EURO Journal on Transportation and Logistics manuscript No.
(will be inserted by the editor)

Multi-period vehicle loading with stochastic release dates

Yasemin Arda · Yves Crama · David Kronus · Thierry Pironet · Pascal Van Hentenryck

Received: date / Accepted: date

Abstract This paper investigates a multi-period vehicle loading problem with stochastic information regarding the release dates of items to be transported. The deterministic version of the problem can be formulated as a large-scale set covering problem. Several heuristic algorithms are proposed to generate decision policies for the stochastic optimization model over a long rolling horizon. The resulting policies have been extensively tested on instances which display the main characteristics of the industrial case-study that motivated the research. The tests demonstrate the benefits of the multi-period stochastic model over simple myopic strategies. A simple and efficient heuristic is shown to deliver good policies and to be robust against errors in the estimation of the probability distribution of the release dates.

Keywords transportation planning · production-transportation coordination · vehicle loading · stochastic programming · set covering · heuristics · consensus algorithms · robustness

Yasemin Arda · Yves Crama · Thierry Pironet
QuantOM, HEC Management School, University of Liège, Rue Louvrex 14, 4000 LIEGE, BELGIUM
Tel.: +32-42327211
Fax: +32-42327240
E-mail: yasemin.arda@ulg.ac.be
E-mail: yves.crama@ulg.ac.be
E-mail: thierry.pironet@ulg.ac.be

David Kronus
Komunardu 31, Prague 7, CZECH REPUBLIC
E-mail: dave@matfyz.cz

Pascal Van Hentenryck
Optimization research Group NICTA, Victoria research Laboratory, and Computing and Information Systems Department, University of Melbourne, AUSTRALIA
E-mail: pvh@nicta.com.au

1 Introduction

1.1 Coordination of production and transportation

Production and outbound transportation are consecutive tasks in the supply chain of most manufacturing companies. The associated planning processes, such as production scheduling and transportation planning, have been thoroughly investigated in the operations management literature, and numerous optimization models tackle production and transportation decisions. Some of these models integrate production-transportation decisions, with a view towards better coordination of the supply chain; representative examples of this trend of research can be found, for instance, in Boudia et al. (2007), Chen (2004), Chen (2010), Fumero & Vercellis (1999), Melo & Wolsey (2010), Sarmiento & Nagi (1999), Erengüç et al. (1999), Stecke & Zhao (2007), Tang et al. (2007), Vidal & Goetschalckx (1997), Wang & Cheng (2009), Yung et al. (2006), and Zegordi et al. (2010). As pointed out in the literature surveys by Chen (2004), Chen (2010), and Vidal & Goetschalckx (1997), most models focus on coordination either at the strategic level or at the tactical level: the objective is then to design efficient supply networks, or to take transportation costs into account in the framework of production planning. Some authors, like Boudia et al. (2007), Wang & Cheng (2009), or Zegordi et al. (2010), deal with operational decisions, but this stream of literature is quite recent and typically places the emphasis on production scheduling issues (see Chen (2010)).

Actually, a most common situation in practice is that transportation decisions are largely disconnected from production decisions at the operational level: when production items or batches have been completely processed by the manufacturing plant, they become available for shipping, and they are subsequently dispatched by the transportation department. This may be explained by several factors: low complexity of this uncoordinated approach, "rule-of-thumb" according to which immediate shipment tends to decrease the cost of inventory, and, in many cases, lack of communication between various departments.

From a global supply chain perspective, aiming at better coordination of the product flows and at increased customer satisfaction, this is not an ideal process. It would be by far preferable, indeed, to set up an integrated production-transportation plan taking into account, among other constraints, the capacity of the plants and the customer due-dates. But even when such a plan exists, many events can concur to create significant differences between the provisions of the plan and the actual situation faced by transportation managers on a day-to-day basis. Production delays caused by late deliveries of materials, equipment failures, non-conformities, urgent orders, and other obstacles to lean plant management, are still frequent in many industrial environments. Similarly, on the demand side, customer orders may not coincide, either in time or in quantity, with the forecasts that have been used to establish the plans.

As a consequence, operational shipping decisions often rely solely on available (deterministic) data about items in physical inventory, and transportation is managed in a purely reactive mode with respect to production. The main objective of this paper is to examine whether and how transportation decisions can be improved when they account for *forecasts* about future releases of items from production. As such, our paper takes an intermediate position between a purely *sequential approach*, whereby production and transportation decisions are independently optimized, and a *fully coordinated approach*, whereby both types of processes are simultaneously handled.

From the methodological point of view, taking forecasts into accounts leads us to consider *stochastic optimization* models of transportation planning. We will show that relatively

simple, but efficient and robust methods can be used to improve the quality of transportation plans, without necessarily resorting to advanced methods from stochastic programming, which may be much more demanding in terms of implementation effort and computational requirements.

1.2 Industrial setting

Our "production-transportation" problem formulation is inspired by a real-world situation encountered in the steel industry. A large number of different steel coils manufactured on a make-to-order basis must be dispatched everyday to customers. At the operational level, the shipping department is informed of a production plan, including confirmed and projected release dates announced by the plants, and of a delivery schedule based on the delivery time windows requested by the customers.

The shipping department charters trucks according to confirmed coil releases from production, and according to customers' requirements. Each truck typically holds one to four coils, depending on their total weight. The economic objective is to minimize the total cost consisting of transportation costs, penalties for early or late deliveries, and inventory costs incurred when available coils are stored in the warehouse before expedition. Shipping decisions are made and are implemented on a daily basis knowing the coils available to be shipped on the first day and the expected release dates of upcoming coils, meaning that only feasible plans are generated and executed at each period. Then, data such as coil releases from production and coil departures from inventory are updated, and this sequential process is repeated day per day over a rolling horizon (see Sethi & Sorger (1991) for the corresponding theoretical framework).

The daily trade-off faced by dispatchers can be expressed as follows: "Is it better to ship a given item today, or to wait in the hope of being able to ship it together with another item which is expected to be released from production in the near future, as this would reduce transportation costs at the risk of incurring delay penalties?" The optimization issue arises from the large number of potential combinations of coils into truckloads, as well as from the selection of the shipment periods. In view of the uncertainty that surrounds the releases of items from production, we consider both expected cost and robustness to be important criteria when evaluating the quality of shipment decisions.

Although our models have not been fully implemented in an industrial environment, they are meant to reflect the major characteristics (objective function, constraints) of the situation that motivated this work. Similarly, the values of the parameters used in all instances are derived from real-world data, as explained in Section 6.

1.3 Outline

In Section 2, we present a stochastic programming formulation of the multi-period vehicle loading problem with stochastic release dates, and we briefly review the relevant literature in Section 3. Then, Section 4 examines a deterministic version of the problem which underlines its combinatorial aspects. This analysis is put to use in Section 5, where several heuristic algorithms are proposed for the solution of the stochastic problem. These algorithms have been tested on numerical instances whose structure is described in Section 6. Since we are dealing with stochastic optimization problems, comparing the performance of different algorithms on a same instance is not a trivial task: we explain in Section 7 how we have

carried out the analysis. The computational results are discussed in Section 8. Interestingly, this discussion leads to the development of an additional, rather simple algorithm, which is extensively tested in Section 9. Section 10 contains the conclusions of our research.

2 Formulation

We consider the following *multi-period vehicle loading problem with stochastic release dates*, or *MVLSRD* problem for short. A set of N items must be delivered by trucks to several customers over a discrete (rolling) horizon $\{1,2,\ldots\}$. Each item $i = 1,\ldots,N$ has several deterministic attributes:
- its weight w_i;
- a delivery time window $[a_i,b_i] \subseteq \{1,2,\ldots\}$;
- the warehouse location where the item must be picked up;
- the customer location where the item must be delivered.

A subset of items are ready to be shipped at the beginning of the first (current) period $t = 1$. Moreover, the firm has forecasts about the release dates of items from production for a number of subsequent periods $t = 2,\ldots,L+1$, where L denotes the length of the *look-ahead horizon* (see Sethi & Sorger (1991)). We represent this information by probabilistic distributions of release dates: for all $i = 1,\ldots,N$ and $t = 1,\ldots,L+1$, $p_{it} \in [0,1]$ denotes the probability that item i will be released in period t and hence, can be shipped in periods $t,t+1,\ldots$ We assume that $\sum_{t=1}^{L+1} p_{it} \leq 1$ and $p_{i1} \in \{0,1\}$ for all i (information relative to the first period is fully revealed). There is an unlimited number of trucks. The maximum total weight that can be loaded on any truck is equal to *Cap*. These attributes, together with a number of auxiliary parameters, allow us to compute the cost generated by a truck picking up a given subset of items at their respective warehouses and transporting them to their respective destinations. In our application, all warehouses are located around the same plant, and we are primarily concerned with long-haul transportation. Therefore, the routing aspects are not of primary interest: each truck only visits a couple of warehouses and customers, so that the optimal route can be easily computed for any truck. The total cost generated by a truckload only depends on:
- the composition of the load;
- the total distance driven by the truck;
- the transportation cost per ton and per kilometer;
- an inventory cost, or opportunity cost, depending on the number of periods that each item spends in the warehouse after it has been released from production;
- penalties linked to the period of delivery of items to customers.

