Planning the tasks of an autonomous mobile robot fleet for internal logistics of production systems

Elouan Blanchard, Arthur Bit-Monnot, Cyrille Briand, Grégoire Milliez, Mohamed Amine Abdeljaouad

Planning the tasks of an autonomous mobile robot fleet for internal logistics of production systems

E. Blanchard[1,2], Arthur Bit-Monnot[1], C. Briand[1], M.A. Abdeljaouad[2], G. Milliez[2]

[1] LAAS-CNRS, Université de Toulouse, CNRS, UT3, Toulouse, France
{elouan.blanchard,cyrille.briand,arthur.bit-monnot}@laas.fr
[2] Direction de l'Innovation – Groupe ALTEN, Toulouse, France
{elouan.blanchard,mohamedamine.abdeljaouad,gregoire.milliez}@alten.com

Abstract. The problem of planning the activities of a fleet of autonomous mobile robots in the context of performing a production plan is tackled in this paper. Three kinds of tasks are considered: the ones related to supplying workstations with the components or tools that are used in the operations, the ones related to the evacuation of empty containers or garbage collecting, and the latter ones that aim at moving semi-finished production from workstations to others. This paper shows that the planning of these activities can be modeled homogeneously as a particular pickup-and-delivery problem with time windows. To solve the latter problem, we propose a greedy heuristic as well as a mixed-integer linear programming approach. An illustration of the interest in the approach is provided in a production context, demonstrating its validity and highlighting its advantages and limitations. A benchmark of the literature is also considered to show how our approach performs in the context of a job-shop problem with transportation constraints.

Keywords: Internal logistics, pick-up and delivery with time windows, inventory routing, job-shop with transportation constraints.

1 Introduction

This work focuses on planning at a tactical decision level the activities of a fleet of robots in the context of the internal logistics of production systems. Two types of robots are usually distinguished for internal logistics: Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs). AGVs can only follow fixed paths or tracks to transport materials, which usually requires infrastructure modifications such as magnetic strips or wires. On the other hand, AMRs are a newer family of robots that exploit sophisticated sensors (cameras, lidars) and more advanced navigation algorithms, in particular the well-known Simultaneous Localization and Mapping (SLAM) algorithm, that allows the robots to navigate in uncontrolled environments without the need for fixed paths or tracks.

AMRs are undoubtedly much more flexible than AGVs since they no longer have to follow fixed routes. They can detect obstacles and bypass them, choosing in real time the best alternative route. Their intelligent navigation allows them to move safely in dynamic environments. For that reason, in the context of the new industrial revolution, AMRs are playing an important role in the development of smarter production systems (see [1]). They are capable of carrying out a wide range of logistical operations such as supplying workstations with component kits, moving specific production tools, collecting empty containers, or evacuating waste. AMRs can also be used to transport semi-finished products between production workstations.

In this new context, the ability to synchronize the planning of robot fleet operations with the design of the production plan is becoming increasingly crucial to fostering the agility of the production system. Indeed, the lateness of a production task, a machine failure, or a quality default may need to adapt the production plan, which will consequently impact the robot routing plan as well since component supplying should possibly be delayed, semi-product moves have to be rescheduled, or new components have to be delivered. Conversely, during the implementation of the routing plan, AMRs can also experience unexpected delays when ensuring collision avoidance (in case of congestion for instance), if a route is temporally closed, or if the loading/unloading operations made on the robot take longer as expected, which may require to adapt the production plan. Therefore, in a highly dynamic environment, the challenge is to maintain in real-time the consistency between production and routing plans such that the work in progress (a basic production efficiency indicator) is kept minimized.

Many attempts have been made in the literature to integrate these two complex problems. The workstation supplying problem being very similar to the Inventory Routing Problem (IRP), a first category of approaches is related to this problem. IRP consists in determining the tours to be carried out for each vehicle of the fleet, defining for each tour the order in which the workstations are visited, the date of the visit and the quantity to be delivered (or picked up). The aim is to avoid stock-outs and minimize logistics costs. Another category of approaches is related to Pickup-and-Deliveries Problems (PDP) where each logistic operation is characterized by a good quantity to move from a specific origin to a particular destination. The problem of coupling transportation constraints with production scheduling is also present in the scheduling literature with the Job-Shop scheduling Problem with Transport (JSPT), introduced in the 90's.

In this paper, we show how internal logistic operations, such as supplying workstations with component kits, moving specific production tools, collecting empty containers, or evacuating waste can be advantageously modeled as pick-up and delivery tasks to be carried out in time windows. Then, aiming at minimizing the work in progress, a two-step approach is proposed to determine an efficient routing of the robots such that pick-up and delivery time windows, and the capacity of workstations and vehicles are satisfied. We do not consider so far a dynamic environment as the production plan is assumed known in advance, the

goal being to build up a routing plan that satisfies the internal logistics required to implement the production plan.

