TRANSPORTATION SCIENCE
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0041-1655 | EISSN 1526-5447 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

Randomized local search for real-life inventory routing

Thierry Benoist, Frédéric Gardi, Antoine Jeanjean
Bouygues e-lab, 40 rue Washington, 75008 Paris,
{tbenoist,fgardi,ajeanjean}@bouygues.com

Bertrand Estellon
Laboratoire d'Informatique Fondamentale - CNRS UMR 6166,
Faculté des Sciences de Luminy - Université Aix-Marseille II,
163 avenue de Luminy - case 901, 13288 Marseille cedex 9,
bertrand.estellon@lif.univ-mrs.fr

In this paper, a real-life routing and scheduling problem is addressed. The problem, which consists in optimizing the distribution of fluids by tank trucks in the long run, is a generalization of the vehicle routing problem with vendor managed inventory replenishment. The particularity of this problem is that the vendor monitors the customers' inventories, deciding when and how much each inventory should be replenished by routing trucks. Thus, the objective of the vendor is to minimize the logistic cost of the inventory replenishment for all customers in the long run. Having detailed the modeling of the real-life problematic, the practical short-term planning approach adopted for optimizing the long-term objective is presented. Then, a pure and direct local-search heuristic is described for solving the short-term planning problem, using a surrogate objective function based on long-term lower bounds. The design and engineering of this algorithm, which is central to the approach, follows the three-layers methodology for "high-performance local search" recently introduced by some of the authors. An extensive computational study shows that our solution is effective, efficient and robust, providing long-term savings exceeding 20 % on average compared to solutions built by expert planners or even a classical urgency-based constructive algorithm. Confirming the promised long-term savings in operations, the resulting decision support system is going to be deployed worldwide.

Key words: logistics; inventory routing; decision support system; stochastic local search; high-performance algorithm engineering
History:

The problem addressed in this paper is a real-life inventory routing problem (IRP) occurring in one of the world's leading company in its field. In order to familiarize the reader with the whole problematic, an informal description is given before presenting our contributions.

Fluid products are produced by the vendor's plants and are consumed at customers' sites. Both plants and customers store the product in tanks. Reliable forecasts of production at plants are known over a short-term horizon. On the customer side, two kinds of resupply are managed by the vendor. The first one, called "forecasting-based resupply", corresponds to clients for which reliable consumption forecasts are available over a short-term horizon. The inventory of each customer must be replenished by tank trucks so as to never fall under its safety level. The second one, called "order-based resupply", corresponds to customers which send orders to the vendor, specifying the desired quantity and the time window in which the delivery must be done. Some customers can ask for the both types of resupply management: their inventory is replenished by the vendor using monitoring and forecasting, but they keep the possibility of ordering (to deal with an unexpected increase of their consumption, for example). The constraints consisting in satisfying orders (no missed orders) and in maintaining inventory levels above safety levels (no stock out) are defined as soft, since the existence of an admissible solution is not ensured in real-life conditions.

The transportation is performed by vehicles composed of three kinds of heterogenous resources: drivers, tractors, trailers. Each resource is assigned to a base. A vehicle is formed by associating one driver, one tractor and one trailer. Some triplets of resources are not admissible (due to driving licences, for example). The availability of each resource is defined through a set of time windows. Each site (plant or customer) is accessible to a subset of resources (special skills or certifications are required to work on certain sites). Thus, scheduling a shift consists in defining: a base, a triplet of resources (driver, tractor, trailer), and a set of operations each one defined by a triplet (site, date, quantity) corresponding to the pickups or deliveries performed along the tour. A shift must start from the base to which are assigned the resources composing the vehicle and end by returning to this base. The working and driving times of drivers are limited; as soon as a maximum duration is reached, the driver must take a rest with a minimum duration (Department of Transportation rules). In addition, the duration of a shift cannot exceed a maximal value depending on the driver. The sites visited along the tour must be accessible to the resources composing the vehicle. A resource can be used only during one of its availability time windows. The date of pickup/delivery must be contained in one of the opening time windows of the visited site. Finally, the inventory dynamics, which can be modeled by flow equations, must be respected at each time step, for each site inventory and each trailer; in particular, the sum of quantities delivered to a customer (resp. loaded at a plant) minus (resp. plus) the sum of quantities consumed by this customer (resp. produced by this plant) over a time step must be smaller (resp. greater) than the capacity of its storage (resp. zero). Note that here the duration of an operation does not depend on the delivered or loaded quantity; this duration is fixed in function of the site where the operation is performed, the resulting approximation being covered by the uncertainties lying on the traveled times.

In our case, reliable forecasts (for both plants and customers) are available over a 15-days horizon. Thus, shifts are planned deterministically day after day with a rolling horizon of 15 days. It means that each day, a distribution plan is built for the next 15 days, but only shifts starting at the current day are fixed. The objective of the planning is to respect the soft constraints described above over the long run (satisfying orders, maintaining safety levels). In practice, the situations where these constraints cannot be met are extremely rare, because missed orders and stockouts are unacceptable for customers (of course, safety levels must be finely tuned according to customer consumptions). Then, the second objective is to minimize over the long term a logistic ratio defined as the sum of the costs of shifts (which is composed of different terms related to the usage of resources) divided by the sum of the quantities delivered to customers. In other words, this logistic ratio corresponds to the cost per unit of delivered product.

Large-scale instances have to be tackled. A geographic area can contain up to 1500 customers, 50 sources, 50 bases, 100 drivers, 100 tractors, 100 trailers. All temporal data have to be managed in continuous time, except for consumptions of customers (resp. productions of plants) which are discretely represented. Concretely, all dates and durations are expressed in minutes (on the whole, the short-term planning horizon counts 21600 minutes); the inventory dynamics for plants and customers are computed with time steps of one hour (because forecasts are computed with this accuracy). The execution time for computing a short-term planning is limited to 5 minutes on standard computers.

1. Related works and contributions

Since the seminal work of Bell et al. (1983) on a real-life inventory routing problem, a vast literature has emerged on the subject. In particular, a long series of papers was published by Campbell et al. (1998, 2002), Campbell and Savelsbergh (2004a), Savelsbergh and Song (2007a,b, 2008), motivated by a real-life problematic encountered in the gas industry. However, in many companies, inventory routing is still done by hand or supported by basic softwares, with rules like: serve “emergency” customers (that is, customers whose inventory is near to run out) using as many “full deliveries” as possible (that is, deliveries with quantity equal to the trailer capacity or, if not possible, to the customer tank capacity). For more references, the interested reader is referred to the recent papers by Savelsbergh and Song (2007a, 2008), which give a comprehensive survey of the research done on the IRP over the past 25 years.

1.1. Contributions to IRP modeling

The problem addressed here is very close to the one treated by operational planners. To our acquaintance, such broad inventory routing problems have been rarely addressed in the operations research literature. Indeed, many real-life features described here have not been treated in past studies, allowing a more global and accurate optimization of the replenishment logistics. Some of these features have been reported as important practical issues in the survey by Campbell et al. (1998). First, our inventory routing model integrates both kinds of resupply: forecasting-based and order-based. Besides, several subproblems related to the scheduling of shifts and the allocation of resources to shifts become computationally hard in the present case. Another interesting feature, enabling to go further in logistic optimization while making the problem harder, is what Savelsbergh and Song (2007a, 2008) called "continuous moves". The vehicles can arbitrarily load or deliver some product along their routes, and loadings can be done at multiple plants. Moreover, when a driver reaches its working or driving time limit, he can continue his route after a layover. This allows to design shifts spanning several days and covering huge geographic areas. Finally, the expected forecasts of consumption for customers and of production for plants are given for each hour on a 15-days horizon, allowing nonlinear consumptions/productions; here forecasts are assumed to be reliable, inducing a deterministic optimization problem (contingencies on the customer consumption are considered to be covered by the defined safety level). Customers (resp. plants) may have different consumption (resp. production) profile, asking several deliveries (resp. pickups) per day or only one per month. Note that one feature generally addressed in the IRP literature (e.g. Campbell et al. 1998, 2002, Savelsbergh and Song 2007a, 2008) is not included in our IRP model: loading or delivery times depending on the quantity. Indeed, fixed-time loadings and deliveries depending on sites were judged sufficient to approximate reality (full loadings/deliveries are performed in almost half an hour), because several other approximations making this detail negligible are done about temporal aspects due to real-life uncertainties (in particular about traveled times). Nevertheless, we shall see later that our solution could be modified to manage this feature without significantly affecting its performance.

As mentioned by Campbell et al. (1998) and Campbell et al. (2002), the first difficulty arising in modeling IRP is to define appropriate short-term objectives leading to good long-term solutions. But how to define good long-term solutions? A popular and sensible objective, used by Campbell et al. (1998, 2002), Savelsbergh and Song (2007a, 2008), is to maximize the volume per mile over the long term, obtained by dividing the total quantity delivered to all customers by the total distance traveled. Instead of the sole traveled distance, we take into account the actual cost of the routes, thanks to a precise modeling of the cost of each shift in function of its traveled distance, its traveled time, its number of loadings, its number of deliveries, and its number of rests. The resulting generalized objective is the minimization of the cost per unit of delivered product, called *logistic ratio* throughout the paper. This was made possible by modeling the cost of a shift in function of its traveled distance, its traveled time, its number of loadings, its number of deliveries, and its number of rests. Then, our first contribution is to introduce a surrogate objective for short-term optimization (here done over a 15-days horizon) ensuring long-term improvements. This surrogate objective, which shall be detailed later in the paper, is based on lower bounds for the logistic ratio (this extends observations made by Savelsbergh and Song (2007b) on performance measurement). Computational experiments with real-life data show that significant gains are obtained in the long run by optimizing this short-term surrogate objective, compared to a direct short-term minimization of the logistic ratio.

1.2. Contributions to IRP resolution

To our knowledge, the sole papers describing practical solutions for similar problems are the ones described by Campbell et al. (2002), Campbell and Savelsbergh (2004a), Savelsbergh and Song

(2007a, 2008). Before presenting our solution approach, we outline the ones implemented by Campbell et al. (2002), Campbell and Savelsbergh (2004a) for solving the single-plant IRP, and by Savelsbergh and Song (2007a, 2008) for solving the multiple-plant IRP.

The solution approaches described by Campbell et al. (2002) and Campbell and Savelsbergh (2004a) are the same in essence; because integrating additional realistic constraints, the single-plant IRP addressed by Campbell and Savelsbergh (2004a) is more complex than the one by Campbell et al. (2002). The methodology developed by the authors is deterministic and proceeds in two phases. In the first phase, it is decided which customers are visited in the next few days, and a target amount of product to be delivered to these customers is set. In the second phase, vehicle routes are determined taking into account vehicle capacities, customer delivery windows, drivers restrictions, etc. The first phase is solved heuristically by integer programming techniques, whereas the second phase is solved with specific insertion heuristics (Campbell and Savelsbergh 2004c), as done for vehicle routing problems with time windows by Solomon (1987). In Campbell et al. (2002), a planning is constructed on a rolling horizon by considering 5 days in full detail plus 4 weeks in aggregated form beyond this. Computational experiments are made on two instances with 50 customers and 87 customers respectively, with 4 vehicles as resources. The authors compare their short-term solutions to the ones obtained by a greedy algorithm based on the rules of thumb commonly used in practice (like the one cited at the beginning of this section). They obtain an average gain of 8.2 % for the volume per mile (running times are not reported). In Campbell and Savelsbergh (2004a), the authors simulate the use of a rolling-horizon approach covering one month. At each iteration of the rolling-horizon framework, they solve the first-phase integer program on 3 days in full detail plus 1 week in aggregated form beyond this, and run the second-phase insertion heuristic with the information from the solution of the integer program for the first two days. Then, the resulting routes are fixed and the clock is moved forward two days in time. The running time to perform one iteration is limited to 10 minutes (with a 366 MHz processor). The authors compare their approach to a greedy algorithm similar to the one described in Campbell et al. (2002). The benchmarks are composed of two instances with almost 100 customers and 50 customers respectively (the available resources are not detailed). The average gain over one month is of 2.7 % for the volume per mile, but a better utilization of ressources is observed (larger average percentage of trailer capacity delivered on routes, shorter average length of shifts).

In Savelsbergh and Song (2007a, 2008), the authors develop two approaches for solving the multiple-plant IRP. Many realistic features taken into account in Campbell and Savelsbergh (2004a) are relaxed in the model addressed by the authors. In particular, simple resources are considered (that is, a vehicle is reduced to a trailer) allowing an integer multi-commodity flow formulation of the problem. The first approach (Savelsbergh and Song 2007a) is based on an insertion heuristic which delivers customers ordered by urgency (that is, the time remaining before the first stockout) while minimizing stockout and transportation costs. This approach is declined into three greedy algorithms: a basic one (called BGH) where insertions are only performed at the end of shifts, a enhanced one (EGH) where insertions can be performed at any point in the shift after the last pickup, and a randomized enhanced one (RGH) where the EGH algorithm is embedded into a greedy randomized adaptive search procedure (Feo and Resende 1995). Then, a postprocessing is performed using linear programming for maximizing delivered quantities on the resulting shifts (in order to maximize the volume per mile). The authors present computational results made on 20 benchmarks derived from an instance with 200 customers, 7 plants, 7 vehicles (with a 2.4 GHz processor). On a 10-days horizon, the average improvement for stockout and transportation costs from BGH to EGH (resp. from EGH to RGH) is of 15.2 % (resp. 6.8 %); the average running time is about a few seconds for BGH and EGH, and about 12 minutes for RGH. The postprocessing optimization is shown to increase the total delivered quantity by 2.8 % on average on the same benchmarks (with a running time lower than one second). Other experiments made on a rolling horizon of 5 months (with 10 days planned, 5 days fixed) show that the delivery volume post optimization helps to reduce costs of about 3 % (using RGH as reference algorithm). The second approach (Savelsbergh and Song 2008) consists in solving heuristically the integer multi-commodity flow program (by using customized integer programming techniques). The authors present computational results made on 25 benchmarks derived from the instance with 200 customers used as basis in Savelsbergh and Song (2007a). The average improvement over RGH for stockout and transportation costs is of 4.1 %, whereas the average running time is greater than 31 hours (with a 900 MHz processor). Since such computational requirements are too large for a practical use, the authors use the integer program for exploring large neighborhoods in a local search scheme (see Estellon et al. (2006, 2008) for an application of this technique to car sequencing problems). This consists in re-optimizing the schedules of two vehicles in the planning by solving the integer program with the other schedules fixed. In this way, all pairs of vehicles are re-optimized iteratively. The authors report an average improvement over RGH of 3.1 %, with an average running time lower than 3 minutes and an average number of improving iterations of 3. Unfortunately, no precise statistic is given in Savelsbergh and Song (2007a, 2008) about the resulting volume per mile over a long term.

Our second contribution concerns the resolution of the short-term planning problem with the surrogate objective. The short-term planning is built for 15 days in full details and only shifts starting the first day are fixed before rolling the horizon. In this paper, a pure and direct local-search heuristic is described for solving the short-term planning problem, whose design and engineering follows the three-layers methodology recently formalized by Estellon et al. (2009) and successfully implemented for solving other large-scale business optimization problems (car sequencing with paint colors at Renault by Estellon et al. (2006, 2008), task scheduling with human resource allocation at France Télécom by Estellon et al. (2009)). A local-search approach is outlined by Lau et al. (2002) for solving an inventory routing problem with time windows, but their solution remains based on a decomposition of the problem (distribution and then routing). We insist on the fact that no decomposition is done here: the 15-days planning is directly optimized by local search. An extensive computational study demonstrates that our solution is both effective, efficient and robust, providing long-term savings exceeding 20 % on average, compared to solutions computed by expert planners or even a classical urgency-based constructive heuristic.

Following the methodology of Estellon et al. (2009), our local-search heuristic is designed according to three layers. The first layer corresponds to the search strategy; here a first-improvement descent heuristic with stochastic selection of transformations is employed (an initial solution is computed using an urgency-based insertion heuristic). The second layer corresponds to the pool of transformations which defines the neighborhood; here more than one hundred transformations are defined on the whole, which can be grouped into a dozen of types (for operations: insertion, deletion, ejection, move, swap; for shifts: insertion, deletion, rolling, move, swap, fusion, separation). Finally, the third layer, corresponding to the "engine" of the local search, consists of three main procedures common to all transformations: evaluate (which evaluates the gain provided by the transformation applied to the current solution), commit (which validates the transformation by updating the current solution and the associated data structures), rollback (which clears all the data structures used to evaluate the transformation). Since the duration of an operation does not depend on the quantity loaded or delivered, the evaluation procedure is separated into two routines: first the scheduling of shifts and then the assignment of volumes. These routines, whose running time is critical for performance, relies on incremental algorithms supported by special data structures for exploiting invariants of transformations. On average, our algorithm visits more than 10 million solutions in the search space during 5 minutes of running time, with a diversification rate of almost 5 % (that is, the number of committed transformations over the number of attempted ones), which allows to reach quickly high-quality local optima.

An abstract of this work appears in Benoist et al. (2009). For an introduction to local search techniques and their applications in combinatorial optimization, the reader is referred to the book edited by Aarts and Lenstra (1997).

2. The inventory routing model

First are detailed the input and output data of the problem. Then, the constraints and objectives of the model will be exposed.

2.1. Input data

The different units of measurement for quantity, time and distance can be chosen freely, but must be consistent. For measuring quantities, weights are generally preferred to volumes in bulk logistics.

The time is represented as a continuous line with horizon T. In other words, any instant is given by a point in the interval $[0, T]$. Thus, all dates defined in the model can be expressed with the desired precision. In effect, we work with a value of T equal to 15 days and the time unit is the minute. Due to physical restrictions, forecasts cannot be available continuously. Thus, consumptions and productions are given discretely for time steps of size U, such that $U \times H = T$ with H the number of time steps over the horizon. In our case, the granularity adopted for U is one hour. Except contrary mention, any interval of time (in particular time windows defined in input) is such that the starting date is included and the ending date is excluded.

The size of the input data of the problem is essentially defined by the number of customers, the number of plants, the number of bases, the number of drivers, the number of tractors, the number of trailers, the number of orders, and the number of time steps for which are defined consumptions/productions over the horizon.

2.1.1. Resources. Here are described the attributes for each kind of resources: drivers, tractors, trailers. A driver resource can represent one driver or a pair of drivers, as encountered in geographic areas like Canada for making very long trips. Then, a driver d is defined by: the $base(d)$ to which he is located, the set $timeWindows(d)$ of availability time windows over the horizon, the set $tractors(d)$ of tractors matchable to the driver, the maximum amplitude $maxAmplitude(d)$ of each shift performed by the driver, the maximum driving duration $maxDrivingDuration(d)$ after which a layover is required (e.g. 11 hours in the USA), the maximum working duration $maxWorkingDuration(d)$ after which a layover is required (e.g. 14 hours in the USA), the minimum duration $minLayoverDuration(d)$ of any layover (e.g. 10 hours in the USA), the cost $timeCost(d)$ per unit of working time, the cost $loadingCost(d)$ for each loading operation performed by the driver, the cost $deliveryCost(d)$ for each delivery operation performed by the driver, the cost $layoverCost(d)$ for each layover taken by the driver.

Note that if a driver represents in reality a pair of drivers, then the driving/working rules must match what is allowed for this team. For example, a pair of drivers whose each one is subject to the 11/14/10 DOT rules could have $maxDrivingDuration(d) = maxWorkingDuration(d) = maxAmplitude(d)$ considering that the two drivers alternate the driving/working periods (the second driver takes a rest during the duty of the first one, and vice versa). Note that in this case the duration of the shift shall remain constrained by the parameter $maxAmplitude(d)$.

Then, a tractor tr is defined by: the $base(tr)$ to which it is located, the set $timeWindows(tr)$ of availability time windows, the set $trailers(tr)$ of trailers matchable to the tractor, its speed $tractorSpeed(tr)$ (an integer between $[0, 9]$ used as index in the time matrix), the cost $distanceCost(tr)$ per unit of traveled distance. Finally, a trailer tl is defined by: the $base(tl)$ to which it is located, the set $timeWindows(tl)$ of availability time windows, its $capacity(tl)$ (that is, the maximal quantity that can be loaded in the trailer and delivered to customers), the quantity $initialQuantity(tl)$ of product in the trailer at the beginning of the period.