This broad definition allows us to integrate various specific features of the cost function (see Section 6.1).

The decisions to be made represent the truckloads to be composed and shipped in period $t = 1$. As a general rule, grouping items on a same truck is beneficial, and a good shipping decision is based on the following insights: it may be appropriate to ship an item early (with respect to its delivery time window), or conversely, to wait before shipping it (even though it has been released or its time window will be missed) if this results in a reduction of the expected number of trucks required and, more generally, in smaller expected total logistical costs.

Since the horizon is rolling, we actually want to solve an (infinite) sequence of optimization problems, one for each look-ahead horizon $\{\ell,\ldots,\ell+L\}$, where $\ell = 1,2,\ldots$ This results in successive decisions regarding the shipments to be made in each period. The objective of

the MVLSRD problem is to minimize the expected cost per unit amount shipped (say, by ton) over the long term:

$$\min_{\pi} Z(\pi) = \lim_{T\to\infty} E\left[\frac{\sum_{t=0}^{T} C(S_t, A^{\pi}(S_t))}{\sum_{t=0}^{T} W(S_t, A^{\pi}(S_t))}\right], \tag{1}$$

where π denotes the decision policy, S_t denotes the state of the system at time t (defined by the collection of items available for shipment at time t, together with their release date), $A^{\pi}(S_t)$ denotes the action taken in state S_t according to policy π, $C(S_t, A^{\pi}(S_t))$ is the associated cost incurred in period t, and $W(S_t, A^{\pi}(S_t))$ is the total weight shipped in period t. This problem can be viewed as a Markov decision process with very large state space and action space, Powell (2011), although we will not explicitly refer to the MDP framework in this paper.

3 Literature review

To the best of our knowledge, the multi-period vehicle loading problem with stochastic release dates has not been previously investigated.

If the release dates are deterministic and the length of the horizon is fixed, then the problem essentially boils down to a variant of the bin packing problem (see Dyckhoff (1990), Wascher et al. (2007), and Section 4 hereunder). As such, this version of the problem does not pose any specific difficulty and has not attracted the attention of researchers. Online variants of bin packing are closer to our multi-period framework (see Coffman et al. (1983) for a survey), but they do not assume any probabilistic information about incoming items, and the bin packing literature mostly focuses on the worst-case analysis of simple heuristics.

Transportation planning under uncertainty has been considered in several papers, from different points of view (see Crainic (2003)). The stress is usually placed on uncertainty on the demand side and on routing aspects. For instance, many authors examined *vehicle routing problems with dynamic demands* arising in a single period, as evidenced by the surveys Cordeau (2007), Gendreau & Potvin (1998), Pillac et al. (2011) and Psaraftis (1995). More recently, Angelelli et al. (2007), Angelelli et al. (2009) and Wen et al. (2010) describe dynamic *multi-period* routing problems where a set of requests need to be served by a fleet of uncapacitated vehicles over a finite discrete horizon. Some of the requests are known initially but more may arrive over time, and each request has a deadline, so that the firm may decide to postpone it or not to a later period. The authors discuss the benefits drawn from the multi-period framework with short look-ahead periods. Here again, as in the online bin packing problem, complete uncertainty is assumed about the requests to be served: this hypothesis makes sense in the case, for instance, of requests placed by private customers of a courier company. But it appears to be inappropriate in the case that we consider, where the production plan provides a fair amount of information about the features of the items that will have to be transported, and about their expected release dates. Also, the focus in Angelelli et al. (2007), Angelelli et al. (2009) and Wen et al. (2010) is on routing, rather than optimal vehicle loading (the items are assumed to be small parcels, so that the capacity of the vehicle is not a binding constraint).

Many papers investigate stochastic optimization models for *fleet management*, see Crainic (2003), Frantzeskakis & Powell (1990), Powell (2011), and Powell & Topaloglu (2003). Here, the uncertainty is mostly due again to customer demands that arise randomly over time, and the models emphasize issues related to the repositioning of empty vehicles and to

the acceptance or rejection of incoming orders. Multi-period problems of this nature can be handled as sequences of two-stage problems with recourse, where the second stage accounts for all future periods.

None of the papers cited above, however, simultaneously considers the three defining features of our problem, namely, multiple periods, stochastic release dates, and (small) capacity of the vehicles.

4 Deterministic optimization

In order to better understand the structure of the problem and to prepare the ground for subsequent developments, we first consider the special case of the vehicle loading problem where all release dates are deterministic: $p_{it} \in \{0,1\}$ for all $i = 1,\ldots,N$ and $t = 1,\ldots,L+1$.

In this case, the problem shares some similarity with a bin packing problem, since we have to "pack" all items into bounded-capacity vehicles so as to minimize a cost function which heavily depends, in practice, on the number of required vehicles (Coffman et al. (1983) and Wascher et al. (2007)). We can formulate this loading problem as a large set covering problem, where each column, or *pattern*, corresponds to a feasible truckload. Each pattern is represented by binary parameters q_{ip} with the interpretation that $q_{ip} = 1$ if pattern p contains the item i, and $q_{ip} = 0$ otherwise. Here, feasibility means that the load does not exceed the capacity of the truck and, possibly, that other relevant constraints are satisfied as well.

If we know the starting time t of the truck, then the corresponding cost c_p^t of pattern p can be simply computed as the sum of transportation costs, inventory costs and auxiliary cost elements. Moreover, for each pattern p, there exists a "best possible starting time" which minimizes the cost c_p^t, and we can accordingly define $c_p = \min_{1\le t\le L+1} c_p^t$. We call c_p the *cost of pattern* p.

If Ω denotes the set of feasible patterns and $\theta_p \in \{0,1\}$ represents the decision to use a pattern or not, then we obtain the following model for the *multi-period vehicle loading problem with deterministic release dates (MVLDRD)*:

$$\min Z^* = \min \sum_{p\in\Omega} c_p \theta_p \tag{2}$$

subject to

$$\sum_{p\in\Omega} q_{ip}\theta_p \ge 1 \quad \forall i = 1,\ldots,N, \tag{3}$$

$$\theta_p \in \{0,1\} \quad \forall p \in \Omega. \tag{4}$$

This set covering representation is usually more efficient, from a computational point of view, than an ILP model based on assignment variables (where $y_{ip} = 1$ if item i is included in load p) and explicitly expressing the feasibility constraints (see, e.g., Vanderbeck (1999)). It allows multiple parameters (item weights, truck capacity, penalties, complex transportation costs, etc.) and side-constraints (heterogeneous fleet, load-vehicle compatibility, route length, etc.) to be "hidden away" in the definition of the patterns, thus leading to a generic model that is suitable for customized applications.

Generating the set of patterns Ω gives rise to an auxiliary task that can be either performed in a preprocessing phase, or embedded in the solution phase by use of a dedicated column generation technique (Vanderbeck (1999) and Vanderbeck & Wolsey (1995)). The former approach becomes rapidly prohibitive if the size of Ω is large. In our case study,

however, this size remains manageable: short look-ahead horizons, small numbers of available items per period, and small number of items per truckload concur to limit the set of feasible patterns. Moreover, exhaustive generation of the set of patterns allows us to compute rather complex, nonlinear cost functions c_p^t, such as those encountered in practice (see Section 6.1), and to derive the pattern cost c_p.

Thus, whenever we rely on model (2)–(4) in the remainder of the paper, we always assume that the complete set of columns is generated in a preliminary phase. This allows us, in particular, to solve the resulting model to optimality by branch-and-bound. (In our computational experiments, we simply feed the model to a generic IP solver.)

