A constraint programming approach for planning items transportation in a workshop context

Valentin Antuori[1,2], Emmanuel Hebrard[1], Marie-José Huguet[1],
Siham Essodaigui[2], Alain Nguyen[2]

[1] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
{vantuori,hebrard,huguet}@laas.fr
[2] Renault, France
{valentin.antuori, siham.essodaigui, alain.nguyen}@renault.fr

Keywords: Constraint Programming, Single-machine Scheduling, TSP.

1 Introduction and related works

We are interested in transporting items in a workshop, from their production spots, to their consumption spots. For each type of items, there is one consumption machine, one production machine and four dedicated carriage trolleys, of limited capacity. Initially, there is one full trolley and one empty trolley in front of each machine. This problem comes from a real industrial problem, in the context of a car manufacturer workshop. When a trolley is full, an operator must transport it. He can transport several trolleys at the same time, making a train with them, but there is a limit on the maximal size on the train he can form. The production rate, and the consumption rate for a type of item is the same, therefore, when a carriage is full at a production spot, another carriage is empty at the associated consumption spot. The operator is also in charge of bringing back empty trolleys from their consumption machines, to production machines. The production rate, and the capacity of the trolley, permit to define a production cycle for each pair of machine. A production cycle is the time taken by a production (resp. consumption) machine, to fill (resp. empty) a trolley. The end time of a production cycle correspond to the beginning of the next cycle. For each pair of machines, and for each of their cycles considered until the time horizon, the goal is to deliver the trolleys which have been filled during the previous cycle to their consumption spots, and the trolleys which have been emptied during the previous cycle to their production spots.

The aim is to plan a route of the operator in order to satisfy pickup and delivery requests with time windows constraints. A particularity of this problem is the periodicity of the requests. Indeed, all the pickups and deliveries of a given item are almost completely sequenced. However, the relative order of pickups and deliveries of distinct items still need to be decided, potentially over a very long planning horizon.

This problem can be seen as a Pickup and Delivery Problem, with a single capacited vehicle and time windows. It is part of the one-to-one pickup and delivery family (which means that each pickup has a unique destination and conversely, each delivery has a unique source)(Cordeau et al. 2008). The single vehicle case comes from a Traveling Salesman Problem (TSP) in which precedences were added between clients. Since then, several additional constraints have been considered, such as time windows, limited capacity or LIFO loading (deliveries must respect a last-in-first-out rule). One can find complete survey on pickup and delivery problem in (Parragh et al. 2008) and (Berbeglia et al. 2007). There are temporal specificities in our problem. Indeed, each request has a twin request with the same time window: for each request from point A to point B to transport a full (resp. empty) trolley, there is a request from B to A for the empty (resp. full) trolley to do during

the same time. Moreover, each request is repeated over time until the horizon, that is, each due date corresponds to a release date for the next request on the same pair of machine.

Routing problems can often be seen as scheduling problems with sequence-dependent setup times. In fact, time windows and precedences constraints, as well as the fact that there is no objective to optimize might suggest that the problem more adapted to scheduling technologies (Beck et al. 2003). From a scheduling point of view, it is a single resource (the driver of the train) scheduling problem. We have 4 types of activities: pickup and delivery of a full and an empty trolley which can't overlap. There is a precedence between a pickup and its delivery, and each activity has a time window. Travel times between two activities are then sequence-dependent. Finally the length of the train can be seen as a reservoir resource with limited capacity, which is filled by pickups, and emptied by deliveries.

2 Constraint Models

We propose two constraint models to deal with this problem. One TSP-oriented based on *next* variables, and one scheduling oriented. In the scheduling model, for each operation $i \in T = \{1, 2, ..., n\}$, we define the variable $start_i$ as the starting date of the operation i. Moreover we introduce for each pair of operations $i, j \in T$, a Boolean variable x_{ij} which represent the relative ordering of the two operations. The link between these two sets of variables is made with with the following constraint:

$$x_{ij} = \begin{cases} 1 \Leftrightarrow start_j \geq start_i + tt_{ij} + p_i \\ 0 \Leftrightarrow start_i \geq start_j + tt_{ji} + p_j \end{cases}$$

with tt_{ij}, the travel time between operation i and j, and p_i the processing time of i. In fact, we do not need a Boolean variable for every pair of tasks, and can easily reduce the model. For all pair of operation $i, j \in T$, such as i precede j, we avoid creating the variable x_{ij}, and directly post the following constraint : $start_j \geq start_i + tt_{ij} + p_i$.