2.1.2. Locations. A location on the map is either a base, a customer, or a plant. Any location p has two $x(p), y(p)$ coordinates to be located on the map. Bases are just locations to which are assigned resources; they are used as starting and ending locations of the shifts. Any customer p has the following attributes: $\textit{capacity}(p)$ which represents the size of its inventory (that is, the maximum quantity that can be delivered to the customer), $\textit{safetyLevel}(p)$ corresponding to the quantity of product which must be maintained in the inventory to avoid stockout costs, the $\textit{initialQuantity}(p)$ of product in the tank of the customer at the beginning of the period, $\textit{forecast}(p,h)$ which gives for each time step h the consumption of the customer, the set $\textit{timeWindows}(p)$ of availability time windows, the set $\textit{allowedDrivers}(p)$ of drivers which are allowed to enter to this customer (some drivers may be forbidden due to inadequate skills), the set $\textit{allowedTractors}(p)$ of tractors which are allowed to enter to this customer (some tractors may be forbidden due to their large size), the set $\textit{allowedTrailers}(p)$ of trailers which are allowed to enter to this customer (some trailers may be forbidden due to inadequate equipments), the fixed duration $\textit{setupTime}(p)$ taken to perform a delivery to this customer (here set to the average delivery time), the cost $\textit{missedOrderCost}(p)$ paid for each missed order, the cost $\textit{runoutCost}(p)$ per time step spent in stockout, the list $\textit{orders}(p)$ of orders asked by the customer, the flag $\textit{callIn}(p)$ which is true if unsolicited deliveries are forbidden for this customer (that is, this one works in pure order-based resupply mode), and finally the flag $\textit{firstAfterSource}(p)$ which is true if this customer must be delivered just after a loading operation in the shift (used to check the purity of the product before the delivery).

An order r is characterized by: the $\textit{quantity}(r)$ asked by the customer and the $\textit{earliestTime}(r)$ and $\textit{latestTime}(r)$ which define the time window for delivering it (more precisely, the starting date of the delivery operation must be contained into this interval). Note that if $\textit{callIn}(p)$ is true, the attributes $\textit{capacity}(p)$, $\textit{safetyLevel}(p)$, $\textit{initialQuantity}(p)$, $\textit{forecast}(p,h)$ and $\textit{runoutCost}(p)$ are not relevant for the customer p. Plants are modeled similarly, without attributes $\textit{runoutCost}(p)$, $\textit{orders}(p)$, $\textit{callIn}(p)$, $\textit{firstAfterSource}(p)$. Note that consumptions of customers (resp. productions of plants) are represented with positive (resp. negative) values.

Finally, some distance and time matrices are provided: $\textit{distMatrix}(p,q)$ gives the distance between locations p and q, $\textit{timeMatrix}(p,q,r)$ corresponds to the traveling time from p to q using a tractor tr with $\textit{tractorSpeed}(tr)$ index equal to r. Both matrices are not necessarily symmetric, but are assumed to satisfy the triangular inequality. Some checking operations must be performed at the start and the end of any shift, as well as before and after any layover; these fixed durations are respectively denoted by $\textit{preTripTime}$ and $\textit{postTripTime}$.

2.2. Output data

A solution consists in a set of shifts. A shift s is defined by: its $\textit{driver}(s)$, its $\textit{tractor}(s)$, its $\textit{trailer}(s)$, its $\textit{base}(s)$, its starting date $\textit{start}(s)$ from the base, its ending date $\textit{end}(s)$ to the base, the quantity $\textit{startTrailerQuantity}(s)$ of product in the trailer at the beginning of the shift, the quantity $\textit{endTrailerQuantity}(s)$ of product at the end of the shift, and the chronological-ordered list $\textit{operations}(s)$ of performed operations.

Then, an operation o is defined by: the shift $\textit{shift}(o)$ to which the operation o belongs, the site $\textit{point}(o)$ where the operation takes place, the order r satisfied by the operation (if any), the $\textit{quantity}(o)$ delivered or loaded (positive for delivery, negative for loading), its starting date $\textit{arrival}(o)$, its ending date $\textit{departure}(o)$, the list $\textit{layoversBefore}(o)$ of layovers taken since the previous operation (in practice, several layovers are rarely set between two operations). Note that the list $\textit{operations}(s)$ contains a final fake operation (with null quantity) used for storing layovers between the last site visited and the base. A layover l, which represents a resting interval for the driver between two locations, is defined by: its starting date $\textit{start}(l)$, its ending date $\textit{end}(l)$, the driving time $\textit{drivingBefore}(l)$ from the previous location or the previous layover (in case of multiple layovers between two locations) in the shift. The value $\textit{drivingBefore}(l) = 0$ means that the layover is taken at the previous location.

The inventory levels (for customers, plants, trailers) can be computed from the quantities delivered or loaded in shifts. We denote by $tankQuantity(p,h)$ the quantity of product in the tank of site p at time step h, and by $trailerQuantity(tl,o)$ the quantity of product in the trailer tl at the end of operation o. If the operation o is the last of the shift s, we must have $endTrailerQuantity(s) = trailerQuantity(tl,o)$. Note that all input and output data related to volumes are integers.

2.3. Constraints

As noted previously, the present IRP can be decomposed into two subproblems: routing/scheduling shifts and assigning volumes.

2.3.1. Routing constraints. Here are listed the constraints bearing on shifts, called routing constraints. The three resources (driver, tractor, trailer) assigned to the shift must be located at the base of the shift. The tractor assigned to the shift must be compatible with the driver of the shift, that is, it must belong to the list $tractors(d)$ which can be driven by this driver. In the same way, the trailer of the shift must be compatible with the tractor of the shift. The interval $[start(s), end(s)[$ induced by any shift s must be contained into an availability time window for each resource assigned to s. Finally, the shifts performed by a resource cannot overlap in time (that is, the time intervals induced by the shifts are pairwise disjoint).

In addition, there are constraints specific to drivers. For each driver d, two consecutive shifts assigned to d must be separated by at most $minLayoverDuration(d)$ and the duration of a shift cannot exceed $maxAmplitude(d)$. For any driver d, $cumulatedDrivingTime(d,t)$ at time t corresponds to the driving time cumulated since the end of the last layover or the start of the shift. In the same way, $cumulatedWorkingTime(d,t)$ corresponds to the cumulated working time since the end of the last layover or the start of the shift; it includes the driving time, the time to perform operations at each site, the $preTripTime$ after each layover (or start from the base), and the $postTripTime$ before each layover (or return to the base). At any time t of a shift, $cumulatedDrivingTime(d,t)$ (resp. $cumulatedWorkingTime(d)$) cannot exceed $maxDrivingDuration(d)$ (resp. $maxWorkingDuration(d)$). In other words, one layover must be set once one of the two maximal durations is reached. The duration of any layover must be greater than $minLayoverDuration(d)$.

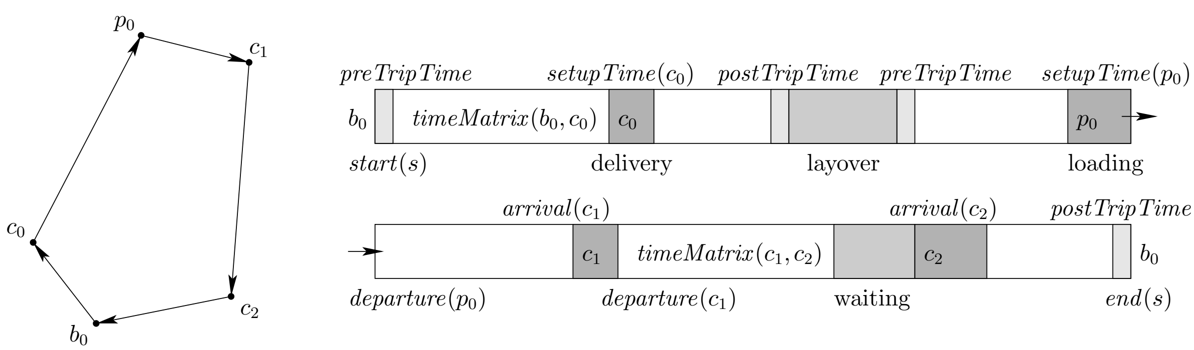

Figure 1 **Two views of the shift** $s = (b_0, c_0, p_0, c_1, c_2, b_0)$**: the route and the schedule.**

Any shift starts from a base and must return to this base (see Figure 1 for two graphical views of a shift). The departure (resp. arrival) of the vehicle from (resp. to) the base must be preceded (resp. followed) by the $preTripTime$ (resp. $postTripTime$) checking. Then, the arrival at a location in the shift requires traveling from the previous location to this one; in other words, the time spent between two consecutive locations p and q is greater than $timeMatrix(p,q,r)$, with r the index of speed of the tractor assigned to the shift. Note that the time spent between two consecutive locations can be greater than (and not only equal to) traveling time to allow waiting time during the shift, for example between the end of the travel and the real entry on the site that may be delayed because of opening hours. Any operation at site p takes a constant time equal to $setupTime(p)$. An operation cannot be stopped for resting (operations are not preemptive), as well as checking operations at start and end of the shift ($preTripTime$ and $postTripTime$). An operation must be performed during the opening hours of the site: the interval $[arrival(o), departure(o)[$ induced by operation o at site p must be contained into one of the intervals $timeWindows(p)$. Note that if a vehicle arrives at a site which is closed, then the vehicle can wait the opening of the site. More generally, a vehicle can stop and wait at any moment during its travel between two operations; the resulting waiting time is assimilated to working time. In addition, all sites of a shift must be accessible to the three resources assigned to the shift: for each site p, the driver (resp. tractor, trailer) of the shift must belong to the list $allowedDrivers(p)$ (resp. $allowedTractors(p)$, $allowedTrailers(p)$). Finally, any delivery performed at customer p with flag $firstAfterSource(p)$ equal to true must be immediately preceded by a loading operation at a plant.

2.3.2. Inventory constraints. Three kinds of inventories have to be managed: tanks of customers, tanks of plants, and trailers. The tank level of a site p at time step h, denoted by $tankQuantity(p,h)$, must remain between zero and its capacity. For customers (except call-in customers which work in pure order-resupply mode), the tank quantity at each time step h is equal to the tank quantity at the previous time step $h-1$, minus the forecasted consumption over h, plus all the deliveries performed over h. Note that the quantities delivered to customers must be positive (loading is forbidden at customers). More formally, the inventory dynamics for customers are expressed as follows: $tankQuantity(p,-1) = initialTankQuantity(p)$ and for all $h \in \{0,\dots,H-1\}$,

$$\begin{cases} tankQuantity(p,h) = tankQuantity(p,h-1) - forecast(p,h) + \sum\limits_{o \in operations(p,h)} quantity(o) \\ if\ \ tankQuantity(p,h) < 0,\ \ then\ tankQuantity(p,h) = 0 \end{cases}$$

with $operations(p,h)$ corresponding to the set of operations performed at site p whose starting date belongs to time step h. Then, $tankQuantity(p,h) < safetyLevel(p)$ implies one stockout for customer p. The inventory dynamics is symmetric for plants, since the forecasted productions and loading quantities have negative values (delivery is forbidden at plants). Thus, the underflow condition is changed into an overflow condition:

$$if\ \ tankQuantity(p,h) > capacity(p),\ \ then\ tankQuantity(p,h) = capacity(p)$$

Note that overflows (which corresponds to venting product) are not penalized in the objective function, because production aspects are assumed to be not managed in this model.

For trailers, the inventory dynamics is realized in continuous time. At any time, the trailer quantity must remain between zero and the capacity of the trailer. The quantity in a trailer at the beginning of a shift is equal to the quantity in this trailer at the end of its previous shift (or the initial quantity if the shift is the first performed by the trailer). Then, the trailer quantity after one operation is equal to the trailer quantity before the operation, plus the delivered or loaded quantity at the site concerned by the operation.

2.4. Objective

The objective, defined *over the long term* (more than 90 days), is three-fold: first to avoid missed orders, then to avoid customer stockouts, and finally to minimize the logistic ratio. As noted earlier, production costs (like venting costs) are not integrated in the model; only distribution costs are taken into account.

The first term MO of the objective function deals with missed orders. An order r is considered as non missed if an operation o is assigned to this one satisfying $quantity(o) \geq quantity(r)$ and

$arrival(o) \in [earliestTime(r), latestTime(r)[$. For each customer p, the number of missed orders is denoted by $nbMissedOrders(p)$. Then the total missed order cost MO is given by:

$$MO = \sum_{p \in customers} missedOrderCost(p) \times nbMissedOrders(p)$$

The second term SO concerns stockouts. A stockout appears at customer p (with flag *callIn* to false) at time step h if $tankQuantity(p, h) < safetyLevel(p)$. For each customer p, the number of time steps in stockout is denoted by $nbStockouts(p)$. Then, the total stockout cost SO is given by:

$$SO = \sum_{p \in customers} stockoutCost(p) \times nbStockouts(p)$$

To avoid end-of-horizon side effects, missed orders and stockouts are counted over a shorter horizon T', defined as $T - \max_{d \in drivers} maxAmplitude(d)$, in order to be sure that demands arising at the end of the horizon can always be satisfied.

The third term is the logistic ratio $LR = SC/DQ$, with SC the total shift cost and DQ the total delivered quantity over the considered horizon (with the exception that if $DQ = 0$, then $LR = 0$). The latter is simply computed as the sum of delivered quantities (that is, positive quantities) for all shifts. The distance of a shift s, denoted by $distShift(s)$, corresponds to the sum of arc lengths of the tour induced by the shift; the duration of a shift s, denoted by $timeShift(s)$, corresponds to the time spent by the driver to work over the shift (that is, $end(s) - start(s)$ minus the sum of layover durations). The total number of deliveries (resp. loadings, layovers) over a shift s is denoted by $nbDeliveries(s)$ (resp. $nbLoadings(s)$, $nbLayovers(s)$). Then, the cost $SC(s)$ of a shift s is composed of five terms:

$$\begin{aligned} SC(s) = & \; distanceCost(tractor(s)) \times distShift(s) \\ & + timeCost(driver(s)) \times timeShift(s) \\ & + deliveryCost(driver(s)) \times nbDeliveries(s) \\ & + loadingCost(driver(s)) \times nbLoadings(s) \\ & + layoverCost(driver(s)) \times nbLayovers(s) \end{aligned}$$

Hence, the total shift cost SC is given by $SC = \sum_{s \in shifts} SC(s)$.

To conclude, we emphasize on the fact that the three terms of the global objective function are optimized in *lexicographic order*: $MO \succ SO \succ LR$. As mentioned in introduction, solutions with $MO = 0$ and $SO = 0$ must be (easily) found in practice.

3. The short-term surrogate objective

One of the main difficulties encountered in IRP problems is to take short-term decisions ensuring long-term improvements. Optimizing the logistic ratio LR over a short-term horizon does not lead necessarily to long-term optimal solutions. For example, assume that a faraway customer has no stockout over the next days. A good short-term decision is to avoid delivering this customer (because delivering necessarily increases the global logistic ratio). More generally, deliveries shall only be triggered due to the appearance of shortages over the horizon. But such short-term decisions may be highly suboptimal in the long run, especially if some near-optimal deliveries are possible over these next days due to the availability of resources. In fact, the short-term goal can be summarized into the following rule: "never put off until tomorrow what you can do optimally today".

This lack of anticipation when minimizing directly the logistic ratio over the short term motivates us to introduce a surrogate objective function. Then, the short-term goal shall be to minimize the global extra cost per unit of delivered product, compared to the optimal logistic ratio LR^*. Denote by $LR^*(p)$ the optimal logistic ratio for delivering the customer p and then by

$$SC^*(s) = \sum_{\substack{customer\,p \\ delivered\ over\ s}} LR^*(p) \times quantity(p)$$

the optimal cost of the shift s according to the quantities delivered at each customer over s. Then, the surrogate logistic ratio LR' is defined as:

$$LR' = \frac{\sum_s (SC(s) - SC^*(s))}{DQ}$$

Unfortunately, it requires to tackle another problem: the computation of lower bounds of $LR^*(p)$ for each customer p (and thus of the global logistic ratio LR^*). In the following subsection is described how to compute such bounds.

3.1. Computing lower bounds

We describe how to compute a lower bound $LR_{\min}$ for the optimal logistic ratio LR^*, assuming that missed orders and stockouts are avoided. Hence, the values $MO = 0$, $SO = 0$, $LR = LR_{\min}$ induces a global lower bound. For now, assuming that all orders are satisfied and no stockout appear, our goal is to compute $LR_{\min}$.

First, a lower bound for $LR^*(p)$ is given. A trip is defined as a subpart of a tour (see Figure 2): it is a sequence of visits starting at a plant (or a base), delivering one or more customers, and finishing at a plant (or a base). In other words, a trip t in the shift s corresponds to an interval $[start(t), end(t)[$ with $start(t)$ (resp. $end(t)$) the starting date from the plant or the base (resp. the starting date from the plant or the base visited in the next trip). Then, the cost of a shift can be decomposed according to its trips, in such a way that the cost of a trip corresponds to the costs (distance, time, deliveries, loadings, layovers) accumulated over $[start(t), end(t)[$. Besides, the cost of each trip can be dispatched to visited customers proportionally to the delivered quantities. For each customer p, a lower bound $LR_{\min}(p)$ is obtained by dividing the cost of the cheapest trip visiting p by the maximum capacity of a trailer able to perform this trip. Since the distance matrix satisfies the triangular inequality, the cheapest trip consists in visiting solely the customer p. Consequently, $LR_{\min}(p)$ is computed in $O((B+P)^2)$ time for each customer p, with B the number of bases and P the number of plants.

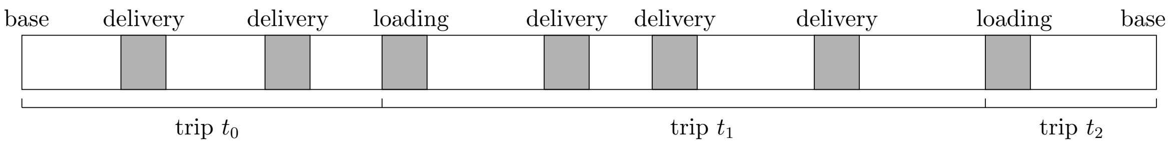

Figure 2 The trips of a shift.

Now, the local lower bounds $LR_{\min}(p)$ are used for computing a global lower bound $LR_{\min}$. For each customer p, denote by $Q_{\min}(p)$ the minimum quantity to deliver to p to prevent it from falling under its safety level over the planning horizon. On the other hand, denote by $Q_{\max}(p)$ the maximum amount of product deliverable along the planning horizon without overfilling its tank. Thus, the quantity $q(p)$ which can be delivered to each customer p in order to avoid stockout is between $Q_{\min}(p)$ and $Q_{\max}(p)$ with:

$$\begin{aligned} Q_{\min}(p) &= safetyLevel(p) - (initialQuantity(p) - \textstyle\sum_h forecast(p,h)) \\ Q_{\max}(p) &= capacity(p) - (initialQuantity(p) - \textstyle\sum_h forecast(p,h)) \end{aligned}$$

If $Q_{\min}(p) = 0$ for all customer p, then the empty solution (no shift) is optimal. Now, assume that at least one customer p exists such that $Q_{\min}(p) > 0$. A first lower bound $LR_{\min}$ is given by:

$$LR_{\min} = \frac{\sum_p (LR_{\min}(p) \times Q_{\min}(p))}{\sum_p Q_{\max}(p)}$$

Each term of the numerator corresponds to the minimum cost of the shifts needed for delivering the quantity $Q_{\min}(p)$ to customer p over the planning horizon. But a better lower bound can be obtained by solving the following mathematical program:

$$\textit{Minimize} \ \frac{\sum_p (LR_{\min}(p) \times q(p))}{\sum_p q(p)}$$

$$q(p) \in [Q_{\min}(p), Q_{\max}(p)] \quad \forall p$$

A solution vector of this program is denoted by $Q = (q(1), \ldots, q(n))$ and its cost by $f(Q)$. First, an optimum solution Q^* of this program is shown to be extremal: the vector Q is such that $q(p) = Q_{\min}(p)$ or $q(p) = Q_{\max}(p)$ for each component p. Indeed, a solution Q^* is optimal if and only if no solution Q exists such that $g(Q) = \sum_p ((LR_{\min}(p) - f(Q^*)) \times q(p)) < 0$. Suppose that an index p exists such that $q(p)$ is not an extreme of $[Q_{\min}(p), Q_{\max}(p)]$. If $LR_{\min}(p) - f(Q^*) \leq 0$ (resp. $LR_{\min}(p) - f(Q^*) \geq 0$), then setting $q(p) = Q_{\max}(p)$ (resp. $q(p) = Q_{\min}(p)$) leads to a solution having a cost lower than or equal to $f(Q^*)$. Since this operation can be performed independently for each index p (because $g(Q)$ is additively separable), our initial claim is proved.

