Online recoverable robustness based on groups of permutable jobs for integrated production scheduling and delivery routing

Azeddine Cheref, Christian Artigues, Jean-Charles Billaut

▶ **To cite this version:**

Azeddine Cheref, Christian Artigues, Jean-Charles Billaut. Online recoverable robustness based on groups of permutable jobs for integrated production scheduling and delivery routing. 2016. hal-01351496

HAL Id: hal-01351496
https://hal.science/hal-01351496

Preprint submitted on 3 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Online recoverable robustness based on groups of permutable jobs for integrated production scheduling and delivery routing

Azeddine Cheref * Christian Artigues † Jean-Charles Billaut ‡

August 3, 2016

Abstract: It is well known that the magnitude and the impact on schedule quality of uncertainty in problem data both increase when the integration of scheduling decisions takes place. This paper aims at comparing the standard robust optimization and the recoverable robust optimization frameworks to tackle scenario-based uncertainty in integrated production and distribution scheduling. While the standard robust optimization aims at providing an integrated baseline schedule that optimizes a performance criterion for the worst scenario, the recoverable robustness framework adds a repairing/recovering component, which is able to react to the realized scenario by modifying in a limited way the issued baseline schedule. Hence, 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. This paper proposes new MILP models and heuristics for a recoverable robust optimization framework based on the concept of groups of permutable jobs and carries out such a comparison on a one-machine production facility coupled with a single vehicle distribution system. For this problem, the experimental results exhibit conditions under which a recoverable robust approach is likely to yield better results than the standard robust approach.

Keywords: Integrated production and distribution scheduling, Recoverable robustness, Groups of permutable jobs

1 Introduction

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 the literature, more and more research deal with integrated approaches, involving production and distribution decisions, at strategic, tactical and operational levels. The relation between production and distribution may involve an intermediate storage area and, in this case, inventory levels are considered as part of the production and distribution problems. In the more general terms and depending on the decision level, the

*cheref@laas.fr
Université François-Rabelais Tours / CNRS, 64 av. J. Portalis, 37200 Tours, France
LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
†artigues@laas.fr
LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
‡jean-charles.billaut@univ-tours.fr
Université François-Rabelais Tours / CNRS, 64 av. J. Portalis, 37200 Tours, France

production problem is a planning problem and/or a scheduling problem, where a manufacturer has to decide in which location the orders will be processed, at which time period and in which quantity. Then, the *delivery problem* is an assignment and/or a routing problem, where the manufacturer has to decide how many vehicles will be used at each location, how orders are grouped into delivery batches and when they will be delivered. We refer to [Chen, 2004, Chen, 2010a, Wang et al., 2014] for a global overview of integrated production and distribution problems.

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 criterion* is minimized, under a scenario-based uncertainty modeling. In contrast with standard robust optimization approaches for scheduling [Kouvelis and Yu, 1997], 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, as in stochastic programming except that no probabilities are attached to scenarios and that the possibilities for recovery are limited by a predefined set of recovery algorithms [Liebchen et al., 2009]. Following this scheme, we propose for the first-stage a set of solutions by using the concept of *groups of permutable jobs* presented in [Billaut and Roubellat, 1996, Artigues et al., 2005]. During second stage, a recovery algorithm uses the information from the revealed scenario to compute a complete schedule from the set of solutions issued from the first-stage. However, we consider that the scenario is unrevealed progressively during schedule execution, so the recovery algorithm has to be both greedy and *online*, exploiting only known information about the jobs available to be scheduled or to be delivered at decision time. This appear to be a practical characteristic of many applications both on production scheduling (where for example the exact release date of a job is known only when the job arrives) and on vehicle routing (where the traveling time from a location to another is only accurate in a short term basis due to traffic fluctuation). We will refer to this framework as *online recoverable robustness*.

Despite its practical appeal, the online recoverable robustness framework raises two major issues, one linked to modeling difficulties and another one linked to tractability in solution methods. For the first issue, a basic question is to know whether is it possible model a non trivial online recoverable robust integrated scheduling and vehicle routing problem such as the one considered in this paper in a standard optimization paradigm such as the mixed-integer linear programming framework. In this paper, we answer positively to this question by providing new MILP models that extends the one proposed in [Kouvelis and Yu, 1997] for the standard robust one machine scheduling problem. The second issue is to design, under limited CPU time, recoverable robustness methods that are able to actually reach better performance than standard robust methods for reasonable problem instance size. In this work, a comparison of a robust heuristic approach with an online recoverable robust heuristic is carried out on a one-machine production facility coupled with a single vehicle distribution system. The experimental results allow to determine conditions under which the proposed recoverable robust approach is likely to yield better results than the standard robust approach.

The paper is organized as follows. Section 2 presents a literature review on integrated production scheduling and vehicle routing problems on the one hand and robust scheduling and vehicle routing on the other hand. In Section 3, we give a formal description of the problem, introducing the proposed online recoverable robustness scheme. In Section 4, we provide a (single stage) robust mixed-integer linear programming formulation of the problem. Section 5 presents a two-stage robust mixed-integer linear programming formulation, instantiating the concept of online recoverable

robustness and making use of the group-of-permutable-jobs structure. As both the single-stage and two-stage robust MILP quickly become intractable as instance size grows, we present in Section 6 a heuristic based on tabu search to build a robust solution on realistic instances. In Section 7, we provide computational experiments to validate our proposal. Concluding remarks and directions for further research are pointed out in Section 8.

2 Related work

To our knowledge, no approach for integrated production scheduling and vehicle routing under uncertainty can be found in the literature. Hence we divide this state-of-the-art section into deterministic integrated production and distribution scheduling (Sect. 2.1), robust scheduling (Sect. 2.2) and robust vehicle routing (Sect. 2.3).

2.1 Deterministic integrated production and distribution scheduling

Few papers deal with an integrated model of production and distribution at an operational level, i.e. with vehicle routing as a part of the decision. In [Chang and Lee, 2004], the authors consider a single machine or a two-parallel machines environment, a single vehicle and one or two customer areas. Each job has a size and the capacity of the vehicle is limited. The authors give NP-hardness proofs and algorithms with worst-case analysis. The same problem is considered by [He et al., 2006] and [Zhong et al., 2007], in which some improvement are given. In [Lu et al., 2008], the authors consider a single machine problem with release dates and a single vehicle to deliver the products to one customer area. The vehicle has a bounded capacity and the objective is to minimize the maximum delivery time of the jobs. The authors prove the NP-hardness of the problem and propose an algorithm with worst-case analysis. They consider also the case when preemption is allowed and propose a polynomial time algorithm for this problem. In [Condotta et al., 2013], the authors treat a similar problem where due dates for delivery are integrated. Multiple vehicles are available for the delivery and the objective is to minimize the maximum lateness. The authors propose a mixed linear program, lower bounds and a tabu search algorithm for solving the problem. In order to minimize the maximum delivery time of the jobs, [Dong et al., 2013] consider the two-machines open shop problem and a single vehicle with bounded capacity to deliver the products to one customer area. The authors consider the special case when the vehicle can take only one job and a general capacity case. The authors propose a polynomial time algorithm with a worst case ratio for the problem. In [Li et al., 2005a], the authors consider a single machine and single vehicle problem. The vehicle has a bounded or unbounded capacity and the objective is to minimize the sum of delivery completion times. The authors prove the NP-hardness of the problem. They propose an efficient dynamic programming algorithm for the single-customer case and, for the general case, they propose a polynomial time dynamic programming algorithm when the number of customers is fixed. A similar problem is considered in [Levin and Penn, 2008], with an arbitrary number of customers and an uncapacitated delivery vehicle. The objective function is to minimize the *total latency*. The authors propose a $\cong$16.3-approximation algorithm. In [Li and Ou, 2005b], the authors consider a factory (single machine) and a warehouse with two different locations. The raw materials (jobs) are located at the warehouse before processing, have to be delivered to the factory for their processing and then delivered back to the warehouse when completed (there is no vehicle routing problem). Only one vehicle is available with a limited capacity (in terms of number of jobs) for transporting unprocessed jobs, and another, also of limited capacity, for transporting processed jobs. The authors prove the NP-completeness of the problem and propose a polynomial time dynamic programming algorithm when the sequence of processing jobs is fixed. A heuristic algorithm with performance guarantee is

proposed for the general case. This model is extended in [Wang and Cheng, 2009] by considering one warehouse for material supply and another warehouse for finished products. There are two vehicles, one for the travels between the first warehouse and the factory and one for the travels between the factory and the second warehouse. The authors derive some optimal properties, study some polynomially solvable cases and propose heuristic algorithms with performance guarantee for the general case and some particular cases. In [Li and Vairaktarakis, 2007], the authors consider a machine processing environment called "*bundling operations*", where each job is composed by two tasks that are processed on dedicated machines. A job is ready when both tasks are completed and can then be delivered to the customer location. There is an unlimited number of vehicles with limited capacity. The objective is to schedule the jobs and to define the routes of the vehicles in order to minimize a delivery cost plus a weighted sum of the delivery completion times. The authors propose polynomial-time heuristic algorithms and dynamic programming-based approximation schemes. In [Scholz-Reiter et al., 2010], the authors consider a global supply chain context, and focus on the interface between manufacturing and logistic systems along the supply chain. Orders are produced in an hybrid flowshop (flowshop with assignment problem) and then assigned to tours. The objective function is a cost function, composed by a production cost, a storage cost (for intermediate and finished products), a tardiness cost, the cost of conducting a tour and the cost of a tour. The authors propose a mathematical programming formulation, tested on a small instance. In [Chen, 2010b], the author considers a problem with a single machine and multiple vehicles. To each job is associated with a processing time and a volume. All the jobs have to be delivered at only one customer location. Vehicles have different limited capacities and different routing times. The objective is to minimize the makespan. The author proves NP-completeness and propose a MILP formulation. For the problem with a single machine, an on-line version is considered by [Ng and Lu, 2011], in which the processing time of each job becomes known at its arrival time. The authors consider the case where preemption is allowed and algorithms with worst case analysis are proposed for both of them. In [Wang and Liu, 2013], the authors consider a problem with a single machine and a single vehicle with limited capacity. The customers are located at the nodes of a star-shaped network (disjoint branches with a common endpoint corresponding to the production center). Each job has a physical space and the vehicle capacity is bounded. The objective is to find a processing sequence and a delivery rule in order to minimize the makespan (maximum delivery completion time). The problem is NP-hard and the authors propose approximation algorithms in the identical-job-size case and in the general case. In [Ullrich, 2013], the author considers a parallel machine scheduling environment with machine ready times. The jobs have to be assigned to certain tours while the routes and the schedule of the tours have to be decided, so that the total tardiness is minimized. A fleet of several vehicles is available and a ready time is associated with each vehicle. Jobs have a size and the vehicles capacity is limited. The author proposes MILP formulations for the scheduling problem and the routing problem and for the integrated problem. Two decomposition heuristics and a genetic algorithm are proposed and compared on specific benchmark instances. Many work on integration of production and distribution scheduling concern perishable products. [Armstrong et al., 2008] consider a zero-inventory production and distribution problem with a single transporter and a fixed sequence of customers. They propose a branch-and-bound procedure for the problem. [Geismar et al., 2008] develop lower bounds and heuristics for an integrated production and transportation scheduling problem with capacity constraints and also no inventory, which means that a completed production job must be transported immediately to the customer. In [Gao et al., 2015], the authors propose a heuristic for the problem of minimizing the makespan on a single machine scheduling problem with a unique capacitated vehicle and a no wait constraint. In [Lee at al., 2014], the authors consider the problem of nuclear medicine production and delivery. Each job has to be assigned to a production machine and has to be delivered in a

given time window. Several vehicles with different capacities are available. The objective is to minimize a cost function composed of a production cost, a fixed cost for the use of a vehicle and a travel cost. The authors propose an MILP formulation and a large neighborhood search heuristic, tested on benchmark instances issued from the VRP literature. [Viergutz and Knust, 2014] pointed out and corrected an error in the branch-and-bound method of [Armstrong et al., 2008]. They also extend the models to variable production and delivery sequences and propose several heuristics.

2.2 Robust scheduling

"*Uncertainty affects a wide range of decisions managers, engineers, and other decision makers have to make*" [Kouvelis and Yu, 1997]. This is specially true for production managers, where data such as processing times or release dates are generally not known with an absolute certitude, and for delivery problems, where 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. This is the field of *robust approaches*. There are several ways to understand and to structure uncertainty and several ways to define a robust approach [Billaut et al., 2008]. The aim of such an approach is to produce a solution that will have a "*reasonable objective value*" under any input data scenario [Kouvelis and Yu, 1997].