The paper is organized as follows. Section 2 proposes an analysis of the literature related to our problem, highlighting the differences between the three previously mentioned categories of approach, to motivate our positioning. Section 3 shows how our problem can be defined generically as a PDP with time windows. For the solving methodologies, we propose in Sections 4 and 5 a mixed integer linear program (MILP) and a polynomial greedy heuristic, respectively. Section 6 provides some first results and illustrates the benefits of our approach on several medium-sized instances. In Section 7, using a benchmark of the literature related to job-shop with transportation constraints, we show how the heuristic and the MILP performs aiming at sizing the AMR fleet. Some conclusions and work perspectives are drawn in Section 8.

2 Literature review

Vehicle Routing Problems (VRP) are a family of NP-hard problems widely studied in combinatorial optimization. They have many practical applications, as they model a wide variety of transportation problems, whether for people, goods, or information. Generally speaking, they involve organizing the delivery of goods to customers over a period of time, according to their needs, using vehicles located in one or more depots. The goods are initially stored in depots. Vehicle loads, as well as their routes, must then be determined to minimize the total cost of transport while respecting logistical constraints (return to depot, on-time delivery, vehicle capacities). The book by Toth and Vigo [2] provides a comprehensive survey of the state of the art, as well as a description of methods and solutions for this problem and its variants. The Capacitated VRP (CVRP) is the simplest and most widely studied of the VRPs. It first appeared in 1959 [3]. In this version of the CVRP, the vehicles are all identical and have the same transportation capacity. There is a single central depot where the requested products are initially stored. Vehicles pass through each customer at most once (they complete an elementary cycle), and each customer is visited at most once. The aim is to minimize transportation costs while ensuring the required deliveries. An important extension of the VRP is the VRP with Time Windows (VRPTW) problem, where each delivery must be made within a specified time interval, possibly including an unloading time [4].

Inventory Routing Problems (IRP) are a generalization of VRP over several periods. Customer requests are no longer data, but decision variables, and customer inventories must be managed aiming at finding a compromise between transport costs and storage costs over a range of time periods. Knowledge of customer inventories enables the delivery driver to be more efficient but makes the problem more complex. A detailed introduction to IRP is given in [5], while a literature review is provided in [6] and [7]. Due to their complexity, IRPs are often solved with heuristics or hybrid methods, involving heuristics and exact methods. In [8], for example, the problem is decomposed into two parts: first, the

customers to be served are determined for each period, and then a VRP is solved at each period. In this way, existing heuristics for the VRP can be exploited [9]. More recently, heuristic, metaheuristic, and hybrid (matheuristic) methods have been developed for these problems. In [10], a matheuristic is proposed for solving IRP, mixing integer linear programming (MILP), and adaptive extended neighborhood search (ALNS). In [11], an approach combining linear programming and GRASP (Greedy Randomized Adaptive Search Procedure) is presented. In [12], a genetic algorithm is used for a general variant of IRP. In [13], a two-part iterative metaheuristic is described, coupling variable neighborhood search (VNS) and linear programming to minimize transport and storage costs at each stage. In [14], a method is proposed that integrates a fixed-sequence mathematical program, two randomized greedy algorithms, and a column-generation-based heuristic, which can solve realistic industrial IRP problems arising in gas distribution. Finally, in [15], the authors propose a mixed integer linear program to minimize the energy consumed by vehicles that supply the workstations of a mixed-model automotive assembly line.

VRP with Pickup and Delivery (or Pickup-and-Delivery Problem, PDP) is another important extension of CVRP [16, 17]. It differs from CVRP in that goods or people can have several origins, and several destinations, unlike CVRP where all goods are stored at the depot. So some customers have negative requests to indicate that they are producing items. There may be several types of items to collect and deliver. As the items can be loaded during the tour, the precedence constraints are not checked a priori, and the items have to be differentiated, which is not necessary in CVRP. There are two main types of PDP, depending on whether goods or people are transported. PDPs for goods involve finding elementary routes for each vehicle that minimize the total cost of transport while respecting the constraints of collection and delivery to customers [18]. A distinction is made between: i) many-to-many problems, where each item is available at several locations and has to be delivered to several destinations, ii) one-to-many-to-one problems, where some items originate from a single depot and have to be delivered to customers, and other items have to be picked up from customers and then delivered to the depot, and iii) one-to-one problems, where each item has a single origin and a single destination. Among PDP variants, we are particularly focusing on those where pickup and deliveries have to be completed inside specific time windows (PDPTW). Solving approaches using column generation have been very successful both for tackling realistic size PDPTW instances [19], as well as dynamic ones [20]. Li and Lim [21] proposes a metaheuristic to solve the PDPTW and six large-size data sets to evaluate their algorithm. More recently, in [22], a matheuristic is proposed that works well on Li-and-Lim's instances, as well as on a new set of instances.