Now, any extremal optimum vector can be normalized by ordering its components such that the corresponding constants $LR_{\min}(p)$ are nondecreasing. Following the previous discussion, an extremal optimum Q^* has a normal form $(Q_{\max}(1), \ldots, Q_{\max}(p^*), Q_{\min}(p^*+1), \ldots, Q_{\min}(n))$, with $LR_{\min}(1) \leq \cdots \leq LR_{\min}(n)$. Consequently, the computation of an extremal optimum is reduced to the computation of an index $p^* \in \{1, \ldots, n\}$ for which the normal form has a minimum cost, which is done in linear time. In conclusion, computing an optimum vector Q^* of the above program is done in $O(n \log n)$ time and linear space, with n the number of components of the the vector.

To summarize, the local lower bounds $LR_{\min}(p)$, defined for each customer p, are computed in $O(C(B+P)^2)$ time and $O(C)$ space, with C (resp. P, B) the number of customers (resp. plants, bases). Then, the global lower bound $LR_{\min}$ is computed in $O(C \log C)$ time and $O(C)$ space.

4. Urgency-based constructive heuristic

In order to quickly build an initial solution, a constructive heuristic was designed, based on a classical urgency approach. The goal of this algorithm is to serve orders and to avoid stockouts. Basically, it repeatedly picks the next order or stockout and tries to create a new delivery for this customer. The deadline of a demand (order or stockout) is defined as the latest start of a shift that would reach the customer on time to perform the desired delivery, taking travel time and opening hours into account.

Algorithm GREEDY;
Input: an instance of the IRP;
Output: a solution S to the IRP;
Begin;
  $S \leftarrow \emptyset$;
  initialize the set D of demands with orders and stockouts for each customer;
  while D is not empty **do**
    pick the demand d with earliest deadline in D;
    create the cheapest delivery o to satisfy d (inside a new shift, possibly);

if o exists **then**
add o to S (with the new shift, if any);
compute the next stockout after $start(o)$ and update the deadline of d accordingly;
else remove d from D;
return S;
End;

At each step of the algorithm, the newly created delivery can be either appended at the end of an existing shift of included in a new shift. In the first case, the extension of a shift can be made impossible due to accessibility or resources constraints (for example, the resulting duration of this shift may extend the maximum allowed amplitude). For each existing shift, this feasibility is tested in constant time. However, inserting a loading operation can be required for refilling the trailer before performing the delivery, in which case all plants are tested. Therefore, this stage runs in $O(SP)$ time, with S the number of shifts returned by the greedy algorithm and P the number of plants. In the second case, all bases and all possible triplets of resources are considered. Here again, all plants are considered if a loading is needed. The worst-case time complexity of this enumeration is in $O(BRP)$, with B the number of bases and R the number of triplets of resources (drivers, tractors, trailers). But in effect, this running time can be reduced by cutting strongly the search tree, in particular once a feasible shift has been found.

The choice of the delivery date impacts the delivered volume (since the available space in the customer tank increases with time) and the availability of resources ("packing" shifts to the left is preferable when possible). Thus, the possible delivery interval is split into two parts: we first consider delivery dates allowing a full-drop delivery with an earliest scheduling strategy and then apply a latest scheduling strategy for other dates. All deliveries considered in these two cases are compared by dividing the cost of the shift (or the increase of the cost of the existing shift) by the quantity of the delivery. If no feasible delivery could be created for solving a stockout, then another search is attempted trying to deliver product to this customer as early as possible after the stockout. Finally, each time a delivery is created, the inventory levels for this customer are updated and its next stockout is set to the first time step under safety level after the start of the created delivery.

By construction, this urgency-based insertion heuristic never backtracks on decisions taken about dates nor quantities. Practically, the running time of this greedy algorithm is about a dozen of seconds (on standard computers), even for the largest instances of our benchmarks. Even if the local-search heuristic described in the next section is able to start from an empty set of shifts, the use of the initial solution obtained by this constructive algorithm yields a significant speed-up in the convergence toward high-quality solutions (in particular, when finding a solution without missed order and stockout is hard).

5. High-performance local search

In this section, the main ingredients of the local-search heuristic are detailed. The exposition follows the three-layers methodology by Estellon et al. (2009) for designing and engineering high-performance local-search algorithms: heuristic & search strategy, transformations, algorithms & implementation.

5.1. Heuristic & search strategy

Below is outlined the skeleton of the whole heuristic, which is a simple first-improvement stochastic descent. We insist on the fact that no metaheuristic is used, avoiding the use of too much tuning parameters. For more details on metaheuristics and their applications in combinatorial optimization, the reader is referred to the book edited by Aarts and Lenstra (1997).

Algorithm STOCHASTIC-DESCENT;
Input: an instance I of the IRP;
Output: a solution S to the IRP;
Begin;
  $S \leftarrow$ GREEDY(I);
Missed order optimization:
  while $MO > 0$ and $timeLimit_{MO}$ is not reached **do**
    choose stochastically a transformation T in the pool $\mathcal{T}_{MO}$;
    evaluate the gain of the application of T to S;
    if the gain is not negative **then** commit T; **else** rollback T;
Stockout optimization:
  while $SO > 0$ and $timeLimit_{SO}$ is not reached **do**
    choose stochastically a transformation T in the pool $\mathcal{T}_{SO}$;
    evaluate the gain of the application of T to S;
    if the gain is not negative **then** commit T; **else** rollback T;
Logistic ratio optimization:
  while $timeLimit_{LR}$ is not reached **do**
    choose stochastically a transformation T in the pool $\mathcal{T}_{LR}$;
    evaluate the gain of the application of T to S;
    if the gain is not negative **then** commit T; **else** rollback T;
  return S;
End;

The heuristic is divided into three optimization phases: the first one (MO) consists in minimizing the cost related to missed orders, the second one (SO) consists in minimizing the cost related to stockouts, and the third one (LR) consists in optimizing the objective related to logistic ratio. In practice, the total execution time is divided as follows: 10 % for MO optimization, 40 % for SO optimization, 50 % for LR optimization. In the same way, the procedure which evaluates the gain of a transformation is staged into three parts (see Figure 3). Note that accepting solutions with equal cost at each optimization phase is crucial for ensuring the diversification of the search and thus the convergence toward high-quality solutions.

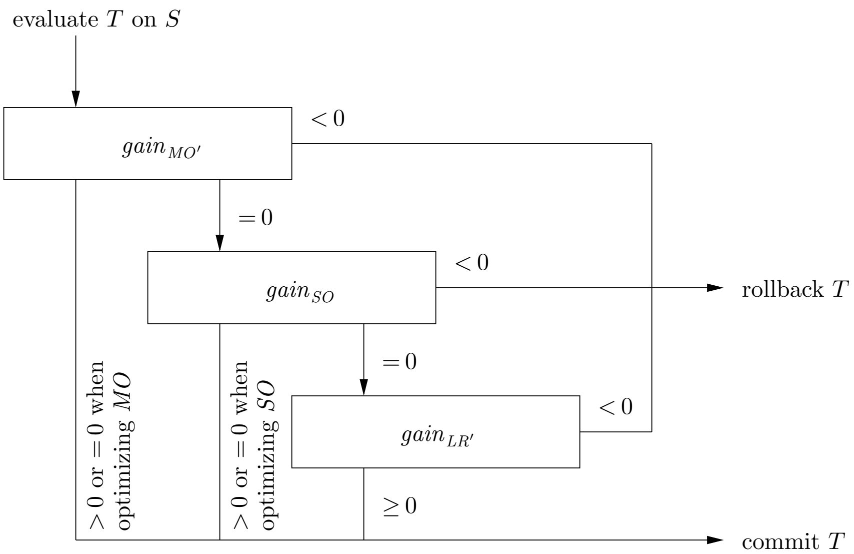

Figure 3 **Evaluation scheme of a transformation.**

Roughly speaking, the gain resulting of the application of a transformation T is the difference between the value of the cost before the application of T to the current solution S (old) and the value of this one after its application (new). Now, we explain how are computed the gains at each stage of the evaluation scheme.

A surrogate cost MO' is defined to smooth the real objective MO for facilitating the convergence of the local search. This is done by introducing for each order an intermediate state called "unsatisfied" between the states "missed" and "satisfied". An order is unsatisfied if an operation exists satisfying the time window of the order, but not its quantity. In this way, an order is satisfied (resp. missed) when both the dates and the quantity are respected by at least one operation (resp. by no operation). Having denoted the number of unsatisfied orders by UO, the value of $gain_{MO'}$ is computed as follows:

$$gain_{MO'} = \begin{cases} MO_{\text{old}} - MO_{\text{new}} & \textit{if } MO_{\text{old}} \neq MO_{\text{new}} \\ UO_{\text{old}} - UO_{\text{new}} & \textit{otherwise} \end{cases}$$

Consequently, a transformation cannot be accepted if either the number of missed orders or the number of unsatisfied orders is deteriorated. Then, the gain related to stockouts is computed as $gain_{SO} = SO_{\text{old}} - SO_{\text{new}}$. Finally, the sign of $gain_{LR'}$ is obtained by evaluating the expression

$$\frac{SC_{\text{old}} - SC^*_{\text{old}}}{DQ_{\text{old}}} - \frac{SC_{\text{new}} - SC^*_{\text{new}}}{DQ_{\text{new}}}$$

or equivalently $DQ_{\text{new}}(SC_{\text{old}} - SC^*_{\text{old}}) - DQ_{\text{old}}(SC_{\text{new}} - SC^*_{\text{new}})$ which avoids imprecisions due to floating-point arithmetic when the expression tends toward zero.

Even if high performance relies on many implementation details, the practical efficiency of the present heuristic relies on two main points: the transformations and the algorithms employed for making their evaluation fast.

5.2. The transformations

The transformations are classified into two categories: the first ones work on operations, the second ones work on shifts. Having introduced the different transformations, their main instantiation shall be described. An instantiation corresponds to the way the objects modified by the transformation are selected. While defining orthogonal transformations (that is, transformations inducing disjoint neighborhoods) enables to diversify the search and then reach better-quality solutions, specializing transformations according to specificities of the problem (because random choices are not the most appropriate in all situations) enables to intensify the search and then speed up the convergence of the heuristic.

The transformations on operations are grouped into the following types: insertion, deletion, ejection, move, swap (see Figure 4). Two kinds of *insertion* are defined: the first kind consists in inserting an operation (pickup or delivery) into an existing shift; the second consists in inserting a pickup followed by a delivery into a shift (the inserted plant is chosen to be one of the nearest from the inserted customer). The *deletion* consists in deleting a block of operations (that is, a set of consecutive operations) in a shift. An *ejection* consists in replacing an existing operation by a new one on a different site. The *move* transformation consists in extracting a block of operations from a shift and reinserting it at another position. Two kinds of moves are defined: moving operations from a shift to another one, or moving operations inside a shift. A *swap* exchanges two different blocks of operations. As for moves, several kinds of swaps are defined: the swap of blocks between shifts, the swap of blocks inside a shift, or the "mirror" which consists in a chronological reversal of a block of operations in a shift. The mirror transformation corresponds to the well-known 2-opt improvement used for solving traveling salesman problems (see Aarts and Lenstra (1997) for more details).

The transformations on shifts are grouped into the following types: insertion, deletion, rolling, move, swap. As for operations, two kinds of *insertion* are defined: insertion of a shift containing one

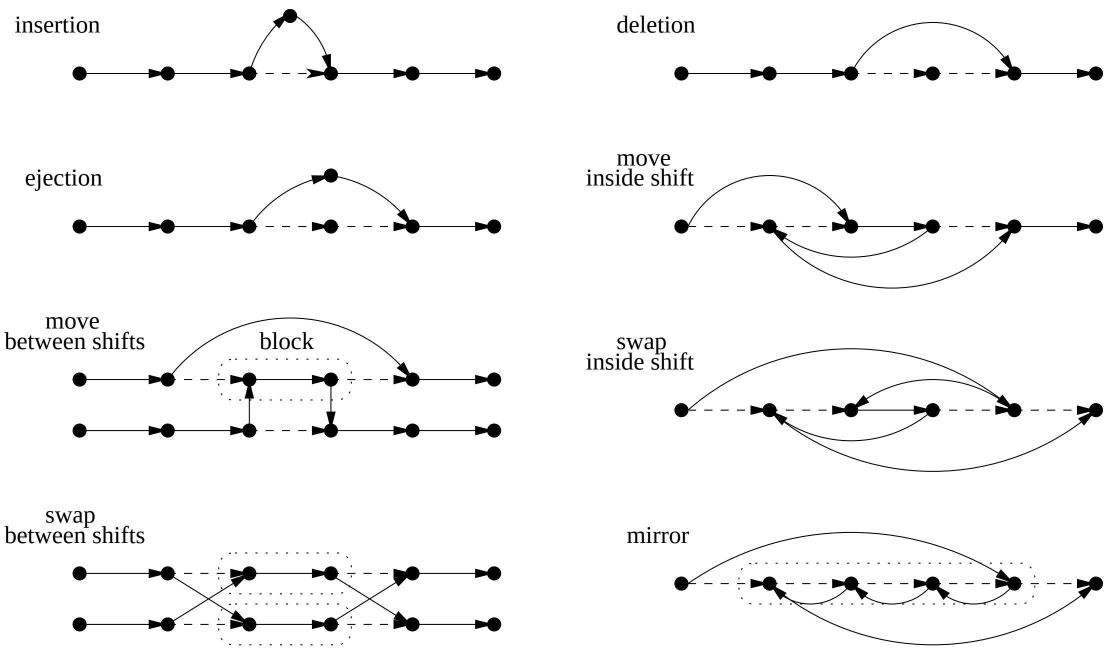

Figure 4 The transformations on operations.

Note. Original tours are given by straight arcs, dashed arcs are removed by the transformation, curved and vertical arcs are added by the transformation.

operation (pickup or delivery), insertion of a shift with a pickup followed by a delivery. *Deletion* consists in removing an existing shift. The *rolling* transformation translates a shift over time. The *move* consists in extracting a shift from the planning of some of its resources and reinserting it into the planning of other ones (such a transformation allows to change some of the ressources of the shift and its starting date). The *swap* is defined similarly: the ressources of the shifts are exchanged and their starting dates can be translated over time. The *fusion* of two shifts into one new shift as well as the *separation* of one shift into two new ones are also available.

Now, these transformations are declined from different ways. The first option concerns the maximal size of blocks for transformations where blocks of operations are involved. In this way, more generic transformations are defined allowing a larger diversification if needed: the (k,l)-ejection which consists in replacing k existing operations by l new ones on different sites, the k-move which consists in moving a block of k operations, the (k,l)-swap which consists in exchanging a block of k operations with a block of l operations, or the k-mirror which consists in reversing a block of k operations.

Then, the second option allows to specialize some transformations when optimizing one of the three objectives. These derivations involve the choice of the sites affected by the transformations. For example, inserting a delivery serving a customer without missed order (resp. stockout) is not interesting when minimizing the number of missed orders (resp. stockouts). In the same way, exchanging two operations which are performed on sites which are very distant is unlikely to succeed when optimizing the logistic ratio. Several derivations have been designed, which differ slightly from one transformation to another. Here are given the three main derivations, essentially used when inserting/ejecting operations or inserting/rolling shifts: "missed order" which positions the delivery so as to (try to) satisfy an order, "stockout" which places the delivery so as to solve a stockout, "nearest" where the customers to insert or exchange are chosen among the nearest ones.

The third option corresponds to the direction used to recompute all the dates of the modified shifts: backward over time by considering the ending date of the shift as fixed, or forward over time by considering its starting date as fixed. This option is available for all transformations, except the deletion of shifts. For the transformations modifying two shifts at once (for example, move operations between shifts), this results in four possible instantiations: backward/backward, backward/forward, forward/backward, forward/forward. Finally, the fourth option allows to augment the number of operations whose quantity is modified during the volume assignment. Recomputing operation quantities during the volume assignment increases its running time but allows repairing stockouts possibly introduced by the transformation, increasing the acceptation rate of the transformations (more details are given in the next section about volume assignment).

The reader shall note that no very large-scale neighborhood is employed. Roughly speaking, the neighborhood explored here has a size $O(n^2)$ with n the number of operations and shifts in the current solution, but the constant hidden by the O notation is large. The number of transformations in $\mathcal{T}_{MO}$, $\mathcal{T}_{SO}$, $\mathcal{T}_{LR}$ used respectively in optimization phases MO, SO, LR are of 47, 49, 71. These ones are exhaustively listed at the end of the paper for the interested reader (Tables 17 and 18). For each optimisation phase, the transformation to apply is chosen randomly with equal probability over all transformations of the pool (improvements being not really significant, further tunings with non-uniform distribution have been abandoned to facilitate maintenance and evolutions).

5.3. Algorithms & implementation

Finally, the kernel of the local search is outlined. Playing a central role in the efficiency of the local-search heuristic, only the evaluation procedure is detailed here. This one is separated into two routines: scheduling shifts, and then assigning volumes. Roughly speaking, the objective of the scheduling routine is to build shifts with smallest costs, whereas the volume assignment tends to maximize the quantity delivered to customers. Even approximately, this leads to minimize the surrogate logistic ratio.

Although conceptually simple, the practical implementation of these routines are considerably complicated by incremental aspects. First, the evaluation is implemented so as to work only on objects (operations, shifts, sites, resources) impacted by the transformation. Besides, all dynamic data associated to these objects are duplicated into backup ones, which correspond to the solution before transformation, and current ones, which correspond to the solution after transformation. This duplication allows to have simpler and faster rollback/commit procedures, whose efficiency is also of importance in the present context.

5.3.1. Scheduling shifts. The transformations modify some shifts in the current solution (at most two actually). When a shift is impacted by a transformation (for example, an operation is inserted into the shift), the starting and ending dates of its operations must be computed anew. Consider the shift $s = (o_1, \ldots, o_n)$ and assume that an operation $\bar{o}$ is inserted into s between operations i and j. The resulting shift $\bar{s}$ is now composed of operations $(o_1, \ldots, o_i, \bar{o}, o_j, \ldots, o_n)$. Then, we have two possibilities: rescheduling dates forward or rescheduling dates backward. The forward (resp. backward) scheduling consists in fixing the ending date of o_i (resp. the starting date of o_j) in order to recompute the starting (resp. ending) dates of $(\bar{o}, \ldots, o_n)$ (resp. $(o_1, \ldots, \bar{o})$). Here computing dates can be done without assigning volumes to operations, because the durations of operations do not depend on delivered/loaded quantities. Since computing dates backward or forward is made completely symmetric by representing shifts with doubly-linked lists, the discussion shall be reduced to the forward case.

More formally, we have to solve the following decision problem, called Shift-Scheduling: given a starting date for the shift $s = (o_1, \ldots, o_n)$, determine the dates of each operation such that the shift is admissible. Two equivalent optimization problems are: having fixed its starting date, build a shift with the earliest ending date or with the minimum cost. A similar problem, called Truckload-Trip-Scheduling, has been recently studied by Archetti and Savelsbergh (2007). This latter problem is more restricted in the sense that only one opening time window is considered for each location to visit and that the rest time must be equal to (not greater than) the legal duration. Archetti and Savelsbergh (2007) sketch a $O(n^2)$-time algorithm for solving the truckload trip scheduling problem, with n the number of locations to visit. For the sake of efficiency, a linear-time and space algorithm has been designed for solving heuristically the SHIFT-SCHEDULING problem.

Algorithm SCHEDULE-SHIFT-GREEDY;
Input: an instance of SHIFT-SCHEDULING;
Output: an admissible shift if any, null otherwise;
Begin;
  define an empty shift;
  for each location to visit **do**
    drive to next location (by taking rests as late as possible if needed);
    if waiting time is needed (due to opening time windows) **then**
      if rest time has been taken on current arc **then**
        lengthen one of the rests to absorb waiting time;
      else if a rest is needed (due to waiting time) or waiting time is larger than rest time **then**
        take a rest (absorbing additional waiting time if any);
      else
        wait for the opening of location;
    perform operation at next location and add it to the shift;
    if maximal amplitude of the shift is exceeded **then** return null (infeasibility);
  return the shift (feasibility);
End;

This algorithm is greedy in the sense that operations are chronologically set without backtracking. Each loop is performed in constant time and space (if rests are not stored explicitly) and the whole algorithm runs in $O(n)$ time and space. The correctness of the algorithm is ensured by construction. The key of the SHIFT-SCHEDULING problem is to minimize unproductive time over the shift. Thereby, the main idea behind the algorithm is to take rests as late as possible during the trip and to avoid waiting time due to opening time windows of locations as much as possible. Here we try to remove waiting time by converting it into rest time (see Figure 5), but only on the current arc, which is suboptimal. Indeed, the algorithm could be reinforced by trying to convert waiting time into rest time on previous arcs (as done by Archetti and Savelsbergh (2007)). But such a modification would lead to a quadratic-time algorithm, which is not desired here, while not guaranteeing optimality because of multiple opening time windows. On the other hand, we have observed that waiting time is rarely generated in practice since many trips are completed in a day or even half a day, ensuring optimality of the algorithm SCHEDULE-SHIFT-GREEDY in most cases. Note that to our knowledge, the complexity of the SHIFT-SCHEDULING problem remains unknown.

