A new robust approach for a production scheduling and delivery routing problem

Azeddine Cheref, Christian Artigues, Jean-Charles Billaut

A new robust approach for a production scheduling and delivery routing problem ⋆

Azeddine Cheref [*,**] **Christian Artigues** [**]
Jean-Charles Billaut [*]

[*] *Université François-Rabelais Tours/CNRS, 64 av. J. Portalis, 37200 Tours, France (e-mail: {azeddine.cheref,jean-charles.billaut}@univ-tours.fr)*
[**] *LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France (e-mail: artigues@laas.fr)*

Abstract: In a supply chain, two essential elements are the production and the distribution. These two problems used to be solved separately, but it it well known that considering them jointly may lead to a global optimization of the supply chain performances. In this paper, we incorporate the delivery plan of a vehicle into a single machine scheduling problem, representing a single manufacturing facility. We consider the simplified case where there is only one vehicle available to serve the customers, with infinite capacity. However, we assume that the data are known with uncertainty. The objective is to find a schedule and a delivery plan so that a *robustness criteria* is minimized, under a scenario-based uncertainty modeling. In contrast with standard robust optimization approaches for scheduling, we do not propose a single complete solution to the problem that has to be feasible w.r.t. all scenarios and minimizes a worst-case criterion over the scenarios. Instead, we adopt the recoverable robustness framework that considers first-stage decisions and second-stage recovery options. We propose for the first-stage a set of solutions by using the concept of groups of permutable jobs. At second stage, a greedy and *online* recovery algorithm exploits the revealed information about the jobs available to be scheduled or to be delivered at decision time. We propose two tabu search algorithms, one based on the standard robust optimization scheme and one based on the new approach. We compare the two robust heuristics on a set of randomly generated problem instances.

Keywords: Integrated production scheduling and delivery routing, robust optimization, recoverable robustness, groups of permutable jobs, tabu search

1. INTRODUCTION

More and more research is devoted to integrated production scheduling and delivery problems, see for instance Chen (2010). However most of the work that has been done in the literature consider deterministic problems. However such integrated problems are highly subject to uncertainty. Data such as processing times or release dates are generally not known with an absolute certitude, and for delivery problems, transportation durations are time dependent and subject to fluctuations due to traffic. In this context, it is generally important to find a solution to the problem which is not only a good solution, but also a solution that keeps a good performance in presence of uncertainties. Robustness considerations in scheduling are frequently treated in the literature and it is not possible to make here an exhaustive review of the state-of-the-art. We voluntarily restrain our work to robust scheduling approaches where no probability distribution is available for the uncertain data. Instead there is a set of possible scenarios for the input data. In particular we consider, as a reference method, the standard robust discrete optimization framework proposed by Kouvelis and Yu (1997) (see also Bertsimas and Sim (2003)) . In this framework, a complete sequence of jobs on each machine is the output of the robust scheduling method. The uncertainty is represented by a discrete finite set of scenarios, each scenario corresponding to a deterministic value of the problem parameters. On a particular scenario, the performance of the sequence is generally measured by computing the earliest start schedule compatible with the prescribed sequence and the realized scenario. Hence, the worst case performance of a sequence on the scenario set can be computed a priori. The well-known problem with this framework is its conservativeness, as mentioned by Bertsimas and Sim (2004), due to the fact that issuing a complete sequence lacks flexibility w.r.t. the realized scenario.

As an alternative we propose to extend these framework by integrating two other families of approaches for scheduling under uncertainty. The first family originates from production scheduling, and was described in Billaut and Roubellat (1996). It aims at computing on a machine a sequence of groups of permutable jobs such that any permutation of the job inside each group yields a feasible job sequence. The final sequence is then selected in real time so as to better react to disruptions. In Wu et al. (1999), the author

⋆ This work was supported by the financial support of the ANR ATHENA project, grant ANR-13- BS02-0006 of the French Agence Nationale de la Recherche.

have shown through simulation of various processing time disturbances that computing a static way such a solution structure and then allowing the remaining decisions to be taken dynamically via dispatching rules yields promising results in a job-shop scheduling setting. The second family of approaches is based on recoverable robustness. This framework, defined in Liebchen et al. (2009), aims at reducing the conservativeness of robust optimization. Recoverable robustness considers two sets of decision variables and consists in providing an assignment for the first-stage decision variables and a family of recovery algorithms such that, for each scenario, there exists one recovery algorithm in the family that computes feasible assignments for the second-stage decision variables. Recently the recoverable robustness framework has been successfully applied as such to a scheduling problem by Caprara et al. (2014) and to vehicle routing problem by Agra et al. (2013) that used the close concept of adjustable robustness as proposed by Ben-Tal et al. (2004).