The existence of transportation operations between production activities has also been investigated in the scheduling literature. In [23], Ulusoy et al. consider the JSPT, a job-shop problem that involves to simultaneous schedule the machines and route the AGVs. Classically, the goal is to minimize the schedule makespan. The authors propose a genetic algorithm that was evaluated on a set

of freely available problem instances, which we used in this work. Deroussi et al. [24] propose three different metaheuristics (an iterated local search, a simulated annealing procedure, and a hybrid approach) that exploit a neighboring structure relying on the routes followed by the AGVs. An extended disjunctive graph and a local search are introduced by Lacomme et al. in [25] to efficiently model both problem and solutions, which can be also used to tackle flexible job-shop with several transport robots. In [26], Gondran et al. show that, when quality of service (QoS) is taken into account additionally with the makespan, the problem becomes close to the Dial-A-Ride Problem (DARP) where pickup and delivery operations have to be scheduled. They propose an iterative solving approach that first minimizes the makespan and then the QoS. The flexible job-shop scheduling problem with transportation resources is considered in [27] where, using a disjunctive graph modeling, the authors propose a tabu search procedure that exploits a neighborhood function, which explores a large set of moves in constant time. New benchmark instances are also proposed. Note that a recent survey related to scheduling problems with transportation constraints is proposed in [28], which reviews over 160 papers and introduces a new three-field classification scheme for characterizing them.

Let us discuss the pros and cons of the previous approaches in the context of internal production logistics. First, we remark that the problem of supplying a production line invalidates several classical assumptions of the IRP. Indeed, loading/unloading times are no longer negligible compared with transportation times as the routes between workstations can now be very short. Additionally, the time granularity to be considered is very fine, with operation times generally expressed in minutes, which means that it is no longer possible to split the decision-making horizon into time periods lasting longer than one vehicle tour.

Besides, the problems of transferring semi-finished products and eliminating waste falls clearly outside the scope of classic IRP, since it no longer just involves supplying workstations, but also collecting items. The JSPT environment also seems too restrictive since conversely, it only considers the transportation of semi-finished products, but not the problem of supplying workstations with components. Furthermore, as indicated in [24], for reasons of complexity, the number of vehicles taken into account in JSPT is often limited to 1 or 2, and their capacity is assumed to be unitary, which seems too restrictive with regard to real industrial practices.

Finally, while internal logistics is classically concerned with minimizing the work-in-progress (WIP) and respecting the production plan as closely as possible, both IRP and PDPTW aim at minimizing the transportation costs, which essentially depends on the length of the routes followed by the vehicles and the number of vehicles. The JSPT also addresses the minimization of makespan, which, although important, is not central in the context of internal logistics.

In the following sections, we propose to model our problem using the PDPTW formalism, which allows us to consider the different types of internal logistics operations generically.

3 Problem definition

Knowing the production schedule and the topology of the production workshop (i.e., the list of buffers B, their lower and upper capacities $[l_B, U_B]$, and the inter-buffer transportation times), it is possible to model all logistical activities as pickup-and-delivery tasks that must be carried out inside some specific time windows to ensure the smooth running of production. We can distinguish between two kinds of buffers: the output buffers that produce goods, which have to be picked up by the AMRs, and the input buffers, which consume goods that have been delivered by the AMRs. Assuming a deterministic production process, the knowledge of the production schedule allows us to predict for each input/output buffer the evolution of its inventory along the time horizon L. We refer to $I_B(t)$ as the inventory level of Buffer B at time t. Note that the notion of good picked-up or delivered to buffers is generic as it could correspond to components required for assembly operations, tools needed to carry out a production task, wastes produced during the production, intermediary products, and so on.

Algorithm 1 details how to compute the set of delivery time intervals of an input buffer B. The way to compute the picked-up intervals of the output buffers being quite similar, we do not detail it further. The algorithm starts with the computation of the number N of mandatory deliveries (see line (2)). Note that $I_B(L)$ refers to as the state of the inventory at the end of the production horizon L, without assuming any replenishment of the buffer. Note also that although every delivery activity is unitary, it does not forbid to supply several goods at the same time to the buffer, in which case several delivery activities are simply assumed completed at the same time. In lines (3)-(10), the time t_i where the delivery activity i can be started is computed (if the inventory level is initially lower than $u_B - 1$ then the activity i can immediately start). In lines (11)-(14), the time window $[a_i, b_i]$ associated with each delivery i is computed: b_i is simply the time where, if i is not yet performed then the inventory level will get lower than l_B.

Algorithm 1 functioning is illustrated in the example of Figure 1. The dotted yellow lines represent the lower and upper value $[l_B, u_B] = [2, 6]$ of the inventory of the buffer B. $I_B(t)$ is represented with the plain blue decreasing curve. The initial value of the inventory $I_B(0) = 4.7$ being lower than 5, the first item can be delivered at time $a_1 = 0$ and has to be delivered at the latest at time 9.6, when I_B becomes equal to l_B. Using the same reasoning, one can deduce that the second item can be delivered at the earliest at time a_1 when I_B becomes equal to 4 and at time b_1, at the latest, when $I_B(b_1) = l_B - 1$. In that example, $N = 4$ items must be delivered to avoid any stock-out.

Eventually, each task (i, j) is characterized by a pick-up buffer $B_i \in B$, a delivery buffer B_j, a pick time window $[a_i, b_i]$ and a delivery time window $[a_j, b_j]$. Two tasks (i, j) and (k, l) can involve identical buffers (i.e., $B_i = B_k$ or $B_j = B_l$).