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. Under this framework, based on uncertainty scenarios, the concept of robust scheduling appeared in [Daniels and Kouvelis, 1995] (in 1992 for a working paper version). The authors consider a single machine environment, and processing time uncertainty is represented by interval data, which yields an infinite number of scenarios. The authors consider the total flow time as the deterministic objective function and the robustness is based on absolute or relative deviations from an optimal performance. Since this seminal paper, a lot of papers have addressed uncertainty in scheduling. Two survey papers deal with the generation of proactive (robust) schedules for the resource constrained project scheduling problem: [Herroelen and Leus, 2004, Herroelen and Leus, 2005]. Aytug et al. [Aytug et al., 2005] review the literature on production scheduling in the presence of unexpected events/unforeseen disruptions and propose a taxonomy for uncertainty in production manufacturing. Another recent survey on general approaches for scheduling under uncertainty can be found in [Verderame et al., 2010].

In this paper we integrate three families of approaches for scheduling under uncertainty. The first family is based on implicit consideration of uncertainty via the concept of computing a set of solutions rather than a single solution for a scheduling problem in which data is subject to unmodeled disruptions. In production scheduling, a family of methods described notably in [Billaut and Roubellat, 1996, Artigues et al., 2005] aimed at computing on each machine a sequence of groups of permutable jobs such that any permutation of the job inside each group yields a feasible job sequence with a guaranteed value on the objective function as soon as the job durations do not exceed a predefined upper bound. In other words, a sequence of groups of permutable jobs is a structured partial solution such that each complete solution issued from it has a guaranteed performance. The drawback of these approaches is that there is no indication on how to select the job order inside each group in response to a particular disruption. However, in [Wu et al., 1999], the author have shown through simulation of various processing time disturbances that computing a static way such a solution structure[1] and then allowing the remaining decisions to be taken

[1]They called this structure an "ordered sequence of subsets", which is equivalent to the "sequence of groups of

dynamically via dispatching rules yields superior results.

The second family of approaches is based on robust discrete optimization for scheduling as stated in [Kouvelis and Yu, 1997]. In this framework (see also several other surveys or seminal works on robust discrete optimization [Bertsimas and Sim, 2003, Bertsimas and Sim, 2003, Aissi et al. 2009, Gabrel et al. 2014]), a complete sequence of jobs on each machine is the output of the robust scheduling method. The uncertainty is represented by a continuous or discrete 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, at least for a finite scenario set, the worst case performance of a sequence on the scenario set can be computed a priori. Note that most robust scheduling problems (aiming at finding the optimal sequence, having the best worst-case performance) are NP-hard even for single-machine scheduling problems [Aloulou and Della Croce, 2008].

The third family of approaches is based on recoverable robustness. This framework, defined in [Liebchen et al., 2009], aims at reducing the conservativeness of robust optimization as described in [Bertsimas and Sim, 2003]). 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. Thanks to this algorithm, the baseline solution adapts to the realized scenario. Note that we can consider that the classical robust scheduling framework (second family of approaches) falls into this scheme, with a simplified recovery algorithm. Indeed, its outcome is made of both the prescribed sequence (which can be defined as an assignment of first-stage decision variables) and the earliest scheduling algorithm (which can be defined as the prescribed algorithm for assigning the start times considered as second-stage decision variables). Recently, recoverable robustness has been successfully applied to scheduling problems [Caprara et al., 2014].

2.3 Robust vehicle routing

Robustness considerations in vehicle routing have been addressed since 1990s. Bertsimas and Simchi-Levi in [Bertsimas and Simchi-Levi, 1996] present a survey on papers dealing with uncertainty issues in vehicle routing. Most of the problems consider that the demand is stochastic (demands arrive randomly, demands have a random size, etc.), and the authors propose robust algorithms, i.e. algorithms "independent of the specific environment and the variability of the data". The same year, in [Gendreau et al., 1996], the authors propose a summary of the literature on stochastic vehicle routing problems. In their survey, stochastic demands, stochastic travel times and stochastic customers are considered. Besides this literature on vehicle routing based on stochastic programming approaches, robust optimization methods have been more recently addressed. A nice state-of-the-art review and a robust optimization approach can be found in [Agra et al., 2013] for the vehicle routing problem with time windows. In this paper, the authors consider an uncertainty polytope for the travel times. Along with standard robust optimization, they also consider the adjustable robustness framework [Ben-Tal et al., 2004], which is close to the recoverable robustness framework, except that the second-stage variables can be adjusted without any algorithmic restriction to the realized scenario. Notably, they allow arrival times to be adjustable to the realized scenario. Other authors considered robust optimization for the capacitated vehicle routing with uncertain demand [Gounaris et al., 2013].

permutable jobs" structure defined in [Billaut and Roubellat, 1996]

3 Problem description and online recoverable robustness

Following the classical three-field notation of scheduling problems [Graham et al., 1979], Z-L. Chen introduces in [Chen, 2010a] 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 ($\forall i, j,\ 1 \leq i, j, \leq n$). Decision variable C_j denotes the *production completion time* of J_j at the plant ($\forall j,\ 1 \leq j \leq n$). Decision variable D_j denotes 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 $\gamma \in \{D_{\max}, L_{\max}\}$ with $D_{\max}$ the maximum delivery completion time, defined by $D_{\max} = \max_{1\leq j\leq n} D_j$ and $L_{\max}$ the maximum delivery lateness, defined by $L_{\max} = \max_{1\leq j\leq n}(D_j - d_j)$. The 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 in addition that uncertainty affects the input data and that our objective is to find a robust solution. To represent 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 following objective functions are defined:

$$\begin{array}{ll} \text{the } \textit{delivery makespan} & D_{\max} = \max_{s\in\mathcal{S}}(\max_{1\leq j\leq n} D_j^s), \\ \text{the } \textit{maximum lateness} & L_{\max} = \max_{s\in\mathcal{S}}(\max_{1\leq j\leq n} L_j^s). \end{array}$$

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, 2003]. 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 denote by π a complete or partial sequence 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 set of complete or partial routing $\sigma(B)$, i.e. the complete or partial job delivery sequence inside each batch $B \in \mathcal{B}$). Let C^s (D^s) denote the vector of second-stage job completion times (delivery times, respectively). We assume that given a first stage solution $x = (\pi, \mathcal{B}, \sigma)$ the recovery algorithm $A(x, s)$ output a feasible solution $y = A(x, s) = (C^s, D^s)$. Function $f(y, s)$ is either $D_{\max}$ or $L_{\max}$ as defined above. However, as both the scheduling and routing problems are time-constrained problems, we assume that algorithm $A(x, s)$ must be an *online* algorithm, i.e. called during schedule execution. At a given decision time t the algorithm must take quickly a basic decision (i.e. what is the next job to be scheduled at time t or what is the next job to be delivered at time t) with the only knowledge on the scenario that has been revealed up to time t (i.e. what are the jobs available for scheduling at time t and what are the estimated travel times at time t). Sections 4 and 5 presents two different definitions of the first-stage decision variables $x = (\pi, \mathcal{B}, \sigma)$, the first one, inspired by [Kouvelis and Yu, 1997], defining complete sequences of jobs both in production and delivery and the second one defining partial sequences based on the concept of groups of permutable jobs [Billaut and Roubellat, 1996, Artigues et al., 2005] under the recoverable robustness framework. Associated with these two different first-stage decisions, two different online scheduling algorithms are defined to output the two-stage decision variables $y = (C^s, D^s)$.

In Sections 4 and 5, we extend the robust scheduling mixed-integer linear programming models proposed by [Kouvelis and Yu, 1997] to the fixed-sequence robust scheduling problem and to the online recoverable robust scheduling problem, respectively. Section 6 considers a particular case of the second model to devise a tabu search heuristic able to deal with large-size instances.

4 The fixed sequence robust model

This Section presents mixed integer linear programs (MILPs) for the robust integrated scheduling and vehicle routing problem using the concept proposed by [Kouvelis and Yu, 1997]. At first stage

a complete production sequence π, a set of batch $\mathcal{B}$ and, for each batch $B \in \mathcal{B}$, a complete routing $\sigma(B)$ are computed. The online recovery algorithm $A(x, s)$ aims at strictly following the prescribed sequences and routings, assigning the production and delivery completion times as early as possible according to the prescribed sequence on the realized scenario. In section 4.1, we first consider the particular case in which the delivery part of the problem is ignored. Three MILPs are proposed to model the optimal assignment of the first-stage variables and second-stage variables according to the on-line algorithm. In Section 4.2, a MILP is proposed for the integrated scheduling and vehicle routing problem. In both cases, an illustrative example is given.

4.1 MILP for the fixed sequence robust scheduling problem

In this section, we consider the scheduling problem with release dates and maximum lateness minimization under several scenarios based on fixed sequences and earliest start online recovery algorithm. Note that the problem is NP-hard with only one scenario. The objective is thus to find a single machine sequence π that is the best possible for all scenarios, given that the online algorithm will adjust at the earliest the completion times C_i^s to the realized scenario. The following model (SM) is based on the one proposed by [Kouvelis and Yu, 1997] for the $1||\sum C_j$ problem. We define positional variables $x_{j,k}$ equal to 1 if job J_j is in position k in sequence π and 0 otherwise.

(SM)

$$\min L_{\max}$$

$$\text{s. t.} \sum_{j=1}^{n} x_{j,k} = 1, \qquad \forall k \in \{1, ..., n\} \qquad (1)$$

$$\sum_{k=1}^{n} x_{j,k} = 1, \qquad \forall j \in \{1, ..., n\} \qquad (2)$$

$$L_{\max} \geq \sum_{j=1}^{n} r_j^s x_{j,k} + \sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q} - \sum_{j=1}^{n} d_j^s x_{j,k'}, \qquad \forall k, k' \in \{1, ..., n\}, k \geq k', \forall s \in \mathcal{S} \qquad (3)$$

$$x_{j,k} \in \{0, 1\}, \qquad \forall j, k \in \{1, ..., n\} \qquad (4)$$

$$L_{\max} \geq 0, \qquad (5)$$

Constraints (1) and (2) guarantee the feasibility of the sequence, assigning exactly one job at each position. Constraints (3) express the fact that the worst-case maximum lateness ($L_{\max}$) has to be larger than the maximum lateness under scenario s, given by the right hand side. Indeed $\sum_{j=1}^{n} r_j^s x_{j,k}$ gives the release date of the job in position k under scenario s, and $\sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q}$ is equal to the total duration of the jobs between positions k and k' under scenario s, while $\sum_{j=1}^{n} d_j^s x_{j,k'}$ gives the due date of the job at position k' in scenario s. These constraints are based on the fact that, according to the earliest time online recovery algorithm, there exists a block of activities starting by a job scheduled at its release date, followed by jobs scheduled without idle times and ending by the job that fixes the $L_{\max}$. Hence the explicit usage of the second-stage completion time variables is not even necessary. This model contains n^2 binary variables, 1 continuous variable and $O(n^2|\mathcal{S}|)$ constraints.

Example

Let us consider an instance with $n = 5$ jobs and the following data.

$s=1$	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	3	14	4	6	10

$s=2$	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	6	11	2	5	14

The optimal robust sequence for this example is $(J_3, J_1, J_4, J_5, J_2)$ with a maximum lateness $L_{\max} = 5$. Fig. 1 provides the second-stage completion time variables and maximum lateness of this fixed sequence for each scenario.

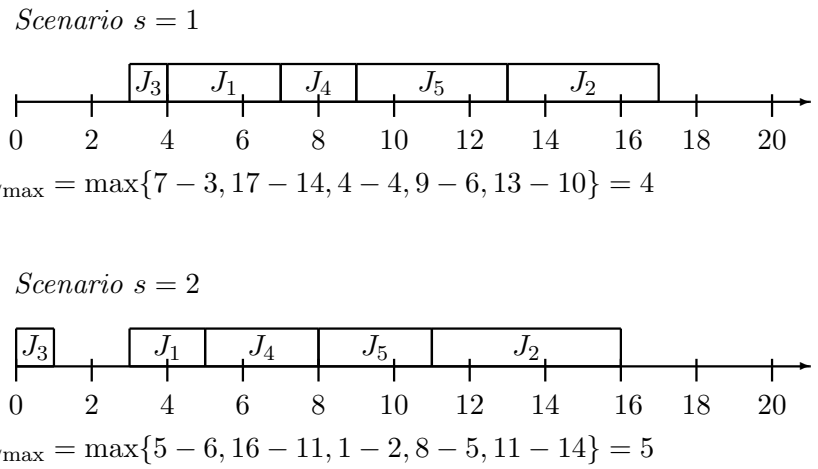

Figure 1: Sequence $(J_3, J_1, J_4, J_5, J_2)$ under the two possible scenarios

4.2 MILP for the fixed sequence robust integrated scheduling and vehicle routing problem

In this section, we a present a new MILP model in which the robust scheduling and the vehicle routing problems are integrated. For these models, the scheduling part is based on the previous model. Here the first stage variables are the complete production sequence π, the batching $\mathcal{B}$ and the routings $\sigma(B)$ for each batch $B \in \mathcal{B}$.

