Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Matheuristic Algorithm for the Inventory Routing Problem

Zhouxing Su, Zhipeng Lü, Zhuo Wang, Yanmin Qi, Una Benlic

To cite this article:
Zhouxing Su, Zhipeng Lü, Zhuo Wang, Yanmin Qi, Una Benlic (2020) A Matheuristic Algorithm for the Inventory Routing Problem. Transportation Science

Published online in Articles in Advance 05 Mar 2020

. https://doi.org/10.1287/trsc.2019.0930

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Articles in Advance, pp. 1–25
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

A Matheuristic Algorithm for the Inventory Routing Problem

Zhouxing Su,[a] Zhipeng Lü,[a] Zhuo Wang,[a] Yanmin Qi,[a] Una Benlic[b]

[a] SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China; [b] Tesco Plc, Holborn, London EC1R 5AR, United Kingdom

Contact: suzhouxing@hust.edu.cn, https://orcid.org/0000-0002-4794-9833 (ZS); zhipeng.lv@hust.edu.cn (ZL); wang_zhuo@hust.edu.cn (ZW); yanminq@hust.edu.cn (YQ); una.benlic@tesco.com (UB)

Received: August 31, 2017
Revised: August 6, 2018; June 17, 2019
Accepted: July 4, 2019
Published Online in Articles in Advance: March 5, 2020

https://doi.org/10.1287/trsc.2019.0930

Abstract. This work addresses a challenging inventory routing problem that arises from a practical application faced by air-product companies, including Air Liquide. Given its computational complexity and industrial importance, this problem (denoted as IRP-Challenge2016) was presented as the topic of the French Operational Research and Decision Support Society/European Operational Research Society (ROADEF/EURO) Challenge 2016. The IRP-Challenge2016 seeks an optimal delivery schedule to minimize the unit distribution cost, while satisfying various hard constraints. It involves a single product, a heterogeneous fleet, heterogeneous drivers, multiperiods, a deterministic consumption forecast, and time-window constraints. We present a new mathematical formulation of the problem and introduce a matheuristic algorithm that integrates a local search-based metaheuristic with mathematical programming. Our algorithm combines mixed integer programming and linear programming as slave methods to optimize timing and delivery and embeds these procedures within a multineighborhood search metaheuristic to adjust routes. The method extends and enhances a preliminary version of our algorithm, which ranked third in the final round of the ROADEF/EURO Challenge 2016. Computational results for 20 challenge benchmark instances demonstrate the value of the proposed algorithm in terms of both effectiveness and efficiency with respect to the results reported in the competition. We additionally analyze several key components of our matheuristic to gain an insight into its operation.

History: This paper has been accepted for the *Transportation Science* Special Section on the ROADEF/EURO Challenge 2016.
Funding: This work was supported by the National Natural Science Foundation of China [Grants 61370183 and 71850410545].
Supplemental Material: The e-companion is available at https://doi.org/10.1287/trsc.2019.0930.

Keywords: ROADEF/EURO Challenge 2016 • inventory routing problem • matheuristic • local search • mixed integer programming • linear programming

1. Introduction

Organized by the French Operational Research and Decision Support Society (ROADEF) jointly with the European Operational Research Society (EURO), the ROADEF/EURO Challenge is an international operations research competition that features real-world optimization problems and gives researchers an opportunity to compare their algorithms on real-life challenging optimization problems. The topic of the Challenge for 2016 focuses on an Inventory Routing Problem (IRP), which occurs in the healthcare business of Air Liquide. In total, 41 teams from 16 countries registered for participation in the challenge, out of which 12 were qualified as finalists. Although the problem formulation was simplified for the qualification phase, the complete problem with extended and enriched features was made available in the final phase. Moreover, two sets of benchmark instances were introduced in the final phase. The first set was published with the problem specification (public set B), whereas the other set was hidden to the participants until the end of the challenge (hidden set X), with the aim to favor solvers that exhibited robust performance on different data sets.