The problem can then be represented by a complete directed graph $G = (V, A)$ where $V = 0, ..., 2n + 1$ is the set of vertices and A is the set of arcs. V is partitioned into two subsets: pick-up vertices $P = \{1, ..., n\}$ and delivery vertices

Algorithm 1: Compute delivery intervals

Data: $I_B(t)$ the inventory level of Buffer B (without replenishment), $[l_B, u_B]$ the lower and upper capacity of B.
Result: The set of delivery intervals $\{[a_i, b_i]\}$.

```
begin
    N ⟵ l_B − ⌊I_B(L)⌋
    for i ⟵ 1 to N + u_B − l_B − 1 do
        if I_B(0) < u_B − 1 then
            t_i ⟵ 0
        else
            Find t such that I_B(t) = u_B − i
            t_i ⟵ t
        end
    end
    for i ⟵ 1 to N do
        a_i ⟵ t_i
        b_i ⟵ t_{i+u_B−l_B−1}
    end
    return {[a_i, b_i]}
end
```

$D = \{n+1, ..., 2n\}$ (each node i is associated with node $n+i$ by one task). The fictitious vertices 0 and $2n+1$ model the origin and end of each vehicle's tour respectively (they do not correspond to any real location, as a vehicle tour can include as many trips to the depot as necessary). A transportation time t_{ij} is associated with each stop $(i, j) \in A$. Each node $i \in V$ must be served during a time interval $[a_i, b_i]$, the service time lasts s_i and the quantity required is q_i. This quantity is positive for the pick-up vertices, and negative for delivery ones. It is assumed that $s_0 = s_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$ and $q_{n+i} = -q_i, \forall i \in P$. The fleet is made up of a set K of K homogeneous AMRs ($K = \{1, ..., K\}$) of capacity Q_k, expressed in number of part kits.

Finding one feasible solution to the above problem consists in determining a route for each vehicle such that i) any vertex i is served once during the time interval $[a_i, b_i]$ (which ensures that any buffer never runs out of goods or never overpasses its maximum capacity) and ii) the vehicle's capacity is never exceeded. Note that, depending on the number of vehicles and their capacity, the problem may not admit any feasible solution. As an objective function, we choose to maximize the sum of the times of service, which tends to deliver the good as late as possible, while respecting the feasibility constraints. This criterion is in line with a WIP minimization viewpoint as it tends to maintain the buffers' inventories as low as possible.

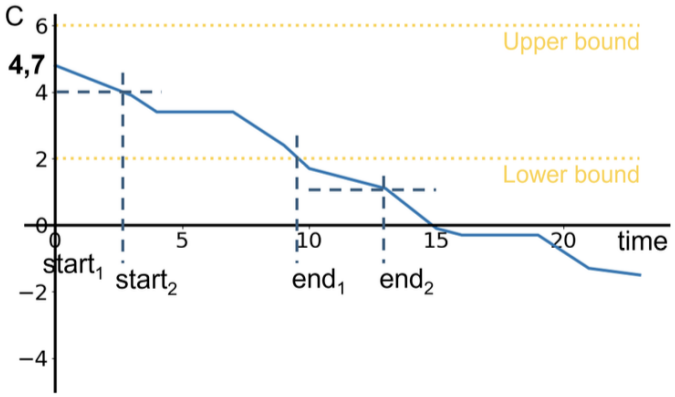

Fig. 1. Example of computation of delivery intervals

4 MILP model

The problem can be modeled as the MILP below, which is partly inspired by the non-linear formulation of PDPTW proposed by Toth and Vigo [29]. Decision variables T_i indicate the time of visit of vertex $i \in P \cup D$. The variable x_{ijk} is equal to 1 if vehicle k uses arc (i, j), 0 otherwise. Q_{ij} indicates the quantity transported on the arc $(i, j) \in A$. The objective function (1) maximizes the sum of the item delivery dates in a just-in-time logic. Constraints (2)-(5) are unchanged from Toth's original formulation. They indicate that at most K disjoint paths, each corresponding to the path of a vehicle, must start from vertex 0 to reach vertex $2n + 1$ and cover all the vertices. The constraints (6)-(8) make the link between T_i and x_{ijk} variables. Note that s_i refers to as the service time in i and t_{ij} to as the time to travel from i to j. Constraints (6) model minimum transportation times. Constraints (7) take into account time windows. Finally, constraints (8) express that, if there is a vehicle k traveling on the arc (i, j), then $T_j - T_i \geq s_i + t_{ij}$. The constant M_{ij} is a lower bound of $T_j - T_i$. The constraints (9)-(10) link the quantities carried on the arcs to the variables x_{ijk}. Constraints (9) ensure that the quantity conveyed on the arc is zero if no AMR uses the arc. Constraints (10) ensure flow conservation.

