations*, *11*(3), 359-376.

Pappas, I., Kourouthanassis, P., Giannakos, M.,& Lekakos, G. (2017). The interplay of online shopping motivations and experiential factors on personalized e-commerce. *Telematics and informatics*, *34*(5), 730-742.

Parveen, S., & Ullah, H. (2010). Review on job-shop and flow-shop scheduling using. *Journal of Mechanical Engineering*, *41*(2), 130-146.

Prins, C. (2009). A GRASP× evolutionary local search hybrid for the vehicle routing problem. In *Bio-inspired algorithms for the vehicle routing problem*, 35-53.

Quinton, F., Hamaz, I., & Houssin, L. (2021). A Benders decomposition for the flexible cyclic jobshop problem. In *17th Internatinal Workshop on Project Management and Scheduling*.

Saidat, S., Junoh, A. K., Wan Muhamad, W. Z. A., & Yahya, Z. (2022). Modified job shop scheduling via Taguchi method and genetic algorithm. *Neural Computing and Applications*, 1-18.

Smutnicki, C., & Pempera, J. (2022). Job Shop Scheduling with Transport by Automated Guided Vehicles. In *16th International Conference on Soft Computing Models in Industrial and Environmental Applications* ,789-799.

Stein, D. M. (1978). Scheduling dial-a-ride transportation systems. *Transportation Science*, *12*(3), 232-249.

Sun, X., & Noble, J. S. (1999). An approach to job shop scheduling with sequence-dependent setups. *Journal of Manufacturing Systems*, *18*(6), 416-430.

Türkyılmaz, A., Şenvar, Ö., Ünal, İ., & Bulkan, S. (2020). A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, *31*, 1949-1983.

Ulusoy, G., Sivrikaya-Şerifoğlu, F., & Bilge, Ü. (1997). A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. *Computers & Operations Research*, 24(4), 335-351.

Vancheeswaran, R., & Townsend, M. A. (1993). Two-stage heuristic procedure for scheduling job shops. *Journal of Manufacturing Systems*, *12*(4), 315-325.

Vital-Soto, A., Azab, A., & Baki, M. F. (2020). Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility. *Journal of Manufacturing Systems*, *54*, 74-93.

Xie, J., Gao, L., Peng, K., Li, X., & Li, H. (2019). Review on flexible job shop scheduling. *IET collaborative intelligent manufacturing*, *1*(3), 67-77.

Yang, L., Li, J., Chao, F., Hackney, P., & Flanagan, M. (2021). Job shop planning and scheduling for manufacturers with manual operations. *Expert Systems*, *38*(7), e12315.

Zeng, C., Tang, J., & Yan, C. (2015). Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles. *Journal of Intelligent Manufacturing*, *26*, 845-859.

Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., & Zhang, M. (2021). Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, *52*(10), 10515-10528.

Zhang, H., Yang, Y., & Wu, F. (2022). Just-in-time single-batch-processing machine scheduling. *Computers & Operations Research*, *140*, 105675.

Zhang, M., Tao, F., & Nee, A. Y. C. (2021). Digital twin enhanced dynamic job-shop scheduling. *Journal of Manufacturing Systems*, *58*, 146-156.

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.

Zhang, Q., Manier, H., & Manier, M. A. (2014). A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints. *International Journal of Production Research*, *52*(4), 985-1002.

Zhou, B., & Liao, X. (2020). Particle filter and Levy flight-based decomposed multi-objective evolution hybridized particle swarm for flexible job shop greening scheduling with crane transportation. *Applied Soft Computing*, *91*, 106217.