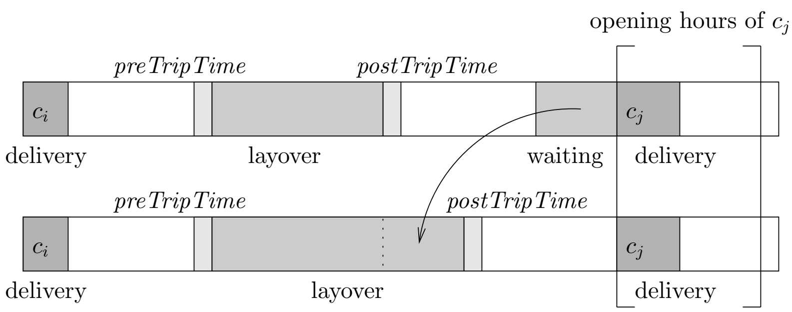

Figure 5 An example with waiting time converted into rest time.

5.3.2. Assigning volumes. Having rescheduled modified shifts, we have to reassign quantities to impacted operations. Having fixed the dates of all operations, the problem consists in assigning volumes such that inventory constraints are respected, while maximizing the total delivered quantity over all shifts. A similar problem, called Delivery-Volume-Optimization, has been addressed by Campbell and Savelsbergh (2004b). In this problem, the authors consider only deliveries on routes and not loadings, but this one is complicated by the fact that the duration of an operation depends on the quantity delivered.

From the theoretical point of view, the present problem, called Volume-Assigning, is not so hard once observed that it can be formulated as a maximum flow problem (in a directed acyclic network). Then, this one can be solved in $O(n^3)$ time by using a classical maximum flow algorithm (Cormen et al. 2004, pp. 625–675), with n the number of operations. As mentioned previously, such a time complexity is not desirable here, even if guaranteeing an optimal volume assignment. Practically, naive implementations having a time complexity depending on the number H of time steps (360 in practice) are prohibited too; indeed, when the granularity becomes smaller than one day, the number of time steps exceeds largely the number of operations at a site (two per day in the worst case).

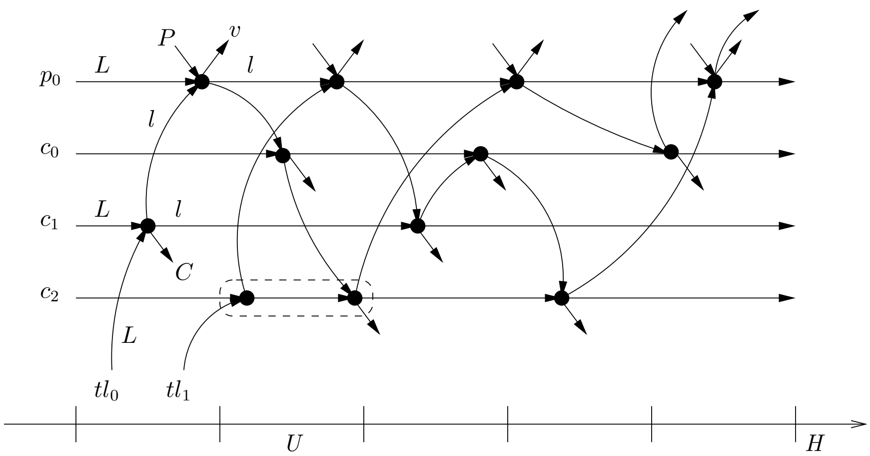

Figure 6 An example of flow network for assigning volumes.

Note. Operations are represented by nodes, input flows L correspond to initial levels for each inventory (trailer, customer, plant), input flow C (resp. P) corresponds to consumption of customer c_1 (resp. production of plant p_0) over the time steps between the current operation and the previous one, flows l correspond to inventory levels (trailer, customer, plant) between two operations, flow v allows an overflow at plant (venting). Flows on arcs representing inventory levels are upper bounded by the capacity of the inventory; for customers, flows are also lower bounded by safety levels. Note that if some consecutive operations appear over the same time step (like the ones dotted around), input flows corresponding to consumption or production are cumulated at the last operation of this time step.

Thus, a $O(n \log n)$-time greedy algorithm has been designed to solve approximately the Volume-Assigning problem. The main idea behind the algorithm is simple: having ordered operations chronologically (that is, according to increasing starting dates), quantities are assigned to operations in this order following a greedy rule. Here we use the basic rule consisting in maximizing the quantity delivered/loaded at each operation, which is a good policy for minimizing the surrogate logistic ratio (this joins the ideas developed by Campbell and Savelsbergh (2004b)). Note that the chronological ordering is crucial for ensuring the respect of constraints related to inventory dynamics (flow conservation, capacity constraints). In graph-theoretical terms, the algorithm consists in pushing flow in the induced directed acyclic network following a topological order of the nodes (ensuring that no node is visited twice).

Because the number of operations may be large (as worst case in practice, one can imagine that the 1500 customers must be delivered two times per day, leading to $n = 45\ 000$), a tradeoff must be found between the time complexity (even linear) and the quality of the volumes assignment. To introduce flexibility on this point, the greedy algorithm has been designed for computing partial reassignments, from the minimal one to the complete one. The minimal reassignment consists in changing only the volumes on impacted operations (that is, operations whose starting dates are modified by the transformation); then, it suffices to tag as impacted some additional operations to expand the reassignment. This complicates notably the practical implementation of the greedy algorithm. Indeed, changing the quantity delivered at an operation is delicate since increasing (resp. decreasing) the quantity may imply overflows (resp. stockouts) at future operations. Then, determining the (maximum) quantity to deliver/load at each operation is not straightforward.

For each site p, denote by $\bar{n}_p$ the number of operations between the first impacted operation (that is, whose quantity can be modified by the transformation) in the chronological ordering and the last one over the horizon. If no operation is impacted at site p, then $\bar{n}_p = 0$. Hence, we define $\bar{n} = \sum_p \bar{n}_p$. When the set of impacted operations consist only in operations whose dates are modified by the transformation, one can observe in practice that $\bar{n} \ll n$, since each transformation modify at most two shifts (the number of sites visited by one shift is generally small). Consequently, it is important to provide algorithms whose running time is linear in $O(\bar{n})$, and not only in $O(n)$. Below is outlined an $O(\bar{n}\log\bar{n})$-time algorithm for assigning volumes. But before, more explanations are given on how the maximum deliverable quantity is computed (the maximum loadable quantity can be obtained in a symmetric way).

Denote by $\textit{customerLevel}(c,i)$ (resp. $\textit{trailerLevel}(r,i)$) the level of customer c (resp. trailer r) before starting the operation i and by $\textit{avoidOverflow}(c,i)$ the maximum quantity that can be delivered to customer c at operation i without inducing overflows until the end of the horizon. In this way, the deliverable quantity at operation i, denoted by $\textit{deliverable}(i)$, is upper bounded by $\min\{\textit{trailerLevel}(r,i), \textit{avoidOverflow}(c,i)\}$. Then, this bound is reinforced in such a way that the quantity remaining in the trailer after a delivery is sufficient to avoid stockouts at customers visited by the shift until the next loading. Denote by $\textit{avoidStockout}(c,i)$ the minimum quantity to deliver at operation i to avoid stockout until the end of the horizon. Now, the minimum quantity $\textit{neededAfter}(r,i)$ which must remain in the trailer r after operation i to avoid stockout later is obtained by summing $\textit{avoidStockout}(c,i)$ for all operations between the current one and the next loading. Then, we have

$$\textit{deliverable}(i) \leq \min\{\textit{trailerLevel}(r,i) - \textit{neededAfter}(r,i), \textit{avoidOverflow}(c,i)\}$$

Given the chronological-ordered list of operations for each trailer, customer and plant, all the data structures mentioned above can be computed in $O(\bar{n})$ time. Updating $\textit{customerLevel}(c,i)$ (resp. $\textit{trailerLevel}(r,i)$) for any operation i is done by sweeping forward the operations delivering customer c (resp. performed by the trailer r). Then, updating $\textit{avoidOverflow}(c,i)$ and $\textit{avoidStockout}(c,i)$ for any operation i is done by sweeping backward the operations performed at customer c (note that the consumption between two operations is obtained in constant time by storing cumulated consumptions over the horizon). Finally, computing $\textit{neededAfter}(r,i)$ for any operation i is done by sweeping backward the operations of shifts performed by r. Below is given a sketch of algorithm.

Algorithm Assign-Volumes-Greedy;
Input: the set $\mathcal{E}$ of $\bar{n}$ impacted operations;
Begin;
  sort the set $\mathcal{E}$ chronologically;
  update *customerLevel*, *trailerLevel*, *avoidOverflow*, *avoidStockout*, *neededAfter*;
  for each operation in $\mathcal{E}$ **do**

assign the maximum deliverable/loadable quantity to the operation;
End;

According to the previous discussion, the five data structures which serve to the calculation of the maximum deliverable/loadable quantity are updated in $O(\bar{n})$ time. Then, sorting the set $\mathcal{E}$ is done in $O(\bar{n}\log\bar{n})$ time in the worst case; in effect, the heapsort algorithm is used (see Cormen et al. (2004, pp. 121–137)). Finally, the loop is done in linear time, since the calculation of the maximum deliverable/loadable quantity requires only constant time by using the adequate data structures. Consequently, the whole algorithm runs in $O(\bar{n}\log\bar{n})$ time.

In theory, the greedy algorithm is far from being optimal. Figure 7 gives the smallest configuration for which the greedy algorithm fails to find an optimal assignment. On the other hand, two sufficient conditions hold for which the greedy assignment is optimal. The proofs are not detailed here because easy to establish. The first condition corresponds to the case where each customer is served at most once over the planning horizon. This condition is interesting because likely to be met in practice. The second condition corresponds to the case where each shift visits only one customer. For example, this condition is satisfied when customers have infinite storage capacity.

Experiments have been made for evaluating the practical performance of this critical routine. In practice, its running time is shown to be constant with respect of the total number of operations: it is 100 times faster than the full application of the greedy algorithm (that is, considering that all operations are impacted, implying that $\bar{n}=n$) and 2000 times faster than exact algorithms (tests have been realized with the simplex algorithm of the linear programming library GLPK 4.24). On the other hand, the total volume delivered by the routine is close to the optimal assignment, in particular when no stockout appears (the average gap between the greedy assignment and an optimal one is lower than 2 %).

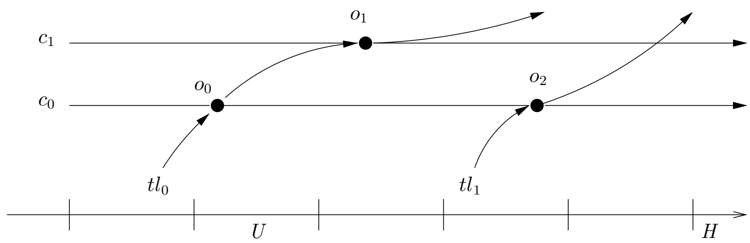

Figure 7 Bad configuration for the greedy volume assignment.

Finally, having assigned volumes, computing the gain of the transformation is done efficiently. The (variation of) cost of shifts is computed during the scheduling, and the (variation of) total delivered quantity is obtained during the assignment of volumes, without increasing the complexity of the algorithms. The (variation of) missed orders and stockouts costs are also computed during the assignment of volumes. Note that computing the number of time steps in stockout between two consecutive operations requires $O(\log H)$ time, with H the number of time steps over the horizon, since it is equivalent to the problem of searching the zero of a discrete non-increasing function.

5.3.3. Implementation details. All sets (unordered or ordered, fixed or dynamic) are implemented as arrays, in order to improve the cache memory locality. Memory allocation is avoided as much as possible during local-search iterations: all the data structures are allocated before starting the local search; an array of capacity n representing a dynamic list is extended if necessary by reallocating a larger block of memory of size $n+k$ (with $k\approx 10$).

Since the success rate of the transformations is low on average (a few percents), the rollback routine must be very efficient. In this way, the decision variables of the problem (for example, the starting and ending dates of an operation) are duplicated in such a way that only temporary data are modified by the transformation. In this way, the rollback routine consists simply in overwriting temporary data by current ones (that is, corresponding to the current solution). But this is complicated by the fact that during one transformation, several objects (in particular operations or shifts) are likely to move in the arrays in which they are stored. In order to ensure the (temporary) insertion or deletion of one object in $O(1)$ time, the objects of the array are doubly linked (see Figure 8 which illustrates the exchange of operations between shifts).

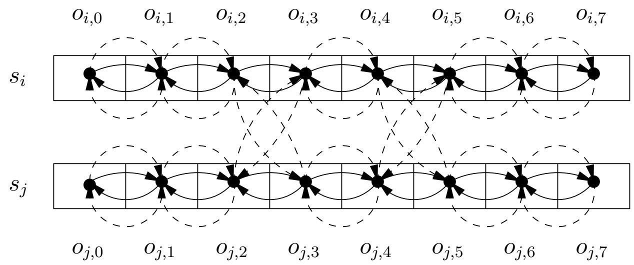

Figure 8 Representation of shifts and operations.

Note. Operations $o_{i,3}, o_{i,4}$ of the shift s_i are swapped with operations $o_{j,3}, o_{j,4}$ of the shift s_j: the current links (before transformation) are plain, the temporary links (after transformation) are dashed.

The main data structures are designed to support basic routines (find, insert, delete, clear) in $O(1)$ time, even if they are implemented as arrays. An example is the classical data structure used to implement an unordered list of objects. For example, this one is used to store the customers experiencing stockouts in the current solution or the missed orders for each customer. The unordered list is implemented as an array L, with $L.size$ the current number of elements in L and $L.capacity$ the capacity of L (that is, the maximum number of elements which can be stored in the list without exceeding allocated memory). Any element e stored in L has a pointer $e.indexL$ to its position in the array L. If the maximal number of elements e stored in L is known a priori and is not too large (a few thousand), then L can be allocated with a capacity equal to this number (to avoid frequent memory allocations), otherwise the extension of L is done by increasing its capacity by $L.increase$ elements when needed. Classically, the routines Find, Insert, Delete, and Clear are implemented as follows to run in $O(1)$ time (note that the elements in L are indexed from 0):

Algorithm Find;
Input: the array L, the element e to find;
Output: true if e belongs to L, false otherwise;
Begin;
  $i = e.indexL$;
  if $i \geq 0$ and $i < L.size$ and $L[i] = e$ **then return** true;
  return false;
End;

Algorithm Insert;
Input: the array L, the element e to insert;
Begin;
  if Find(L,e) **then return**;
  if $L.size = L.capacity$ **then**;
    $L.capacity = L.capacity + L.increase$;
    reallocate L with the new $L.capacity$;
  $L[L.size] = e$, $e.indexL = L.size$, $L.size = L.size + 1$;
End;

Algorithm DELETE;
Input: the array L, the element e to delete;
Begin;
  if not FIND(L,e) **then return**;
  $i = e.indexL$, $e' = L[L.size - 1]$;
  $L[i] = e'$, $e'.indexL = i$, $L.size = L.size - 1$;
End;

Algorithm CLEAR;
Input: the array L;
Begin;
  $L.size = 0$;
End;

To end the section, we focus on a data structure which is particularly critical for efficiency. Operations or shifts correspond to intervals over the horizon, for which we have the following need: given a date over the horizon, find the previous, current, or next operation performed at a site if any. The same problem arises for situating shifts performed by a resource. For this, the following data structure is employed. Assume that the n operations are stored into an ordered list L. The horizon is divided into m intervals $U_0, \ldots, U_{m-1}$ of given length u (with u dividing T). Then, an array I is defined such that I_i refers to the first operation whose starting date is larger than the left endpoint of U_i. The next operation after date d is found by searching the operations between the one pointed by I_i with $i = \lfloor d/u \rfloor$ and the one pointed by I_{i+1}. In this way, the search is done in $O(k)$ time in the worst case, with k the number of operations contained in the interval U_i. If u corresponds to the entire horizon ($m = 1$), then $k = n$; on the other hand, if u corresponds to the smallest granularity for expressing time (here the minute, leading to $m = 21600$), then $k = 1$. Assuming that starting dates of operations performed at a site are uniformly distributed over the horizon, the number k is equal to n/m. In this case, searching takes $O(n/m)$ time (when evaluating the transformation), but the array I requires $O(m)$ space to be stored and $O(m)$ time to be updated (when committing the transformation). This implies two compromises: time to evaluate vs. time to commit, time to evaluate vs. space.

Theoretically, the best value m^* for solving the compromise on running time corresponds to the minimum of the function $T(m) = E(N/m) + Cm$, with N the average number of operations per customer and E (resp. C) a coefficient relative to the proportion of calls of the evaluate routine (resp. commit routine) per customer. A simple calculation using differentiation yields $m^* = \sqrt{(EN)/C}$. Table 1 shows the values of m^* for different realistic configurations of parameters N, E, C. In practice, we have chosen $m^* = 15$, which corresponds to interval U_i of one day: such a value of m offers a good compromise for running time (even in worst-case situations) and leads to a small memory footprint.

Table 1 Theoretical values of m^*.

N	2	2	2	10	10	10	30	30	30
E	90	95	99	90	95	99	90	95	99
C	10	5	1	10	5	1	10	5	1
m^*	4.2	6.2	14.1	9.5	13.8	31.5	16.4	23.9	54.5

6. Computational experiments

The whole algorithm was implemented in C# 2.0 programming language (for running on Microsoft .NET 2.0 framework). The resulting program includes nearly 30 000 lines of code, whose 6 000 lines (20 %) are dedicated to check the validity of all incremental data structures at each iteration (only active in debug mode). The whole project (specifications, implementation, tests), realized during the year 2008, required nearly 300 man-days. All statistics and results presented here have been obtained (without parallelization) on a computer equipped with a Windows Vista operating system and a chipset Intel Xeon X5365 64 bits (CPU 3 GHz, L1 cache 64 Kio, L2 cache 4 Mio, RAM 8 Go). The interested reader is invited to contact the authors to obtain some benchmarks to work on this problem. Note that the urgency-based constructive heuristic used to compute an initial solution is also called "greedy algorithm" below.

Since the local-search heuristic is stochastic, 5 runs have been performed with different seeds for each benchmark. Except contrary mention, all the statistics presented below correspond to average results obtained for these 5 runs. Note results requiring particular explanations are marked with asterisks (*) in figures presenting numerical experiments; these explanations could be found in the text below.

The analysis of IRP solutions was facilitated by the use of a visualization tool, developed specifically for the project. This tool allows to visualize shifts from several points of view, as well as sites' inventories. Figures 10, 11, 12 give an overview of this visualization tool.

6.1. Short-term benchmarks

The local-search algorithm has been extensively tested on short-term benchmarks (15 days) with different characteristics: realistic (that is, matching the operational conditions), pathological (for example, with plants whose production is stopped several days), large-scale (for example, with 1 500 sites and 300 resources). Some results are presented for 61 short-term benchmarks decomposed into 3 kinds A, B, C. Table 2 gives the characteristics of each instance: the number of customers, the number of plants, the number of bases, the number of drivers, the number of tractors, the number of trailers, the number of call-in customers (that is, customers in pure "order-based resupply" management), the number of orders over the short-term horizon. When the number of call-in customers is zero whereas the number of orders is not zero, this means that all the orders are asked by customers which share the two modes of replenishment (forecasting-based and order-based). Benchmarks A and B include no orders (all customers are in pure "forecasting-based resupply" mode), whereas benchmarks C include some customers asking orders; the base A contains "easy" instances, that is, instances for which our greedy algorithm finds a solution without missed order and stockout. The results obtained by the greedy algorithm on benchmarks A, B, C are shown on Tables 3 and 4. Two kinds of results are presented for the local-search heuristic: the results obtained by optimizing directly the logistic ratio LR (denoted by LS-LR and reported on Tables 5 and 6) and the ones obtained by optimizing the surrogate ratio LR' (denoted by LS-LR' and reported on Tables 7 and 8). In order to compare solutions with missed orders and stockouts, a global cost $GC = MO + SO + LR$ is introduced.