$$\text{(PDPTW)} \qquad \max \sum_{i \in P} T_{n+i} \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \qquad \forall i \in P \tag{2}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{n+i,jk} \qquad \forall i \in P, k \in K \tag{3}$$

$$\sum_{j \in V} x_{0jk} \leq 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} \qquad \forall i \in P \cup D, k \in K \tag{5}$$

$$T_{n+i} \geq T_i + s_i + t_{i,n+i} \qquad \forall i \in P \tag{6}$$

$$a_i \leq T_i + s_i \leq bi \qquad \forall i \in V \tag{7}$$

$$T_j - T_i \geq M_{ij} + (s_i + t_{ij} - M_{ij}) x_{ijk} \qquad \forall i \in V, j \in V, k \in K \tag{8}$$

$$0 \leq Q_{ij} \leq \sum_{k \in K} Q_k x_{ijk} \qquad \forall i \in V, j \in V \tag{9}$$

$$\sum_{j \in V} Q_{ji} - \sum_{j \in V} Q_{ij} = q_i \qquad \forall i \in V \tag{10}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i \in V, j \in V, k \in K \tag{11}$$

5 A greedy heuristic

To solve the MILP efficiently, it is often worthwhile to provide the solver with a feasible initial solution. For that purpose, we propose a greedy algorithm that tries to find a feasible solution. The solution is constructed iteratively by the algorithm, in a greedy manner: at each iteration, the list of tasks to be performed is updated and the algorithm enumerates all the tuples constituted by an activity (i, j), a vehicle k, and an insertion position in the route followed by k, r_k. Such a tuple is said feasible if:

1. there is enough time at the insertion position between the previous task $(i-1, j-1)$ and the next one $(i+1, j+1)$ to insert (i, j), i.e.: the time to travel from $j-1$ to i, then pick-up one item at i in the time window $[a_i, b_i]$, then to move from i to j, deliver the item in the time window $[a_j, b_j]$, to move from i to j, and travel finally to j to $i+1$ (obviously, the situations where $j-1$ and i, or j and $i+1$, are located at the same workstation, are particular cases);
2. There is enough capacity left in the vehicle to load an additional item.

Once all possible tuples have been checked, the best tuple is selected, i.e. the one minimizing the increase of the sum of the delivery times, and the insertion is performed. If at a given iteration, none of the combinations turns out to be feasible, then the heuristic fails. In that case, the number of vehicles K or their capacity Q_k should be increased. To evaluate the condition (1) efficiently, note that we actualize dynamically the earliest and latest starting time of the tasks belonging to the route r_k each time a new task is inserted (which can be done with a linear complexity). The overall complexity of the procedure is $\mathrm{O}((2n+1)^3)$, n being the number of pick-up and delivery tasks. Note that it is also possible to guide the heuristic using another criterion that

the one related to the WIP, e.g. the total distance covered by the robots, their load, and the lateness (if we allow it).

6 First results

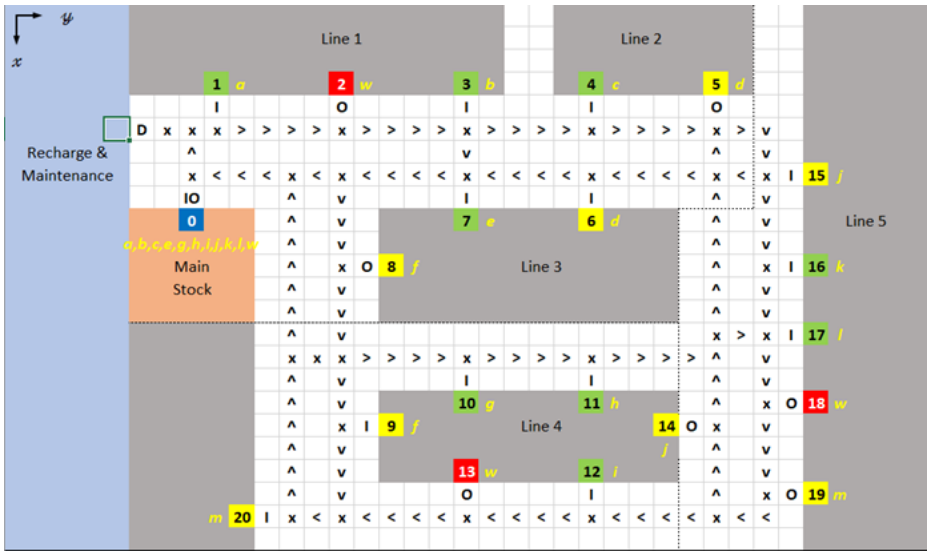

Fig. 2. Schematic representation of the workshop topology

To test the quality of the greedy heuristic, one of our MILP model, and determine its added value, we set up 3 problem instances having 4 vehicles and 8 buffers for the first, 6 vehicles and 14 buffers for the second, and 8 vehicles and 20 buffers for the last. In these instances, the depot is an additional workstation having an infinite input and output capacity. Each workstation corresponds to a specific buffer in the MILP model and has a minimum and maximum storage capacity. The consumption inside the buffers is spread over 20 minutes, time during which every workstation can consume up to around twice their respective maximum capacities.

Some workstations have a positive consumption meaning that their local inventory monotonically decreases, i.e. they are associated with the input buffers. The others have a negative consumption so that their local inventory monotonically increases over the planning horizon, i.e. they are associated with the output buffers. Let us recall that this modeling enables us to represent any internal logistic task (as long as the consumption of vehicle capacity by tasks can be expressed using the same scale of size): e.g., the components supplying tasks starting from the depot, the collecting and supplying of semi-finished products, and the retrieval of finished products or waste.