To integrate these frameworks, we consider as in the standard robust scheduling approach a set of finite scenarios such that a scenario corresponds to a realization of uncertain parameters. We address a single machine scheduling problem, representing a single manufacturing facility in which a set of jobs with release dates is scheduled non preemptively. In addition, there is one vehicle available to deliver the jobs, with infinite capacity which has to be routed according to a travel time matrix between job locations. The vehicle takes a batch of completed jobs and delivers the customers at each job location preferably before the job delivery time and then return to the manufacturing facility to select another batch and, so on until all jobs are delivered. All parameters are assumed to be uncertain, i.e. job release dates, job processing times, job delivery dates and vehicle travel times The objective is to find a schedule and a delivery plan so that the maximum lateness over all jobs and all scenarios is minimized.

In the recoverable robustness framework, we propose to compute, as the output of the first decision level, a set of groups of permutable jobs on the machine, a set of batches and, inside each batch, a set of permutable job deliveries. To evaluate this first set of decision, a second level decision set adapts to the realized scenario in a greedy and online manner : by scheduling the production jobs inside each production group according to the earliest release date criterion (jobs are schedule as soon as they are available) and by scheduling the delivery jobs according to the nearest neighbor criterion (the vehicle always delivers the nearest job). This limited recovery algorithms simulates a real time decision making process which implies that a decision has to de made as soon as a portion of the scenario is revealed in an online framework. This means that when a group of production or delivery jobs is completed, the recovery algorithms wait for the next job to be released in the new group in the realized scenario for the production part or for the nearest job to visit according to the realized scenario for the delivery part.

In this work we formally present the standard robust and the online recovery robust integrated scheduling an delivery routing problem in Section 2. Then we present two tabu search heuristic, one implementing the standard robust scheme and one implementing the online recovery scheme in Section 3. In Section 4 we provide computational experiments to validate our proposal. Concluding remarks and directions for further research are pointed out in Section 5.

2. STANDARD ROBUST VS ONLINE RECOVERABLE ROBUST APPROACH FOR INTEGRATED PRODUCTION SCHEDULING AND DELIVERY ROUTING

Following the classical three-field notation of scheduling problems **?**, Z-L. Chen introduces in Chen (2010) a five-field notation $\alpha|\beta|\pi|\delta|\gamma$ for the problems where scheduling and routing are integrated at an operational level. In this notation, α, β and γ correspond to the machine configuration, scheduling constraints and objective function respectively, similarly to the notation of scheduling problems. Fields π and δ specify the characteristics of the delivery process and the number of customers, respectively. In the following, we focus on a single machine environment ($\alpha = 1$), with *routing* as a delivery problem (i.e. solving a vehicle routing problem is part of the decision), we have a single vehicle with unbounded capacity and n customers to deliver ($\pi = \{routing, V(1, \infty)\}$, $\delta = n$).