Since production cycles are often much shorter than the planning horizon, the problem involves a lot of precedences. Therefore, we can drastically reduce the size of the model. Moreover, we observe that there are only two sensible ways to schedule the four activities of a production cycle (pickup of the full trolley, pickup of the empty trolley, and the corresponding deliveries). First, obviously pickups must precede their respective deliveries. Second, since the second pickup and the first delivery take place at the same spot, it is always preferable (w.r.t. the capacities and the time windows) to do the first delivery before the second pickup. Therefore, there are only two possible orders: the operator may either first pickup and deliver the full trolley, or first pickup and deliver the empty trolley. Hence, only one variable is needed for these four tasks. In order to deal with the constraint on the length of the train, we use the balance constraint introduced in (Laborie 2003).

The other model is inspired by the constraint model for TSP with time windows proposed in (Ducomman et al. 2016). For each operation i, there is a variable $next_i$ that indicates which operation directly follows i. Additional variables are needed in order to post redundant constraints. pos_i indicates the position of the operation i in the sequence of operations, and x_{ij} has the same semantic than in the first model. We add another variable $trainL_i$ which represent the length of the train before the operation i. Our model only differ by the adding of the following constraints :

$$trainL_{next_i} = trainL_i + l_i \qquad \forall i \in T \cup \{0\} \tag{1}$$

$$trainL_0 = 0 \tag{2}$$

$$pos_{del_i} > pos_i \qquad \forall i \in P \tag{3}$$

with l_i, the length of the trolley of the operation i (negative length for delivery), del_i is the associate delivery of the operation i, and P is the set of pickup operations. Constraint (1) and (2) represent the accumulation on the train length during the sequence of operations and use the ELEMENT constraint. The last constraint (3) ensures that each pickup precedes its delivery. In addition, we use the CIRCUIT constraint to enforce the Hamiltonian circuit.

3 Experiments

In order to compare the two models, and in addition to the industrial instances, we generated random instances [1]. We tried to generate only feasible instances, but we cannot guaranty that all instances are. There are 4 categories of instances (A, B, C, D). Instances A contain the least dense instances, i.e. with the least number of operations. They have also more pairs of machines with the same production cycle. Conversely, instances D are the most dense, and most asynchronous. We generate 10 instances of each category, and for each of them we consider 3 different temporal horizons, leading to 120 instances in total. The two models were implemented in the constraint solver Choco [2].

We observed that variable orderings following the chronological sequence tend to be more efficient. For instance, in the TSP model, building a tour by selecting the nodes from the first to be visited (pos_1) to the last to be visited (pos_n) was more effective than assigning the positions in a different order. In the case of the scheduling model, we use the following strategy: we say x_{ij} dominate x_{kl}, iff $min(start_i) \leq min(start_k)$ and $min(start_j) \leq min(start_l)$ with at least one strict inequality or $min(start_i) \leq min(start_l)$ and $min(start_j) \leq min(start_k)$ with at least one strict inequality, with $min(start_i)$ the minimum value of $start_i$. When choosing a variable to assign, we look for the boolean variables which is not dominated by any other variables, ties are broken randomly. We noticed a slight improvement when in addition we gave priority to the variables which order operations in the same pair of machines before the other ones. That is, we don't deal with a variable x_{ij} if the variable ordering the three remaining activities linked to operation i, and the variable ordering the three remaining activities linked to operation j are not instantiated. Generic heuristics such as *dom/wdeg* (Boussemart et al. 2004) were significantly less effective than these simple orderings.

Similarly, we explore first the branch that minimizes the start time of the next operation. In the TSP model, it means assigning pos_i to j such that the minimum value of $start_j$ is minimal. In the scheduling model, it means assigning x_{ij} to 1 iff minimum value of $start_i$ is lower than the minimum value of $start_j$. For both, and thanks to the propagation on the *start*'s variables, this heuristic acts more or less like a nearest neighborhood approach and takes advantage of the travel time between activities.