For the following model, the binary variables associated to the vehicle routing part of the problem are the following. Let B_r denote the r^{th} batch. r varies from 1 to n and some batches can be empty. A binary first-stage variable $z_{i,j,r}$ is equal to 1 if tour $\sigma(B_r)$ visits the site of job i and then the site of job j consecutively, and 0 otherwise, while another binary first-stage variable $y_{j,r}$ is equal to 1 if tour $\sigma(B_r)$ visits site j (in other words if j belongs to batch B_r). We denote by $D_{j,r}^s \geq 0$ the second-stage delivery completion time of job J_j ($D_{0,r}^s$ is the start time of tour $\sigma(B_r)$) if it is in tour $\sigma(B_r)$, for scenario s. The dummy site $(n+1)$ is introduced as a finishing site of a tour ($D_{n+1,r}^s$ is the finishing time of tour $\sigma(B_r)$ in scenario s).

The model called SRM comes from model SM but introduces the second stage completion times of the kth job in scenaro s ($\tilde{C}_k^s$) with constraints (6) that is still based on the concept of blocks, and uses constraints (7)-(13) to set the worst-case $L_{\max}$ relatively to the worst case delivery times. Note that variable $\tilde{C}_k^s$ is necessary to serve as an earliest start time in scenario s of any tour that delivers the site of a job scheduled at position k.

(SRM)

$$\min L_{\max}$$

$$\text{s. t. } \sum_{j=1}^{n} r_j^s x_{j,k} + \sum_{q=k}^{k'} \sum_{j=1}^{n} p_j^s x_{j,q} \leq \tilde{C}_{k'}^s, \qquad \forall k, k' \in \{1,...,n\}, k \geq k', \forall s \in \mathcal{S} \tag{6}$$

$$\sum_{j=0, j\neq i}^{n} z_{i,j,r} = y_{i,r}, \qquad \forall i \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{7}$$

$$\sum_{i=0, i\neq j}^{n} z_{i,j,r} = y_{j,r}, \qquad \forall j \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{8}$$

$$\sum_{r=1}^{n} y_{j,r} = 1, \qquad \forall j \in \{1,...,n\} \tag{9}$$

$$D_{0,r}^s \geq D_{n+1,r-1}^s, \qquad \forall r \in \{2,...,n\}, \forall s \in \mathcal{S} \tag{10}$$

$$D_{j,r}^s \geq D_{i,r}^s + t_{i,j}^s - M(1 - z_{i,j,r}), \qquad \forall i \in \{0,...,n\}, \forall j, r \in \{1,...,n\}, i \neq j, \forall s \in \mathcal{S} \tag{11}$$

$$D_{0,r}^s \geq \tilde{C}_k^s - M(2 - x_{j,k} - y_{j,r}), \qquad \forall j, k, r \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{12}$$

$$L_{\max} \geq D_{j,r}^s - d_j^s, \qquad \forall j, r \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{13}$$

$$z_{i,j,r} \in \{0,1\} \qquad \forall i, j \in \{0,...,n\}, i \neq j, \forall r \in \{1,...,n\} \tag{14}$$

$$y_{i,r} \in \{0,1\} \qquad \forall i, j \in \{0,...,n\}, \forall r \in \{1,...,n\} \tag{15}$$

$$D_{i,r}^s \geq 0 \qquad \forall i, j \in \{0,...,n\}, i \neq j, \forall r \in \{1,...,n\} \tag{16}$$

$$C_k^s \geq 0 \qquad \forall k \in \{1,...,n\}, \forall s \in \mathcal{S} \tag{17}$$

and Constraints (1), (2), (4), (5)

Constraints (7) and (8) indicate that if a site i belongs to a tour $\sigma(B_r)$, then there exists a unique ingoing arc and a unique outgoing arc selected for i in tour $\sigma(B_r)$. Constraints (9) ensure that a site j can belong only to one tour. Constraints (10)-(13) are the scenario-dependent constraints. Constraints (10) ensure that the tour $\sigma(B_r)$ starts after the end of previous tour $\sigma(B_{r-1})$ in scenario s. Constraints (11) sets for each scenario s the delivery completion time of job J_j as the completion time of the delivery of a job J_i plus the traveling time from site i to site j, provided that arc (i,j) has been selected in a tour $\sigma(B_r)$. These constraints also automatically eliminate subtours. Constraints (12) ensure that tour $\sigma(B_r)$ starts after the execution of all the jobs which are delivered by $\sigma(B_r)$. Constraints (13) gives the expression of the maximum lateness for each scenario s. This model contains $O(n^3)$ binary variables, $O(n^2|\mathcal{S}|)$ continuous variables and $O(n^3|\mathcal{S}|)$ constraints.

Example

For this example, we change the delivery due dates d_j and data routing are added to the previous example. Release dates and processing times are similar to those of the previous example.

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

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

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

Let us consider the schedule $(J_3, J_4, J_1, J_5, J_2)$ for the production phase and then two batches composed by jobs $B_1 = \{J_1, J_3, J_4\}$ and $B_2 = \{J_2, J_5\}$ with routings $(J_4 \to J_3 \to J_1)$ and $(J_5 \to J_2)$, respectively. The production and transportation phases are illustrated in Fig. 2 for each scenario. In the figure, the scheduling part is displayed with a standard Gantt chart while the routing part is symbolized via arrows whose length gives the transportation time. Visited sites are displayed in parenthesis, where (j) denotes the site of job J_j for $j > 1$ and (0) denotes the depot. The start time of each batch is marked with a double bar ($||$). While the visit of each site is marked with a simple bar ($|$). In scenario 1, site J_3 is delivered at time 12 while the deadline of J_3 for scenario 1 is $d^1_3 = 9$. In scenario 2, site J_2 is delivered at time 20 and $d^2_2 = 20$. For both scenario we have $L_{\max} = 3$.

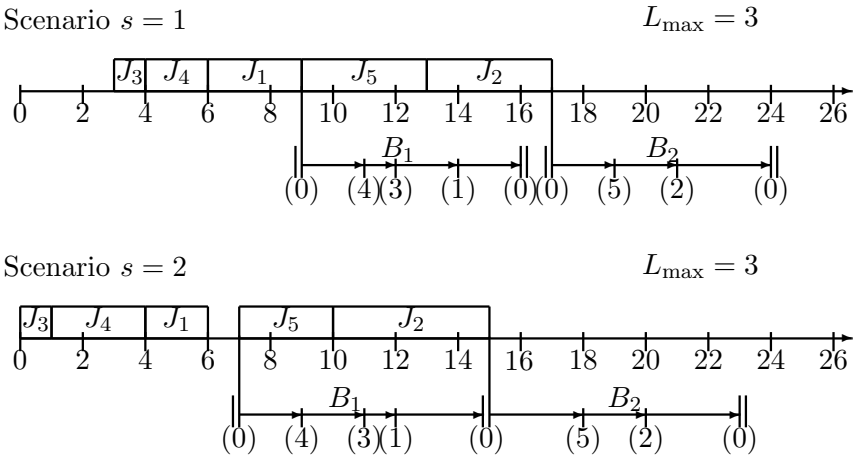

Figure 2: Robust solution for two scenarios

5 The online recoverable robust model

In this section we present a more flexible online recoverable robustness model for the integrated scheduling and vehicle routing problem. We now suppose that the first-stage solution $x = (\pi, \mathcal{B}, \Sigma)$ is made of a set of batches $\mathcal{B}$ that partitions the jobs (just as in the previous model) but the production and routing sequences π and $\sigma(B)$ for each batch $B \in \mathcal{B}$ are now *partial* sequences. Furthermore, we consider only partial sequences that can be expressed as a *sequence of groups of permutable jobs*. Hence a partial sequence involving a set E of jobs is of the form $g_1, g_2, \ldots, g_q$ where $\cup_{h=1}^{q} g_h = E$ and $\cap_{h=1}^{q} g_h = \emptyset$ and represent the precedence relation $J_i \prec J_j$, $\forall i \in g_x$, $\forall j \in g_y$, $\forall x < y$. Consequently π is a sequence of groups of permutable jobs on $\mathcal{J}$, while $\sigma(B)$ is a sequence of groups of permutable jobs on B for each batch $B \in \mathcal{B}$. Hence the relative ordering of the jobs inside each group is left undetermined during the first-stage. It follows that the recovery algorithm $A(x, s)$ has to determine the total order besides assigning the completion and delivery times. We

will specify $A(x, s)$ and provide an online recovery robust MILP for scheduling in Section 5.1. We will extend the approach to the integrated scheduling and routing problem in Section 5.2.

5.1 MILP for the online recoverable robust scheduling problem

In this subsection, we only consider production scheduling without delivery routing. Delivery routing will be considered in the next subsection. As already mentioned, we assume that the recovery algorithm is a greedy online algorithm. When a job J_j completes at a given time t, the online recovery algorithm $A(x, s)$ has only the knowledge on the operations available at time t for scenario s (i.e. such that $r_i^s \leq t$) and no other information. If there are operations available in the group of J_j, the algorithm selects one of them. Otherwise, the $A(x, s)$ scans the available operations of the next group. To summarize, $A(x, s)$ follows the sequence of groups and schedule the jobs inside a group following the earliest release date (ERD) rule according to the realized scenario.

To simulate this rule in a MILP formulation, let us introduce the list of predecessors and successors of each job for each scenario according to the ERD rule. We denote by $Succ_j^s$ and by $Prec_j^s$ the list of successors and of predecessors of J_j under scenario s according to tule ERD, defined as follows.

$$Prec_j^s = \{J_i \in \mathcal{J} / (r_i^s < r_j^s) \text{ or } (r_i^s = r_j^s \text{ and } i < j)\}$$
$$Succ_j^s = \{J_i \in \mathcal{J} / (r_i^s > r_j^s) \text{ or } (r_i^s = r_j^s \text{ and } i > j)\}$$

These lists are used to ensure that the ERD rule is respected in each group for each scenario. To model the group sequence, we also define variables $\chi_{j,k}$ equal to 1 if job J_j is in group G_k^J, and 0 otherwise, $1 \leq j \leq n$, $1 \leq k \leq n$. Note that the number of groups cannot exceed n (There cannot be more than one group per job). The online recoverable robust MILP for the one machine scheduling problem (OSM) can now be proposed.

$$\text{(OSM)} \quad \min L_{\max} \tag{18}$$

$$\text{s. t.} \quad \sum_{k=1}^{n} \chi_{j,k} = 1, \qquad \forall j \in \{1, ..., n\} \tag{19}$$

$$r_i^s + p_i^s + \sum_{l \in (Succ_i^s \cap Prec_j^s)} p_l^s \chi_{l,k} + p_j^s - M(2 - \chi_{i,k} - \chi_{j,k}) \leq C_j^s, \qquad \forall i, j, k \in \{1, ..., n\}, j \in Succ_i^s, \forall s \in \mathcal{S} \tag{20}$$

$$r_i^s + p_i^s + \sum_{l \in Succ_i^s} p_l^s \chi_{l,k} + \sum_{q=k+1}^{k'-1} \sum_{l=1}^{n} p_l^s \chi_{l,q} + \sum_{l \in Prec_j^s} p_l^s \chi_{l,k'} + p_j^s - M(2 - \chi_{i,k} - \chi_{j,k'}) \leq C_j^s, \quad \forall i, j, k, k' \in \{1, ..., n\}, k' > k, \forall s \in \mathcal{S} \tag{21}$$

$$L_{\max} \geq C_j^s - d_j^s, \qquad \forall j \in \{1..n\}, \forall s \in \mathcal{S} \tag{22}$$

Constraints (19) assign each job to exactly one group. Constraints (20) and (21) are used to compute the minimum value of the job completion time under each scenario as set by the online algorithm. For a given sequence, at the earliest, the completion time of a job J_j is equal to the release date of some job J_i plus the duration of the jobs that are between J_i and J_j in the sequence,

these two jobs included. Constraints (20) considers the case were J_i and J_j are in the same group. In this case $\sum_{l\in(Succ_i^s\cap Prec_j^s)} p_l^s\chi_{l,k}$ is the total duration of the jobs that are scheduled between J_i and J_j under scenario s according to the ERD rule. Constraints (21) considers the case where J_i and J_j are in different groups. The expression $\sum_{l\in Succ_i^s} p_l^s\chi_{l,k} + \sum_{q=k+1}^{k'-1}\sum_{l=1}^{n} p_l^s\chi_{l,q} + \sum_{l\in Prec_j^s} p_l^s\chi_{l,k'}$ is the total duration of the jobs between jobs J_i and J_j. Constraints (22) bound the worst-case $L_{\max}$. This model contains n^2 binary variables, $n|\mathcal{S}|$ continuous variables and $O(n^4|\mathcal{S}|)$ constraints.