The reader shall note the following remarkable instances: A11, B01, B28, B29, B30, and particularly B31 are classified as large-scale instances (more than 500 customers); B28 and B29 are some instances where the production has been stopped at plants (no product is loadable); B30 contains many customers in stockout at the beginning of the horizon; the customers defined in instances C03 and C04 have tight opening hours (for example, a customer can be open only during 6 hours over the 15 days); almost all the customers of C12 are call-in customers (implying more than 600 orders to satisfy).

The running time of the greedy algorithm is of the order of few seconds for large-scale instances (12 seconds for instance B31). Statistics about the performance of the local search are given on Table 9. The column "attempt" corresponds to the number of transformations attempted by the local-search heuristic. The columns "accept" (resp. "improve") corresponds to the number of accepted (resp. strictly improving) transformations; in addition, the corresponding rate for 100 (resp. 10 000) attempted transformations is specified. Note that average values given at the bottom of the table are calculated by omitting exceptional results obtained on C04 instance (tight opening hours cause early rejections, resulting in more attempts). The local-search algorithm attempts more than 10 000 transformations per second, even for large-scale instances. On average, our algorithm visits more than 10 million solutions in the search space during 5 minutes of running time (which is the desired time limit in operational conditions). When planning over a 15-days horizon, the memory allocated by the program does not exceed 30 Mo for medium-size instances (hundred sites, ten resources), and 300 Mo for large-scale instances (thousand sites, hundred resources). The acceptance rate, which corresponds to the number of accepted transformations (that is, transformations not strictly improving the current solution) over the number of attempted ones, varies essentially between 1 and 10 %, with an average value of 5 % over all the instances of A, B, C. Note that this rate is quasi constant all along the search (that is, during the 5 minutes of running time), allowing a large diversification of the search (without the use of metaheuristics). On the other hand, the number of strictly improving transformations is of several hundreds, which corresponds to a rate of nearly 2 improvements for 10 000 attempts. One can observe that the choice of objective (LR or LR') does not affect the performance of the local-search heuristic.

The column "gain GC" (resp. "gain LR", "gain LR'") on Tables 5 and 7 reports the gain for GC (resp. LR, LR') in comparison to the solution found by the greedy algorithm. The gain for GC allows to evaluate in a global way the gain on MO and SO. Note that the gain for LR can be negative when the minimization of MO and SO imposes to deteriorate sharply the quality of the solution found by the greedy algorithm in term of logistic ratio. In both cases (LS-LR or LS-LR'), the local-search heuristic improves drastically the quality of solutions provided by the greedy algorithm. On instances of base A (for which the comparison between the greedy algorithm and the local search is fair concerning the logistic ratio), the average gain obtained by LS-LR (resp. LS-LR') is of 29.2 % (resp. 22.6 %), and of 24.1 % (resp. 20.0 %) by considering the global logistic ratio (that is, the sum of shift costs for all instances divided by the sum of delivered quantities for all instances). This last measure is interesting but not completely fair, because all costs are not always expressed according to the same currency unit; on the other hand, quantities are always expressed in kilograms here. On Table 7, one can observe that gains on LR' are correlated to gains on LR; but important gains on LR' are necessary to obtain some gains on LR comparable to the ones obtained by LS-LR. A possible explanation is that the initial solutions computed by the greedy algorithm are less optimized for LR' than for LR (indeed, the goal of the greedy algorithm is just to resolve missed orders and stockouts while minimizing SC).

Tables 6 and 8 gives more statistics on the solutions found by local search. The column "nb shift" (resp. "nb oper") of the tables reports the number of shifts (resp. operations) of the solution. The column "avg oper" (resp. "avg deliv", "avg load", "avg layov") reports the average number of operations (resp. deliveries, loadings, layovers) per shift. Finally, the column "avg dur" (resp. "avg dist") reports the average traveled distance (resp. duration) per shift. One can observe that more shifts and much more operations are included in both LS-LR and LS-LR' solutions. On average, the number of shifts (resp. operations) is increased of nearly 25 % (resp. 50 %). In this way, the average number of operations per shift is increased from almost 4 to more than 6. The average distance and duration of shifts are decreased slightly in the case of LS-LR, whereas these ones are increased slightly in the case of LS-LR'.

6.2. Long-term benchmarks

The local-search algorithm has been also tested on long-term benchmarks, in particular for verifying that optimizing the surrogate objective leads to better solutions on the long run. Some results are presented for 5 real-life benchmarks, each one with 105 days. The operational planning process is simulated as follows. The simulator starts at day 0 by computing a planning over the next 15 days, with 5 minutes as time limit. Then, only the shifts starting at the first day of this short-term planning are fixed (the levels of plants or customers visited by these shifts are updated, the resources operating on these shifts become unavailable) and the process is iterated the following day.

The characteristics of these 5 benchmarks are presented on Table 11. The long-term solutions found by the three heuristics (greedy, LS-LR, LS-LR') include no missed orders. The complete statistics are given for each heuristic on Tables 12, 13, and 14. The main remark is that the average number of operations per shift is increased in local-search solutions. Indeed, the number of shifts in local-search solutions is slightly smaller than in greedy solutions, whereas the number of operations is increased. Besides, local-search solutions are characterized by a larger average traveled duration per shift, thanks to an increased use of layovers. The average logistic ratios marked by an asterisk are computed as the sum of shift costs for the 5 benchmarks divided by the sum of all delivered quantities.

The gains obtained by LS-LR' are reported on the right part of Table 11. The column "wst 1 mn" reports the worst LR gain in % obtained over the 5 runs for 1 minute of running time per planning iteration. The column "avg 1 mn" (resp. "avg 5 mn", "avg 1 h") reports the average gain in % for LR obtained by local search limited to 1 minute (resp. 5 minutes, 1 hour) of computation per planning iteration. Note that the solutions found by the greedy algorithm include some stockouts. On average, the LR gain obtained by LS-LR' with only 1 minute of running time per planning iteration is of nearly 20 % on average. More than providing high-quality solutions, these statistics demonstrate that our local-search heuristic is robust and fast (with an exponential-inverse convergence).

Solutions provided by logistic experts are reported on Table 16. Note that the comparison between the experts and the three heuristics is not completely fair. Indeed, because finding solutions with no stockout (with actual safety levels) was difficult and fastidious, experts were allowed to modify the initial long-term benchmarks in order to provide solutions without missed order and stockout. This could explain the negative gain of greedy algorithm on instances L1, L2, L4. Moreover, the solution provided by experts for instance L2 is considered as a "best-effort" solution, in the sense that many more time has been spent to optimize carefully the solution.

6.3. Impact of the surrogate objective

The key figures for comparing LS-LR and LS-LR' are given on Table 10 (for short-term benchmarks) and on the right part of Table 15 (for long-term benchmarks). "avg DQ" corresponds to the average total delivered quantity, and "avg delivq" to the average delivered quantity (per operation). On short-term benchmarks, the total delivered quantity in both LS-LR and LS-LR' solutions is increased by more than 50 % on average. But note that the average quantity per delivery is increased by almost 5 % in LS-LR' solutions compared to LS-LR solutions.

On long-term benchmarks, one shall observe that LS-LR' aims at increasing the average quantity per delivery, which results in better long-term solutions. On instances L1 (resp. L2), the augmentation is of 21 % (resp. 13 %), leading to +8 % (resp. +5 %) of gain for LR. Moreover, LS-LR' is able to produce solutions without stockout on the instance L4, contrary to LS-LR. The values marked by an asterisk on Tables 15 and 16 are computed globally, for the 5 benchmarks (as done for logistic ratios on Tables 13 and 14).

The tables on the left part of Tables 3 and 15 gives lower bounds for SO and LR. The computation of the lower bound $SO_{\min}$ for the number of stockouts is based on two basic observations. Some stockouts appear at a customer if: the earliest date of arrival at the customer during opening hours is greater than the date of the first stockout; the sum of consumptions of the customer during

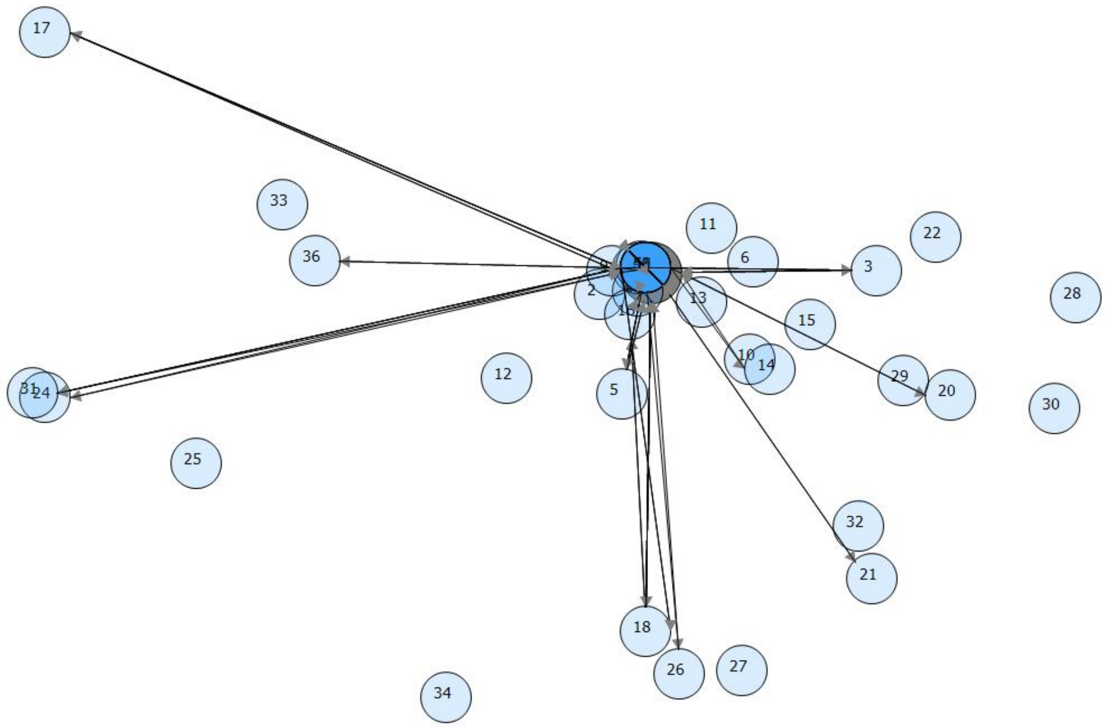

Figure 9 **A near-optimal IRP solution, with a majority of round-trips with full drops.**

closed hours exceeds the capacity of its tank. The computation of $LR_{\min}$ is done as described in Section 3.1. On Tables 3, 5, 7, an asterisk is set in the column SO if the value equals the lower bound $SO_{\min}$.

The logistic ratio obtained after local-search optimization remains far from the lower bound $LR_{\min}$ (the gap is lower than 10 % for only 3 instances B06, B20, B25). Figure 9 shows a near-optimal solution obtained on instance B06, with a majority of round-trips with full drops. Nevertheless, the reader shall note that this bound has been obtained by relaxing strongly the constraints of the problem (researches are planned in order to reinforce this lower bound).

7. Conclusion

Having introduced a real-life IRP problem encountered in a worldwide industry, two contributions have been presented in this paper. First, a surrogate objective based on local lower bounds was defined for ensuring a long-term optimization when building a planning over the short term. Then, a local-search heuristic has been described for solving effectively and efficiently the real-life IRP over the short term (15 days in full details), even when some large-scale instances (thousand sites, hundred resources) are considered. An extensive computational study shows that our solution yields long-term savings exceeding 20 % on average compared to solutions built by expert planners or even a classical urgency-based constructive algorithm. Since the promised long-term savings have been confirmed in operations, a decision support system integrating this high-performance local-search heuristic is going to be deployed worldwide.

New researches are still conducted in several directions:

- enlarging the scope of the IRP problem addressed in the paper (in particular, refining costs and managing drivers' desiderata);
- improving the existing local-search heuristic (adding transformations with larger neighborhoods for speeding up convergence);
- reinforcing the global lower bound by integrating tours visiting several sites and constraints on resources.

Another prospective, but promising, line of research is to proceed step by step toward a global optimization of the supply chain, by tackling jointly the production and distribution problems and finally by integrating purchasing issues. Indeed, we think that local-search approaches like the one developed presently are best suited for solving such very large-scale problems.

References

Aarts, E., J. Lenstra, eds. 1997. *Local Search in Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, England.

Archetti, C., M. Savelsbergh. 2007. The truckload trip scheduling problem. *TRISTAN VI, the 6th Triennial Symposium on Transportation Analysis*. Phuket Island, Thailand.

Bell, W., L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, P. Prutzman. 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* **13**(6) 4–23.

Benoist, T., B. Estellon, F. Gardi, A. Jeanjean. 2009. High-performance local search for solving inventory routing problems. H. Hoos T. Stützle, M. Birattari, ed., *SLS 2009, the 2nd International Workshop on Engineering Stochastic Local Search Algorithms*, *Lecture Notes in Computer Science*, vol. 5752. Springer, 105–109.

Campbell, A., L. Clarke, A. Kleywegt, M. Savelsbergh. 1998. The inventory routing problem. T. Crainic, G. Laporte, eds., *Fleet Management and Logistics*. Kluwer Academic Publishers, Norwell, MA, 95–113.

Campbell, A., L. Clarke, M. Savelsbergh. 2002. Inventory routing in practice. P. Toth, D. Viego, eds., *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications 9, SIAM, Philadelphia, PA, 309–330.

Campbell, A., M. Savelsbergh. 2004a. A decomposition approach for the inventory-routing problem. *Transportation Sci.* **38**(4) 488–502.

Campbell, A., M. Savelsbergh. 2004b. Delivery volume optimization. *Transportation Sci.* **38**(2) 210–223.

Campbell, A., M. Savelsbergh. 2004c. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Sci.* **38**(3) 369–378.

Cormen, T., C. Leiserson, R. Rivest, C. Stein. 2004. *Introduction à l'Algorithmique*. Dunod, Paris, France. French 2nd edition.

Estellon, B., F. Gardi, K. Nouioua. 2006. Large neighborhood improvements for solving car sequencing problems. *RAIRO Operations Research* **40**(4) 355–379.

Estellon, B., F. Gardi, K. Nouioua. 2008. Two local search approaches for solving real-life car sequencing problems. *Eur. J. Oper. Res.* **191**(3) 928–944.

Estellon, B., F. Gardi, K. Nouioua. 2009. High-performance local search for task scheduling with human resource allocation. H. Hoos T. Stützle, M. Birattari, ed., *SLS 2009, the 2nd International Workshop on Engineering Stochastic Local Search Algorithms*, *Lecture Notes in Computer Science*, vol. 5752. Springer, 1–15.

Feo, T., G. Resende. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization* **6**(2) 109–133.

Lau, H., Q. Liu, H. Ono. 2002. Integrating local search and network flow to solve the inventory routing problem. *AAAI 2002, the 18th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, CA, 9–14.

Savelsbergh, M., J.-H. Song. 2007a. Inventory routing with continuous moves. *Computers and Operations Research* **34**(6) 1744–1763.

Savelsbergh, M., J.-H. Song. 2007b. Performance measurement for inventory routing. *Transportation Sci.* **41**(1) 44–54.

Savelsbergh, M., J.-H. Song. 2008. An optimization algorithm for the inventory routing with continuous moves. *Computers and Operations Research* **35**(7) 2266–2282.

Solomon, M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35**(2) 254–265.

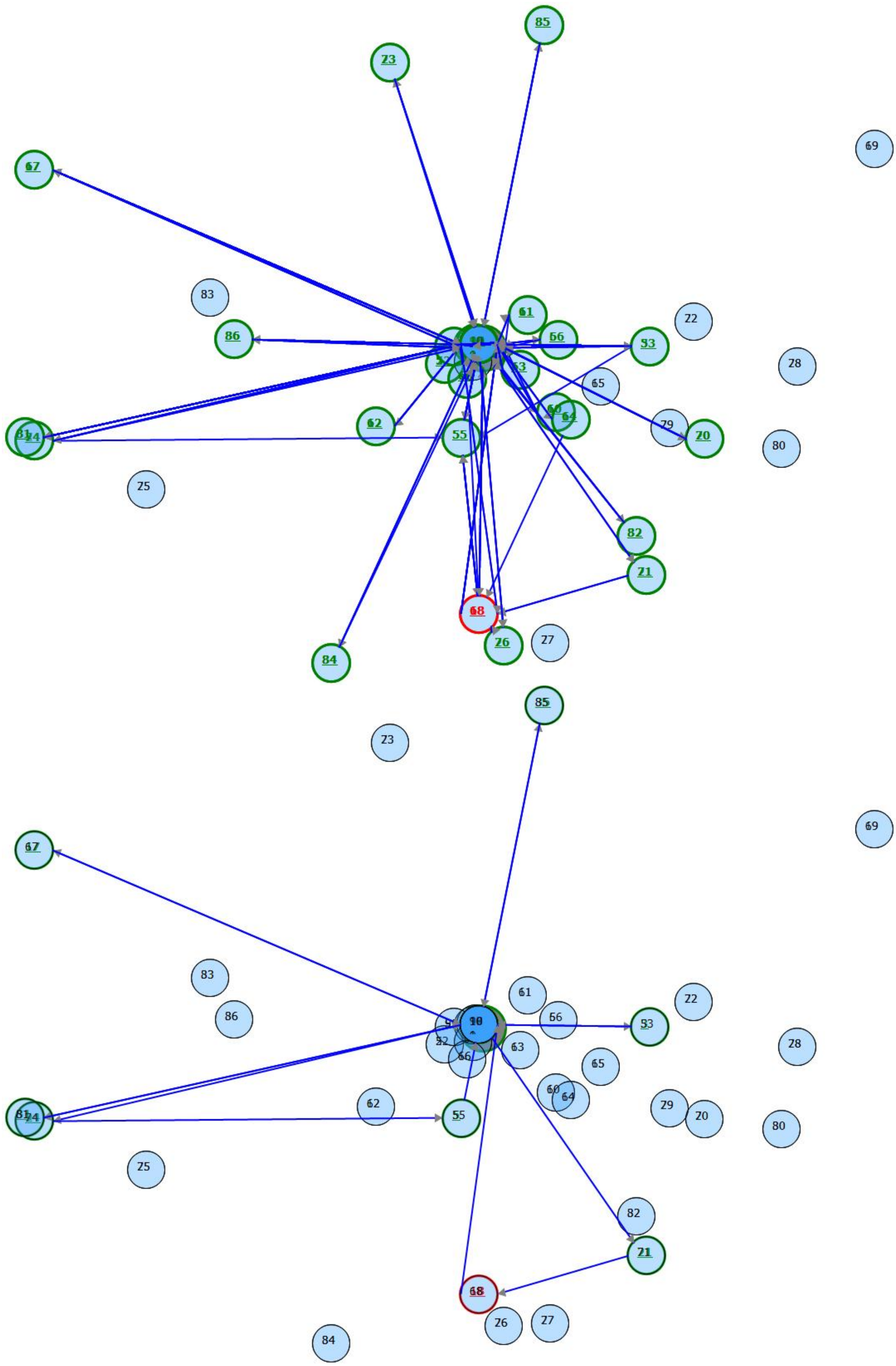

Figure 10 **The geographical view of an IRP solution (top) and of a "long" shift (18 operations: 10 deliveries, 8 loadings, 2 layovers) built by local search (bottom). Note that the customer 68 has stockouts.**