Procedure: Rolling deterministic
For each period $\ell = 1, 2, \ldots$, successively, do

1. Let $t(i)$ be the release date of item $i = 1, \ldots, N$.
2. Consider all items such that $t(i) \in \{\ell, \ldots, \ell + L\}$, and solve the associated set covering problem (2)–(4). Let Θ be the collection of patterns selected in the optimal solution of (2)–(4) and for which the best possible starting time is the current period ℓ: $c_p = c_p^\ell$ for $p \in \Theta$.
3. Remove all items contained in the patterns of Θ (that is, constitute and ship the corresponding truckloads), let $\ell := \ell + 1$ (increase the time counter to the next period), update all data, and repeat steps 1–2.

Fig. 1 Procedure *Rolling deterministic*

Putting together the pieces of the previous discussion, we conclude that the deterministic MVLDRD problem over a rolling horizon can be handled by the procedure in Figure 1. This sequential process can be used to generate a policy over an indefinitely long rolling horizon. As L increases, we expect it to provide an increasingly better approximation of the long term optimal policy, but its computational complexity grows accordingly.

5 Stochastic optimization

Let us now return to the stochastic version of the MVLSRD problem. We can view a decision policy for this problem as a mapping π which, for every instance described as in Section 2, selects the patterns to be loaded and shipped in period 1. The quality of a policy is evaluated by the objective function (1). We denote the optimal policy by π^* and its value by $Z(\pi^*)$. It is the best expected value that can be achieved in the probabilistic environment. From an algorithmic point of view, the policy π^* cannot be easily computed, due in particular to the very large size of the state space, and to the inherent complexity of the deterministic version of the problem (it is easy to infer from Section 4 that the deterministic loading problem is NP-hard, just like set covering and bin packing problems).

In this section, therefore, we propose several heuristic methods that produce "good" policies π_i; these heuristics are based on generic schemes discussed for instance in Van Hentenryck & Bent (2006) and Powell (2011). Several of them rely on the generation of *scenarios*: a scenario is a joint realization of the random release dates of items $i = 1, \ldots, N$.

5.1 Deterministic equivalent: Scenario tree

For every instance of MVLSRD, the set S of potential scenarios over the look-ahead horizon $\{1,\ldots,L+1\}$ is finite, since any item i can only be released in one of the periods $1,\ldots,L+1$, or not at all. This set S can be represented by a scenario tree, which can be used in turn to set up an *equivalent deterministic formulation* of the problem as an integer programming problem based on the binary decision variables $\theta_{p,t,s}$, where $\theta_{p,t,s}=1$ if pattern p is shipped at time t in scenario s. As usual, non-anticipativity constraints ($\theta_{p,t,s_1}=\theta_{p,t,s_2}$) are imposed in this model to enforce the consistency of decisions regarding each pattern p for all pairs of scenarios $s_1,s_2\in S$ which coincide over an initial subhorizon $\{1,\ldots,t\}$ (see, e.g., Birge & Louveaux (1997)). Taking into account the whole set of scenarios S leads to a very large-scale IP model, which turns out to be intractable in practice. An approximation of this model can be obtained by restricting the tree to a subset of scenarios. This approach is called "reduced tree approximation" (see Heitsch & Romisch (2007a), Heitsch & Romisch (2007b) and Shapiro (2003)). We have conducted preliminary experiments with reduced trees built on random samples of scenarios (Monte Carlo and stratified sampling consistent with the probability distributions p_{it}, $t=1,\ldots,L+1$, have been used). Even with as few as 20 items and 5 look-ahead periods, the resulting models proved very hard to solve to optimality by a commercial software package (IBM ILOG CPLEX 11). Therefore, we have abandoned this approach and we have considered alternative, more efficient heuristics.

5.2 Local Optimization Algorithm *(LO)*

This simple myopic heuristic reduces the multi-period stochastic problem to a sequence of mono-period deterministic ones. At the current period $t=1$, *LO* solves the deterministic set covering model (2)–(4) associated with the set of items that are initially available for shipment, thus disregarding any information about future releases of items in periods $2,\ldots,L+1$. The same process is repeated in the second period, when the information pertaining to this period is revealed, and so forth, to produce a policy with expected value $Z(LO)$.

From the industrial perspective, heuristic *LO* mimics the procedure implemented by many companies, whereby transportation decisions are made with the objective to ship all available items as soon as possible, without consideration for combinations of items that may potentially bring additional benefits in the future; see Section 1.1. We do not expect *LO* to be very effective, but it provides a benchmark against which other approaches can be evaluated.

5.3 Mean Release Date Algorithm *(Mean)*

An intuitive and popular approach to stochastic optimization problems consists in replacing all uncertain parameters by their expected values. In our case, this can be translated into selecting a unique scenario for which the release date of item i is equal to the mean value $t(i)=\sum_t p_{it}t$ (rounded to the nearest period). The procedure *Rolling deterministic* defined in Figure 1 can be used with these release dates to define an algorithm *Mean* that generates a policy with expected value *Z(Mean)*. (Note that when the data is updated in Step 3 of the procedure, the expected value of the release dates may change, since the probability distributions p_{it} must be conditioned by the occurrence or non-occurrence of certain events.)

When compared with the myopic *LO* heuristic, *Mean* takes into account the look-ahead horizon L, at least to some extent. Thus, the difference between $Z(LO)$ and $Z(Mean)$ provides a measure for the value of using a multi-period model with a rough estimate of the future. Our computational results will demonstrate that this simplistic improvement provides significant benefits.

5.4 Modal Release Date Algorithm *(Mod)*

As an alternative to the previous single scenario heuristic, the scenario based on the modal value of the release dates can be used, together with the procedure *Rolling deterministic*, to define an algorithm *Mod* that generates a decision policy with value $Z(Mod)$; here, $t(i)$ is the value of t which maximizes p_{it}, that is, $t(i) = \text{argmax}_t\, p_{it}$ for all i (ties are arbitrarily broken).

5.5 Consensus Algorithm *(Cons)*

A more complex heuristic to approximate the optimal value $Z(\pi^*)$ is based on *consensus approaches* described in Van Hentenryck & Bent (2006). This family of algorithms provides a generic framework for multi-period stochastic optimization problems. It relies on a simple intuitive idea, namely: if a same decision is frequently made in the optimal solutions associated with a large number of scenarios, then this decision is presumably "good" and can be adopted more generally. For our specific vehicle loading problem, we implement this principle as follows: First, a sample $\mathscr{S} = \{1, \ldots, K\}$ of scenarios is randomly generated. The scenarios in $\mathscr{S}$ are called *calibrating scenarios*. Next, the set covering model (2)–(4) corresponding to each calibrating scenario is independently solved: this yields K sets of patterns $\Theta^1, \ldots, \Theta^K$, where Θ^j is the optimal set of patterns to be shipped in period 1 for scenario $j \in \mathscr{S}$.

Notice that, as compared with the reduced tree approach sketched in Section 5.1, we do not impose here any non-anticipativity constraints, so that the decisions $\Theta^1, \ldots, \Theta^K$ are usually different, even though the set of items released in period 1 is identical in all scenarios. We define D to be the set of those items i that appear in at least $\lceil K/2 \rceil$ of the solutions $\Theta^1, \ldots, \Theta^K$ (that is, D is the set of items that are shipped in period 1 in a majority of the calibrating scenarios). In the consensus algorithm *Cons*, all items contained in D will be shipped in period 1.

The reader should observe that, at this point, it is usually not clear how the items in D should be loaded (i.e., how they should be grouped into truckloads); in fact, the set D may not have been selected for transportation, as a whole, in any of the independent solutions $\Theta^1, \ldots, \Theta^K$. Moreover, optimizing the transportation of D may lead to less-than-full truckloads; therefore, it seems reasonable to complete the set D with additional items that can be loaded without increasing the total cost.

To account for these observations, in a second phase, we solve the following mono-period optimization model, where N_1 is the set of items available in period 1, and Ω stands here for the set of patterns that only involve items from N_1:

$$\min M \sum_{p \in \Omega} c_p \theta_p - \sum_{i \in N_1 \setminus D} \sum_{p \in \Omega} a_{ip} \theta_p \tag{5}$$

subject to

$$\sum_{p \in \Omega} a_{ip} \theta_p \geq 1 \quad \forall i \in D \tag{6}$$

$$\theta_p \in \{0,1\} \quad \forall p \in \Omega. \tag{7}$$

Constraint (6) expresses that all the items in D must be included in the truckloads. For M large enough, the first term of the objective function (5) ensures that the total cost of the solution does not exceed the optimal cost incurred for D, whereas the second term tends to maximize the number of items included in the truckloads.