Example

We consider the same example as in Section 4.1. An optimal solution for this example gives two groups $\{J_1, J_3, J_4\}$ and $\{J_2, J_5\}$ with a maximum lateness $L_{\max} = 0$ instead of $L_{\max} = 5$ in the version without groups. Fig. 3 illustrates the solution for each scenario (in each group the jobs are presented in ERD order).

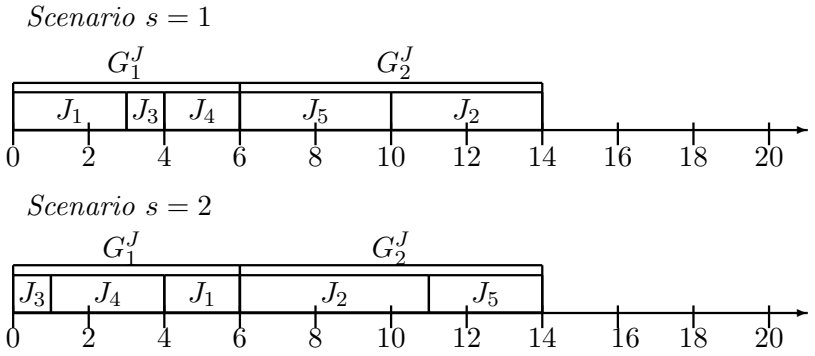

Figure 3: Groups $G_1^J = \{J_1, J_3, J_4\}$ and $G_2^J = \{J_2, J_5\}$ under the two possible scenarios

5.2 MILP for the online recoverable robust scheduling and vehicle routing problems

In the first-stage solution $x = (\pi, \mathcal{B}, \Sigma)$ of the integrated robust scheduling and vehicle routing problem, π is a sequence of groups of permutable delivery jobs while $\sigma(B)$ for each batch $B \in \mathcal{B}$ is a sequence of groups of permutable delivery jobs. The online recovery algorithm $A(x, s)$ as the same behavior on production partial sequence π as in the previous section. For the delivery part, $A(x, s)$ follows the order of the groups and when a delivery is completed inside a group it scans the remaining deliveries and select the nearest one according to the traveling time matrix of the realized scenario, as this travel time is assumed to be known with a sufficient precision at time of departure. This amounts to perform the routing inside a group according to the nearest neighbor rule. The following model is called OSRM and is based on model OSM for the scheduling part. The binary variables associated to the vehicle routing part of the problem are the following:

$\lambda_{j,k}^s$ is equal to 1 if the site J_j is in position k (i.e. the job J_j is the k^{th} job delivered) for scenario s, and 0 otherwise. $\delta_{k,r}$ is equal to 1 if the tour r (i.e. the r^{th} tour) visits the site in position k, and 0 otherwise, $\gamma_{k,h}$ is equal to 1 if the site in position k is in the group of permutable deliveries G_h^D, and a binary variable $\theta_{h,r}$ equal to 1 if the group G_h^D is in the route r. For a scenario s, continuous variables D_j^s and α_r^s represent respectively the arrival date to the site J_j and the departure date of the tour r.

According to these decision variables, we detail model ORSM below.

Objective (18) and constraints (19) - (21) are taken from OSM. As already mentioned, Constraints (19) - (21) set the production job completion times w.r.t. the production group orderings and the ERD rule inside each production group. The constraints related to the routing part are as follows. Constraints (23) and (24) bound from below the departure dates of the tours. Constraints (23) state that a tour r cannot start in scenario s before the completion of all the jobs which are to be delivered by r.

$$\alpha_r^s \geq C_j^s + M(2 - \delta_{kr} - \lambda_{jk}^s), \quad \forall j, r, k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{23}$$

Constraints (24) prevent, for a given scenario s, tour r to start before the end of previous tour $r-1$, which is equal to the delivery date of the last visited site J_j plus the return time to the depot.

$$\alpha_r^s \geq D_j^s + t_{j0}^s + M(2 - \delta_{k,r-1} - \lambda_{jk}^s), \quad \forall j, k \in \{1, ..., n\}, \forall r \in \{2, ..., n\}, \forall s \in \mathcal{S} \tag{24}$$

Constraints (25) to (31) below set the delivery dates of the jobs based on the computed batches, delivery group ordering inside each batch via the decision variables and on the nearest neighbor rule to schedule deliveries inside each groups. Constraints (25) prevent a job J_i at the position $k+1$ in scenario s to be delivered before the delivery of the job J_j at the position k in scenario s.

$$D_j^s \geq D_i^s + t_{ij}^s + M(2 - \lambda_{i,k}^s - \lambda_{j,k+1}^s), \quad \forall i, j \in \{1, ..., n\}, j \neq i, \forall k \in \{1, ..., n-1\}, \forall s \in \mathcal{S} \tag{25}$$

Constraints (26) prevent in turn to deliver a job J_j at position k for scenario s before the departure time of the tour r that delivers it.

$$D_j^s \geq \alpha_r^s + \omega_{0j}^s - M(2 - \delta_{kr} - \lambda_{j,k}^s), \quad \forall j, r, k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{26}$$

Constraints (27) ensure that the nearest neighbor rule is respected into each group, for each scenario. This is achieved by enforcing that the job J_j delivered at position k in scenario s in a group h is the nearest in this scenario from the previous job J_i (delivered at position $k-1$ in the same group h for the same scenario s) among all jobs J_l delivered in group h.

$$\begin{aligned} D_j^s \leq D_i^s + t_{il}^s + M(6 - \gamma_{k-1,h} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s), \\ \forall i, j, l, h \in \{1, ..., n\}, j \neq i, l \neq i, \forall k \in \{2, ..., n\}, \forall k' \in \{k, ..., n\}, \forall s \in \mathcal{S} \end{aligned} \tag{27}$$

Constraints (28) considers a job J_j at position k in scenario s which is the first delivered job of its group h for tour r. Then, if there is a preceding delivery group $h-1$ for the same tour r, J_j must be the nearest among all other delivery jobs J_l in group h in scenario s from the last job J_i of group $h-1$.

$$\begin{aligned} D_j^s \leq D_i^s + t_{il}^s + M(8 - \gamma_{k-1,h-1} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s - \theta_{h-1,r} - \theta_{h,r}), \\ \forall i, j, l, r \in \{1, ..., n\}, j \neq i, l \neq i, \forall h, k \in \{2, ..., n\}, \forall k' \in \{k, ..., n\}, \forall s \in \mathcal{S} \end{aligned} \tag{28}$$

$$\tag{29}$$

Constraints (30) considers again a job J_j at position k in scenario s which is the first delivered job of its group h for tour r, but in the case that group h is the first delivery group of tour r. In that case, if there exists a preceding tour $r-1$, which is assessed by the fact that some job J_i is delivered at position $k-1$, then job J_i must be the nearest from the depot.

$$\begin{aligned} D_j^s \leq \alpha_r^s + t_{0l}^s + M(8 - \gamma_{k-1,h-1} - \lambda_{i,k-1}^s - \gamma_{k',h} - \lambda_{l,k'}^s - \gamma_{k,h} - \lambda_{j,k}^s - \theta_{h-1,r-1} - \theta_{h,r}), \\ \forall i, j, l \in \{1, ..., n\}, j \neq i, l \neq i, \forall r, h, k \in \{2, ..., n\}, \forall k' \in \{k, ..., n\}, \forall s \in \mathcal{S} \end{aligned} \tag{30}$$

Constraints (31) considers the last case for a job J_j at position k in scenario s which is the first delivered job of its group h for tour r. In the case that group h is the first delivery group of tour r but when there is no preceding tour, which is assesed by the fact that job J_j is delivered at position 1, then job J_j must also be the nearest from the depot among all other delivery jobs l of its group.

$$D_j^s \leq \alpha_1^s + t_{0l}^s + M(4 - \gamma_{1,1} - \lambda_{j,1}^s - \gamma_{k,1} - \lambda_{l,k}^s) \quad \forall j, l, k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{31}$$

Since the variables $\gamma_{k,h}$ depend to the positions of the jobs, Constraints (32) ensure that a group contains the same jobs at each scenario.

$$\gamma_{k,h} \geq 1 - M(3 - \lambda_{j,k}^{s'} - \lambda_{j,k'}^{s''} - \gamma_{k',h}), \quad \forall j, h, k, k' \in \{1, ..., n\}, k \neq k', \forall s', s'' \in \mathcal{S}, s'' > s' \tag{32}$$

Constraints (33)-(41) ensure the feasibility of the tours. For a scenario s, each site is assigned to exactly one position (33) and each position to one site (34). A site is assigned to exactly one tour (35) and one group (36), and a group can belong to only one tour (37). Constraints (38) serve to have consecutive tours, stating that if a job is delivered at position k by a tour r, the job delivered at position $k+1$ must be either in tour r or $r+1$. Constraints (38) serve to have consecutive delivery groups using the same principle. Constraints (40) ensure consistency between position assignment to groups and tours, stating that if a group h is assigned to a tour r and if group h contains a job delivered at position k, then tour r must also deliver the job at position k. Constraints (41) assign the job at the first position to the first group and to the first tour, and assign the first group to the first tour. Last, Constraints (42) bound $L_{\max}$.

$$\sum_{j=1}^{n} \lambda_{j,k}^s = 1, \forall k \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{33}$$

$$\sum_{k=1}^{n} \lambda_{j,k}^s = 1, \forall j \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{34}$$

$$\sum_{h=1}^{n} \gamma_{k,h} = 1, \forall k \in \{1, ..., n\} \tag{35}$$

$$\sum_{r=1}^{n} \delta_{k,r} = 1, \forall k \in \{1, ..., n\} \tag{36}$$

$$\sum_{r=1}^{n} \theta_{h,r} = 1, \forall h \in \{1, ..., n\} \tag{37}$$

$$\delta_{kr} + \delta_{k+1,r'} \leq 1, \forall k \in \{1, ..., n-1\}, \forall r, r' \in \{1, ..., n\}, r' \neq r, r' \neq r+1 \tag{38}$$

$$\gamma_{kh} + \gamma_{k+1,h'} \leq 1, \forall k \in \{1, ..., n-1\}, \forall h, h' \in \{1, ..., n\}, h' \neq h, h' \neq h+1 \tag{39}$$

$$\delta_{kr} \geq \gamma_{kh} + \theta_{hr}, \forall k \in \{1, ..., n-1\}, \forall h, h' \in \{1, ..., n\}, h' \neq h, h' \neq h+1 \tag{40}$$

$$\delta_{11} = 1, \gamma_{11} = 1, \theta_{11} = 1, \tag{41}$$

$$L_{\max} \geq D_j^s - d_j^s, \forall j \in \{1, ..., n\}, \forall s \in \mathcal{S} \tag{42}$$

This model contains $O(n^2|\mathcal{S}|)$ binary variables, $n|\mathcal{S}|$ continuous variables and $O(n^7|\mathcal{S}|)$ constraints.

Example

We consider here the same example as in Section 4.2 . We define 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 $\{J_2, J_5\}$. In the first batch, there are two groups of permutable deliveries $G_1^D = \{J_3\}$ and $G_2^D = \{J_1, J_4\}$ and in the second batch there is only one group $G_3^D = \{J_2, J_5\}$. Delivery phases are illustrated in Fig. 4 for each scenario and the global solutions that include the production and delivery phases are illustrated in Fig. 5 for each scenario. For this example, the maximum lateness over all scenarios is $L_{\max} = 0$.