The vehicles are assumed all identical and have a capacity $Q_k = 4, \forall k$. Pick-up and delivery tasks are created thanks to Algorithm 1. Figure 2 shows the topology of the workshop that has been used to estimate the traveling time on the routes.

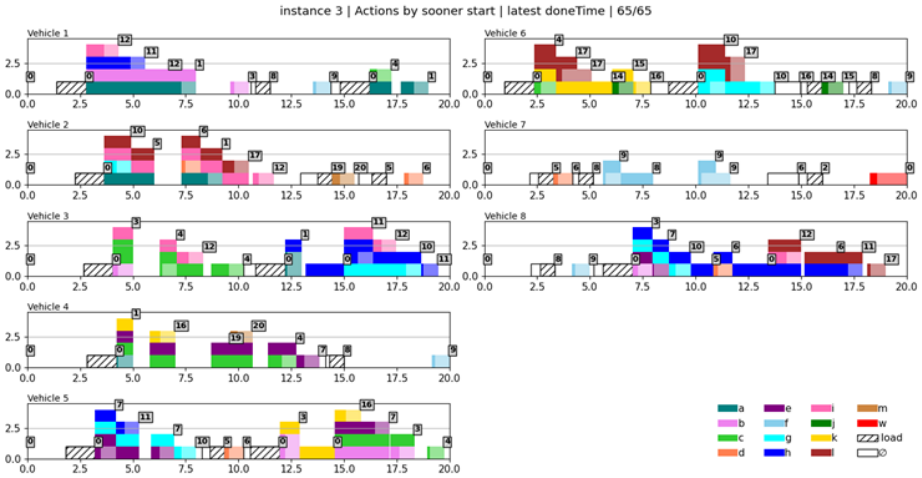

Fig. 3. A greedy solution for the third instance. Tasks are processed in ascending order of their earliest start. The sum of service times is maximized. The load factor is 70%.

The solution obtained by the heuristic for the third instance is displayed on the diagram of Figure 3, which provides an overview of the pick-up and delivery schedule for each vehicle, i.e. , the schedule of the transportation capacity. The number of units loaded (in this case, up to 4) into the vehicles over time is represented by colored rectangles. In order to better understand such diagram, let us analyze more tightly vehicle 2's schedule, which has been detailed in figure 4. The numbers framed in gray rectangles (1) indicate the location of the vehicle at the start of each task. Each colored rectangle (2) represents a different item. A bright-colored rectangle (3) represents movement from one location to another. A dull rectangle (4) represents the service time of a task to be performed. A hatched rectangle (5) represents the loading of an empty vehicle. A white rectangle (6) represents an empty trip. Eventually, no rectangle (7) means that the vehicle is waiting. Here, vehicle 2 starts empty in 0, it first waits, then loads 4 units a time 3: one green, one cyan, one pink, and one red. It then moves to workstation 10 to unload the cyan unit at time 5, then to workstation 5 where it waits to load an orange unit, which is unloaded at workstation 6 at time 8. It unloads the green unit in workstation 1, then the red unit in workstation 17. He waits a little before unloading in 12 the pink unit. It waits before loading a brown unit in 19, which it then unloads in 20. Finally, it loads an orange unit in 5 and waits a little before unloading it in 6, at time 19.

If we now launch the MILP solver (GUROBI in our case) with a time limit of 30 minutes, initializing it with the solutions found by the heuristic, we do observe an improvement in the performance of the WIP score. For the first instance, the heuristic score is 657 seconds, which is increased to 716 seconds (+9%) by the MILP (with a final gap equal to 4.85%). For the second instance, the heuristic score is 1032 seconds, which is increased to 1087 (+5.3%) by the MILP (with a gap equal to 8.21%). For the third instance, the new routes are shown in Figure **??**. While the heuristic offers a WIP score of 1315 seconds, the MILP increases this score to 1368 seconds (i.e., +4%), with a final gap equal to 12%. The MILP succeeded in systematically improving the WIP

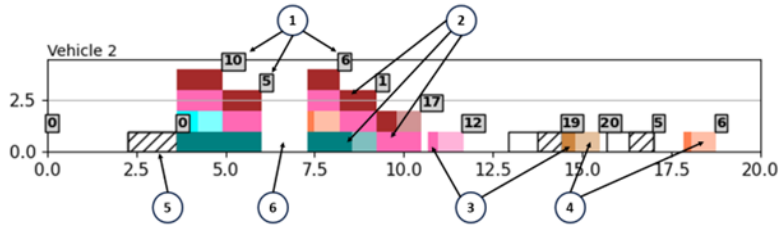

Fig. 4. The schedule of vehicle 2 in the greedy solution of the third instance

criterion in our three instances. However, we observe that the improvement is rapid at the start of the optimization, but tends to slow down after a few minutes, and may even freeze. We suspect that the solutions found are very close to being optimal and that the solver spends its time to prove the optimality and close the instance.