We consider a set $\mathcal{J} = \{J_1, ..., J_n\}$ of n jobs to schedule on a single machine representing a manufacturing facility (plant). We denote by p_j the processing time of J_j, r_j the release date of J_j that corresponds to the earliest possible start time of production and d_j the delivery due date of J_j, i.e. the date at which J_j is supposed to be delivered to the customer. It is assumed that the storage capacity of jobs after production is not limited. However, we also consider that the jobs have strict due dates, which is equivalent to say that the maximum lateness is bounded. Each job J_j is associated with a customer location j, and 0 denotes the location of the plant. $t_{0,j}$ denotes the travel time between the plant and the customer location j and $t_{i,j}$ is the travel time between customer locations i and j ($\forall i, j,\ 1 \leq i, j, \leq n$). Variables C_j denote the *production completion time* of J_j at the plant ($\forall j,\ 1 \leq j \leq n$). Variables D_j denote the *delivery completion time* of J_j, i.e. the date of delivery at the customer location ($\forall j,\ 1 \leq j \leq n$). Remember that jobs are delivered by a single vehicle that may return to the plant several times (as in the multi-trip traveling salesman problem) with unlimited capacity (capacity $K = n$). Without uncertainty considerations, the problem can be denoted by $1|r_j|routing, V(1, \infty)|n|\gamma$ where γ is the objective function. In the following, we have $\gamma = L_{\max}$ where $L_{\max}$ denotes the maximum delivery lateness, defined by $L_{\max} = \max_{1 \leq j \leq n}(D_j - d_j)$. The deterministic problem is to find a production sequence of jobs (equivalently, to determine jobs production completion times), a batching of jobs and a route for each batch, so that the objective function is minimized. A batch can only start when all the jobs of the batch are completed.

We assume that some uncertainty affect the input data and that our objective is to find a robust solution. To represent the uncertainty, we use a scenario based approach similar to the one introduced in Kouvelis and Yu (1997). The data associated to the jobs and to the transportation times between sites vary according to predefined scenarios. We denote by $\mathcal{S}$ the set of possible scenarios over the planning

horizon. Now, we say that to each job J_j is associated a release date r_j^s, a processing time p_j^s and a due date of delivery d_j^s for each scenario $s \in \mathcal{S}$. The matrix of travel times $\mathcal{T}^s = (t_{i,j}^s)_{0 \leq i \leq n, 0 \leq j \leq n}$ gives the transportation time from any site i to any site j under scenario $s \in \mathcal{S}$.

We denote by C_j^s the production completion time of job J_j under scenario s and by D_j^s its delivery completion time under scenario s. The maximum lateness of J_j under scenario s is denoted by $L_j^s = D_j^s - d_j^s$. Under uncertainty, the maximum lateness is $L_{\max} = \max_{s \in \mathcal{S}}(\max_{1 \leq j \leq n} L_j^s)$.

We now formally define the concept of online recoverable robustness. Let us consider the standard robust optimization problem that we can denote as

$$\min_{x \in \cup_{s \in \mathcal{S}} \mathcal{X}_s} \max_{s \in \mathcal{S}} f(x, s)$$

where $\mathcal{S}$ is the set of scenarios, $\mathcal{X}_s$ is the set of solutions feasible for scenario s and $f(x, s)$ is some performance indicator of solution $x \in \mathcal{X}_s$ under scenario $s \in \mathcal{S}$. This definition implies that x has to be feasible for any scenario, inducing a high degree of conservativeness Bertsimas and Sim (2004). The concept of recoverable robustness Liebchen et al. (2009) is often considered in the literature as a two-stage extension to the robust optimization problem, in which the (first-stage) solution x can be modified in a limited way in the second-stage to fit with the realized scenario to obtain solution $y = A(x, s)$ where $A(x, s)$ is the application-dependent recovery algorithm of solution x under scenario s. Hence the recoverable robust optimization problem becomes:

$$\min_{x \in \mathcal{X}} \max_{s \in \mathcal{S}} f\left(A(x, s), s\right)$$

where $\mathcal{X}$ is the scenario-independent set of first-stage solutions, which is a more realistic definition in practical applications. Note that x does not necessarily assign all the decision variables of the problem. As in stochastic programming x can be only the subset of first-stage decision variables, while algorithms $A(x, s)$ returns a complete solution, giving values to both first-stage and second stage decision variables.

Instantiating this concept to our integrated scheduling and vehicle routing problem, we define two different robust problems by using different definition of the first stage decisions x and of the second-stage online algorithm A.

The standard robust integrating scheduling and delivery routing problems defines $x = (\pi, \mathcal{B}, \Sigma)$ where π a complete of the jobs on the machines $\mathcal{B}$ a batching (i.e. a partition of the jobs) and $\Sigma = \{\sigma(B), B \in \mathcal{B}\}$ the complete ojob delivery sequence inside each batch $B \in \mathcal{B}$). Algorithm $A(x, s)$ is a simple completion time adjustment algorithm that sets feasible completion times C_j^s and D_j^s according to the realized parameters following the prescribed production and delivery sequences.