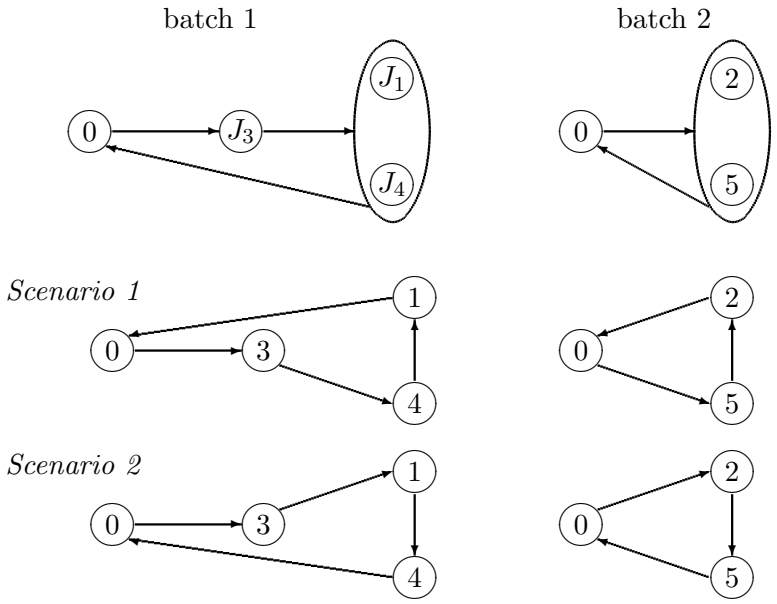

Figure 4: Delivery group sequence $\{J_3\}, \{J_1, J_4\}$ and $\{J_2, J_5\}$ yielding different routes on two possible scenarios

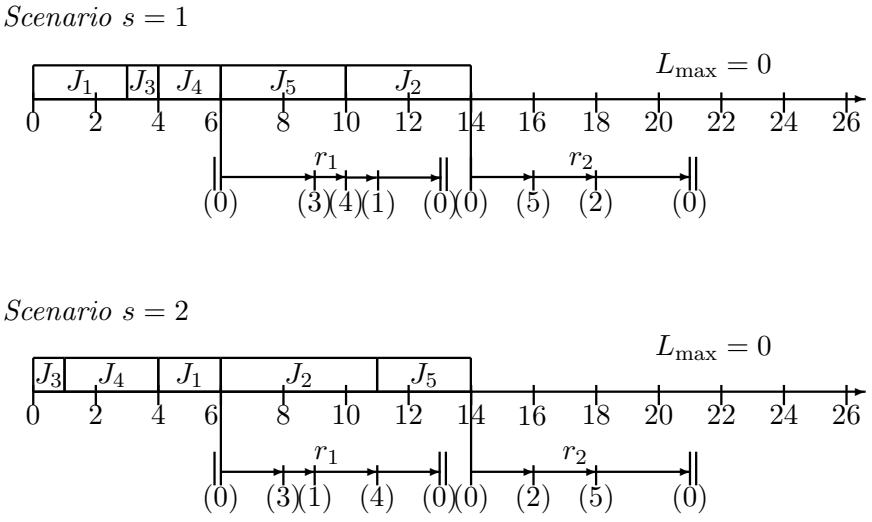

Figure 5: Production and transportation phases under the two possible scenarios

6 Fixed sequence robust and online recoverable robust tabu search

Although we were able to propose MILPs for both the standard robust and the online recovery robust integrated scheduling and routing problems, they can only be used to solve very small problem instances. To overcome this difficulty, we propose in this section two tabu search methods. The first one is called TSR (Tabu search for robust Scheduling & Routing) and solves the fixed sequence robust problem while the second one, called TOSR (Tabu search for Online recoverable robust Scheduling & Routing), deals with the online recovery robust problem. To make a fair comparison of the two robust solution paradigms, the two proposed algorithms are based on the same principles and can be presented jointly.

For TSR, a solution is a production job sequence on the machine, a sequence of delivery batches and, for each batch a delivery sequence (a routing). A solution is evaluated by computing for each scenario, the earliest start time of each production and delivery jobs following the prescribed production and delivery sequences, yielding a maximum delivery lateness over all scenarios.

For TOSR, a solution is a sequence of groups of permutable production jobs, a sequence of delivery batches and, for each batch a sequence of groups of permutable deliveries. However, to simplify the method, we restrict the solution space to the case where there is exactly one group of deliveries per batch, which means that no routing decisions have to be made in advance apart from batching. A solution is evaluated by computing for each scenario an earliest start time for each production job (following the prescribed production group sequence and the earliest release date first rule for scheduling the jobs inside each production group) as well as an earliest delivery time for each delivery job (following the nearest neighbor rule on the realized scenario for delivering the jobs inside each delivery group/batch). The maximum lateness over all scenarios can also be computed from these earliest start times.

In both cases, a quite standard version of the the Tabu Search algorithm has been implemented [Glover, 1990]. This metaheuristic starts from an initial complete 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 an aspiration criterion. The chosen neighbor becomes tabu and the search process is then repeated using this chosen neighbor as a new basic solution.

We propose a decomposition of the neighborhood search into two neighborhoods for TOSR and three neighborhoods for TSR that we search sequentially.

The two algorithms differ in the scheduling sub-problem where the neighbors are different as TOSR includes group-based neighborhoods and in the routing part where there is no decision to take in TOSR as the scenario-aware greedy nearest neighbor heuristic is used, while TSR ultimately optimizes the routes via the scenario-unaware 2-opt heuristic. We selected this policy (i.e. having an optimization process for the routing in TSR and not in TOSR) to have a fair comparison between the two approaches that have different strengths and weaknesses. On the one hand, the standard robust approach has the advantage of optimizing the routing part, while the recovery robust approach can not optimize any routing decision apart from the batching decision as only one group is allowed inside each batch. On the other hand the recovery robust approach has the possibility to adapt to the realized scenario inside each production group and each batch with the above-described online heuristics while the standard robust approach can only use worst-case evaluations. In the following sub-sections we details some key elements of both algorithms.

TOSR and TSR are sketched in Algorithms 6 and 6, respectively. Algorithm TSR uses the concept of reference scenario, which is one of the scenarios as explained later in Section 7.1. The following sections present different key components of the methods.

Algorithm 1 Tabu search algorithm (Online recoverable robust version - TOSR)

COMPUTE INITIAL SOLUTION

- Generate a feasible solution (sequence of groups of permutable jobs and batching) using a greedy algorithm (details in Section 6.1.2).

WHILE THE GLOBAL STOP CONDITION IS NOT SATISFIED DO

- Production sequencing neighborhood
  - For a fixed batching and starting with the current production group sequence, search for the best non-tabu production group sequence move (details in Section 6.1.4).
  - Store the newly selected solution in the scheduling tabu list.
- Batching neighborhood
  - For a fixed production group sequence and starting with the current batching, search the best batching move to get the best non-tabu neighbor solution (details in Section 6.1.4).
  - Store the newly selected solution in the batching tabu list.

Algorithm 2 Tabu search algorithm (Robust version - TSR)

COMPUTE INITIAL SOLUTION

- Generate a feasible solution (sequence of production jobs, batching, routing) using a greedy algorithm (details in Section 6.1.1).

WHILE THE GLOBAL STOP CONDITION IS NOT SATISFIED DO

- Production sequencing neighborhood
  - For a fixed batching and starting with the current production sequence, search the best production sequencing move to get the best non-tabu neighbor solution (routing inside each batch is computed with the NN heuristic on the reference scenario, details in Section 6.1.4).
  - Store the newly selected solution in the scheduling tabu list.
- Batching neighborhood
  - For a fixed production sequence and starting with the current batching, search the best batching move to get the best non-tabu neighbor solution (routing inside each batch is computed with the NN heuristic on the reference scenario, details in Section 6.1.4).
  - Store the newly selected solution in the batching tabu list.
- Routing sub-neighborhood
  - For a fixed production sequencing and batching and starting with the current routing, search the best permutation inside each route using the 2-opt heuristic (descent method only, details in Section 6.1.4).

6.1 Initial solutions

To initialize the methods, two constructive approaches are used. We first present the constructive approach for initializing the TSR algorithm called GSR (Greedy, Scheduling, Routing) then a second approach adapted to the TOSR algorithm called GOSR (Greedy, Online, Scheduling, Routing) is presented.

6.1.1 GSR: initial solution for TSR

In order to build an initial solution for the TSR algorithm, we first find a sequence of the jobs on the machine for the scheduling part. Then, from this sequence, we build batches. First, we sort the jobs according to their average due dates over the scenarios (Earliest Due Date rule -EDD) and from this sequence, batches are built by inserting the job in the current batch or by creating a new batch for this job. More precisely, Algorithm GSR is as follows:

step 1: Sort the jobs according to their average due dates over the scenarios.

step 2: There are two options for each job, the best option being selected at each iteration:

- Insert the job in the current batch.
- Create a new batch and insert the job.

6.1.2 GOSR: Initial solution for TOSR

A second constructive approach called GOSR is used to obtain an initial solution for TOSR. This time, we build simultaneously the groups for the scheduling part and the batches for the routing part. First we sort the jobs according to EDD and, from this sequence, production groups and batches are built at the same time by inserting the job in the current group or by by creating a new group for this job. The same procedure is done for batching: the job is inserted in the current batch or in a new batch. Finally, for each job, we keep the best solution found and we proceed to the next job. Heuristic GOSR is as follows:

step 1: Sort the jobs according to their average due dates over the scenarios.

step 2: There are four options for each job, the best option being selected at each iteration:

- Insert the job in the current production group; insert the job in the current routing group/batch.
- Insert the job in the current production group; create a new routing group/batch with the job.
- Create a new production group with the job; insert the job in the current routing group/batch.
- Create a new production group and a new current routing group/batch with the job.

6.1.3 Encoding

Encoding a solution for TOSR: In the TOSR algorithm, the problem of routing the jobs of a batch is systematically solved by the nearest neighbor algorithm (NN) on the realized scenario. In fact, since we enforce in addition that there is only one group of permutable delivery for each batch, it suffices to have the scheduling part and the composition of the batches to encode a solution. A

solution of the problem S is consequently encoded directly by a vector of size $2n$. The first part of size n contains the scheduling part, whereas the second part of size n contains the batching part. Let v denote the used vector for encoding a solution of the TOSR method. For the first part, we consider that v_j for $1 \leq j \leq n$ is the index of the production group to which job J_j is assigned, where the groups are sequenced in the order of their numbering. In the second part of the vector, we find the assignment to delivery groups/batches, where v_j, for $n+1 \leq j \leq 2n$, is the index of the delivery group/batch to which job J_{j-n} is assigned. The delivery groups/batches are sequenced in the order of their numbering.

The example in Fig. 6 shows a solution encoding corresponding to production groups $G_1^J = \{J_1, J_3, J_4\}$ and $G_2^J = \{J_2, J_5\}$ and delivery batches/groups $G_1^D = \{J_3, J_4\}$, $G_2^D = \{J_1, J_5\}$, $G_3^D = \{J_3\}$.

Encoding a solution for TSR: the encoding for TSR is more subtile as for the first two neighborhoods, the routing is generated with NN on the reference scenario from a fixed production sequence and batching and, consequently, does not require an encoding. For the third neighborhood the routing requires an encoding for the 2-opt neighborhood. Hence for the first two neighborhoods, the solution is encoded with a vector v' of size $2n$ where the first part of v' corresponds to the position of the production jobs in the sequence where (v'_i is the position of job J_i in the schedule) and the second part represents the batching, as in the TOSR algorithm. For the third neighborhood, the vector has a third part of size n representing the routing, where $v'[2n + k]$ corresponds the the job delivered at position k in the concatenation of all batch routings. The encoding of the solution displayed on top of Figure 6 in TSR would be vector $(3, 5, 1, 2, 4|2, 3, 1, 1, 2)$ for the first two neighborhood, with the adjunction of vector $(3, 4, 5, 1, 2)$ to represent the routing in the third neighborhood.