7 Experiments

In this section, the goal is to determine whether our approach can work on the benchmark instances proposed by Ulusoy et al. (see [23]) for the JSPT. This benchmark consists of 10 instances, named *jobset1* .. *jobset10*, having from 5 to 7 jobs visiting a set of 7 machines. The benchmark includes a matrix indicating the transport between each pair of machines. Two unit-capacity vehicles are responsible for transferring products from one machine to the other.

In our experiments, we aim to optimize the QoS expressed in terms of WIP minimization, like Gondran et al. in [26], assuming that the production schedule has been built up in an earlier phase and must now be respected. To establish an initial production schedule, we use an efficient hybrid approach, named ARIES, that mixes SAT/SMT and CP solvers for solving to optimality the pure job-shop part of the problem (see [30]), to minimize the makespan. Note that we simply take into account in solving the minimum time of transportation between the machines, assuming an infinite number of robots. Then, in the second step, we use our approach to solve the routing part of the problem, progressively increasing the number of AMRs, until the fleet size becomes large enough to carry out the various transportation tasks without causing any delay in the implementation of the optimal schedule. The objectives are twofold. First, we want to determine whether our approach is efficient enough to deal with this benchmark. Secondly, as the benchmark considers 2 single capacity vehicles, we would like to determine whether this number is really constraining, as it impacts the production makespan, or not, which would mean that the computational complexity of the instances will mainly lie on solving the job-shop part of the problem (not the routing part).

Results of the experiments are displayed in Table 1, in which we report for each instance: the time $T_{\text{CPU}}^{\text{JSPT}}$ spent by our job-shop solver to compute the optimal makespan, the optimal makespan $C_{\max}$ (assuming an infinite number of AMRs), the number of AMRs found by our heuristic to follow the production without any makespan deterioration, the time spent for computing the AMRs routes and improving them with our MILP, the value of our WIP indicator, and eventually the WIP GAP.

We first observe that our SAT/SMT and CP solver is very efficient in finding the optimal makespan, since it always returns an optimal schedule in less than 20 ms. Our heuristic helps us to initially determine the number of vehicles needed to implement the initial schedule without deteriorating the makespan. Interestingly, this number, which is limited to 2 in JSPT instances, must be increased to 4 or 5 to avoid any degradation of the makespan. We also observe that our MILP solves JSPT instances quite quickly compared to the time experienced in instances presented in Section refFirstResults. One explanation is that the heuristic already provides routing plans that are almost optimal so that the MILP therefore does not need to modify them much to make them optimal. Nevertheless, it should be pointed out that our heuristic gives us no guarantee regarding the number of vehicles. In a future study, it might be interesting to study the same problem in the context of bi-objective optimization, where decision-makers are looking for a compromise between minimizing the makespan and minimizing the number of vehicles.

Instance name	$T_{\text{CPU}}^{\text{JSPT}}$(ms)	$C_{\max}$(s)	#AMR	$T_{\text{CPU}}^{\text{MILP}}$ (s)	WIP	GAP
jobset1	7	220	5	9.95	5536	0%
jobset2	6	172	3	4.46	3329	0%
jobset3	12	166	4	0.95	3286	0%
jobset4	13	127	4	3.3	2686	0%
jobset5	11	116	4	0.99	2216	0%
jobset6	13	204	4	12.18	4909	0%
jobset7	14	154	5	34.9	4636	0%
jobset8	16	307	4	7.28	7668	0%
jobset9	12	194	4	3.52	4485	0%
jobset10	15	238	4	7.16	6114	0%

Table 1. Results obtained for the JSPT instances.

8 Conclusion

Adopting the formalism of a pick-up and delivery problem with time windows, we provide in this paper a generic way to model the various internal logistics activities that can appear in a production environment. We propose two solving procedures (that can be sequenced): a greedy insertion heuristic and a MILP formulation. The correctness of the greedy heuristic and MILP has been established on medium-sized instances. The greedy heuristic allows for determining feasible solutions quickly, which can eventually speed up the MILP-solving process. The use of a MILP solver gives interesting results in our first experiments, although the optimization process can be very long. These impressions were confirmed by the results obtained on the JSPT benchmark. Our heuristic succeeded in quickly sizing the fleet and obtaining a fairly good initial routing plan. MILP also enables us to improve this initial plan, if necessary, and guarantee its optimality.

As far as future research directions are concerned, although our MILP enables us to improve the quality of some initial solutions (the number of vehicle being fixed),

we do not believe that it will be effective for solving large-scale problem instances, or even for tackling the problem in its dynamic version. Indeed, the number of logistical operations involved in implementing a production plan during a production campaign can be very high, which could cause the MILP to be very inefficient. Furthermore, as mentioned above, production is often disrupted by many unpredictable events, and it is therefore necessary to maintain real-time consistency between production plans and routing plans, which falls outside the scope of our method. It should also be noted that since the implementation of the routing plan is also subject to uncertainties (for example, it may take longer than expected for a robot to move), the production plan may also be impacted. Therefore, production scheduling and vehicle routing should really be considered in an integrated manner. Decomposition methods such as dynamic branch-and-price approaches, similar to those used in the literature for dynamic PDP-TW, could be a promising direction of research to achieve this integration.