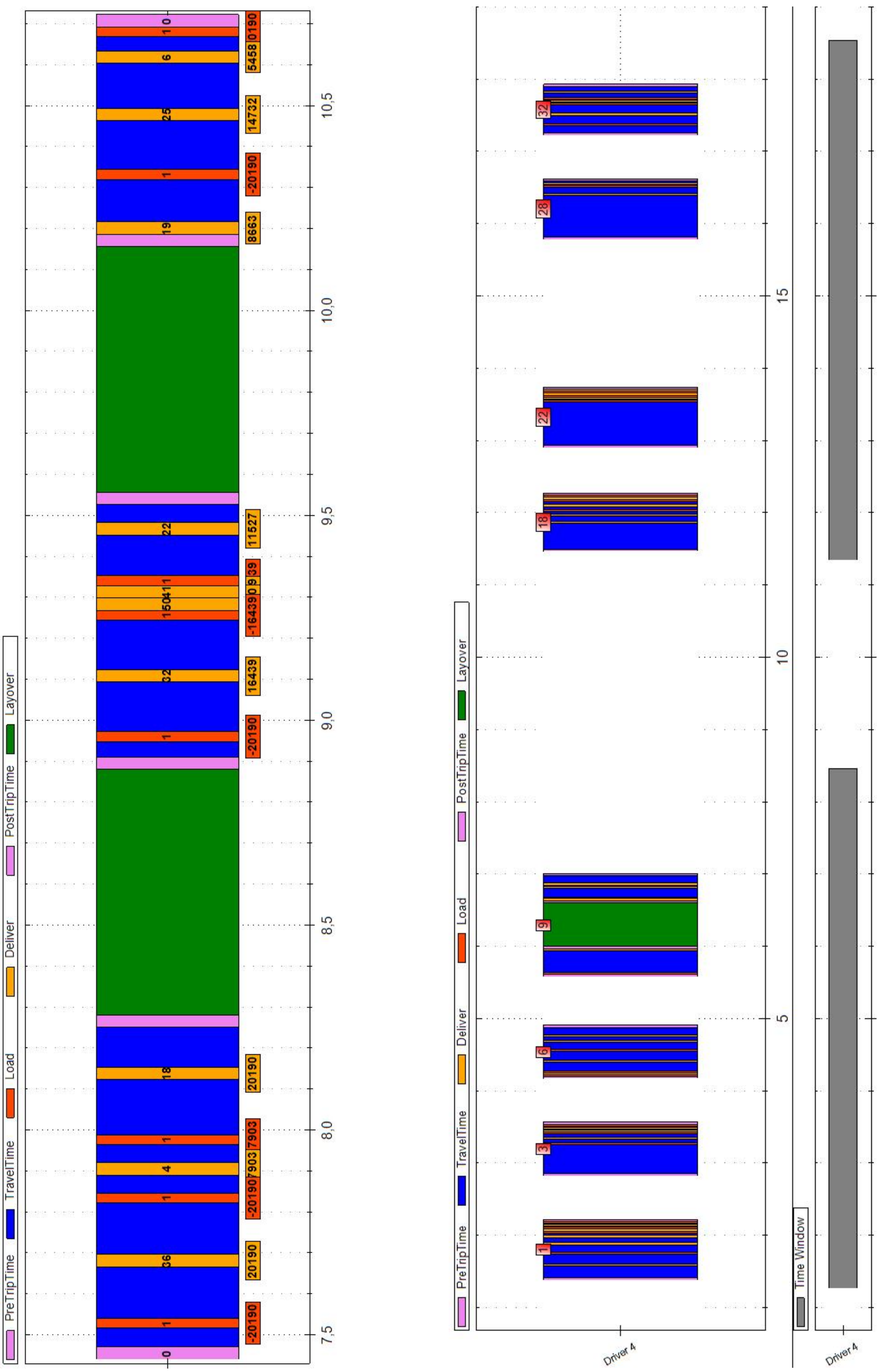

Figure 11 **The chronological view of the long shift of Figure 10 (left) and of the set of shifts performed by a driver (right). Note that the second layover of the long shift was anticipated for waiting the opening hours of a customer.**

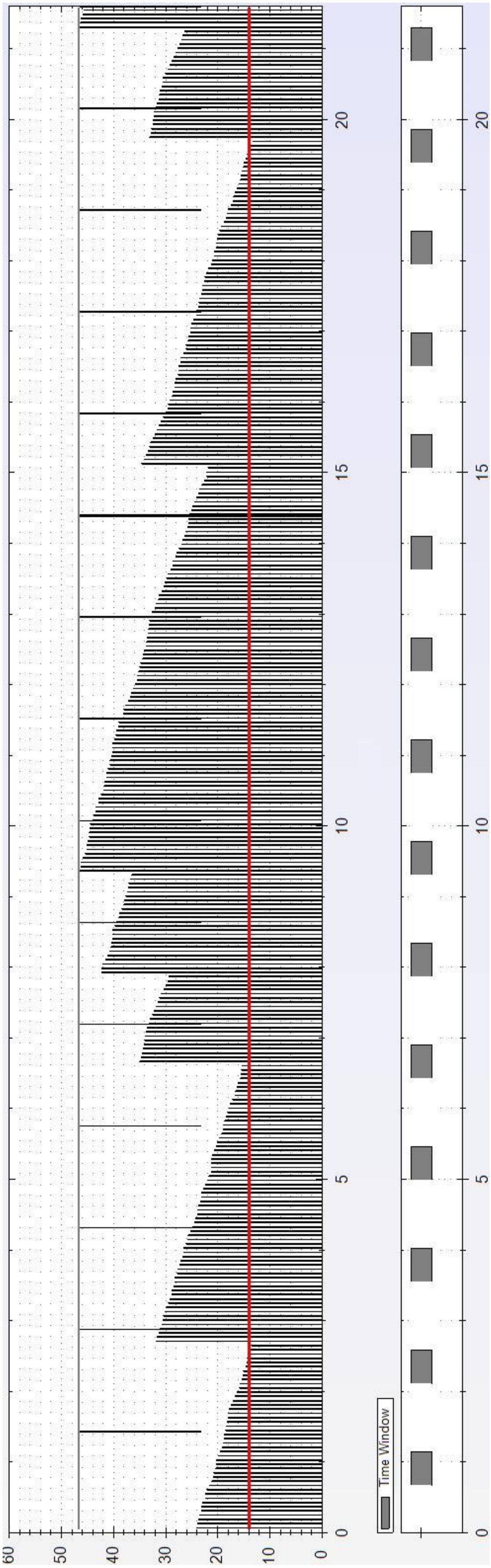

Figure 12 The inventory level at a customer over 15 days.

Note. The safety level is marked by the horizontal red line, and the days of the horizon by the vertical black lines. All deliveries, which are marked by a rise in the inventory level, appear well during opening hours.

Table 2 Short-term benchmarks: characteristics and greedy results (statistics on costs).

data	customers	plants	bases	drivers	tractors	trailers	callins	orders
A01	80	2	2	20	10	20	0	0
A02	108	1	1	35	18	35	0	0
A03	132	1	1	20	17	15	0	0
A04	130	2	1	17	10	20	0	0
A05	125	2	1	20	18	20	0	0
A06	46	1	2	50	50	50	0	0
A07	80	2	2	20	10	20	0	0
A08	75	1	1	10	10	10	0	0
A09	150	2	1	20	20	20	0	0
A10	250	5	1	30	30	30	0	0
A11	500	4	2	50	20	50	0	0
A12	108	1	1	35	18	32	0	0
A13	100	1	1	35	35	35	0	0
A14	70	1	1	50	5	10	0	0
A15	132	1	1	20	17	15	0	0
A16	130	2	1	17	10	20	0	0
A17	135	3	1	20	18	20	0	0
B01	500	4	2	50	40	50	0	0
B02	200	1	1	13	9	12	0	0
B03	100	1	1	35	35	35	0	0
B04	70	1	1	10	5	10	0	0
B05	135	3	1	20	18	20	0	0
B06	50	1	1	5	5	5	0	0
B07	200	1	1	13	9	12	0	0
B08	100	2	1	20	15	15	0	0
B09	124	2	1	20	18	20	0	0
B10	99	3	2	20	14	30	0	0
B11	75	1	1	10	10	10	0	0
B12	75	1	1	10	10	10	0	0
B13	75	1	1	10	10	10	0	0
B14	75	1	1	10	10	10	0	0
B15	198	3	1	10	10	8	0	0
B16	198	3	1	10	10	8	0	0
B17	198	3	1	10	10	8	0	0
B18	198	3	1	10	10	8	0	0
B19	50	1	1	5	5	5	0	0
B20	50	1	1	10	10	10	0	0
B21	50	1	1	5	5	5	0	0
B22	50	1	1	5	5	5	0	0
B23	50	1	1	5	5	5	0	0
B24	50	1	1	5	5	5	0	0
B25	50	1	1	5	5	5	0	0
B26	20	1	1	1	1	1	0	0
B27	99	3	1	20	20	20	0	0
B28	500	1	3	60	60	60	50	0
B29	500	1	3	60	60	60	50	0
B30	783	16	4	78	49	42	191	0
B31	1 500	50	50	100	100	100	0	0
C01	75	6	1	35	21	3	0	9
C02	75	6	1	34	21	3	0	4
C03	75	6	1	35	21	5	20	9
C04	75	6	1	35	21	5	20	2
C05	75	6	1	35	21	5	19	6
C06	75	6	1	35	21	5	19	6
C07	122	1	1	6	6	6	0	10
C08	122	1	1	6	6	6	0	15
C09	175	8	1	35	21	12	38	20
C10	175	8	1	31	21	12	38	28
C11	215	1	5	18	15	34	48	25
C12	272	17	9	57	27	27	265	616
C13	100	1	1	50	50	50	0	41
average	171	4	2	25	19	20	-	-

data	$SO_{\min}$	$LR_{\min}$
A01	0	0.014 630
A02	0	0.009 551
A03	0	0.013 192
A04	0	0.039 239
A05	0	0.043 731
A06	0	0.031 722
A07	0	0.015 470
A08	0	0.194 255
A09	0	0.194 276
A10	0	0.147 122
A11	0	0.032 681
A12	0	0.010 978
A13	0	0.013 444
A14	0	0.016 950
A15	0	0.009 364
A16	0	0.048 491
A17	0	0.032 739
B01	45	0.006 406
B02	0	0.012 301
B03	16	0.009 767
B04	10	0.006 043
B05	21	0.040 597
B06	7	0.014 001
B07	0	0.009 816
B08	297	0.050 168
B09	7	0.047 395
B10	3	0.028 094
B11	0	0.157 607
B12	0	0.159 432
B13	12	0.159 432
B14	26	0.159 362
B15	0	0.060 412
B16	0	0.061 183
B17	0	0.060 412
B18	0	0.061 692
B19	278	0.006 026
B20	23	0.196 649
B21	0	0.036 080
B22	0	4.302 140
B23	0	0.036 080
B24	0	0.036 080
B25	0	2.895 112
B26	4	0.022 161
B27	0	0.078 626
B28	37	0.013 677
B29	37	0.013 677
B30	968	0.004 329
B31	47	0.057 477
C01	57	0.445 947
C02	57	0.411 399
C03	0	0.038 796
C04	37	0.027 532
C05	0	0.041 293
C06	0	0.042 624
C07	0	0.024 255
C08	140	0.023 984
C09	13	0.217 204
C10	13	0.223 018
C11	0	0.005 883
C12	0	0.900 584
C13	0	0.035 972

data	*MO*	*SO*	*SC*	*DQ*	*LR*
A01	-	0	57 178	2 317 462	0.024 673
A02	-	0	64 472	3 534 883	0.018 239
A03	-	0	77 651	3 531 172	0.021 990
A04	-	0	202 829	2 951 746	0.068 715
A05	-	0	199 190	2 173 870	0.091 629
A06	-	0	172 410	1 095 217	0.157 421
A07	-	0	45 092	1 602 654	0.028 136
A08	-	0	307 783	875 106	0.351 710
A09	-	0	788 566	2 100 705	0.375 382
A10	-	0	1 310 183	3 758 893	0.348 556
A11	-	0	595 733	9 410 181	0.063 307
A12	-	0	89 696	5 020 209	0.017 867
A13	-	0	64 197	2 429 315	0.026 426
A14	-	0	41 885	1 468 461	0.028 523
A15	-	0	54 049	2 945 628	0.018 349
A16	-	0	376 060	3 070 358	0.122 481
A17	-	0	334 667	3 247 942	0.103 040
B01	-	51	196 307	13 909 999	0.014 113
B02	-	196	131 242	3 905 673	0.033 603
B03	-	*16	160 038	13 809 196	0.011 589
B04	-	*10	23 082	1 379 000	0.016 738
B05	-	*25	291 179	3 583 893	0.081 247
B06	-	10	16 638	509 922	0.032 629
B07	-	58	84 933	3 565 791	0.023 819
B08	-	8 312	475 665	5 595 653	0.085 006
B09	-	55	362 308	3 227 554	0.112 255
B10	-	*3	117 994	1 361 526	0.086 663
B11	-	3	293 817	742 473	0.395 728
B12	-	7	237 479	588 415	0.403 590
B13	-	*12	277 024	586 285	0.472 508
B14	-	69	315 868	769 653	0.410 404
B15	-	7 600	471 325	1 619 909	0.290 958
B16	-	4 356	505 750	2 322 558	0.217 756
B17	-	8 802	405 570	1 213 195	0.334 299
B18	-	9 458	418 735	1 315 944	0.318 201
B19	-	6 971	10 822	46 100	0.234 751
B20	-	1 244	292 095	685 681	0.425 993
B21	-	262	55 080	631 287	0.087 250
B22	-	262	6 557 325	609 803	10.753 186
B23	-	357	60 865	612 436	0.099 382
B24	-	258	98 020	605 441	0.161 899
B25	-	265	3 723 050	572 758	6.500 215
B26	-	1 088	23 470	328 763	0.071 389
B27	-	927	529 644	1 038 549	0.509 985
B28	-	46 322	155 168	1 986 943	0.078 094
B29	-	46 298	149 518	1 986 944	0.075 250
B30	-	74 094	124 298	9 880 399	0.012 580
B31	-	*47	4 206 864	33 035 582	0.127 343
C01	6	2 698	414 736	379 165	1.093 814
C02	3	819	258 656	251 252	1.029 469
C03	0	309	25 570	240 408	0.106 362
C04	0	*37	1 464	17 827	0.082 148
C05	0	31	30 147	244 708	0.123 194
C06	0	41	22 566	225 504	0.100 069
C07	0	1 789	366 799	214 009	1.713 942
C08	0	5 271	281 550	205 571	1.369 601
C09	0	1 290	1 043 204	1 972 743	0.528 809
C10	3	4 085	1 496 640	2 488 223	0.601 490
C11	1	10 449	24 271	1 317 292	0.018 425
C12	95	14	3 226 530	2 036 466	1.584 377
C13	4	5 318	221 965	3 610 200	0.061 483

Table 3 Short-term benchmarks: lower bounds (left) and greedy results (right).

Table 4 Short-term benchmarks: greedy results (statistics on shifts).

data	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
A01	98	433	2.4	1.4	1.1	0.2	179	316
A02	76	481	4.3	2.4	1.9	0.6	254	540
A03	86	573	4.7	2.5	2.1	0.0	273	415
A04	46	370	6.0	3.3	2.8	0.1	501	598
A05	51	335	4.6	2.6	2.0	0.1	417	549
A06	87	413	2.7	2.2	0.6	0.2	929	857
A07	61	316	3.2	1.8	1.4	0.0	227	381
A08	25	159	4.4	2.2	2.1	0.2	493	651
A09	47	341	5.3	2.8	2.4	0.3	699	920
A10	70	626	6.9	3.8	3.2	0.5	933	1 332
A11	116	1 183	8.2	4.5	3.7	0.2	545	776
A12	69	632	7.2	3.8	3.4	0.0	391	581
A13	66	451	4.8	3.0	1.8	0.0	430	542
A14	53	273	3.2	1.7	1.5	0.0	244	362
A15	109	570	3.2	1.8	1.5	0.0	146	272
A16	54	433	6.0	3.4	2.6	0.6	810	1 300
A17	52	463	6.9	3.8	3.1	0.5	720	1 113
B01	335	2 084	4.2	2.3	1.9	0.7	167	644
B02	125	780	4.2	2.4	1.8	0.6	321	594
B03	112	1 839	14.4	7.5	6.9	0.0	550	864
B04	48	246	3.1	1.7	1.4	0.0	141	259
B05	65	500	5.7	3.2	2.5	0.1	482	699
B06	23	105	2.6	1.4	1.1	0.0	225	312
B07	49	588	10.0	5.7	4.3	0.2	517	834
B08	151	958	4.3	2.3	2.0	0.0	245	380
B09	52	446	6.6	3.7	2.8	0.6	788	1 312
B10	46	226	2.9	2.0	0.9	0.1	400	475
B11	24	149	4.2	2.2	2.0	0.2	453	704
B12	22	127	3.8	2.0	1.8	0.1	425	574
B13	24	133	3.5	2.0	1.6	0.2	407	695
B14	29	175	4.0	2.3	1.7	0.3	398	770
B15	76	357	2.7	1.5	1.2	0.3	587	810
B16	79	446	3.6	1.9	1.7	0.3	602	817
B17	63	284	2.5	1.4	1.1	0.3	617	824
B18	78	313	2.0	1.1	1.0	0.2	498	729
B19	16	64	2.0	1.0	1.0	0.2	182	586
B20	65	691	8.6	4.3	4.3	0.4	239	925
B21	15	109	5.3	2.9	2.3	0.3	259	713
B22	12	97	6.1	3.4	2.7	0.4	319	887
B23	16	114	5.1	2.8	2.3	0.3	254	639
B24	16	108	4.8	2.6	2.2	0.4	238	802
B25	15	101	4.7	2.7	2.1	0.4	245	884
B26	18	71	1.9	1.0	0.9	0.1	36	254
B27	75	351	2.7	1.9	0.8	0.8	400	1 849
B28	63	207	1.3	1.3	0.0	0.0	398	458
B29	62	206	1.3	1.3	0.0	0.0	389	408
B30	307	1 184	1.9	1.0	0.8	0.0	197	417
B31	360	3 975	9.0	5.8	3.3	0.1	474	566
C01	28	112	2.0	1.1	0.9	0.3	452	833
C02	20	77	1.9	1.1	0.8	0.2	387	696
C03	20	78	1.9	1.2	0.8	0.2	379	705
C04	2	8	1.5	1.5	0.0	0.5	196	1 847
C05	14	75	2.7	1.7	1.0	0.7	638	1 356
C06	13	55	2.2	1.4	0.8	0.5	517	1 078
C07	63	252	2.0	1.0	1.0	0.2	260	577
C08	51	204	2.0	1.0	1.0	0.3	254	639
C09	84	406	2.8	1.6	1.2	0.2	298	683
C10	106	524	2.9	1.7	1.2	0.2	364	786
C11	58	279	2.8	1.8	1.0	0.0	348	568
C12	252	1 355	3.4	2.1	1.3	0.0	150	645
C13	70	450	4.4	2.6	1.8	0.0	321	433
average	72	475	4.2	2.4	1.8	0.2	397	722

Table 5 Short-term benchmarks: LS-LR results (statistics on costs).