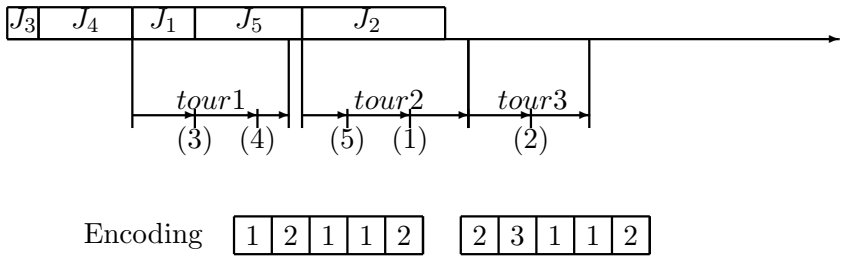

Figure 6: A solution for a scenario and an encoding for the production group sequence/batching

6.1.4 Neighborhood definition

We present in this section the neighborhoods used in each algorithm. The production scheduling neighborhoods are first presented for TOSR and TSR. Then, we present the batching neighborhood, which is the same for both algorithms (although evaluations of a neighbor differ, as already mentioned). Finally we present the 2-opt neighborhood used only in the TSR algorithm. We use a perimeter δ as a limit for the search area from a solution. For each neighborhood, the best non-tabu move is retained over all possible changes.

Production group sequence neighborhood for TOSR:. We denote by $|G|$ the number of groups. The following four neighborhoods are considered:

- 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 production jobs, taking perimeter δ into consideration. Let v_i denote the group of job J_i while v_j is the group of job J_j. The exchange is possible if and only if $v_i \neq v_j$ and $v_i - \delta|G| \leq v_j \leq v_i + \delta|G|$.

- Insertion of a job in a different group: this neighborhood allows a job J_i of group v_i to be inserted in an existing group v_k if $v_i - \delta|G| \leq v_k \leq v_i + \delta|G|$.

- Group split: this neighborhood allows one group to be split into two groups, which postpone the subsequent groups. Since the sequence inside the groups depends on the realized scenario, the number of possible cases to split 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 to the size of the group.

- Group fusion: this neighborhood performs the merging of two consecutive groups into one, which advances the second group.

Production job sequence neighborhood for TSR: Two types of neighborhoods are used:

- 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 we enforce that $|v'_i - v'_j| \leq \delta n$.

- Insertion at a different 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, where, $|v'_i - v'_k| \leq \delta n$. This yields that jobs at positions $v'_{i+1}, .., v'_k$ are advanced in the sequence if $v'_i > v'_k$ or that jobs at positions $v'_k, .., v'_{i-1}$ are postponed if $v'_i < v'_k$.

Delivery batching neighborhood for TSR and Delivery grouping/batching neighborhood for TOSR The neighborhood generation is then the same for determining the delivery batches in TSR and the delivery groups/batches in TOSR. In fact, we have for a delivery batch (TSR) and for a delivery group/batch (TOSR) the same modifications as for a production group for TOSR. This yields four types of neighborhoods that can be derived from the TOSR production group sequence neighborhoods: batches exchange; insertion of a job in a different batch, batch split and batch fusion).

Routing neighborhoods for TSR This neighborhood is the standard 2-opt operator that swaps two jobs at different positions in the third part of the encoding, i.e. for each position j, k with $1 \leq j < k \leq n$, the exchange of jobs $v'[2n + j]$ and $v'[2n + k]$ is evaluated.

6.1.5 Tabu list

To discourage the tabu algorithm to select the solutions that have been visited recently, the tabu list contains solutions that cannot be selected. These solutions are called Tabu. In our case, since the scheduling part and the batching part are treated separately, we opted for two tabu lists with a fixed size equal to the number of jobs. The first list contains the part of the vector corresponding to scheduling while the second list contains the part corresponding to the batches. We have no tabu list for the routing part in TSR as the 2-opt algorithm is applied as a descent method.

6.1.6 Aspiration criterion

A solution is considered tabu if either the scheduling part or the routing part is tabu. For this reason, a solution is selected if it is better than the best solution ever found, even if it is tabu as in standard tabu search.

6.1.7 Global stopping condition

The general algorithm stops when a global time limit is reached. However, for the scheduling sub-problem and the batching sub-problem, there is no specific time limit but the algorithm stops after a fixed number of iterations without improvment.

7 Experimentations

7.1 Data generation

To validate our models, we have generated a random data set as follows. We call scenario $s = 1$ the "reference scenario" and we first generate instance (r_j^1, p_j^1, d_j^1). For this, we first generate the p_j^1 for all the jobs in the interval $[1, 50]$ 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. Due 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 distance matrix 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.

$$t_{i,j}^1 = t_{j,i}^1 = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$$

For modeling the uncertainty on the data, each scenario s ($s \geq 2$) is generated as follows:

- r_j^s is generated in $[(1 - \omega^r)r_j^1, (1 + \omega^r r_j^1)]$ with a given coefficient ω^r
- p_j^s is generated in $[(1 - \omega^p)p_j^1, (1 + \omega^p p_j^1)]$ with a given coefficient ω^p
- d_j^s is generated in $[(1 - \omega^d)d_j^1, (1 + \omega^d d_j^1)]$ with a given coefficient ω^d
- X_j^s is generated in $[(1 - \omega^X)X_j^1, (1 + \omega^X X_j^1)]$ with a given coefficient ω^X
- Y_j^s is generated in $[(1 - \omega^Y)Y_j^1, (1 + \omega^Y Y_j^1)]$ with a given coefficient ω^Y

The changes of the coordinates of the sites can be explained, if we consider that a site is a city and the change of position corresponds to different possible locations in the same city but it can also be seen as a simple way to vary the traveling time between two sites. For practical reasons, all the values of the coefficients ω are identical.

7.2 Computational results

In this section, we evaluate our tabu search algorithms on the instances presented above. The number of jobs varies from 10 to 100, the parameters α, β and γ are fixed to 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 a tuple (n, ω), and, for each n and ω, 10 instances are generated with a number of scenarios $\mathcal{S}' = 50$. The practical limitations of the proposed MILP formulations has led us to omit the presentation of the results. However, for the

solved instances with a number of jobs $n = 10$, the obtained results are equal to that obtained by the tabu search heuristic for both cases (TSR and TOSR). Note that, the fact that in the TOSR algorithm it is assumed that there are exactly one group of permutable delivery in each batch led us to adapt our formulation for the comparison.

The comparison takes place in two phases. In a "training" phase, we define a subset of scenarios (the training set) $\mathcal{S} \subset \mathcal{S}'$. We compare on $\mathcal{S}$ the GSR and TSR algorithms for the fixed sequence robust approach with the GOSR and TOSR algorithms for the online robust approach. In a second phase, we keep the best production job sequence, batching and routing found by GSR and TSR on set $\mathcal{S}$, and the best production group sequence and batching found by GOSR and TOSR on the same set. Then, we evaluate and compare the performance of these solutions on the scenario set $\mathcal{S}'$. This allows to evaluate these solutions on a scenario that was not anticipated and hence to provide additional informal information on the robustness of the methods.

The instances are solved on a 2.4 GHz Intel i3 computer with 3 GB of RAM.

7.2.1 Comparisons on the training scenario set $\mathcal{S}$

In this section, the tabu search algorithms TSR and TOSR as well as the greedy algorithms GSR and GOSR presented above are tested on instances having a number of jobs from 10 to 100, a deviation $\omega \in \{20, 40, 60\}$ and 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 is set to 0.4. For each $(n, \mathcal{S}, \omega)$, the algorithms are tested for the ten instances and the results displayed in Table 1 give the average worst-case maximum lateness on the ten instances. The presented results are divided into three blocks. The first block provides the results of the fixed sequence robust approaches (GSR and TSR). The second one concerns the online recoverable robust approaches (GOST and TOSR) and the last one compares the fixed sequence and the online recoverable robust approaches. Let $z(\mathcal{A})$ denote the objective function value (i.e. the maximum lateness over scenario set $\mathcal{S}$) returned by algorithm $\mathcal{A} \in \{$TSR, TOSR, GSR, GOSR$\}$ on a particular instance. Columns *ttb* give the average CPU time needed. Δ is the average gap between the initial solution found the the greedy approach and the tabu search solution and it is equal to $(z(GSR) - z(TSR))/z(GSR)$ in the first block and $(z(GOSR) - z(TOSR))/z(GOSR)$ for the second. In the last block, $\Delta^1 = (z(GSR) - z(GOSR))/z(GSR)$ is the average gap between the initial solutions found by GSR and GOSR and $\Delta^2 = (z(TSR) - z(TOSR))/z(TSR)$ is the average gap between the solutions found by TSR and TOSR.

First, Column Δ^1 reveals that the greedy online recoverable robust algorithm GOSR is far more better than the greedy fixed sequence robust algorithm GSR. This could be expected as the optimization level is very low in GSR whereas GOSR can react to the realized scenario thanks to the group structure. Second, the first and the second block show that for both approaches the tabu search algorithm improves significantly the performances of the greedy algorithms. Interestingly, TSR and TOSR have comparable CPU times on comparable instances, which means that considering an additional decision level for grouping does not yield a computational overhead. Finally, column Δ^2 shows that in most cases the online recoverable robust approach improves upon the robust approach. However, this is not always the case and it can be seen that the improvement increases with the number of jobs and the deviation parameter, which indicates the uncertainty level.

7.2.2 Assessing the robustness of the obtained job/group sequences and batching on scenario $\mathcal{S}'$

In this part, we fix, for each instance, the best production job sequence, batching and routing found by GSR and TSR (denoted x^{GSR} and x^{TSR}), and the best production group sequence and batching

			TSR		TOSR		TSR Vs TOSR	
n	$\mathcal{S}$	ω	$ttb(s)$	$\Delta(\%)$	$ttb(s)$	$\Delta(\%)$	$\Delta^1(\%)$	$\Delta^2(\%)$
10	2	20	0.01	50.42	0.08	23.02	38.31	5.58
		40	0.01	50.09	3.67	23.01	36.08	1.95
		60	0.01	51.39	0.20	26.98	28.75	-4.32
	5	20	7.79	50.27	0.18	22.78	35.77	1.30
		40	0.29	41.71	10.36	18.65	29.67	0.54
		60	0.05	33.46	0.13	17.32	27.46	10.24
	10	20	0.04	48.41	3.02	17.74	38.26	2.40
		40	0.08	41.12	0.64	16.31	31.19	-0.24
		60	0.21	37.97	8.47	13.74	26.85	-0.83
20	2	20	2.70	70.48	17.76	32.90	56.38	2.44
		40	16.07	61.49	8.74	27.47	47.68	1.32
		60	19.33	61.67	6.30	22.15	51.43	3.78
	5	20	3.68	65.69	2.01	28.20	54.60	3.92
		40	9.52	56.03	10.87	21.66	47.08	4.72
		60	10.45	56.69	9.51	17.87	47.79	2.00
	10	20	28.82	62.76	14.32	30.32	48.98	4.88
		40	30.38	53.34	16.68	20.46	45.03	5.62
		60	23.60	50.09	5.55	14.81	44.12	5.01
30	2	20	17.36	68.89	9.96	27.44	55.82	0.88
		40	30.93	64.96	11.63	22.54	55.30	4.50
		60	12.51	67.87	6.78	26.11	57.20	1.44
	5	20	25.88	66.36	27.59	28.28	54.56	3.59
		40	22.26	58.97	7.51	16.81	52.19	4.22
		60	42.83	59.24	24.24	22.11	49.20	3.43
	10	20	15.28	63.25	22.79	23.42	54.50	5.01
		40	25.35	56.45	16.90	16.47	48.80	2.49
		60	21.26	54.39	15.54	20.25	45.54	5.46
50	2	20	24.47	71.33	2.49	21.29	64.22	2.31
		40	25.45	67.42	5.26	21.25	59.17	3.10
		60	14.56	65.27	10.02	18.55	59.16	4.59
	5	20	30.13	68.33	16.19	24.15	59.45	4.27
		40	24.16	61.62	27.54	19.81	54.49	5.26
		60	32.86	56.53	31.75	16.09	51.00	5.95
	10	20	50.60	65.60	32.79	21.16	59.24	6.76
		40	58.49	59.56	56.41	21.99	49.48	3.19
		60	47.13	54.79	55.31	18.01	48.37	6.44
100	2	20	52.73	70.49	21.22	21.14	62.70	1.30
		40	59.31	64.46	25.72	14.83	60.54	5.41
		60	56.97	62.26	9.65	8.61	61.09	4.90
	5	20	109.51	65.84	64.50	15.02	61.21	4.39
		40	110.73	60.82	66.68	20.45	52.08	3.13
		60	113.04	54.41	64.40	13.80	49.11	3.77
	10	20	119.02	58.36	116.47	16.91	58.37	16.15
		40	120.63	55.47	117.90	16.01	51.64	9.28
		60	122.56	44.55	127.80	13.58	47.90	17.82

Table 1: Comparisons of GSR, GOSR, TSR and TOSR methods for $|\mathcal{S}| \in \{2, 5, 10\}$

found by GOSR and TOSR (denoted x^{GOSR} and x^{TOSR}), found during the optimization on the training set. We evaluate their performance by computing the worst case maximum lateness on the extended scenario set $\mathcal{S}'$.