We ran each instance 10 times with randomized heuristics: if there are ties, they are broken randomly, and if not, one of the two best choices is chosen randomly with equal probabilities. We also used a restart strategy (the Luby sequence (Luby et al. 1993)) to improve the result. Each run has a timeout of 1 hour.

Table 1 shows the results. The first column denotes the category of the instance and second column denotes the temporal horizon. For the two models (denoted by Scheduling and TSP), 3 indicators are given, the number of solved instances (in average on 10 runs), the average time, and the numbers of fails for solved instances. We observe that the scheduling-based model can solve more instances in every category, and is faster in average. The average number of fails for the TSP model shows the relative slowness of that model. Most of the instances are unsolved.

[1] Available on https://github.com/AntuVal/SPDPTW

[2] Prud'homme, C., Fages, J.-G. & Lorca, X. (2017), Choco Documentation.

Table 1. Comparison of the two models on the generated instances

Cat.	Hor.	Scheduling			TSP			Nb Inst.
		NbSolve	Time	NbFail	NbSolve	Time	NbFail	
A	15000	9.3	44.11	174195	9.0	18.39	13081	10
	25000	8.5	52.74	143564	7.9	140.92	9445	10
	40000	8.0	36.48	29665	7.1	149.71	3803	10
B	15000	4.8	390.61	301231	2.1	943.88	18340	10
	25000	2.5	381.94	203849	0.7	1017.76	2015	10
	40000	2.0	402.88	419479	0.1	1929.71	283	10
C	15000	4.3	534.60	387635	2.0	541.68	14549	10
	25000	1.4	1264.03	352172	0.1	1693.80	1521	10
	40000	0.0	-	-	0.0	-	-	10
D	15000	2.2	495.55	112918	0.8	1271.75	9484	10
	25000	0.2	2565.44	45153	0.0	-	-	10
	40000	0.0	-	-	0.0	-	-	10

4 Conclusions

We introduced a one machine scheduling problem with several additional constraints. That problem is not new in the definition of its constraints, but the temporal structure of the instances makes it challenging. We proposed a randomly generated set of benchmark instances, whose a large number stay unsolved. We observed that a light model based on ordering pair of activities works better than a TSP-based model, which does not scale to industrial instances.

As a large number of the generated instances remains unsolved, there are still works to do. We are currently working on a Large Neighborhood Search (LNS) in order to improve these results. In this scheme, we relax the due date of each task, and instead we minimize the maximum lateness. Then during a LNS move, a subset of operations can be withdrawn from the sequence and re-inserted so as to minimize this objective. We also aim to deal with the entire problem, which is the team sizing for the whole shop. The goal is to plan a route through the pair of machines, minimizing the number of trains in the shop.

References

Beck, J. C., Prosser, P. & Selensky, E. (2003), Vehicle routing and job shop scheduling: What's the difference?, ICAPS, pp. 267–276.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. & Laporte, G. (2007), 'Static pickup and delivery problems: a classification scheme and survey', *TOP* **15**(1), 1–31.

Boussemart, F., Hemery, F., Lecoutre, C. & Sais, L. (2004), Boosting systematic search by weighting constraints, ECAI, pp. 146–150.

Cordeau, J.-F., Laporte, G. & Røpke, S. (2008), Recent models and algorithms for one-to-one pickup and delivery problems, *in* 'The Vehicle Routing Problem', Springer, pp. 327–357.

Ducomman, S., Cambazard, H. & Penz, B. (2016), Alternative Filtering for the Weighted Circuit Constraint: Comparing Lower Bounds for the TSP and Solving TSPTW, AAAI, pp. 3390–3396.

Laborie, P. (2003), 'Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results', *Artificial Intelligence* **143**(2), 151–188.

Luby, M., Sinclair, A. & Zuckerman, D. (1993), 'Optimal speedup of las vegas algorithms', *Information Processing Letters* **47**, 173–180.

Parragh, S. N., Doerner, K. F. & Hartl, R. F. (2008), 'A survey on pickup and delivery problems', *Journal für Betriebswirtschaft* **58**(2), 81–117.