We illustrate the robust optimization scheme with the following example.

The matrices $(t_{i,j}^1)$ and $(t_{i,j}^2)$ are the following:

$s = 1$	J_1	J_2	J_3	J_4	J_5
r_j^1	0	7	3	4	3
p_j^1	3	4	1	2	4
d_j^1	11	18	9	10	17
$s = 2$	J_1	J_2	J_3	J_4	J_5
r_j^2	3	3	0	1	7
p_j^2	2	5	1	3	3
d_j^2	9	17	10	11	18

$$(t_{i,j}^1) = \begin{pmatrix} 0 & 2 & 4 & 3 & 2 & 2 \\ 2 & 0 & 3 & 2 & 1 & 3 \\ 3 & 3 & 0 & 3 & 2 & 2 \\ 3 & 2 & 3 & 0 & 1 & 3 \\ 2 & 1 & 2 & 1 & 0 & 2 \\ 2 & 3 & 2 & 3 & 2 & 0 \end{pmatrix} \quad (t_{i,j}^2) = \begin{pmatrix} 0 & 3 & 2 & 2 & 2 & 3 \\ 3 & 0 & 4 & 1 & 2 & 3 \\ 2 & 4 & 0 & 4 & 3 & 2 \\ 2 & 1 & 4 & 0 & 2 & 4 \\ 2 & 2 & 3 & 2 & 0 & 4 \\ 3 & 3 & 2 & 4 & 4 & 0 \end{pmatrix}$$

Let consider the schedule $(3, 4, 1, 5, 2)$ for the production phase, two batches composed by jobs $\{1, 3, 4\}$ and $\{2, 5\}$ with routings $(4, 3, 1)$ and $(5, 2)$, respectively. The production and transportation phases are illustrated in Fig. 1 for each scenario. t_r denotes the departure time of tour r.

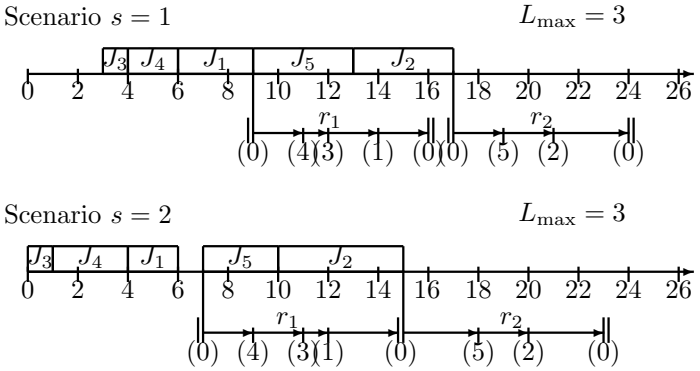

Fig. 1. Robust solution for two scenarios

The online recovery robust integrating scheduling and delivery routing problems defines $x' = (\pi', \mathcal{B}', \Sigma')$ where π' a sequence of groups of permutable jobs of the jobs on the machines $\mathcal{B}$ a batching (i.e. a partition of the jobs) and $\Sigma = \{\sigma(B), B \in \mathcal{B}\}$ the sequence of groups of permutable delivery jobs inside each batch $B \in \mathcal{B}$). Algorithm $A'(x', s)$ is a greedy algorithms that takes the jobs inside a group of permutable production jobs in the earliest release date first order and the jobs inside a group of permutable delivery jobs in the nearest neighbor order, to simulate an online reaction to the revealed scenario.

We consider here the same example as for the standard robust approach.

Let consider the sequence of groups of permutable jobs $G_1^J = \{J_1, J_3, J_4\}$ and $G_2^J = \{J_2, J_5\}$. For the transportation phase, one batch contains jobs $\{J_1, J_3, J_4\}$ and the other batch contains the jobs $\{J_2, J_5\}$ (notice that the composition of batches may be different than the definition of groups). In the first batch, there are two groups of permutable deliveries $G_1^D = \{3\}$ and $G_2^D = \{1, 4\}$ and in the second batch there is only one group $G_3^D = \{2, 5\}$.