The solution of model (5)–(6) defines the consensus decision that comes into effect in period 1. Our Consensus algorithm (*Cons*) is obtained by repeating the same process for each successive period. Note that it requires the solution of $K+1$ integer programming problems per period, as compared to one IP for each of the three previous algorithms.

5.6 Restricted Expectation Algorithm *(RE)*

A drawback of the Consensus algorithm *Cons* is that it relies on the frequency of individual shipping decisions for each item whereas, in practice, the quality of a transportation plan depends to a large extent on the composition of the loads, that is, on the efficient combinations of items within trucks. We now propose a so called *Restricted Expectation algorithm* (*RE*) which avoids the deconstruction of near-optimal truckloads, which is inspired by the general framework described in Van Hentenryck & Bent (2006).

Just like *Cons*, our algorithm *RE* starts by generating a random sample $\mathscr{S} = \{1,\ldots,K\}$ of calibrating scenarios and by computing the corresponding optimal sets of patterns for period 1, say $\Theta^1,\ldots,\Theta^K$. We now look at $\Theta^1,\ldots,\Theta^K$ as candidate decisions, any of which could be implemented in period 1. By definition, Θ^j is optimal for scenario j, but not necessarily for alternative scenarios. In *RE*, the quality of decision Θ^j is evaluated for each scenario $k \in \mathscr{S} \setminus \{j\}$, as follows: all items included in patterns of Θ^j are discarded and model (2)–(4) is solved for the remaining set of items with the release dates of scenario k. This allows us to estimate, the expected cost $E[\Theta^j]$ generated by decision Θ^j over all scenarios in the sample $\mathscr{S}$ (hence the name "restricted expectation"; this is also akin to estimating the expected value of decision Θ^j in the framework of approximate dynamic programming, see Powell (2011), or sample average approximation methods, see Verweij et al. (2002)). Then, *RE* selects and implements in period 1 the decision Θ^j with the smallest value of $E[\Theta^j]$. The same process can be subsequently repeated for periods 2, 3,..., as in the *Rolling deterministic* procedure.

Note that *RE* requires the solution of K^2 integer programming problems for each decision period, which may turn out to be prohibitive for large scale instances and/or for large values of K. In our experiments, we have used $K = 10$ calibrating scenarios. Larger values of K did not appear to yield significant improvements for most instances.

6 Numerical instances

In this section, we describe the numerical instances that we have used in our experiments. Although the instances are randomly generated, their main features are intended to mimic the industrial environment that motivated our study. In the following description, we distinguish between deterministic and stochastic parameters of the instances.

6.1 Deterministic parameters

We consider four classes of instances involving, respectively, $N = 80, 120, 160$, and 200 items. The capacity of each truck is $Cap = 23.5$ tons. The weight w_i of each item i is randomly and uniformly generated, strictly between $0.2 \times Cap$ and $0.8 \times Cap$. This distribution puts an upper-limit of 4 items per truckload, similarly to what happens in real life. (The weight of a coil typically ranges between 4.5 and 19 tons.) The delivery cost per ton varies between 30 and 34 monetary units per ton, depending on the customer.

The data are generated over a "long term" horizon of $T = 20$ periods, and the look-ahead horizon includes $L = 4$ periods beyond the current one, so as to reflect the weekly operational setting. This results in 16 effective planning periods $\{1, 2, \ldots, 16\}$, whereas periods $\{17, \ldots, 20\}$ account for the definition of the delivery time windows and for end-of-horizon effects (see Section 7.5).

For each item i, the delivery time window covers four consecutive periods $a_i, a_i + 1, a_i + 2$, and $a_i + 3$. Delivery before a_i or after $a_i + 3$ is infeasible, meaning that it carries an infinite penalty.

Note that, in view of these assumptions, there is no point in considering release dates that would result in deliveries outside of the time window $[a_i, a_i + 3]$ for item i. Consequently, the release dates of potential interest for item i must be four consecutive periods $r_i, r_i + 1, r_i + 2$, and $r_i + 3$, where r_i is the first possible release period of item i. Based on transportation times, these four periods can easily be calibrated to match the delivery interval $[a_i, a_i + 3]$, or, equivalently, we can set the transportation times to zero and identify the release intervals with the delivery intervals. Accordingly, in the remainder of the paper, we do not distinguish any longer between a_i and r_i. In order to create our instances, we generate a_i uniformly between 1 and T.

The ideal target dates for delivery are $a_i + 1$ and $a_i + 2$. Early delivery in period a_i carries a penalty PE, late delivery in period $a_i + 3$ carries a penalty PL. For each period when an item is available for shipment but the firm decides to postpone its delivery, it incurs an inventory cost PI. We set $PI = 15$, $PE = 40$ and $PL = 70$ monetary units. These values are such that, if the item is available in period a_i or $a_i + 1$, then the cheapest period for delivery is $a_i + 1$, and the next cheapest period is $a_i + 2$. The most expensive delivery period is $a_i + 3$. (These choices reflect a "Just-in-Time" supply chain environment.)

In the industrial setting and, accordingly, in our numerical instances, the computation of the transportation cost involves a number of additional factors which further complicate it, but which do not have a direct impact on the performance of our algorithms. For instance, the transportation cost charged by the transporter is fixed up to a certain minimum load, and increases linearly as a function of the load beyond this minimum quantity. Moreover, when a truck picks up items from several warehouses and/or delivers them to several customers, an additional handling fee must be paid to the transporter, and so on.

6.2 Stochastic release dates

Let us now turn to the stochastic models of uncertain release dates. In view of our previous assumptions, probability distributions of interest for the release date of item i can be expressed in the form $[p_{i,t};\ p_{i,t+1};\ p_{i,t+2};\ p_{i,t+3}]$, where $t, t+1, t+2, t+3$ are the possible release dates of item i, and where p_{ik} is the probability that item i is released in period k.

In our experiments, we use four distributions for each item i, namely:

1. "Early": [40%; 30%; 20%; 10%] translating optimistic forecasts for the release dates.

2. "Late": [10%; 20%; 30%; 40%] translating pessimistic forecasts.
3. "Uniform": [25%; 25%; 25%; 25%] translating maximal uncertainty.
4. "Binomial": [12.5%; 37.5%; 37.5%; 12.5%] translating Just-in-Time production targets.

These distributions represent various stochastic profiles in the output of the industrial production system. They are assumed here to be independent and identical for all items. Note that this assumption would fail if the variability of release dates was caused by special causes that simultaneously affect all produced items, such as machine breakdowns or other structural disturbances of the production process. Although the algorithms could still be applied, it would probably be more appropriate to adapt them specifically to deal with such situations.

7 Algorithmic performance evaluation

When evaluating the performance of optimization algorithms, classical criteria are the quality of the solutions that they deliver (optimality, performance ratio, etc.) and their computing time (see Barr et al. (1995), Brownlee (2007), Hooker (1995), Johnson (2001), Rardin & Uzsoy (2001), Ruml (2010)).

The evaluation of algorithms for multi-period stochastic optimization problems, however, presents numerous specific hurdles. We explain in this section how the performance of the algorithms has been assessed in our study.

7.1 Evaluation scenarios

As explained in the previous section, a numerical instance I of the problem is completely defined by a selection of values for its deterministic parameters (weight w_i of each item, time windows $[a_i, a_i + 3]$, cost parameters, etc.) and by a choice of the probability distribution of release dates (either Early, or Late, or Uniform, or Binomial). In order to assess the performance of an arbitrary algorithm $\mathscr{A}$ on instance I, we should be able to evaluate the expected value $Z(I;\mathscr{A})$ of the cost function over all possible realizations of the random release dates, i.e., all possible scenarios.

In practice, we estimate this expected value over a random sample consisting of *evaluation scenarios* over the long term horizon $\{1,\ldots,T\}$. (These evaluation scenarios are generated independently from the calibrating scenarios used in the Consensus and Restricted Expectation algorithms of Section 5.) In our experiments, we have used $V = 30$ evaluation scenarios for each instance.

Note that the computational burden of this evaluation phase is quite heavy. In particular, with our previous notations, estimating the expected value of the solution generated by the Restricted Expectation algorithm *RE* for a single instance requires the solution of $(T-L) * K^2 * V$ medium scale IP problems. With $T = 20$, $L = 4$, $K = 10$ and $V = 30$, this amounts to 48 thousand IP problems for the estimation of $Z(I;RE)$.

7.2 Statistical analysis of results

Since the cost function $Z(I;\mathscr{A})$ is estimated on the basis of a random sample of evaluation scenarios for each algorithm, small differences between the values obtained for two distinct

algorithms $\mathscr{A}_1$ and $\mathscr{A}_2$ might be due to random effects only, and their statistical significance must be assessed.