References

1. Rodrigo Bernardo, João M. C. Sousa, and Paulo J. S. Gonçalves. Survey on robotic systems for internal logistics. *Journal of Manufacturing Systems*, 65:339–350, October 2022.
2. Vigo Daniele and Toth Paolo, editors. *Vehicle Routing*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, November 2014.
3. G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959.
4. Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. Chapter 5: The Vehicle Routing Problem with Time Windows. In *Vehicle Routing*, MOS-SIAM Series on Optimization, pages 119–159. Society for Industrial and Applied Mathematics, November 2014.
5. Luca Bertazzi and M.Grazia Speranza. Inventory routing problems: An introduction. *EURO Journal on Transportation and Logistics*, 1, 12 2012.
6. Luca Bertazzi and M.Grazia Speranza. Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, 2, 08 2013.
7. Leandro C. Coelho, Jean-François Cordeau, and Gilbert Laporte. Thirty years of inventory routing. *Transp. Sci.*, 48:1–19, 2014.
8. Moshe Dror, M. Ball, and Bruce Golden. A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, 4:3–23, 12 1985.
9. Ann M. Campbell, Lloyd W. Clarke, and Martin W. P. Savelsbergh. 12. Inventory Routing in Practice. In *The Vehicle Routing Problem*, Discrete Mathematics and Applications, pages 309–330. Society for Industrial and Applied Mathematics, January 2002.
10. Leandro Callegari Coelho, Jean François Cordeau, and Gilbert Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270–287, 2012.
11. Yufen Shao, Kevin C. Furman, Vikas Goel, and Samid Hoda. A hybrid heuristic strategy for liquefied natural gas inventory routing. *Transportation Research Part C: Emerging Technologies*, 53:151–171, 2015.
12. Noor Moin, Said Salhi, and Nur Arina Bazilah Aziz. An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *International Journal of Production Economics*, 133:334–343, 09 2011.

13. Anis Mjirda, Bassem Jarboui, Rita Macedo, and Saïd Hanafi. A variable neighborhood search for the multi-product inventory routing problem. *Electronic Notes in Discrete Mathematics*, 39:91–98, 2012.
14. Y. He, C. Artigues, C. Briand, N. Jozefowiez, and S.U. Ngueveu. A Matheuristic with Fixed-Sequence Reoptimization for a Real-Life Inventory Routing Problem. *Transportation Science*, 54(2):355–374, 2020.
15. Cyril Briand, Yun He, and Sandra Ulrich Ngueveu. Energy-efficient planning for supplying assembly lines with vehicles. *EURO Journal on Transportation and Logistics*, 7(4):387, 2018.
16. Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, and François Soumis. 9. VRP with Pickup and Delivery. In *The Vehicle Routing Problem*, Discrete Mathematics and Applications, pages 225–242. Society for Industrial and Applied Mathematics, January 2002.
17. Jan Dethloff. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spektrum*, 23:79–96, 02 2001.
18. Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: Pickup-and-Delivery Problems for Goods Transportation. In *Vehicle Routing*, MOS-SIAM Series on Optimization, pages 161–191. Society for Industrial and Applied Mathematics, November 2014.
19. Stefan Ropke and Jean-François Cordeau. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 43(3):267–286, 2009. Publisher: INFORMS.
20. Martin Savelsbergh and Marc Sol. Drive: Dynamic Routing of Independent Vehicles. *Operations Research*, 46(4):474–490, August 1998. Publisher: INFORMS.
21. H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001*, pages 160–167, 2001. Conference Name: 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001 ISBN: 9780769514178 Place: Dallas, TX, USA Publisher: IEEE Comput. Soc.
22. Carlo S. Sartori and Luciana S. Buriol. A study on the pickup and delivery problem with time windows: Matheuristics and new instances. *Computers & Operations Research*, 124:105065, December 2020.
23. Gündüz Ulusoy, Funda Sivrikaya-Şerifoğlu, and Ümit Bilge. A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. *Computers & Operations Research*, 24(4):335–351, April 1997.
24. L. Deroussi, M. Gourgand, and N. Tchernev. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Research*, 46(8):2143–2164, April 2008. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207540600818286.
25. Philippe Lacomme, Mohand Larabi, and Nikolay Tchernev. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics*, 143(1):24–34, May 2013.
26. Matthieu Gondran, Marie-José Huguet, Philippe Lacomme, Alain Quilliot, and Nikolay Tchernev. A Dial-a-Ride evaluation for solving the job-shop with routing considerations. *Engineering Applications of Artificial Intelligence*, 74:70–89, September 2018.
27. Lucas Berterottière, Stéphane Dauzère-Pérès, and Claude Yugma. Flexible job-shop scheduling with transportation resources. *European Journal of Operational Research*, 312(3):890–909, February 2024.

28. Amir Hosseini, Alena Otto, and Erwin Pesch. Scheduling in manufacturing with transportation: Classification and solution techniques. *European Journal of Operational Research*, 315(3):821–843, 2024. Publisher: Elsevier.
29. P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Society for Industrial and Applied Mathematics, USA, 2014.
30. Arthur Bit-Monnot. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. In *ECAI 2023*, pages 255–262. IOS Press, 2023.