Transportation phases are illustrated in Fig. 2 for each scenario and global solution that includes the production and transportation phases are illustrated in Fig. 3 for each scenario. For this example the maximum lateness found is $L_{\max} = 0$. This illustrates the potential gain

of applying the online recoverable robustness scenario. Indeed, the recoverable robustness framework has two merits: it is more realistic as such repairing elements are often present in practice and it is able in theory to reach better worst-case performances on the scenario set. However the question remains whether under limited CPU time, recoverable robustness algorithms are able to actually reach better performance than robust algorithms for reasonable problem instance size. To make such a comparison we propose two heuristics in Section 3 and we compare them under a fair setting in Section 4.

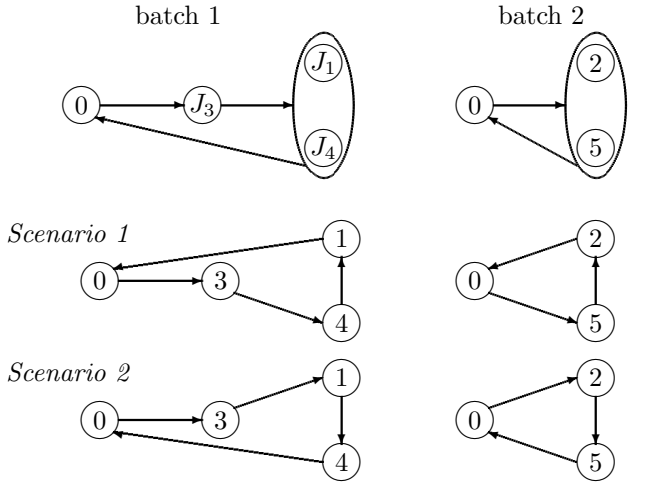

Fig. 2. Tours $\{(3),(1,4)\}$ and $\{(2,5)\}$ under the two possible scenarios

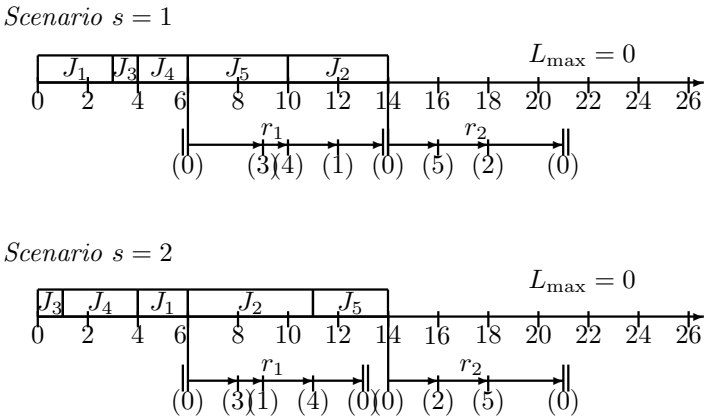

Fig. 3. Production and transportation phases under the two possible scenarios

3. TABU SEARCH HEURISTICS

We propose in this section two tabu search algorithms for the integrated scheduling and vehicle routing problem. The first one is called SRT (Scheduling, Routing, Tabu search) to solve the standard robust approach and, a second called OSRT (Online, Scheduling, Routing, Tabu search) to solve the online recoverable robust approach. In order to compare them, the two algorithms proposed are relatively the same and can be presented together. Note that to simplify the procedure we assume that OSRT makes only one delivery group by delivery batch. This means that the routing of deliveries of each batch is decided by the greedy NN algorithm on the realized scenario. We consider that