For each fixed instance, we generate the same sample of evaluation scenarios for all algorithms, so as to reduce the variance of the tests. This allows us to apply a paired-sample t-test to compare the values $\mu_1 = Z(I;\mathscr{A}_1)$ and $\mu_2 = Z(I;\mathscr{A}_2)$. We use both two-sided and one-sided tests:

1. test the hypothesis $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu_1 \neq \mu_2$;
2. test the hypothesis $H_0 : \mu_1 = \mu_2$ vs. $H_1 : \mu_1 < \mu_2$ (or $H_1 : \mu_1 > \mu_2$).

We say for short that algorithm $\mathscr{A}_1$ *outclasses* algorithm $\mathscr{A}_2$ on a given instance if we can conclude that $\mu_1 < \mu_2$ with a confidence level fixed at 95%.

7.3 Fully or partially revealed information

In general, we are not able to compute exactly the optimal value $Z(\pi^*)$ of an instance I. But an optimistic estimate of $Z(\pi^*)$ can in principle be computed as follows: for each evaluation scenario s, solve the deterministic optimization problem (2)–(4) over the long term horizon $\{1,\ldots,T\}$ to obtain the optimal value $Z^*(s)$, and average this value over all evaluation scenarios. Since this approach amounts to solving the problem with fully revealed information, it yields a benchmark value O^* (as Oracle) which, for practical purposes, can be viewed as a lower bound on the optimal value (except for the fact that we work with a sample of evaluation scenarios).

The value O^*, however, is overly optimistic since in an industrial setting, it is unrealistic to assume that reliable information about the release dates is available over a long horizon. A more relevant benchmark is obtained by assuming that in period ℓ, the firm has "partially revealed" deterministic information regarding the actual release dates during the look-ahead horizon $\{\ell,\ldots,\ell+L\}$ (see, e.g., Sethi & Sorger (1991)). Then, the *Rolling deterministic* procedure of Figure 1 can be used to compute a policy with value $Z^*(s;L)$ for each evaluation scenario s. Averaging these values over all evaluation scenarios yields a value $O^*(L)$, which gives an indication of the best achievable performance in a production environment "under control" over the look-ahead horizon.

7.4 Value of information

In order to reduce its total logistical costs, the firm may want to obtain more precise information about the release dates of items (e.g., by reducing the variability of its production processes, by implementing data collection systems, or by improving its production planning systems), or it may want to improve the performance of its vehicle loading algorithms. Both choices are likely to require additional investments that must be counterbalanced by the benefits that they bring. The following measures bring insights into this trade-off (see, e.g., Birge & Louveaux (1997) and Van Hentenryck & Bent (2006)).

The difference $Z(\pi^*) - O^*$ represents the *Value of Perfect Information*, since $Z(\pi^*)$ is the minimum cost that must be incurred if the firm makes the best possible use of the probabilistic information, whereas O^* is the cost that would be incurred with perfect information. In our computational reports, we cannot compute $Z(\pi^*)$, and we use as a surrogate the best cost value obtained by any of our algorithms: that is, we define $VPI = \min_{\mathscr{A}} Z(\mathscr{A}) - O^*$.

As mentioned in Section 7.3, the bound O^* is overly optimistic, and it is more realistic to assume that the firm could collect perfect information for the next L periods only. Therefore, we also compute the *Value of the Accessible Information*: $VAI = \min_{\mathscr{A}} Z(\mathscr{A}) - O^*(L)$.

The difference between $Z(\pi^*)$ and $Z(Mean)$ represents the value of using the probability distributions in the most effective way, instead of relying on their mean values; we call this difference *Value of the Stochastic Solution* and we approximate it as $VSS = Z(Mean) - \min_{\mathscr{A}} Z(\mathscr{A})$.

Finally, we also propose to compute the *Value of the Multi-Period Model* as $VMPM = Z(LO) - O^*(L)$: this measures the performance deficit incurred by the myopic mono-period policy *LO* when compared with the benchmark where perfect information is available over the look-ahead periods.

7.5 Computational limitations and bias

We briefly comment here on some of the design choices that we made in our experiments, and which are directly related to the dynamic, multi-period setting of our problem.

7.5.1 Start and end of horizon

In a rolling horizon context, the first and last periods of the horizon $\{1,\ldots,T\}$ are somewhat special. In a real-world application, the first, or current period is generally affected by past decisions, such as undelivered items carried over from previous periods. Similarly, since the tests cannot be performed over an infinite horizon, each instance suffers from boundary effects at the end of the horizon, as no further item releases are forecast beyond period T. In order to alleviate these effects, we work with a "sufficiently long" horizon T, which reduces the influence of the initial conditions, and we make shipping decisions for periods 1 through $T-L$ only.

7.5.2 Objective function

The total logistical costs represent the cost of all performed decisions, meaning that only delivered items are taken into account. Because of the look-ahead periods introduced at the end of horizon, and because it is in the nature of our problem that some shipments may be postponed, all algorithms do not ship exactly the same set of items during the last few periods. This potential bias in the comparison of algorithms is mitigated, however, by the length of the horizon, and, mostly, by the fact that the objective function actually is the *average cost per ton shipped* (see Eq. (1)).

Let us also mention that in the industrial application, the total cost incurred for each item is made up of a significant fixed cost (equal to the weight of the item multiplied by a unit cost) which cannot be improved by any algorithm. Therefore, we remove this fixed component from all our estimations, which only include improvable elements such as temporal penalties, multiple pick-up and unloading fees, or opportunity costs for loading the truck under its purchased capacity (see Section 6.1).

7.6 Computing times

Our computational experiments have been performed on a personal laptop computer (Core 2 Duo 2GHz, 2GB of RAM, Windows). The running time of the algorithms is mostly de-

termined by the number of integer programming subproblems that must be solved. In our experiments, these IP subproblems have been solved using IBM ILOG CPLEX 11 with default settings.

For the instance sizes that we consider in this paper, all our algorithms meet the computing time requirements for operational daily use: namely, the decision relating to a single period is obtained in a few seconds. This is essentially due to the fact that the number of items that come into consideration over L look-ahead periods is relatively small, so that the number of feasible patterns and the associated set covering problems remain accordingly manageable. (*Evaluating* the quality of the algorithms is a different matter, as we explained in Section 7.1, since this requires the solution of a huge number of IPs.) Table 1 displays the computing time for solving one instance of problem (2)–(4) with $L = 4$ and N items. For algorithm *RE*, this computing time would be roughly multiplied by a factor K^2, as mentioned in Section 5.6.

Table 1 Problem size and computing speed for problem (2)–(4)

$L = 4$	$N = 50$	$N = 100$	$N = 200$
Average number of patterns	1403	6059	33399
CPU time (seconds)	0.3	1.3	13.0

Therefore, in the following sections, we do not discuss computing times and we concentrate, rather, on a comparison of the quality of the solutions provided by different algorithms.

8 Computational results

8.1 First results

Tables 2 to 5 report the expected value of the objective function (estimated over the evaluation scenarios) for the different algorithms and probability distributions. The best value obtained for each instance is in boldface.

For an easier understanding of the results, all values are expressed as percentages of the lower bound O^* (which corresponds, therefore to the value 100%). Note that this standardization of the objective function masks the decrease of the transportation cost per ton when the number of coils increases from 80 to 200. This scale effect is not surprising: indeed, when there is a large number of items with uniformly distributed weights, we can expect that it is possible to allocate the items to a number of trucks that are loaded to full capacity, or very close to it in each period (see for instance, Rhee (1988)). As a consequence, the transportation cost per ton decreases, and the performance of most algorithms tends to improve as the instances become somewhat easier.