data	MO	SO	SC	DQ	LR	gain GC	gain LR
A01	-	0	61 347	3 140 722	0.019 533	20.8 %	20.8 %
A02	-	0	67 627	5 389 262	0.012 548	31.2 %	31.2 %
A03	-	0	75 443	4 617 278	0.016 339	25.7 %	25.7 %
A04	-	0	214 050	4 255 173	0.050 303	26.8 %	26.8 %
A05	-	0	187 720	3 130 162	0.059 971	34.5 %	34.5 %
A06	-	0	184 128	1 634 342	0.112 662	28.4 %	28.4 %
A07	-	0	49 354	2 357 186	0.020 938	25.6 %	25.6 %
A08	-	0	338 629	1 582 160	0.214 029	39.1 %	39.1 %
A09	-	0	981 784	3 666 094	0.267 801	28.7 %	28.7 %
A10	-	0	1 600 926	6 656 780	0.240 496	31.0 %	31.0 %
A11	-	0	729 420	15 765 833	0.046 266	26.9 %	26.9 %
A12	-	0	93 166	7 084 289	0.013 151	26.4 %	26.4 %
A13	-	0	61 572	3 012 796	0.020 437	22.7 %	22.7 %
A14	-	0	37 755	1 963 049	0.019 233	32.6 %	32.6 %
A15	-	0	47 438	3 824 474	0.012 404	32.4 %	32.4 %
A16	-	0	376 239	4 398 543	0.085 537	30.2 %	30.2 %
A17	-	0	356 575	5 135 668	0.069 431	32.6 %	32.6 %
B01	-	48	208 995	17 147 972	0.012 188	5.9 %	13.6 %
B02	-	69	132 251	4 654 644	0.028 413	64.7 %	15.4 %
B03	-	*16	169 664	15 643 641	0.010 846	0.0 %	6.4 %
B04	-	*10	17 700	1 853 051	0.009 552	0.7 %	42.9 %
B05	-	*21	353 487	5 243 479	0.067 415	16.0 %	17.0 %
B06	-	10	14 348	958 772	0.014 965	13.3 %	54.1 %
B07	-	0	93 094	5 720 379	0.016 274	97.3 %	31.7 %
B08	-	1 152	690 985	9 136 192	0.075 632	86.1 %	11.0 %
B09	-	*7	390 235	4 661 929	0.083 707	76.8 %	25.4 %
B10	-	*3	120 401	2 367 933	0.050 846	30.7 %	41.3 %
B11	-	3	293 645	1 392 672	0.210 850	43.4 %	46.7 %
B12	-	7	245 273	1 217 519	0.201 453	42.7 %	50.1 %
B13	-	*12	264 905	1 300 288	0.203 728	45.4 %	56.9 %
B14	-	*26	346 686	1 592 901	0.217 645	56.6 %	47.0 %
B15	-	1 378	557 875	4 586 864	0.121 624	81.8 %	58.2 %
B16	-	834	535 275	4 374 700	0.122 357	80.7 %	43.8 %
B17	-	1 537	580 345	4 607 614	0.125 953	82.5 %	62.3 %
B18	-	2 580	522 365	3 664 172	0.142 560	72.7 %	55.2 %
B19	-	6 472	6 773	46 400	0.145 970	7.2 %	37.8 %
B20	-	374	392 495	1 872 138	0.209 651	69.9 %	50.8 %
B21	-	0	47 685	1 131 970	0.042 126	100.0 %	51.7 %
B22	-	0	6 438 345	1 157 627	5.561 675	84.9 %	48.3 %
B23	-	0	52 320	1 139 022	0.045 934	99.9 %	53.8 %
B24	-	0	126 960	1 204 818	0.105 377	99.6 %	34.9 %
B25	-	0	3 197 695	1 078 086	2.966 085	91.0 %	54.4 %
B26	-	137	24 405	816 473	0.029 891	87.2 %	58.1 %
B27	-	0	545 077	1 506 760	0.361 754	100.0 %	29.1 %
B28	-	28 834	336 092	1 986 943	0.169 150	37.8 %	-116.6 %
B29	-	28 892	353 603	1 986 944	0.177 963	37.6 %	-136.5 %
B30	-	15 659	141 026	15 978 096	0.008 826	78.9 %	29.8 %
B31	-	*47	5 016 050	46 282 014	0.108 380	0.4 %	14.9 %
C01	3	530	639 297	563 692	1.134 125	72.3 %	-3.7 %
C02	3	74	233 825	322 556	0.724 912	63.5 %	29.6 %
C03	0	0	33 215	533 379	0.062 273	99.8 %	41.5 %
C04	0	*37	2 901	87 268	0.033 239	1.3 %	59.5 %
C05	0	0	29 851	505 713	0.059 028	98.2 %	52.1 %
C06	0	0	24 280	466 894	0.052 003	98.8 %	48.0 %
C07	0	0	319 365	341 328	0.935 655	95.2 %	45.4 %
C08	0	2 407	266 655	350 180	0.761 480	54.1 %	44.4 %
C09	0	33	1 285 520	3 469 495	0.370 521	94.8 %	29.9 %
C10	0	149	2 063 977	5 304 062	0.389 131	95.8 %	35.3 %
C11	1	235	53 706	3 827 663	0.014 031	97.7 %	23.8 %
C12	89	0	3 020 317	2 114 443	1.428 422	6.5 %	9.8 %
C13	1	7	372 072	6 513 105	0.057 127	98.0 %	7.1 %

Table 6 Short-term benchmarks: LS-LR results (statistics on shifts).

data	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
A01	115	565	2.9	1.7	1.2	0.2	159	298
A02	135	790	3.9	2.1	1.7	0.2	143	342
A03	133	830	4.2	2.4	1.8	0.0	163	341
A04	54	503	7.3	4.1	3.3	0.0	416	528
A05	53	404	5.6	3.3	2.4	0.1	348	513
A06	102	523	3.1	2.4	0.7	0.1	795	751
A07	78	449	3.8	2.3	1.5	0.0	188	394
A08	25	254	8.2	4.4	3.8	0.1	490	677
A09	55	546	7.9	4.3	3.6	0.3	712	975
A10	83	994	10.0	5.3	4.6	0.5	895	1 360
A11	153	1 865	10.2	5.4	4.8	0.2	456	744
A12	138	988	5.2	2.7	2.4	0.0	194	380
A13	77	548	5.1	3.2	1.9	0.0	327	466
A14	81	398	2.9	1.6	1.3	0.0	138	302
A15	157	821	3.2	1.8	1.4	0.0	81	246
A16	61	586	7.6	4.4	3.2	0.5	673	1 172
A17	60	659	9.0	5.0	4.0	0.4	617	1 008
B01	386	2 506	4.5	2.4	2.1	0.7	150	638
B02	164	986	4.0	2.3	1.7	0.4	242	488
B03	154	2 140	11.9	6.2	5.7	0.0	403	685
B04	69	357	3.2	1.8	1.4	0.0	66	228
B05	89	736	6.3	3.6	2.6	0.3	388	739
B06	35	188	3.4	2.0	1.4	0.0	116	295
B07	82	993	10.1	5.8	4.3	0.2	317	700
B08	229	1 534	4.7	2.5	2.2	0.0	234	364
B09	58	607	8.5	4.8	3.7	0.5	736	1 193
B10	53	353	4.7	2.9	1.8	0.1	315	462
B11	25	237	7.5	4.1	3.4	0.1	398	596
B12	24	209	6.7	3.7	3.0	0.0	335	487
B13	24	224	7.3	4.0	3.3	0.0	365	525
B14	32	271	6.5	3.7	2.8	0.2	354	664
B15	84	773	7.2	4.1	3.1	0.2	600	843
B16	91	760	6.4	3.6	2.8	0.2	525	739
B17	86	790	7.2	4.1	3.1	0.2	612	847
B18	72	647	7.0	4.1	2.9	0.3	669	908
B19	13	61	2.7	1.5	1.2	0.2	199	420
B20	71	1 580	20.3	10.1	10.1	0.3	274	985
B21	11	161	12.6	7.0	5.6	0.2	283	712
B22	9	156	15.3	8.3	7.0	0.3	378	916
B23	11	164	12.9	7.0	5.9	0.3	294	790
B24	12	165	11.8	6.3	5.5	0.2	297	679
B25	16	189	9.8	6.2	3.6	0.6	197	1 086
B26	7	111	13.9	7.4	6.4	0.0	162	528
B27	78	439	3.6	2.7	0.9	0.7	346	1 937
B28	102	570	3.6	3.6	0.0	0.2	492	666
B29	101	591	3.9	3.7	0.1	0.3	511	795
B30	300	1 529	3.1	2.0	1.1	0.0	206	500
B31	531	5 402	8.2	5.0	3.2	0.1	358	461
C01	29	173	4.0	2.7	1.3	0.7	649	1 451
C02	14	81	3.8	2.6	1.1	0.4	464	981
C03	24	115	2.8	1.8	1.0	0.3	397	935
C04	6	23	1.7	1.3	0.3	0.0	129	268
C05	21	120	3.3	2.2	1.0	0.4	401	934
C06	19	100	3.2	2.2	1.0	0.3	357	862
C07	65	354	3.4	1.7	1.7	0.1	360	585
C08	43	265	4.2	2.1	2.0	0.2	398	771
C09	99	588	3.9	2.4	1.6	0.2	254	737
C10	156	926	3.9	2.4	1.5	0.1	273	690
C11	114	704	4.2	2.7	1.5	0.1	383	738
C12	256	1 389	3.4	2.2	1.2	0.0	138	706
C13	215	1 051	2.9	1.7	1.2	0.0	144	282
average	93	706	6.3	3.6	2.7	0.2	360	694

Table 7 Short-term benchmarks: LS-LR' results (statistics on costs).

data	MO	SO	SC	DQ	LR	gain GC	gain LR'	gain LR
A01	-	0	72 227	3 392 822	0.021 288	13.7 %	37.3 %	13.7 %
A02	-	0	72 213	5 383 578	0.013 414	26.5 %	50.1 %	26.5 %
A03	-	0	96 672	5 112 877	0.018 908	14.0 %	28.5 %	14.0 %
A04	-	0	231 448	4 299 676	0.053 829	21.7 %	51.1 %	21.7 %
A05	-	0	205 122	3 218 972	0.063 723	30.5 %	65.6 %	30.5 %
A06	-	0	191 645	1 647 962	0.116 292	26.1 %	33.5 %	26.1 %
A07	-	0	55 130	2 465 328	0.022 362	20.5 %	47.7 %	20.5 %
A08	-	0	346 527	1 397 463	0.247 969	29.5 %	51.9 %	29.5 %
A09	-	0	1 055 789	3 906 338	0.270 276	28.0 %	54.5 %	28.0 %
A10	-	0	1 881 203	7 387 093	0.254 661	26.9 %	49.8 %	26.9 %
A11	-	0	817 071	16 925 051	0.048 276	23.7 %	51.2 %	23.7 %
A12	-	0	116 102	7 513 201	0.015 453	13.5 %	33.4 %	13.5 %
A13	-	0	67 251	3 174 802	0.021 183	19.8 %	41.1 %	19.8 %
A14	-	0	56 302	2 317 519	0.024 294	14.8 %	40.7 %	14.8 %
A15	-	0	68 054	4 395 307	0.015 483	15.6 %	36.8 %	15.6 %
A16	-	0	423 711	4 807 925	0.088 128	28.0 %	49.7 %	28.0 %
A17	-	0	377 020	5 298 991	0.071 149	30.9 %	48.7 %	30.9 %
B01	-	49	204 693	15 950 475	0.012 833	3.9 %	16.3 %	9.1 %
B02	-	70	132 788	4 660 904	0.028 490	64.3 %	21.3 %	15.2 %
B03	-	*16	169 280	15 211 761	0.011 128	0.0 %	15.9 %	4.0 %
B04	-	*10	32 526	2 084 416	0.015 604	0.2 %	43.9 %	6.8 %
B05	-	*21	398 876	5 606 842	0.071 141	16.1 %	24.7 %	12.4 %
B06	-	10	27 538	1 301 185	0.021 164	5.2 %	66.3 %	35.1 %
B07	-	0	103 538	5 930 657	0.017 458	99.1 %	41.7 %	26.7 %
B08	-	1 172	687 755	9 030 922	0.076 156	85.9 %	22.7 %	10.4 %
B09	-	*7	431 889	4 937 110	0.087 478	82.9 %	41.3 %	22.1 %
B10	-	*3	143 157	2 613 767	0.054 770	37.5 %	59.1 %	36.8 %
B11	-	3	365 400	1 550 112	0.235 725	52.7 %	62.6 %	40.4 %
B12	-	7	279 055	1 217 253	0.229 250	43.4 %	62.5 %	43.2 %
B13	-	*12	347 743	1 499 034	0.231 978	50.8 %	76.8 %	50.9 %
B14	-	*26	458 784	1 906 122	0.240 690	63.6 %	68.1 %	41.4 %
B15	-	1 549	600 530	4 900 575	0.122 543	79.6 %	71.8 %	57.9 %
B16	-	841	555 560	4 406 363	0.126 081	80.6 %	60.8 %	42.1 %
B17	-	1 508	572 385	4 583 682	0.124 875	82.8 %	75.6 %	62.6 %
B18	-	2 648	511 925	3 639 688	0.140 651	72.0 %	68.4 %	55.8 %
B19	-	6 472	7 028	46 400	0.151 466	7.2 %	35.3 %	35.5 %
B20	-	374	356 665	1 595 901	0.223 488	69.9 %	66.7 %	47.5 %
B21	-	0	64 315	1 253 926	0.051 291	100.0 %	72.2 %	41.2 %
B22	-	0	8 364 065	1 342 321	6.231 047	95.3 %	70.0 %	42.1 %
B23	-	0	62 915	1 175 160	0.053 537	100.0 %	75.4 %	46.1 %
B24	-	0	157 870	1 380 915	0.114 323	99.7 %	39.0 %	29.4 %
B25	-	0	6 190 560	1 491 728	4.149 925	99.3 %	83.0 %	36.2 %
B26	-	137	31 785	847 726	0.037 494	87.4 %	83.1 %	47.5 %
B27	-	0	555 017	1 530 477	0.362 643	100.0 %	34.2 %	28.9 %
B28	-	28 846	344 463	1 986 943	0.173 363	37.7 %	-147.5 %	-122.0 %
B29	-	28 831	338 685	1 986 944	0.170 455	37.7 %	-156.5 %	-126.5 %
B30	-	15 354	154 313	16 925 694	0.009 117	79.3 %	37.5 %	27.5 %
B31	-	*47	5 036 543	45 573 555	0.110 515	0.3 %	20.0 %	13.2 %
C01	3	611	650 584	562 753	1.156 073	70.8 %	-22.8 %	-5.7 %
C02	3	82	264 592	303 151	0.872 806	63.7 %	14.7 %	15.2 %
C03	0	0	37 177	511 392	0.072 698	99.9 %	61.7 %	31.7 %
C04	0	*37	13 535	199 939	0.067 696	1.1 %	83.8 %	17.6 %
C05	0	0	44 035	594 459	0.074 075	99.7 %	80.3 %	39.9 %
C06	0	0	42 185	556 139	0.075 854	99.8 %	78.3 %	24.2 %
C07	0	0	350 576	348 089	1.007 144	95.0 %	41.9 %	41.2 %
C08	0	2 383	267 431	350 192	0.763 669	54.6 %	45.2 %	44.2 %
C09	0	33	1 275 703	3 418 723	0.373 152	96.4 %	49.3 %	29.4 %
C10	0	147	1 989 237	5 155 110	0.385 877	96.3 %	54.1 %	35.8 %
C11	1	340	52 607	3 513 443	0.014 973	96.7 %	41.5 %	18.7 %
C12	89	0	3 009 688	2 104 364	1.430 213	6.5 %	10.5 %	9.7 %
C13	1	9	420 256	7 116 669	0.059 052	98.1 %	-8.3 %	4.0 %

Table 8 Short-term benchmarks: LS-LR' results (statistics on shifts).

data	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
A01	128	599	2.7	1.5	1.2	0.1	171	286
A02	103	709	4.9	2.6	2.3	0.4	203	457
A03	117	799	4.8	2.6	2.2	0.0	246	413
A04	46	474	8.3	4.4	3.9	0.1	557	694
A05	46	391	6.5	3.8	2.7	0.1	460	670
A06	101	521	3.2	2.4	0.8	0.1	847	797
A07	88	465	3.3	1.9	1.4	0.0	189	363
A08	19	212	9.2	4.6	4.5	0.1	745	909
A09	52	549	8.6	4.6	4.0	0.3	835	1 116
A10	91	1 083	9.9	5.2	4.7	0.5	991	1 457
A11	160	1 966	10.3	5.4	4.9	0.1	506	766
A12	101	921	7.1	3.7	3.5	0.0	341	559
A13	70	552	5.9	3.7	2.2	0.0	405	556
A14	82	422	3.1	1.7	1.5	0.0	210	356
A15	151	816	3.4	1.8	1.6	0.0	130	281
A16	61	618	8.1	4.6	3.5	0.5	775	1 228
A17	58	671	9.6	5.2	4.3	0.4	684	1 134
B01	357	2 310	4.5	2.4	2.1	0.7	161	651
B02	136	931	4.8	2.7	2.1	0.6	293	603
B03	110	1 987	16.1	8.3	7.8	0.0	587	935
B04	65	351	3.4	1.8	1.6	0.0	146	278
B05	84	748	6.9	3.8	3.1	0.3	485	869
B06	40	223	3.6	2.0	1.6	0.0	208	374
B07	69	942	11.7	6.4	5.2	0.2	430	867
B08	225	1 512	4.7	2.5	2.2	0.0	240	366
B09	65	651	8.0	4.5	3.5	0.6	732	1 225
B10	68	406	4.0	2.4	1.5	0.1	295	416
B11	24	245	8.2	4.3	3.9	0.2	579	815
B12	21	199	7.5	3.9	3.6	0.0	496	659
B13	22	236	8.7	4.6	4.1	0.1	601	833
B14	28	289	8.3	4.4	3.9	0.4	617	1 036
B15	82	808	7.9	4.4	3.4	0.3	662	961
B16	94	768	6.2	3.5	2.7	0.2	530	741
B17	84	763	7.1	4.0	3.1	0.2	623	834
B18	72	638	6.9	3.9	3.0	0.3	655	891
B19	16	67	2.2	1.3	0.9	0.1	165	356
B20	59	1 312	20.2	10.1	10.1	0.3	335	1 008
B21	14	178	10.7	5.8	4.9	0.1	340	671
B22	10	182	16.2	8.6	7.6	0.4	513	1 073
B23	13	164	10.6	5.7	4.9	0.2	351	657
B24	11	177	14.1	7.4	6.7	0.5	560	1 149
B25	19	230	10.1	5.7	4.4	0.5	322	1 051
B26	7	112	14.0	7.6	6.4	0.0	280	621
B27	74	436	3.9	2.9	1.0	0.9	360	2 026
B28	105	582	3.5	3.5	0.0	0.2	486	666
B29	99	576	3.8	3.7	0.1	0.2	494	795
B30	317	1 630	3.1	2.0	1.1	0.0	215	516
B31	479	5 110	8.7	5.3	3.4	0.1	414	507
C01	27	165	4.1	2.8	1.3	0.8	714	1 613
C02	26	101	1.9	1.2	0.7	0.1	282	531
C03	28	118	2.2	1.4	0.9	0.3	388	877
C04	9	39	2.2	1.4	0.8	0.3	445	1 012
C05	21	121	3.3	2.1	1.2	0.6	614	1 413
C06	21	103	2.9	1.8	1.1	0.6	592	1 246
C07	65	356	3.5	1.7	1.7	0.1	378	634
C08	42	257	4.1	2.1	2.0	0.3	422	826
C09	99	579	3.8	2.3	1.5	0.2	255	722
C10	149	889	4.0	2.4	1.6	0.1	273	697
C11	106	665	4.3	3.0	1.3	0.2	401	807
C12	255	1 375	3.4	2.2	1.2	0.0	138	699
C13	225	1 094	2.9	1.7	1.2	0.0	165	293
average	89	695	6.6	3.7	2.9	0.2	435	785

data	attempt	accept		improve	
A01	11.175 M	483 840	4.3 %	497	0.4 h%
A02	5.726 M	373 977	6.5 %	1 035	1.8 h%
A03	5.689 M	305 816	5.4 %	923	1.6 h%
A04	6.786 M	247 082	3.6 %	983	1.4 h%
A05	12.877 M	436 890	3.4 %	1 001	0.8 h%
A06	16.167 M	637 190	3.9 %	787	0.5 h%
A07	8.552 M	540 443	6.3 %	706	0.8 h%
A08	7.898 M	394 354	5.0 %	593	0.8 h%
A09	8.912 M	301 540	3.4 %	845	0.9 h%
A10	7.909 M	203 519	2.6 %	1 467	1.9 h%
A11	4.896 M	141 962	2.9 %	2 067	4.2 h%
A12	4.867 M	322 731	6.6 %	1 068	2.2 h%
A13	8.916 M	376 197	4.2 %	821	0.9 h%
A14	10.698 M	705 213	6.6 %	516	0.5 h%
A15	5.057 M	332 914	6.6 %	703	1.4 h%
A16	10.776 M	347 633	3.2 %	917	0.9 h%
A17	9.237 M	260 568	2.8 %	1 002	1.1 h%
B01	4.118 M	42 669	1.0 %	1 236	3.0 h%
B02	5.190 M	379 446	7.3 %	1 288	2.5 h%
B03	1.972 M	42 669	2.2 %	570	2.9 h%
B04	12.925 M	204 149	1.6 %	447	0.3 h%
B05	6.461 M	631 576	9.8 %	2 024	3.1 h%
B06	25.769 M	304 123	1.2 %	325	0.1 h%
B07	3.992 M	175 528	4.4 %	2 547	6.4 h%
B08	4.822 M	294 494	6.1 %	2 044	4.2 h%
B09	7.446 M	715 529	9.6 %	1 649	2.2 h%
B10	46.834 M	384 442	0.8 %	655	0.1 h%
B11	21.861 M	144 446	0.7 %	461	0.2 h%
B12	21.920 M	196 947	0.9 %	694	0.3 h%
B13	23.449 M	183 145	0.8 %	516	0.2 h%
B14	5.976 M	959 724	16.1 %	906	1.5 h%
B15	9.509 M	695 514	7.3 %	2 178	2.3 h%
B16	8.041 M	651 256	8.1 %	1 467	1.8 h%
B17	9.738 M	782 531	8.0 %	2 048	2.1 h%
B18	13.478 M	772 540	5.7 %	1 665	1.2 h%
B19	19.563 M	1 859 210	9.5 %	188	0.1 h%
B20	3.024 M	411 119	13.6 %	1 136	3.8 h%
B21	10.806 M	470 509	4.4 %	645	0.6 h%
B22	10.262 M	414 098	4.0 %	566	0.6 h%
B23	10.359 M	449 538	4.3 %	718	0.7 h%
B24	10.229 M	481 571	4.7 %	628	0.6 h%
B25	11.157 M	1 349 731	12.1 %	887	0.8 h%
B26	9.502 M	1 235 342	13.0 %	311	0.3 h%
B27	33.932 M	740 801	2.2 %	839	0.2 h%
B28	14.570 M	924 970	6.3 %	2 449	1.7 h%
B29	11.249 M	677 473	6.0 %	2 940	2.6 h%
B30	12.115 M	607 763	5.0 %	5 337	4.4 h%
B31	1.952 M	22 282	1.1 %	2 594	13.3 h%
C01	21.673 M	780 970	3.6 %	539	0.2 h%
C02	26.518 M	2 010 675	7.6 %	337	0.1 h%
C03	44.361 M	1 405 911	3.2 %	221	0.1 h%
C04	298.345 M	919 179	0.3 %	49	0.0 h%
C05	29.659 M	1 593 056	5.4 %	268	0.1 h%
C06	27.087 M	1 175 693	4.3 %	237	0.1 h%
C07	15.622 M	955 990	6.1 %	647	0.4 h%
C08	24.908 M	1 966 737	7.9 %	533	0.2 h%
C09	12.472 M	969 246	7.8 %	1 069	0.9 h%
C10	8.345 M	652 189	7.8 %	2 113	2.5 h%
C11	9.913 M	758 558	7.7 %	1 782	1.8 h%
C12	21.907 M	682 664	3.1 %	607	0.3 h%
C13	5.876 M	582 395	9.9 %	3 855	6.6 h%
average*	13.112 M	619 185	5.5 %	1 168	1.7 h%