Algorithm 1. Tabu search algorithm

(1) Generate an initial solution $(\pi, \mathcal{B}, \Sigma)$ for SRT and $(\pi', \mathcal{B}', \Sigma')$ for OSRT
(2) Perform tabu search on the production scheduling part with fixed batches and routing with NN (on the reference scenario for SRT and on the realized scenario for OSRT).
(3) Perform tabu search on the batching part with fixed production schedule and routing with NN (on the reference scenario for SRT and on the realized scenario for OSRT).
(4) (For SRT only) Post optimize the routing part with the 2-opt heuristic.
(5) Return to step 1 until a stopping criterion (global time limit) is met

among the scenario there is a particular scenario s_0 corresponding to the average case, or to a reference scenario (see Section 4). The routing part of SRT is also simplified in a first step by routing the jobs according to NN on the reference scenario s_0, but the routing is post-optimized in a second phase. The Tabu algorithm implemented is quite standard. This metaheuristic starts from an initial solution and then searches the most appropriate solution among neighbors. A solution can be chosen if it is not tabu (i.e. not in the tabu list) or if it satisfies the aspiration criterion. The chosen neighbor immediately becomes tabu and the search process is then repeated using this chosen neighbor as a new basic solution. Algorithm 3 summarized the common process, with an additional step top post-optimize the routing for SRT. This post-optimization is used to compensate the fact that SRT is not allowed to react to the realized scenario.

We detail the different elements hereafter.

3.1 Inital solution

We first describe how we obtain a feasible solution for SRT. We first find a sequence of production jobs by sorting the jobs according to their increasing due dates on scenario s_0 (Earliest Due Date rule). From this sequence, we build batch by selecting the best solution in terms of maximum lateness over all scenarios between creating a new batch with the current job or inserting this job into the previous batch. The routing is performed by the NN heuristic on the average scenario.

For OSRT, we also first sort the jobs according to their increasing due dates on scenario s_0 but we then build groups of permutable production jobs and batches simultaneously. More precisely, there are four options for each job, the best option is selected at each iteration:

- Insert the job in the current group; insert the job in the current batch
- Insert the job in the current group; create a new batch and insert the job
- Create a new group insert the job; insert the job in the current batch
- Create a new group and insert the job; create a new batch and insert the job

For OSRT, each option is evaluated by ERD on the realized scenario for the groups of permutable production

jobs and by NN on the realized scenario for the routing of each delivery batch.

3.2 Solution encoding

The problem of routing the jobs of a batch is systematically solved by the nearest neighbor algorithm (NN) in the STR algorithm as well as in the OSTR algorithm. In fact, assuming that there is only one group of permutable delivery for each batch fact that for each batch and each scenario the nearest neighbor search is used in OSTR algorithm. Therefore, it suffices to have the scheduling part and the composition of the batches to encode a solution. A solution of the problem S is encoded directly by a vector of size $2n$. The first part of size n contains the scheduling part which is not the same for both algorithms STR and OSTR, whereas, the second part of size n contains the batching part witch is similar in both algorithms. For STR the first vector gives the position of each production job in the sequence while in OSR, the first vector gives the group number of the job.

3.3 Neighbor definition

Starting from this encoding, several neighborhoods have been defined. The neighborhood generation of the scheduling sub-problem are first presented for the two algorithms, then we present those of the batching sub-problem whose are similar in the two algorithms. We use a perimeter δ as the search area from a solution and fix it to 0.4. In each case, some details are given for more accuracy.
In the neighborhood generation of the scheduling sub-problem of STR, two types of neighborhood generation methods are used: (1) Position sequence exchange: this neighborhood performs the exchange of two values v'_i and v'_j in the first part of the encoding. The perimeter δ is taken into consideration in the sense that $|v'_i - v'_j| \leq \delta n$. Then, the best move is retained over all possible changes. (2) Insertion at position in the sequence: this neighborhood performs the exchange of the position v'_i of the job J_i by inserting the job J_i at a position v'_k and where, $|v'_i - v'_k| \leq \delta n$. The best move is retained and the jobs at positions $v'_{i+1}, .., v'_k$ are advanced in the sequence if $v'_i > v'_k$ or, jobs at positions $v'_k, .., v'_{i-1}$ are delayed if $v'_i < v'_k$.