Table 2 Algorithmic performance – Early distribution

Early	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	107.2	105.3	103.8	105.6
LO	193.5	172.5	168.7	159.3
Mean	116.5	113.9	111.1	110.4
Mod	112.2	**108.5**	**107.0**	**107.6**
Cons	122.4	119.5	113.1	117.4
RE	**111.0**	111.7	109.2	111.8
$O^*(4) - O^*$	7.2	5.3	3.8	5.6
VMPM	86.3	67.2	65.0	53.7
VPI	11.0	8.5	7.0	7.6
VAI	3.8	3.2	3.2	2.0
VSS	5.4	5.4	4.1	2.8

Table 3 Algorithmic performance – Late distribution

Late	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	102.7	103.0	102.8	103.8
LO	154.3	144.0	142.3	136.8
Mean	119.6	115.0	112.0	113.2
Mod	120.1	117.0	117.9	115.4
Cons	109.7	**110.1**	109.8	111.2
RE	**109.5**	111.0	**109.0**	**109.7**
$O^*(4) - O^*$	2.7	3.0	2.8	3.8
VMPM	51.6	41.1	39.5	33.0
VPI	9.5	10.1	9.0	9.7
VAI	6.8	7.1	6.2	5.9
VSS	10.0	4.9	3.0	3.5

Table 4 Algorithmic performance – Uniform distribution

Uniform	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	108.1	104.2	102.9	104.9
LO	179.5	159.2	154.6	147.5
Mean	117.7	112.3	**109.6**	109.9
Mod	125.1	118.6	113.9	113.3
Cons	115.7	114.7	111.6	113.3
RE	**112.1**	**112.2**	110.0	**108.7**
$O^*(4) - O^*$	8.1	4.2	2.9	4.9
VMPM	71.4	55.1	51.7	42.6
VPI	12.1	12.2	9.6	8.7
VAI	4.0	8.0	6.7	3.8
VSS	5.6	0.1	**0**	1.2

Table 5 Algorithmic performance – Binomial distribution

Binomial	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	107.2	105.8	104.4	106.1
LO	184.8	179.0	160.9	157.3
Mean	123.4	117.2	114.7	116.7
Mod	123.4	**109.9**	112.4	115.0
Cons	114.7	114.0	113.5	115.3
RE	**113.0**	111.6	**112.1**	**111.9**
$O^*(4) - O^*$	7.2	5.8	4.4	6.1
VMPM	77.6	73.2	56.5	51.2
VPI	13.0	9.9	12.1	11.9
VAI	5.8	4.1	7.7	5.8
VSS	10.5	7.3	2.6	4.8

Here are some observations issued from these empirical tests:

1. The *Value of the Multi-Period Model* $VMPM = Z(LO) - O^*(4)$ is quite large: it varies from 30% to more than 80%. This underlines the important benefit of taking several periods into account for transportation planning, rather than only the current period as in the myopic policy *LO*.
2. On the other hand, although $O^*(4)$ is significantly larger than O^* at a confidence level of 99%, $O^*(4) - O^*$ is not excessively large from a managerial point of view. (See, e.g., Rardin & Uzsoy (2001) for a discussion of statistical vs. practical significance.) This suggests that very good policies might be achieved with relatively short look-ahead horizons, such as those that are likely to be implemented in practice. In fact, in our preliminary experiments, we have also computed $O^*(L)$ for different values of L; the tests showed that $O^*(L)$ decreases rather quickly when L increases from 1 to 4, and decreases at a slower pace for larger values of L.
3. *Mean* and *Mod* perform rather poorly in most cases. An intriguing exception is the good performance of *Mod* for the Early distribution of release dates (we return to it later). $VSS = Z(Mean) - \min_{\mathscr{A}} Z(\mathscr{A})$ ranges from 0 up to 10%, meaning that practitionners may want to invest some efforts into the development of algorithms making effective use of the probability distributions.
4. $VAI = \min_{\mathscr{A}} Z(\mathscr{A}) - O^*(4)$ varies from 4% to 8%: for some instances, there is considerable value in reducing the uncertainty that surrounds the release dates.
5. The Restricted Expectation algorithm *RE* performs well: it provides the best expected values 10 times out of 16, the second best value 5 times, and the third best value once.

The performance of *RE* is substantiated by the results of hypothesis tests displayed in Table 6. Here, we have tested whether there is a significant difference between the mean values obtained by *RE* and by any of three other algorithms, namely, *Mean, Mod,* and *Cons*. (*LO* is clearly not in the same league.) The entries in the table indicate the number of instances for which the null hypothesis $H_0 : \mu_{RE} = \mu_{\mathscr{A}}$ is rejected against the alternative hypothesis H_1, for each algorithm $\mathscr{A}$, at a confidence level of 95% (see Section 7.2).

Table 6 shows that *RE* outclasses all other algorithms on most instances. It is statistically tied with *Mean* for a few instances, and with *Cons* for a majority of instances, but it is never outclassed by any of these two algorithms. An interesting exception is *Mod*, which significantly outclasses *RE* on three instances associated with the Early distribution; we refer to Section 9 for an analysis of this exception.

Table 6 Comparison of means for *RE* vs. alternative algorithms

	$\mathscr{A} = Mean$		$\mathscr{A} = Mod$		$\mathscr{A} = Cons$	
Reject H_0 vs. H_1	Yes	No	Yes	No	Yes	No
$H_1 : \mu_{RE} \neq \mu_{\mathscr{A}}$?	12	4	13	3	7	9
$H_1 : \mu_{RE} < \mu_{\mathscr{A}}$?	12	0	10	0	7	0
$H_1 : \mu_{\mathscr{A}} < \mu_{RE}$?	0	0	3	0	0	0

8.2 Robustness analysis

Robustness may take various meanings in operations research. In broad terms, robust optimization aims at finding the (near)-optimal solution of a mathematical progamming problem under a set of constraints that represent all possible realizations of uncertain parameters (see Bertsimas et al. (2011) and Kouvelis & Yu (1997)).

Thus, typically in robustness analysis, the quality (say, the cost) of a solution is measured under adverse realizations of the uncertain events. This focus on worst-case scenarios, however, gives rise to optimization problems that are quite hard to solve. More importantly, it may be seen as overly pessimistic in our planning framework where decisions are repeatedly made over a long term horizon, and where it is therefore unlikely that adverse scenarios will systematically materialize.

Therefore, in our experiments, we have tested the robustness of the Restricted Expectation algorithm *RE* in a different way: rather than assuming that the worst-case scenario unfolds for a given policy, we have tried to understand what happens when the decision-maker has poorly estimated the distribution of release dates. (This is akin to "distributional uncertainty" in the theoretical framework of Bertsimas et al. (2011).) Let us describe more precisely our experimental setting.

Let $\mathscr{D}$ be a fixed probabilistic distribution of release dates, $\mathscr{D} \in \{$Early, Late, Uniform, Binomial$\}$, and let $RE_{\mathscr{D}}$ be the variant of *RE* where all calibrating scenarios are drawn according to $\mathscr{D}$ (see Section 5.6). This reflects the situation where the decision-maker believes that $\mathscr{D}$ is the true distribution of release dates. Then, for a given instance I, the expected cost of the policy generated by $RE_{\mathscr{D}}$ is estimated by drawing V evaluation scenarios (as in Section 7.1) *according to a different distribution* $\mathscr{R}$, which is meant to represent the real, unknown distribution of release dates.

Our preliminary experiments were carried out with two "extreme choices", namely, RE_{Early} was evaluated over the late distribution $\mathscr{R} = Late$, and conversely, RE_{Late} was evaluated over the early distribution $\mathscr{R} = Early$. These experiments revealed that RE_{Late} performs rather poorly, whereas RE_{Early} seems to resist quite well to a wrong estimation of the distribution. The next section builds on these observations.

9 An optimistic algorithm

The previous results lead us to the hypothesis that an "optimistic" decision-maker, who always assumes that future items will be released as early as possible, may turn out to achieve a very good cost performance. This hypothesis is motivated, in part, by the results of Section 8.2, where RE_{Early} appeared to be a robust algorithm. But it is also supported by the observations made in Section 8.1 regarding the performance of algorithm Mod on the instances with an Early distribution (see Table 2 and Table 6). Indeed, when applied to the Early distribution $[p_{i,t}; p_{i,t+1}; p_{i,t+2}; p_{i,t+3}]$ where $p_{i,t} > p_{i,t+1} > p_{i,t+2} > p_{i,t+3}$, the modal

scenario used by *Mod* is exactly the scenario where each item is released at the earliest possible date, namely, in period t.

Thus, we define a simple algorithm *Optimist* as follows: For each instance, *Optimist* selects a unique scenario in which the release date of item i is equal to its earliest feasible release date: $t(i) = \min\{t : p_{it} > 0\}$. Then, the procedure *Rolling deterministic* of Section 4 is used with these release dates to define the policy *Optimist*.

9.1 Comparative performance of algorithm *Optimist*

Procedures RE_{Early} and *Optimist* have been tested on the 16 instances already used in Section 8. Note that each instance I defines feasible release dates $\{t, t+1, t+2, t+3\}$ for each item, as well as a (real) probability distribution $\mathscr{R}$ over these dates (see Section 6.2). For RE_{Early}, the calibrating scenarios are drawn from the Early distribution over the feasible release dates of each item. For both algorithms RE_{Early} and *Optimist*, the evaluation scenarios are generated from the real distribution $\mathscr{R}$ associated with the given instance. (So, when $\mathscr{R}$ is the Early distribution, RE_{Early} coincides with RE.)