The results presented in Table 2 are also divided into three blocks. The first block concerns the results of the fixed sequence robust solutions (x^{GSR} and x^{TSR}), the second concerns the online recoverable robust solutions (x^{GOSR} and x^{TOSR}) and the last one is also meant to compare the two approaches.

Let $z(x) = \max_{s\in\mathcal{S}'}\{L_{\max}(x,s)\}$ for $x \in \{x^{\mathrm{GSR}}, x^{\mathrm{GOSR}}\}$ the maximum lateness of the fixed sequence robust solutions on scenario set $\mathcal{S}'$ and let $z^A(x) = \max_{s\in\mathcal{S}'}\{L_{\max}(A(x,s),s)\}$ the maximum lateness of the online recoverable sequence robust solutions on the same scenario set, where $A(x,s)$ is the online algorithm used on the realized scenario (i.e. ERD for the production part and NN for the routing part). Columns $\Delta^{'}$ give the average gap between the initial solution and the tabu search solution, which is equal to $(z(x^{\mathrm{GSR}}) - z(x^{\mathrm{TSR}}))/z(x^{\mathrm{GSR}})$ in the first block and $(z^A(x^{\mathrm{GOSR}}) - z^A(x^{\mathrm{TOSR}}))/z^A(x^{\mathrm{GOSR}})$ in the second.

However considering each scenario in $\mathcal{S}'$ as a possible outcome outside the training set $\mathcal{S}$, the maximum performance on this set of each method separately has a limited interest and we would rather like to know how each solution issued from a different method behaves on the same unexpected scenario. Columns $\bar{\Delta}'$ give the following average comparisons where

$$\bar{\Delta}' = average_{s\in\mathcal{S}'}\frac{L_{\max}(x^{\mathrm{GSR}},s) - L_{\max}(x^{\mathrm{TSR}},s)}{L_{\max}(x^{\mathrm{GSR}},s)}$$

for the fixed sequence robust approach and

$$\bar{\Delta}' = average_{s\in\mathcal{S}'}\frac{L_{\max}(A(x^{\mathrm{GOSR}},s),s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{GOSR}},s),s)}$$

for the online recoverable robust approach. The comparison between the two robust approaches is given in the third block where the first two columns $\Delta^{'(1)}$ and $\bar{\Delta}^{'(1)}$ concern the initials solutions and $\Delta^{'(2)}$ and $\bar{\Delta}^{'(2)}$ concern the tabu search solutions. $\Delta^{'(1)}$ and $\bar{\Delta}^{'(1)}$ give the gap between the initials solutions when a solution is the worst case in $\mathcal{S}'$, and the average gap relative to each scenario, respectively. $\Delta^{'(2)}$ and $\bar{\Delta}^{'(2)}$ indicate the same information for the tabu search solutions.

$$\Delta^{'(1)} = \frac{z(x^{\mathrm{GSR}}) - z^A(x^{\mathrm{GOSR}})}{z(x^{\mathrm{GSR}})}$$

$$\bar{\Delta}^{'(1)} = average_{s\in\mathcal{S}'}\frac{L_{\max}(x^{\mathrm{GSR}},s) - L_{\max}(A(x^{\mathrm{GOSR}},s),s)}{L_{\max}(x^{\mathrm{GSR}},s)}$$

$$\Delta^{'(2)} = \frac{z(x^{\mathrm{TSR}}) - z(x^{\mathrm{TOSR}})}{z(x^{\mathrm{TSR}})}$$

$$\bar{\Delta}^{'(2)} = average_{s\in\mathcal{S}'}\frac{L_{\max}(x^{\mathrm{TSR}},s) - L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(x^{\mathrm{TSR}},s)}$$

Looking first at columns $\Delta^{'(1)}$ and $\Delta^{'(1)}$, it appears that the solutions issued from the online recoverable robust approaches (either greedy or tabu) are far more robust than the solutions issued from their standardly robust counterparts, both in the worst case and average setting. Again the ability of the group structure to react to realized scenarios in a controlled way, even with a simple online algorithm, is demonstrated. Looking now at columns TSR and TOSR, if the improvement brought by TSR over GSR is again assessed, the average comparison shows that TOSR may provide solutions that perform worse than those of GOSR in average on scenario $\mathcal{S}'$.

			TSR		TOSR					
n	$\mathcal{S}$	ω	$\Delta^{'}$ (%)	$\bar{\Delta}^{'}$(%)	$\Delta^{'}$ (%)	$\bar{\Delta}^{'}$(%)	$\Delta^{'(1)}$ (%)	$\bar{\Delta^{'(1)}}$(%)	$\Delta^{'(2)}$ (%)	$\bar{\Delta^{'(2)}}$(%)
10	2	20	53.99	36.37	37.42	15.56	50.03	34.69	35.88	12.89
		40	49.35	26.34	40.53	8.30	55.33	28.24	42.86	10.49
		60	47.78	10.73	30.93	-14.09	55.62	27.11	40.89	8.47
	5	20	55.98	40.31	36.21	15.40	51.43	34.96	34.08	8.25
		40	47.46	25.38	36.45	5.89	53.29	30.89	46.62	13.23
		60	39.25	15.26	30.33	-3.77	57.44	29.74	48.83	14.70
	10	20	56.12	42.73	32.25	14.31	52.75	38.37	33.48	9.17
		40	47.07	26.70	39.62	7.31	55.84	31.12	42.55	12.37
		60	49.28	19.73	31.81	1.98	54.60	25.72	39.89	9.30
20	2	20	67.61	52.20	44.61	11.73	67.49	54.92	45.08	18.23
		40	58.68	34.08	32.44	-7.66	65.05	44.36	38.60	9.29
		60	56.85	19.26	26.49	-15.45	62.94	41.24	43.17	16.03
	5	20	68.71	57.24	44.09	14.99	69.62	55.96	40.04	12.70
		40	58.26	39.59	34.45	2.95	65.65	46.85	39.01	14.98
		60	56.95	28.89	30.77	-5.46	65.22	43.61	48.35	17.14
	10	20	67.40	56.96	49.67	22.72	65.79	51.67	40.52	15.22
		40	59.96	42.66	35.04	7.53	66.14	46.46	37.81	13.55
		60	57.03	32.52	30.87	-2.82	62.97	42.93	42.73	13.47
30	2	20	66.30	50.19	33.11	7.12	66.32	55.24	40.95	17.78
		40	58.51	30.80	24.84	-7.24	67.15	48.65	42.33	20.37
		60	60.52	22.25	30.36	-8.97	64.62	43.61	46.64	19.86
	5	20	65.05	53.31	37.01	13.06	68.00	57.21	42.80	19.35
		40	56.70	37.07	27.30	-4.03	69.09	51.82	48.82	21.10
		60	58.24	31.79	27.42	-6.89	65.59	46.74	41.99	17.25
	10	20	64.18	54.10	38.08	14.82	69.08	57.60	44.02	20.33
		40	63.57	45.65	23.76	0.78	68.53	51.69	36.88	12.01
		60	59.58	40.37	33.58	0.84	64.36	45.96	37.97	11.74
50	2	20	64.40	49.50	25.11	4.77	71.88	59.75	43.24	23.48
		40	58.81	31.21	23.70	-2.59	66.27	50.80	48.29	27.06
		60	60.60	22.45	20.20	-6.07	64.60	46.10	49.41	25.98
	5	20	67.36	56.95	33.67	10.06	69.23	59.45	36.70	16.21
		40	58.30	41.19	27.58	1.14	66.47	53.02	45.87	21.72
		60	54.23	31.08	22.62	-4.53	61.08	46.84	39.43	19.08
	10	20	67.76	57.26	30.26	12.48	71.84	60.63	37.61	19.14
		40	62.54	47.83	32.06	8.17	65.45	52.15	37.96	16.15
		60	60.94	41.35	29.50	3.70	63.69	48.43	39.29	15.37
100	2	20	64.35	49.01	22.47	5.25	69.63	60.24	42.98	26.06
		40	59.71	30.93	20.29	1.74	64.07	50.36	46.21	28.27
		60	55.11	15.57	10.77	-1.99	61.98	42.67	46.88	29.74
	5	20	64.12	53.99	17.42	-1.81	70.40	61.88	33.71	15.39
		40	58.39	40.51	21.64	-4.07	63.62	52.98	35.73	17.55
		60	52.19	28.37	16.35	-6.00	61.15	47.53	41.00	21.53
	10	20	58.99	51.58	20.92	3.64	69.68	61.48	37.94	23.04
		40	55.29	43.53	20.31	2.26	64.74	53.56	38.69	19.43
		60	44.24	30.29	16.22	-1.64	59.96	49.21	42.44	25.33

Table 2: Assessing the robustness of the GSR, GOST, TSR and TOSR solutions on scenario set $\mathcal{S}'$

To exhibit a possible explanation of this behavior, Table 3 gives an aggregation of the results presented in Table 2. These results are divided into two blocks in which all algorithms are compared relatively to TOSR. More precisely, $\Delta^{''(1)}$ and $\Delta^{''(4)}$ compare GSR and TOSR by computing for each instance, the gap

$$average_{s\in\mathcal{S}'}\frac{L_{\max}(x^{\mathrm{GSR}},s)-L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

$\Delta^{''(2)}$ and $\Delta^{''(5)}$ compare GOSR and TOSR by computing for each instance, the gap

$$average_{s\in\mathcal{S}'}\frac{L_{\max}(A(x^{\mathrm{GOSR}},s),s)-L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

and $\Delta^{''(3)}$ and $\Delta^{''(6)}$ compare TSR and TOSR by computing for each instance, the gap

$$average_{s\in\mathcal{S}'}\frac{L_{\max}(x^{\mathrm{TSR}},s)-L_{\max}(A(x^{\mathrm{TOSR}},s),s)}{L_{\max}(A(x^{\mathrm{TOSR}},s),s)}$$

The first block (columns $\Delta^{''(1)}$, $\Delta^{''(2)}$ and $\Delta^{''(3)}$) present the above-defined gaps averaged over all ω values (i.e. independently of the uncertainty level), while the second block (columns $\Delta^{''(4)}$, $\Delta^{''(5)}$ and $\Delta^{''(6)}$) give the gaps averaged over over all $|\mathcal{S}|$ values (i.e. independently of the training set size level).

n	$\mathcal{S}$	$\Delta^{''(1)}(\%)$	$\Delta^{''(2)}(\%)$	$\Delta^{''(3)}(\%)$	ω	$\Delta^{''(4)}(\%)$	$\Delta^{''(5)}(\%)$	$\Delta^{''(6)}(\%)$
10	2	63.90	10.13	16.43	20	95.77	21.89	14.34
20		106.61	3.42	23.40		193.07	27.76	25.68
30		110.08	2.82	29.07		179.02	18.63	28.28
50		120.51	1.11	38.31		187.30	13.24	28.61
100		120.05	2.74	42.43		172.49	3.75	30.63
10	5	69.23	11.37	18.33	40	68.10	12.83	20.05
20		128.59	10.81	22.72		98.96	5.25	18.32
30		129.54	5.52	28.77		110.32	0.06	26.66
50		133.52	5.82	26.72		123.56	5.35	31.45
100		120.44	-2.36	24.93		115.30	1.35	31.23
10	10	72.94	12.95	16.35	60	42.20	-0.27	16.72
20		129.86	15.52	20.50		73.04	-3.27	22.62
30		136.92	9.61	20.90		87.21	-0.74	23.79
50		149.37	11.65	23.63		92.53	-0.01	28.59
100		132.97	2.67	32.42		85.68	-2.05	37.92

Table 3: Aggregated comparisons of the solutions on scenario set $\mathcal{S}'$

Figures 7 and 8 synthesize table 3, where color blue represents the columns $\Delta^{''(1)}$ and $\Delta^{''(4)}$, color red represents the columns $\Delta^{''(2)}$ and $\Delta^{''(5)}$, and color green represents the columns $\Delta^{''(3)}$ and $\Delta^{''(6)}$. From top to bottom in Figure 7 we have the results when $s = 2$, $s = 5$ and $s = 10$. In the same way, in Figure 8, we have the results when $\omega = 20\%$, $\omega = 40\%$ and $\omega = 60\%$.