In a neighborhood generation of the scheduling sub-problem of the OSTR, neighbors concerns groups of permutable jobs. We denote by $|G|$ the number of groups and have the following four types of neighborhood generation methods: (1) Groups exchange: this neighborhood performs the exchange of two values in the first part of the encoding, i.e., the swap of the assignment of two jobs to groups with taking into consideration the perimeter δ. Let v''_j the group of the job J_i and v''_i the group of the job J_j, the exchange is possible if and only if $v''_i \neq v''_j$ and $v''_i - \delta|G| \leq v''_j \leq v''_i + \delta|G|$. (2) Insertion group at position: this neighborhood states that a job J_i of the group v''_j can be inserted in an existing group v''_k with $v''_i - \delta|G| \leq v''_k \leq v''_i + \delta|G|$. (3) Group split: this neighborhood states that one group can be split into two groups, which delays the next groups. Since the sequence into the groups depends to the scenario, the number of possible cases to cut a group is exponential. Therefore, we order the jobs of a group according to the average due dates of the jobs over the scenarios, which reduces the number of cases for the sizes of the groups. (4) Group merging: this neighborhood performs the grouping of two consecutive groups into one, which advances the next groups.

Since the second sub-problem is the constitution of the batches, witch is the same for SRT and tOSRT, the neighborhood generation is then the same for the two problems. One can see also that the types of neighbors used for the scheduling part of OSTR algorithm where the goal was to form groups can be used to form batches. Then, we have the previously four types of neighborhood generation methods which are applied to the batches.

3.4 Other elements : tabu list, aspiration criterion and stop condition

To discourage the tabu algorithm to select the solutions that have been visited recently, a tabu list is integrated and it contains solutions can not be selected. These solutions are called Tabu. In our case, since the scheduling part and the routing part are treated separately, we opted for two tabu list with a fixed size equal to the number of jobs. The first list contains the part of the vector corresponding to the scheduling, so that the second list contains the part corresponding to the batches.The fact of having two tabu list can ensure that a solution is considered tabu only if her scheduling part or her routing part is tabu. For this reason, a solution is aspired if it is better than the best solution ever found. We have a fixed time limit, the general algorithm stops when the time limit is reached. However, for the scheduling sub-problem and the batching sub-problem, there is no time limit but the algorithm stops when the solution is not improved.

4. COMPUTATIONAL EXPERIMENTS

4.1 Data generation

To validate our models, we generate random data sets. For the generation of scenarios, we call scenario $s = 1$ the "reference scenario" and we first generate this instance (r_j^1, p_j^1, d_j^1). Given that r_j^1 and d_j^1 depend on the processing times p_j^1, we first generate the p_j^1 for all the jobs. We take $p_j^1 \in [1, 50]$ and we denote by $P = \sum_{j=1}^{n} p_j^1$. The release dates r_j^1 are generated in the interval $[0, \gamma P]$ where γ is a given parameter. Dues dates d_j^1 are generated in the interval $[\alpha - \frac{\beta}{2}P, \alpha + \frac{\beta}{2}P]$, where α and β are given parameters. For the generation of the matrix of distances between sites, we generate randomly the coordinates (X_j^1, Y_j^1) of each site j in the interval $[0, 50]$ and the distance between two sites is the classical euclidian distance. For modeling the uncertainty on the data, scenario s ($s \geq 2$) is generated by a random perturbation of the reference scenario. For a given parameter v_j^s, we perturb the reference scenario $[(1 - \omega^v)v_j^1, (1 + \omega^v v_j^1)]$ where ω^v represents the perturbation magnitude. To have a fair comparison, the same time limit was used for SRT and OSRT.

4.2 Computational Results

In this section, we compare the tabu search algorithms SRT and OSRT, as well as the initials solutions for both cases, on the instances presented above. The number of jobs varies from 10 to 100, the parameters α, β and γ are fixed to 1. There are different classes of problems depending on the number of jobs n and the coefficient of variation ω between the scenarios. An instance is then represented by his type called (n, ω) and for each n and $\omega \in \{20, 40, 60\}$, 10 instances are generated with a number of scenarios $|\mathcal{S}| \in 2, 5, 10$. The size of the tabu lists is equal to the number of jobs n and the perimeter δ to 0.4. For each $(n, \mathcal{S}, \omega)$, the algorithms are tested for the ten instances and the displayed results represent an average for the ten instances. In Table 1, column $\Delta^{(1)} = (GSR - GOSR)/GSR$ is the average gap between the initial solutions GSR and GOSR and $\Delta^{(2)} = (SRT - OSRT)/SRT$ is the average gap between the solutions SRT and OSRT.