The results are displayed in Tables 7 to 10. Quite remarkably, the simple algorithm *Optimist* yields the best solution for 10 instances out of 16, and the second best solution for three additional instances. The algorithm RE_{Early} provides three times the best value and seven times the second best one (beaten only by *Optimist*, but often close to it in such cases).

Table 7 Algorithmic performance – Early distribution

Real $\mathscr{R}$ = Early	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	107.2	105.3	103.8	105.6
RE_{Early}	**111.1**	111.7	109.2	111.8
Optimist	112.2	**108.5**	**107.0**	**107.6**

Table 8 Algorithmic performance – Late distribution

Real $\mathscr{R}$ = Late	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	102.7	103.0	102.8	103.8
RE	**109.5**	111.0	109.0	109.7
RE_{Early}	111.7	**108.6**	**106.9**	108.8
Optimist	110.1	109.2	107.3	**108.3**

Table 9 Algorithmic performance – Uniform distribution

Real $\mathscr{R}$ = Uniform	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	108.1	104.2	102.9	104.9
RE	**112.1**	112.2	110.0	108.7
RE_{Early}	113.7	111.4	107.8	109.5
Optimist	113.6	**109.7**	**107.3**	**108.4**

Table 10 Algorithmic performance – Binomial distribution

Real $\mathscr{R}$ = Binomial	$N = 80$	$N = 120$	$N = 160$	$N = 200$
O^*	100	100	100	100
$O^*(4)$	107.2	105.8	104.4	106.1
RE	**112.9**	111.6	112.1	111.9
RE_{Early}	117.3	113.5	112.1	111.7
Optimist	116.0	**109.0**	**109.5**	**110.3**

These observations are confirmed in Table 11, where we display the conclusions of several tests of hypotheses regarding the performance of *Optimist*. Here, the algorithms competing with *Optimist* are $\mathscr{A} = RE_{Early}$ (evaluated on all instances), and $\mathscr{A} = RE$ (evaluated on the instances associated with the Late, Uniform, and Binomial distributions). The tests show that *Optimist* outclasses the other algorithms on 6 instances, and is never outclassed.

Table 11 Comparison of means for *Optimist* vs. alternative algorithms

	$\mathscr{A} = RE_{Early}$		$\mathscr{A} = RE$ (L-U-B)	
Reject H_0 vs. H_1	Yes	No	Yes	No
$\mu_{Optimist} \neq \mu_{\mathscr{A}}$	6	10	6	6
$\mu_{Optimist} < \mu_{\mathscr{A}}$	6	0	6	0
$\mu_{\mathscr{A}} < \mu_{Optimist}$	0	0	0	0

9.2 Additional results

The performance of *Optimist* has been further assessed on a larger testbed of 80 additional instances. Namely, for each probability distribution of release dates, we have generated 10 random instances of size $N = 120$ and 10 instances of size $N = 160$ with different item characteristics, such as weight and first possible release period. The unit transportation costs are smaller for the instances with 160 items.

The expected value of the solution produced by *Optimist* is reported in Table 12 (averaged over 10 instances of the same class), as a percentage of the lower bound O^*. We see that *Optimist* performs very well in all cases. On average, it comes within 8% of O^* for the instances with 120 items, and within 5% for the instances with 160 items. The standard deviation is around 3% in all cases. We also observe that, not surprisingly, *Optimist* tends to perform slightly better when the release dates follow the Early probability distribution.

Table 12 Algorithmic performance of *Optimist* over 80 instances

Optimist/O^*	$N = 120$	$N = 160$
Early	106.7	103.8
Late	108.7	105.1
Uniform	108.6	105.8
Binomial	109.1	106.8

9.3 Interpretation

The observations in Section 9.1 and Section 9.2 are quite unexpected and interesting. Indeed,

1. *Optimist* is a much simpler and much faster algorithm than either *RE*, or RE_{Early}, or even *Cons*. At every period, it only relies on a single scenario and it solves a single set covering problem (2)–(4) in order to determine the items to be shipped, as opposed to K^2 IP's for *RE* and RE_{Early}, or $(K+1)$ IP's for *Cons*.
2. The complexity of *Optimist* is similar to that of either *Mean* or *Mod*, but the solutions that it produces are significantly better.
3. Most interestingly, perhaps, *Optimist* does not require any information regarding the probabilistic distribution of release dates! It only focuses on the first feasible release date of each item, which essentially depends on its delivery time windows. Therefore, it can easily be implemented in an industrial environment.

A possible explanation for the good performance of this simple algorithm goes as follows. In the current period, say period 1, *Optimist* faces a subset of items that are available for shipment. It establishes a scenario for the upcoming periods based on earliest possible release dates. In comparison with a random scenario, this optimistic scenario foresees relatively many releases over the look-ahead horizon (say, in periods 2 to 5). Hence, this increases the likelihood of postponing the shipment of an item i that is available in the current period, due to the perspective of shipping i together with another item in the near future, at a lower cost.

There are now two possibilities for the next periods. Either the observed releases are close to the optimistic forecast, in which case postponing item i was a good decision, or the releases deviate from the forecast. In the latter case, the transportation cost of i may increase, but there is still a reasonable chance that i will be efficiently combined with other items, so that the total penalty due to postponement may not be too high (in practice, the inventory costs are small relative to the cost of a truck). In summary, according to this tentative explanation, the algorithm *Optimist* would incite the manager to delay the shipments more frequently than, say, a myopic strategy would do, and this postponement strategy should yield benefits.

Although it is not easy to validate completely the various elements involved in this intuitive explanation, we can exhibit some evidence to support it. First, it appears that shipments are indeed more frequently postponed in the solutions generated by *Optimist* than in solutions generated by other algorithms. This is illustrated by two instances from Table 7 on which *Optimist* performs respectively worse ($N = 80$), or better ($N = 200$) than *RE* (the other instances yield similar conclusions). For each instance, we have collected the average expeditions dates d_{Opt} and d_{RE} of all items over the 30 evaluation scenarios, for the solutions generated by the two algorithms. Table 13 shows the average difference $d_{Opt} - d_{RE}$, as well as the standard deviation σ_d of these differences. Although the average difference may appear to be rather small, a t test significantly rejects the hypothesis that $d_{Opt} = d_{RE}$, to the benefit of the alternative hypothesis $d_{Opt} > d_{RE}$ (at high significance levels $>> 0.99$). We also observe that the differences tend to get larger when *Optimist* performs better.

Table 13 Difference in expedition dates for two instances from Table 7

	$N = 80$	$N = 200$
$d_{Opt} - d_{RE}$	2.9	8.1
σ_d	3.6	5.2
t test statistics	4.33	8.37

Furthermore, we note that postponing shipments is unlikely to entail large penalties when the number of items is relatively high in each period, since this increases the number of

favorable combinations of items, and hence reduces the likelihood that a postponed item may require an additional truck. The results in Tables 7 to 10 accordingly show that algorithm *Optimist* often performs better than other algorithms when N gets larger.

These observations, by themselves, cannot fully validate our tentative explanation according to which "postponing decisions tends to produce better outcomes when facing an uncertain future", but they provide partial support for this plausible interpretation.

10 Conclusions

In this article, we have investigated an original multi-period vehicle loading problem including stochastic information about releases from production. A main objective of this research was to examine the benefits that the firm can draw from improving the coordination between production and transportation. Improved coordination means, in particular, that transportation of goods to the customers should not be managed on a purely reactive and myopic mode, whereby items are shipped as soon as they are released from the plant, but that it should be optimized on the basis of forecasts derived from the production plan.

We have proposed a stochastic optimization model with rolling-horizon for this problem. The deterministic version of the model, where all release dates are assumed to be exactly known over a fixed horizon, can be formulated as a set covering problem (closely resembling the bin packing problem) and, for realistic problem sizes, can be solved by a commercial IP solver. The stochastic model, however, is much more difficult. We have proposed several efficient heuristic algorithms for its solution.

The corresponding policies have been extensively tested on randomly-generated instances which share the main characteristics of the industrial application that motivated our study. As we are dealing with stochastic optimization problems, particular attention has been paid to the estimation of the objective function (expected cost over a rolling horizon), to the statistical significance of the comparisons, and to the robustness of the results.