Table 3 and Figures 7, 8 show first that the greedy fixed sequence robust approach GSR is always outperformed by the other methods. Also the improvement brought by the online robust recoverable tabu search TOSR on the standard fixed sequence tabu search TSR is stable independently of ω and $|\mathcal{S}|$. However the improvement brought by TOSR on the greedy recoverable robust approach is stable with the number of training scenarios but it decreases with the number of jobs and with

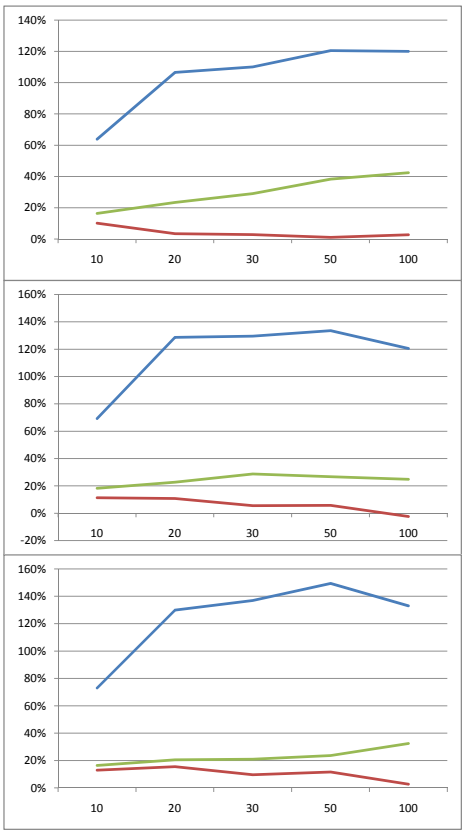

Figure 7: Comparison for different $|\mathcal{S}|$

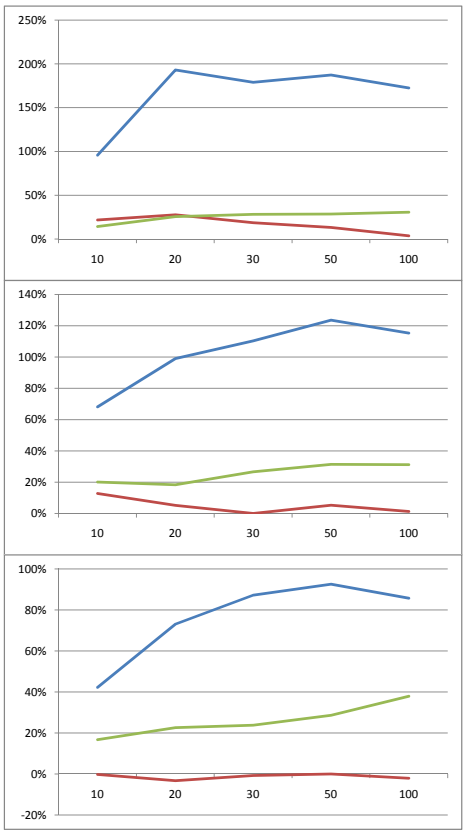

Figure 8: Comparison for different Ω

the uncertainty level ω. GOSR is even slightly better than TOSR for 100 jobs and $|\mathcal{S}| = 5$ as well as for $\omega = 60$. One explanation would be that when the uncertainty level is too high, the effort for optimizing the production group sequence becomes useless.

8 Conclusion and future research directions

In this paper we investigated a novel use of the concept of groups of permutable jobs, introduced in [Billaut and Roubellat, 1996], for an online recoverable robustness approach in an integrated production scheduling and delivery routing problem when uncertainty is modeled via discrete scenarios. The main novelty of the method is that it builds sequences of groups of production and delivery jobs that are fixed for all scenarios, while the job sequence inside each production or delivery group is constructed by an online greedy algorithm that uses in real time the available information on the realized scenario. With the aim to compare the standard robust scheduling approach with the new approach, we have proposed mixed integer linear programming formulations, greedy algorithms and tabu search methods for both approaches. On randomly generated instances of reasonable size, we compared the robustness of the heuristic approaches on a small size training scenario set and we compared the robustness of the solutions issued from the training phase on an extended scenario set. In both cases, the computational experiments show that for comparable CPU times the online recoverable robust approaches outperform the standard robust approaches. However, there seems to to be no more interest in optimizing the group sequences when the uncertainty level becomes too high.

References

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

[Aissi et al. 2009] H. Aissi, C. Bazgan and D. Vanderpooten (2009) Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research* 197(2), pp. 427–438.

[Aloulou and Della Croce, 2008] M. A. Aloulou and F. Della Croce (2008). Complexity of single machine scheduling problems under scenario-based uncertainty. *Operations Research Letters* 36(3), pp. 33!–342.

[Armstrong et al., 2008] R. Armstrong, S. Gao and L. Lei (2008). A zero-inventory production and distribution problem with a fixed customer sequence. *Annals of Operations Research*, 159(1), pp. 395–414.

[Artigues et al., 2005] C. Artigues, J-C. Billaut and C. Esswein (2005). Maximization of solution flexibility for robust shop scheduling *European Journal of Operational Research*, 165 (2), pp. 314-328.

[Aytug et al., 2005] H. Aytug, M. A. Lawley, K. McKay, S. Mohan and R. Uzsoy (2005). Executing production schedules in the face of uncertainties: a review and some future directions, *European Journal of Operational Research*, 161(1), pp. 86-110.

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

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

[Bertsimas and Sim, 2003] D. Bertsimas and M. Sim (2004). The price of robustness. *Operations Research*, 52(1), pp. 35–53.

[Bertsimas and Simchi-Levi, 1996] D. J. Bertsimas and D. Simchi-Levi (1996). A new generation of vehicle routing research: robust algorithms, adressing uncertainty, *Operations Research*, 44(2), pp. 286–304.

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

[Billaut et al., 2008] J-C. Billaut, A. Moukrim and E. Sanlaville, eds. (2008). *Scheduling with Flexibility and Robustness*, ISTE Ltd, Wiley, London.

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

[Chang and Lee, 2004] Y-C. Chang and C-Y. Lee (2004). Machine scheduling with job delivery consideration, *European Journal of Operational Research*, 158, 470-487.

[Chen, 2004] Z-L. Chen (2004). Integrated production and distribution operations: taxonolmy, models and review, in *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, D. Simchi-Levi, S. D. Wu and Z. M. Shen (Eds), Kluwer Academic Publishers.

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

[Chen, 2010b] J-S. Chen (2010). Integration of job scheduling with delivery vehicles routing, *Information technology journal*, 9(6), pp. 1202–1206.

[Condotta et al., 2013] A. Condotta, S. Knust, D. Meier and N. V. Shakhlevich (2013). Tabu search and lower bounds for a combined productiontransportation problem, *Computers & Operations Research*, 40(3), pp. 886–900.

[Daniels and Kouvelis, 1995] R. L. Daniels and P. Kouvelis (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production, *Management Science*, 41 (2), pp. 363-376.

[Dong et al., 2013] J. Dong, A. Zhang, Y. Chen and Q. Yang (2013). Approximation algorithms for two-machine open shop scheduling with batch and delivery coordination, *Theoretical Computer Science*, 491, pp. 94–102.

[Gabrel et al. 2014] V. Gabrel and C. Murat and A. Thiele (2014) Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3), pp. 471–483.

[Gao et al., 2015] S. Gao, L. Qi and L. Lei (2015). Integrated batch production and distribution scheduling with limited vehicle capacity, *International Journal of Production Economics*, 160, pp. 13–25.

[Geismar et al., 2008] H. N.Geismar, G. Laporte, L. Lei and C. Sriskandarajah (2008). The integrated production and transportaion scheduling problem for a product with a short lifespan. INFORMS Journal on Computing, 20(1), pp. 21–33.

[Gendreau et al., 1996] M. Gendreau, G. Laporte and R. Sguin. Stochastic vehicle routing (1996). *European Journal of Operational Research*, 88 (1), pp. 3-12.

[Glover, 1990] F. Glover (1990). Tabu search, Part I, *ORSA Journal on Computing*, 1 (3), 190-206.

[Gounaris et al., 2013] C. E. Gounaris, W. Wiesemann abd C. A. Floudas (2913). The Robust Capacitated Vehicle Routing Problem Under Demand Uncertainty. *Operations Research*, 61(3), pp. 677–693.

[Graham et al., 1979] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals Discrete Mathematics*, 4, pp. 287–326.

[He et al., 2006] Y. He, W. Zhong and H. Gu (2006). Improved algorithms for two single machine scheduling problems, *Theoretical Computer Science* 363, pp. 257-265.

[Herroelen and Leus, 2004] W. Herroelen and R. Leus (2004). Robust and reactive project scheduling: A review and classification of procedures *International Journal of Production Research*, 42 (8), pp. 1599-1620.

[Herroelen and Leus, 2005] W. Herroelen and R. Leus (2005). Project scheduling under uncertainty, survey and research potentials, *European Journal of Operational Research*, 165 (2), pp. 289–306.

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

[Lee at al., 2014] J. Lee, B-I. Kim, A. L. Johnson and K. Lee (2014). The nuclear medicine production and delivery problem, *European Journal of Operational Research*, 236, pp. 461–472.

[Levin and Penn, 2008] A. Levin and M. Penn (2008). Approximation algorithm for minimizing total latency in machine scheduling with deliveries, *Discrete Optimization*, 5, pp. 97-107.

[Li et al., 2005a] C-L. Li, G. Vairaktarakis and C-Y. Lee (2005). Machine scheduling with deliveries to multiple customer locations, *European Journal of Operational Research*, 164 (1), pp. 39–51.

[Li and Ou, 2005b] C-L. Li and J. Ou (2005). Machine scheduling with pickup and delivery, *Naval Research Logistics*, 52, pp. 617-630.

[Li and Vairaktarakis, 2007] C.-L. Li and G. Vairaktarakis (2007) Coordinating production and distribution of jobs with bundling operations *IIE Transactions*, 39 (2), pp. 203-215.

[Liebchen et al., 2009] C. Liebchen, M. Lübbecke, R. Möhring and S. Stiller (2009) 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

[Lu et al., 2008] L. Lu, J. Yuan and L. Zhang (2008). Single machine scheduling with release dates and job delivery to minimize the makespan, *Theoretical Computer Science*, 393, pp. 102–108.

[Ng and Lu, 2011] C. T. Ng and L. Lu (2011). On-line integrated production and outbound distribution scheduling to minimize the maximum delivery completion time, *Journal of Scheduling*, 15(3), pp. 391–398.

[Scholz-Reiter et al., 2010] B. Scholz-Reiter, E. M. Frazzon and T. Makuschewitz (2010). Integrating manufacturing and logistic systems along global supply chains, *CIRP Journal of Manufacturing Science and Technology*, 2, pp. 216–223.

[Ullrich, 2013] C. A. Ullrich (2013). Integrated machine scheduling and vehicle routing with time windows, *European Journal of Operational Research*, 227, pp. 152–165.

[Verderame et al., 2010] P. M. Verderame, J. A. Elia, J. Li and C. A. Floudas (2010) *Industrial & Engineering Chemisty Research*, 49(9), pp. 3993–4017.

[Viergutz and Knust, 2014] C. Viergutz and S. Knust (2014), Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research*, 213(1), pp. 293–318.

[Wang and Cheng, 2009] X. Wang and T.C.E. Cheng (2009). Production scheduling with supply and delivery considerations to minimize the makespan, *European Journal of Operational Research*, 194, pp. 743–752.

[Wang and Liu, 2013] L. Wang and Z. Liu (2013). Single machine scheduling with batch delivery to multiple customers in a star-shaped network, *Asia-Pacific Journal of Operational Research*, 30 (1), art. no. 1250048.

[Wang et al., 2014] Wang, D. Y., Grunder, O., Moudni, A. E. Integrated scheduling of production and distribution operations: a review. *International Journal of Industrial and Systems Engineering*, **19**(1), 94–122.

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

[Zhong et al., 2007] W-Y. Zhong, G. Dosa and Z-Y. Tan (2007). On the machine scheduling problem with job delivery coordination, *European Journal of Operational Research*, 182, pp. 1057–1072.