			SRT Vs OSRT	
n	$\mathcal{S}$	ω	Δ^1(%)	Δ^2(%)
10	2	20	38.31	5.58
		40	36.08	1.95
		60	28.75	-4.32
	5	20	35.77	1.30
		40	29.67	0.54
		60	27.46	10.24
	10	20	38.26	2.40
		40	31.19	-0.24
		60	26.85	-0.83
20	2	20	56.38	2.44
		40	47.68	1.32
		60	51.43	3.78
	5	20	54.60	3.92
		40	47.08	4.72
		60	47.79	2.00
	10	20	48.98	4.88
		40	45.03	5.62
		60	44.12	5.01
30	2	20	55.82	0.88
		40	55.30	4.50
		60	57.20	1.44
	5	20	54.56	3.59
		40	52.19	4.22
		60	49.20	3.43
	10	20	54.50	5.01
		40	48.80	2.49
		60	45.54	5.46
50	2	20	64.22	2.31
		40	59.17	3.10
		60	59.16	4.59
	5	20	59.45	4.27
		40	54.49	5.26
		60	51.00	5.95
	10	20	59.24	6.76
		40	49.48	3.19
		60	48.37	6.44
100	2	20	62.70	1.30
		40	60.54	5.41
		60	61.09	4.90
	5	20	61.21	4.39
		40	52.08	3.13
		60	49.11	3.77
	10	20	58.37	16.15
		40	51.64	9.28
		60	47.90	17.82

Table 1. Comparison of methods on the instances with $|\mathcal{S}| \in \{2, 5, 10\}$

5. CONCLUSION

The results shown in table 1 show that the recoverable robustness framework based on groups of permutable jobs generally allows to find most robust solutions than the standard robust optimization framework, under the same experimental framework. This is especially true for an important number of jobs and the largest number of scenarios (up to near 18% improvement). The improvement is even drastic for the comparison of the simple greedy initial solution procedures. In the future we will compare the proposed methods on a larger scenario set. It would be indeed interesting to compare the robust framework and the recoverable robust framework on a test bed of scenarios that are generated according to the same random perturbation scheme on the reference scenario, but which was not used to compute the robust solutions. More precisely, we would have a training set of scenarios of reduced cardinality to compute the robust solution, while these solutions will be evaluated on a larger set, which would be closer to the real life context.

ACKNOWLEDGEMENTS

This work was supported by the financial support of the ANR ATHENA project, grant ANR-13- BS02-0006 of the French Agence Nationale de la Recherche.

REFERENCES

The robust vehicle routing problem with time windows A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss and C. Requejo. *Computers & Operations Research*, 40(3), pp. 856–866, 2013.

A. Ben-Tal, A. Goryashko, E. Guslitzer and A. Nemirovski. Adjustable robust solutions of uncertain linear programs *Mathematical Programming*, 99(2), pp. 351–376, 2004.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows, 2003. *Mathematical Programming*, 98(1–3), pp. 49–71.

D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1), pp. 35–53, 2004.

J-C. Billaut and F. Roubellat. A new method for workshop real time scheduling, *International Journal of Production Research*, 34 (6), 1555-1579, 1996.

Z-L. Chen. Integrated production and outbound distribution scheduling: review and extensions, *Operations Research*, 58(1), 130–148, 2010.

A. Caprara, L. Galli, S. Stiller and P. Toth. Delay-Robust Event Scheduling, 2014. *Operations Research*, 62(2), pp 274–283.

P. Kouvelis and G. Yu. *Robust discrete optimisation and its applications*, Kluwer Academic Publishers, 1997.

C. Liebchen, M. Lübbecke, R. Möhring and S. Stiller. The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications in R. K. Ahuja, R. H. Möhring and C. D. Zaroliagis (editors), *Robust and Online Large-Scale Optimization, Models and Techniques for Transportation*, Lecture Notes in Computer Science, Volume 5868, Springer Berlin Heidelberg, 2009.

S. D. Wu, E.-S. Byeon and R. H. Storer. A Graph-Theoretic Decomposition of the Job Shop Scheduling Problem to Achieve Scheduling Robustness. *Operations Research*, 47(1), pp. 113–214, 1999.