Our main conclusions are as follows:

1. The multi-period setting provides a clear benefit over the myopic procedures that are frequently used in practice. In our experiments, the value of the multi-period model (VMPM) is very high and the Local Optimization algorithm (*LO*) is dominated by all heuristics that take future scenarios into account.
2. Even though it does not make much use of the special structure of the problem, the generic *Restricted Expectation algorithm (RE)* yields very good results and is quite robust.
3. The simple heuristic *Optimist*, which combines point forecasts based on earliest release dates with truckload optimization over a short look-ahead horizon, performs surprisingly well. It is at least on a par with, and often superior to much more complex algorithms. It allows us to close a large fraction of the gap between the cost of the "optimal policy" (computed ex post under conditions of perfect information) and the cost of the myopic policy *LO*. Moreover, *Optimist* appears to be robust under a variety of assumptions regarding the probability distributions of release dates.
4. The Value of the Accessible Information, i.e., $VAI = \min_{\mathscr{A}} Z(\mathscr{A}) - O^*(L)$, may be relatively large, which suggests that the firm would benefit from reducing the uncertainty that surrounds the release dates, at least over a short look-ahead horizon. In order to attain this ultimate goal of supply chain coordination, the firm may have to rely on improvement practices derived from lean management, total productivity management, or quality management. Such developments are outside the scope of our study.

5. From a methodological point of view, these contributions illustrate the benefits of bridging part of the existing gap between the pragmatic, but suboptimal methods frequently used in the industry, and more sophisticated methods proposed in the scientific literature in order to deal with decision-making situations that involve a significant amount of uncertainty. In particular, we believe that the approaches proposed in this paper provide an interesting response to two of the main limitations encountered in the implementation of stochastic optimization models, namely, their computational hardness, and the difficulty to provide detailed estimates of the relevant probability distributions in an industrial setting. The resulting methods are easily implemented, computationally efficient, robust with respect to the estimation of model parameters, and they deliver considerably better solutions than myopic procedures which do not take forecasts into account.

Acknowledgements We are grateful to the supply chain department of ArcelorMittal Liège for providing us with industrial data, and to Frédéric Semet for his comments during the course of our research. We thank the reviewers for their helpful comments. The project leading to these results was partially supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, Grant P7/36.

References

Angelelli, E., Bianchessi, N., Mansini, R., & Speranza, M. (2009). Short term strategies for a dynamic multi-period routing problem. *Transportation Research*, *17*, 106–119.

Angelelli, E., Savelsbergh, M., & Speranza, M. (2007). Competitive analysis for dynamic multi-period uncapacitated routing problems. *Networks*, *49*, 308–317.

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. C., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, *1*, 9–32.

Bertsimas, D., Brown, D., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, *53*, 464–501.

Birge, J. R., & Louveaux, F. (1997). *Introduction to Stochastic Programming*. Berlin: Springer.

Boudia, M., Louly, M., & Prins, C. (2007). A reactive GRASP and path relinking for a combined production-distribution problem. *Computers and Operations Research*, *34*, 3402–3419.

Brownlee, J. (2007). *A note on research methodology and benchmarking optimization algorithms*. Technical Report Swinburne University of Technology.

Chen, Z.-L. (2004). Integrated production and distribution operations: Taxonomy, models and review. In D. Simchi-Levi, S. Wu, & Z.-J. Shen (Eds.), *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-business Era* chapter 17. (pp. 711–746). Kluwer Academic Publishers.

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

Coffman, J., E. G., Garey, M. R., & Johnson, D. S. (1983). Dynamic bin-packing. *SIAM Journal on Computing*, *12*, 227–258.

Cordeau, J.-F. (2007). The dial-a-ride problem : models and algorithms. *Annals of Operations Research*, *153*, 29–46.

Crainic, T. (2003). Long-haul freight transportation. In R. Hall (Ed.), *Handbook of Transportation Science* International Series in Operations Research and Management Science (pp. 451–516). Boston, MA: Kluwer Academic Publishers.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, *44*, 145–159.

Erengüç, S. S., Simpson, N. C., & Vakharia, A. J. (1999). Integrated production/distribution planning in supply chains: An invited review. *European Journal of Operational Research*, *115*, 219–236.

Frantzeskakis, L., & Powell, W. (1990). A successive linear approximation procedure for stochastic dynamic vehicle allocation problems. *Transportation Science*, *24*, 40–57.

Fumero, F., & Vercellis, C. (1999). Synchronized development of production, inventory and distribution schedules. *Transportation Science*, *33*, 330–340.

Gendreau, M., & Potvin, J.-Y. (1998). Dynamic vehicle routing and dispatching. In T. Crainic, & G. Laporte (Eds.), *Fleet Management and Logistics* (pp. 115–126). Boston: Springer.

Heitsch, H., & Romisch, W. (2007a). A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, *35*, 731–738.

Heitsch, H., & Romisch, W. (2007b). Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, *118*, 371–406.

Hooker, J. N. (1995). *Testing heuristics: We have it all wrong*. Technical Report Carnegie Mellon University.

Johnson, D. S. (2001). *A theoretician's guide to the experimental analysis of algorithms*. Technical Report AT and T Labs.

Kouvelis, P., & Yu, G. (1997). *Robust Discrete Optimization and its Applications*. Dordrecht, The Netherlands: Kluwer Academic Publisher.

Melo, R., & Wolsey, L. (2010). Optimizing production and transportation in a commit-to-delivery business mode. *European Journal of Operational Research*, *203*, 614–618.

Pillac, V., Gendreau, M., Guret, C., & Medaglia, A. (2011). *A revieuw of dynamic vehicle routing problems*. Technical Report CIRRELT-2011-62 CIRRELT.

Powell, W. (2011). *Approximate Dynamic Programming*. Wiley Series in Probability and Statistics (2nd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Powell, W., & Topaloglu, H. (2003). Stochastic programming in transportation and logistics. In *Handbooks in Operations Research and Management Science, Volume 10* (pp. 555–636). Amsterdam: Elsevier.

Psaraftis, H. (1995). Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, *61*, 143–164.

Rardin, R. L., & Uzsoy, R. (2001). Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, *7*, 261–304.

Rhee, W. (1988). Optimal bin packing with items of random sizes. *Mathematics of Operations Research*, *13*, 140–151.

Ruml, W. (2010). *The logic of benchmarking : A case against state-of-the-art performance*. Technical Report Association for the Advancement of Artificial Intelligence.

Sarmiento, A. M., & Nagi, R. (1999). A review of integrated analysis of production-distribution systems. *IIE Transactions*, *31*, 1061–1074.

Sethi, S., & Sorger, G. (1991). A theory of rolling horizon decision making. *Annals of Operations Research*, *29*, 387–416.

Shapiro, A. (2003). Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research (ZOR)*, *58*, 57–68.

Stecke, K., & Zhao, X. (2007). Production and transportation integration for a make-to-order manufacturing company with a commit-to-delivery business mode. *Manufacturing & Service Operations Management*, *9*, 206–224.

Tang, J., Yung, K.-L., Ip, A. W. H., & Liu, S. (2007). Synchronized production and transportation planning using subcontracted vehicles in a production-distribution network. *Transportation Planning and Technology*, *30*, 113–146.

Van Hentenryck, P., & Bent, R. (2006). *Online Stochastic Combinatorial Optimization*. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, *Series A 86*, 565–594.

Vanderbeck, F., & Wolsey, L. A. (1995). An exact algorithm for IP column generation. *Operations Research Letters*, *19*, 151–159.

Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. (2002). The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, *24*, 289–333.

Vidal, C., & Goetschalckx, M. (1997). Strategic production-distribution models: A critical review with emphasis on global supply chain models. *European Journal of Operational Research*, *98*, 1–18.

Wang, X., & Cheng, T. C. E. (2009). Logistics scheduling to minimize inventory and transport costs. *International Journal of Production Economics*, *121*, 266–273.

Wascher, G., Hausner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*, 1109–1130.

Wen, M., Cordeau, J.-F., Laporte, G., & Larsen, J. (2010). The dynamic multi-period vehicle routing problem. *Computers and Operations Research*, *37*, 1615–1623.

Yung, K.-L., Tang, J., Ip, A. W. H., & Wang, D. (2006). Heuristics for joint decisions in production, transportation, and order quantity. *Transportation Science*, *40*, 99–116.

Zegordi, S. H., Abadi, I. N. K., & Nia, M. A. B. (2010). A novel genetic algorithm for solving production and transportation scheduling in a two-stage supply chain. *Computers and Industrial Engineering*, *58*, 373–381.