data	attempt	accept		improve	
A01	9.704 M	372 849	3.8 %	506	0.5 h%
A02	5.288 M	323 798	6.1 %	1 292	2.4 h%
A03	4.564 M	228 900	5.0 %	716	1.6 h%
A04	6.914 M	185 590	2.7 %	710	1.0 h%
A05	11.190 M	313 239	2.8 %	1 315	1.2 h%
A06	15.184 M	488 048	3.2 %	814	0.5 h%
A07	9.844 M	536 327	5.4 %	690	0.7 h%
A08	8.437 M	439 519	5.2 %	389	0.5 h%
A09	8.126 M	263 955	3.2 %	773	1.0 h%
A10	7.001 M	167 872	2.4 %	1 438	2.1 h%
A11	4.287 M	117 333	2.7 %	2 218	5.2 h%
A12	3.706 M	251 338	6.8 %	1 238	3.3 h%
A13	8.096 M	271 741	3.4 %	877	1.1 h%
A14	7.758 M	452 062	5.8 %	434	0.6 h%
A15	4.324 M	278 738	6.4 %	668	1.5 h%
A16	10.093 M	260 081	2.6 %	1 043	1.0 h%
A17	8.599 M	210 142	2.4 %	968	1.1 h%
B01	4.316 M	43 458	1.0 %	934	2.2 h%
B02	5.252 M	380 335	7.2 %	2 241	4.3 h%
B03	2.127 M	48 612	2.3 %	332	1.6 h%
B04	13.186 M	213 735	1.6 %	305	0.2 h%
B05	6.522 M	645 239	9.9 %	2 289	3.5 h%
B06	24.905 M	319 034	1.3 %	371	0.1 h%
B07	3.833 M	153 680	4.0 %	2 905	7.6 h%
B08	4.357 M	272 632	6.3 %	2 033	4.7 h%
B09	7.179 M	687 065	9.6 %	1 866	2.6 h%
B10	46.760 M	323 324	0.7 %	650	0.1 h%
B11	21.805 M	138 727	0.6 %	307	0.1 h%
B12	22.693 M	192 329	0.8 %	444	0.2 h%
B13	23.179 M	155 999	0.7 %	447	0.2 h%
B14	6.190 M	967 508	15.6 %	1 044	1.7 h%
B15	10.292 M	712 879	6.9 %	2 039	2.0 h%
B16	8.429 M	678 137	8.0 %	1 484	1.8 h%
B17	9.777 M	754 940	7.7 %	2 377	2.4 h%
B18	13.651 M	744 369	5.5 %	1 828	1.3 h%
B19	19.429 M	1 874 974	9.7 %	217	0.1 h%
B20	3.189 M	429 622	13.5 %	1 003	3.1 h%
B21	12.828 M	486 984	3.8 %	552	0.4 h%
B22	9.598 M	360 746	3.8 %	557	0.6 h%
B23	12.057 M	518 152	4.3 %	519	0.4 h%
B24	9.442 M	367 871	3.9 %	725	0.8 h%
B25	9.447 M	1 341 553	14.2 %	1 120	1.2 h%
B26	9.398 M	1 247 208	13.3 %	272	0.3 h%
B27	31.473 M	499 668	1.6 %	983	0.3 h%
B28	15.499 M	917 257	5.9 %	2 477	1.6 h%
B29	12.007 M	657 253	5.5 %	3 187	2.7 h%
B30	11.793 M	552 206	4.7 %	5 751	4.9 h%
B31	1.879 M	17 957	1.0 %	2 297	12.2 h%
C01	22.068 M	785 592	3.6 %	525	0.2 h%
C02	31.398 M	2 027 068	6.5 %	323	0.1 h%
C03	56.408 M	1 390 099	2.5 %	181	0.0 h%
C04	264.322 M	887 057	0.3 %	43	0.0 h%
C05	28.102 M	1 127 520	4.0 %	291	0.1 h%
C06	27.010 M	1 539 526	5.7 %	186	0.1 h%
C07	14.234 M	861 659	6.1 %	690	0.5 h%
C08	24.957 M	1 965 849	7.9 %	467	0.2 h%
C09	11.885 M	883 356	7.4 %	1 203	1.0 h%
C10	7.653 M	592 909	7.7 %	2 172	2.8 h%
C11	9.839 M	681 057	6.9 %	1 990	2.0 h%
C12	20.623 M	623 018	3.0 %	644	0.3 h%
C13	5.699 M	564 609	9.9 %	4 320	7.6 h%
average*	13.091 M	581 787	5.3 %	1 211	1.8 h%

Table 9 Short-term benchmarks: statistics on transformations for LS-LR (left) and LS-LR' (right) optimization. M = million, h% = one-hundredths percent.

Table 10 Short-term benchmarks: statistics on volumes.

data	avg DQ greedy	avg DQ LS-LR	avg DQ LS-LR'	avg delivq greedy	avg delivq LS-LR	avg delivq LS-LR'
A	3 031 400	4 565 518	4 861 465	15 992	16 066	17 073
A+B+C	2 897 779	4 398 780	4 517 178	16 770	13 139	13 718

Table 11 Long-term benchmarks: characteristics and LR gains with different time limits.

data	customers	plants	bases	drivers	tractors	trailers	callins	orders	wst 1 mn	avg 1 mn	avg 5 mn	avg 1 h
L1	75	6	1	35	21	5	19	56	23.8 %	24.6 %	26.3 %	26.5 %
L2	75	6	1	35	21	5	20	55	22.3 %	23.5 %	24.9 %	25.2 %
L3	175	8	1	35	21	12	36	189	5.2 %	5.8 %	8.3 %	11.2 %
L4	165	4	1	24	11	7	33	167	9.9 %	11.2 %	14.0 %	18.9 %
L5	198	8	7	12	12	12	3	40	32.5 %	34.2 %	35.7 %	35.9 %
average	138	6	2	28	17	8	22	101	18.7 %	19.9 %	21.8 %	23.5 %

Table 12 Long-term benchmarks: greedy results.

data	SO	SC	DQ	LR	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
L1	652	406 443	3 767 868	0.107 871	189	503	2.7	1.6	1.0	0.6	640	1 320
L2	146	407 379	3 827 560	0.106 433	196	506	2.6	1.6	1.0	0.5	619	1 235
L3	86	1 092 976	31 989 357	0.034 167	790	3 584	4.5	2.7	1.8	0.2	366	954
L4	257	808 887	18 433 289	0.043 882	590	2 249	3.8	2.4	1.4	0.2	395	844
L5	85	145 339	8 830 708	0.016 458	295	1 020	3.5	1.9	1.5	1.2	598	1 760
average	245	572 205	13 369 756	*0.042 798	412	1 572	3.4	2.0	1.3	0.5	524	1 223

Table 13 Long-term benchmarks: LS-LR results.

data	SO	SC	DQ	LR	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
L1	0	340 767	3 840 502	0.088 730	137	590	4.3	3.0	1.3	0.8	721	1 618
L2	0	335 661	3 899 780	0.086 072	148	570	3.9	2.7	1.2	0.7	660	1 445
L3	0	1 019 292	32 079 238	0.031 774	839	3 570	4.3	2.6	1.7	0.2	317	873
L4	17	697 009	18 694 845	0.037 283	605	2 400	4.0	2.6	1.4	0.2	321	735
L5	0	106 326	9 475 562	0.011 221	110	1 324	12.0	7.8	4.3	3.2	1 256	4 286
average	3	499 811	13 597 985	*0.036 756	368	1 691	5.7	3.7	2.0	1.0	655	1 792

Table 14 Long-term benchmarks: LS-LR' results.

data	SO	SC	DQ	LR	nb shift	nb oper	avg oper	avg deliv	avg load	avg layov	avg dist	avg dur
L1	0	321 449	4 045 989	0.079 449	148	542	3.7	2.4	1.3	0.7	632	1 403
L2	0	321 207	4 016 621	0.079 969	140	541	3.9	2.6	1.3	0.7	669	1 471
L3	0	1 012 191	32 320 180	0.031 318	807	3 583	4.4	2.7	1.8	0.2	327	890
L4	0	701 139	18 587 949	0.037 720	602	2 396	4.0	2.6	1.4	0.2	325	744
L5	0	101 913	9 630 979	0.010 582	138	1 352	9.8	6.3	3.5	2.2	945	3 159
average	0	491 580	13 720 344	*0.035 829	367	1 683	5.2	3.3	1.9	0.8	580	1 533

Table 15 Long-term benchmarks: lower bounds (left) and gains obtained by local search against greedy (right).

data	$LR_{\min}$
L1	0.061 025
L2	0.059 855
L3	0.019 176
L4	0.012 256
L5	0.005 576

data	gain LS-LR	gain LS-LR'	avg delivq greedy	avg delivq LS-LR	avg delivq LS-LR'
L1	17.7 %	26.3 %	12 460	9 344	11 391
L2	19.1 %	24.9 %	12 205	9 759	11 035
L3	7.0 %	8.3 %	14 997	14 706	14 833
L4	15.0 %	14.0 %	13 018	11 885	11 876
L5	31.8 %	35.7 %	15 755	11 044	11 078
average	*14.1 %	*16.3 %	*14 146	*12 537	*12 864

Table 16 Long-term benchmarks: gains obtained by local search against logistic experts.

data	SO	SC	DQ	LR	gain greedy	gain LS-LR	gain LS-LR'
L1	0	378 778	3 725 847	0.101 662	-6.1 %	12.7 %	21.9 %
L2	0	328 364	3 567 370	0.092 047	-15.6 %	6.5 %	13.1 %
L3	0	1 257 354	32 667 576	0.038 489	11.2 %	17.4 %	18.6 %
L4	0	788 893	18 683 473	0.042 224	-3.9 %	11.7 %	10.7 %
L5	0	290 921	10 398 050	0.027 978	41.2 %	59.9 %	62.2 %
average	0	608 862	13 808 463	*0.044 093	*2.9 %	*16.6 %	*18.7 %

Table 17 The pools $\mathcal{T}_{MO}$ and $\mathcal{T}_{SO}$ of transformations.

$\mathcal{T}_{MO}$	$\mathcal{T}_{SO}$
OperationDeletionBackwardBlockPropag	OperationDeletionBackwardBlockPropag
OperationDeletionForwardBlockPropag	OperationDeletionForwardBlockPropag
OperationInsertionOrderBackwardPropag	OperationInsertionCustomerRunoutBackwardBlockPropag
OperationInsertionOrderForwardPropag	OperationInsertionCustomerRunoutForwardBlockPropag
OperationInsertionSourceOrderBackwardPropag	OperationInsertionSourceCustomerRunoutBackwardBlockPropag
OperationInsertionSourceOrderForwardPropag	OperationInsertionSourceCustomerRunoutForwardBlockPropag
OperationEjectionCustomerNearBackwardBlockPropag	OperationEjectionCustomerNearBackwardBlockPropag
OperationEjectionCustomerNearForwardBlockPropag	OperationEjectionCustomerNearForwardBlockPropag
OperationEjectionSourceNearBackwardBlockPropag	OperationEjectionSourceNearBackwardBlockPropag
OperationEjectionSourceNearForwardBlockPropag	OperationEjectionSourceNearForwardBlockPropag
OperationEjectionOrderBackwardBlockPropag	OperationEjectionRunoutBackwardBlockPropag
OperationEjectionOrderForwardBlockPropag	OperationEjectionRunoutForwardBlockPropag
OperationMoveBetweenShiftsBackwardBackwardBlockPropag	OperationMoveBetweenShiftsBackwardBackwardBlockPropag
OperationMoveBetweenShiftsBackwardForwardBlockPropag	OperationMoveBetweenShiftsBackwardForwardBlockPropag
OperationMoveBetweenShiftsForwardBackwardBlockPropag	OperationMoveBetweenShiftsForwardBackwardBlockPropag
OperationMoveBetweenShiftsForwardForwardBlockPropag	OperationMoveBetweenShiftsForwardForwardBlockPropag
OperationMoveInsideShiftBeforeBackwardBlockPropag	OperationMoveInsideShiftBeforeBackwardBlockPropag
OperationMoveInsideShiftBeforeForwardBlockPropag	OperationMoveInsideShiftBeforeForwardBlockPropag
OperationMoveInsideShiftAfterBackwardBlockPropag	OperationMoveInsideShiftAfterBackwardBlockPropag
OperationMoveInsideShiftAfterForwardBlockPropag	OperationMoveInsideShiftAfterForwardBlockPropag
OperationSwapBetweenShiftsBackwardBackwardBlockPropag	OperationSwapBetweenShiftsBackwardBackwardBlockPropag
OperationSwapBetweenShiftsBackwardForwardBlockPropag	OperationSwapBetweenShiftsBackwardForwardBlockPropag
OperationSwapBetweenShiftsForwardBackwardBlockPropag	OperationSwapBetweenShiftsForwardBackwardBlockPropag
OperationSwapBetweenShiftsForwardForwardBlockPropag	OperationSwapBetweenShiftsForwardForwardBlockPropag
OperationSwapInsideShiftBackwardBlockPropag	OperationSwapInsideShiftBackwardBlockPropag
OperationSwapInsideShiftForwardBlockPropag	OperationSwapInsideShiftForwardBlockPropag
OperationMirrorInsideShiftBackwardBlockPropag	OperationMirrorInsideShiftBackwardBlockPropag
OperationMirrorInsideShiftForwardBlockPropag	OperationMirrorInsideShiftForwardBlockPropag
ShiftSlidingBackward	ShiftSlidingBackward
ShiftSlidingForward	ShiftSlidingForward
ShiftSlidingOrderBackward	ShiftSlidingRunoutBackward
ShiftSlidingOrderForward	ShiftSlidingRunoutForward
ShiftSlidingUnsatOrderBackward	ShiftSlidingFirstRunoutBackward
ShiftSlidingUnsatOrderForward	ShiftSlidingFirstRunoutForward
ShiftResourcesChangingBackward	ShiftResourcesChangingBackward
ShiftResourcesChangingForward	ShiftResourcesChangingForward
ShiftDeletion	ShiftDeletion
ShiftInsertionOrderBackwardPropag	ShiftInsertionCustomerFirstRunoutBackwardPropag
ShiftInsertionOrderForwardPropag	ShiftInsertionCustomerFirstRunoutForwardPropag
ShiftInsertionSourceOrderBackwardPropag	ShiftInsertionSourceCustomerRunoutBackwardPropag
ShiftInsertionSourceOrderForwardPropag	ShiftInsertionSourceCustomerRunoutForwardPropag
ShiftMoveBackward	ShiftInsertionSourceCustomerFirstRunoutBackwardPropag
ShiftMoveForward	ShiftInsertionSourceCustomerFirstRunoutForwardPropag
ShiftSwapBackwardBackward	ShiftMoveBackward
ShiftSwapBackwardForward	ShiftMoveForward
ShiftSwapForwardBackward	ShiftSwapBackwardBackward
ShiftSwapForwardForward	ShiftSwapBackwardForward
	ShiftSwapForwardBackward
	ShiftSwapForwardForward

The first option, used to enlarge the neighborhood induced by certain transformations, is denoted by the suffix "Block". The second option, used to specialize some of the transformations according to the objective, is denoted by suffixes "Order", "Runout", or "Near". The third option, used to set the direction during the (re)scheduling of shifts, is denoted by the suffixes "Backward" or "Forward". Finally, the fourth option, used to facilitate the propagation of flows during the volume (re)assignment, is denoted by the suffix "Propag".

Table 18 The pool $\mathcal{T}_{LR}$ of transformations.

$\mathcal{T}_{LR}$	
OperationDeletionBackward	OperationMoveInsideShiftBeforeBackward
OperationDeletionForward	OperationMoveInsideShiftBeforeForward
OperationDeletionBackwardBlock	OperationMoveInsideShiftAfterBackward
OperationDeletionForwardBlock	OperationMoveInsideShiftAfterForward
OperationInsertionCustomerBackward	OperationMoveInsideShiftBeforeBackwardBlock
OperationInsertionCustomerForward	OperationMoveInsideShiftBeforeForwardBlock
OperationInsertionSourceBackward	OperationMoveInsideShiftAfterBackwardBlock
OperationInsertionSourceForward	OperationMoveInsideShiftAfterForwardBlock
OperationInsertionSourceCustomerBackward	OperationSwapBetweenShiftsBackwardBackward
OperationInsertionSourceCustomerForward	OperationSwapBetweenShiftsBackwardForward
OperationInsertionSourceCustomerNearBackward	OperationSwapBetweenShiftsForwardBackward
OperationInsertionSourceCustomerNearForward	OperationSwapBetweenShiftsForwardForward
OperationEjectionCustomerBackward	OperationSwapBetweenShiftsBackwardBackwardBlock
OperationEjectionCustomerForward	OperationSwapBetweenShiftsBackwardForwardBlock
OperationEjectionCustomerNearBackward	OperationSwapBetweenShiftsForwardBackwardBlock
OperationEjectionCustomerNearForward	OperationSwapBetweenShiftsForwardForwardBlock
OperationEjectionSourceBackward	OperationSwapInsideShiftBackward
OperationEjectionSourceForward	OperationSwapInsideShiftForward
OperationEjectionSourceNearBackward	OperationSwapInsideShiftBackwardBlock
OperationEjectionSourceNearForward	OperationSwapInsideShiftForwardBlock
OperationEjectionCustomerBackwardBlock	OperationMirrorInsideShiftBackwardBlock
OperationEjectionCustomerForwardBlock	OperationMirrorInsideShiftForwardBlock
OperationEjectionCustomerNearBackwardBlock	ShiftSlidingBackward
OperationEjectionCustomerNearForwardBlock	ShiftSlidingForward
OperationEjectionSourceBackwardBlock	ShiftResourcesChangingBackward
OperationEjectionSourceForwardBlock	ShiftResourcesChangingForward
OperationEjectionSourceNearBackwardBlock	ShiftDeletion
OperationEjectionSourceNearForwardBlock	ShiftInsertionSourceCustomerBackward
OperationMoveBetweenShiftsBackwardBackward	ShiftInsertionSourceCustomerForward
OperationMoveBetweenShiftsBackwardForward	ShiftMoveBackward
OperationMoveBetweenShiftsForwardBackward	ShiftMoveForward
OperationMoveBetweenShiftsForwardForward	ShiftSwapBackwardBackward
OperationMoveBetweenShiftsBackwardBackwardBlock	ShiftSwapBackwardForward
OperationMoveBetweenShiftsBackwardForwardBlock	ShiftSwapForwardBackward
OperationMoveBetweenShiftsForwardBackwardBlock	ShiftSwapForwardForward
OperationMoveBetweenShiftsForwardForwardBlock	