The development of information technologies has brought about more comprehensive data and more precise prediction in supply chain management. Companies are now able to globally optimize their supply chains to reduce logistics costs, while providing better service by using this information in various ways. The Vendor Managed Inventory (VMI) system is a noteworthy example, which allows vendors to monitor their customer inventory levels and directly arrange the replenishment schedules. Apart from such services for VMI customers, there are also call-in orders, where customers manage their stocks themselves and decide the time windows of their orders, as well as the product quantity to acquire.

In this paper, we introduce a complete mathematical formulation and propose a matheuristic algorithm for IRP-Challenge2016. Our proposed algorithm produces high-quality solutions to this complex real-world optimization problem in a very favorable time span compared with previous efforts. Because of the fine-grained time discretization, which makes it very difficult to simultaneously optimize routes, timing, and delivery quantities, our algorithm applies a new decomposition hierarchy, which, unlike other existing schemes, divides the original problem into one master problem and two subproblems. The matheuristic adjusts routes in the master problem and then optimizes timing and decides delivery quantities in the two subproblems. For the master problem, our method adopts a local search algorithm that is based on six neighborhoods, such as adding/removing operations or shifts to adjust routes. For the timing optimization subproblem, we employ a mixed-integer-programming (MIP) model, which we solve with the mathematical programming solver Gurobi. By introducing a linear-programming (LP) model, the matheuristic can properly deal with continuous quantities. These features distinguish our algorithm from previous metaheuristics, such as Cousineau-Ouimet (2002), Guerrero et al. (2013), and Qin et al. (2014).

The remainder of this paper is organized as follows. In Section 2, we present the problem description of IRP-Challenge2016, including problem constraints and objective. The related works of the IRP are discussed in Section 3. Section 4 gives a complete mathematical formulation of the problem. The framework and ingredients of our algorithm are described in Section 5. Section 6 reports computational results and analysis that helps understand the reason for the performance reported with our method. Conclusions are given in Section 7.

2. Problem Description

The IRP-Challenge2016 problem is defined on a complete directed graph, $G = (N, A)$, with a set of nodes $N = B \cup S \cup V \cup C$ and a set of arcs $A = \{(n_1, n_2) | \forall n_1, n_2 \in N, n_1 \neq n_2\}$. B is the set of bases that are both the starting and the ending nodes of shifts. Specifically, there is only one base in the competition. S is the set of sources where a trailer can be loaded with products. $V \cup C$ is the set of customers where a trailer stops and delivers the product. Specifically, V is the set of VMI customers, and C is the set of call-in customers. For any two nodes n_1 and n_2, the travel distance (in kilometers) and time (in minutes) from n_1 to n_2 are given.

In the IRP-Challenge2016, an operation is either a load at a source or a delivery at a customer. Thus, sources and customers are also called operation nodes. An operation is carried out by a trailer driven by a driver. A set of trailers T and a set of drivers D are available for scheduling.

There are two types of discretization of the schedule horizon in IRP-Challenge2016. Hourly timesteps provide the inventory-management granularity of VMI customers, whereas minute timesteps give the time granularity for drivers, trailers, and operations. A solution to IRP-Challenge2016 is a set of shifts. Each shift is a chronological list of operations performed by a driver–trailer pair. A shift starts at a base, performs the scheduled operations one by one, and ends up returning to the base. Therefore, for each shift of a solution, we need to decide on its driver, trailer, start time (in minutes), and list of operations. In turn, for each operation in the list, we need to decide on the node where the operation takes place, the arrival time (in minutes) that represents the start time of the operation, and the quantity to be delivered or loaded in the operation.

The constraints that must be respected in the IRP-Challenge2016 are usually categorized in the following six groups:

1. Constraints related to layovers:

• $C_{1.1}$ *Layover amount restriction.* Some customers, called layover customers, are in distant areas far from bases, which require long travel time for delivery. To enable a driver to travel for an extended duration, a fixed rest time in a shift called a layover (in minutes) is required. More precisely, a layover takes place if and only if the travel between two consecutive nodes in a shift lasts more than the layover time of the driver plus the travel time. A shift can include at most one layover at anywhere on its route if it involves one or more deliveries to layover customers.

2. Constraints related to drivers:

• $C_{2.1}$ *Minimum intershifts duration.* Two consecutive shifts assigned to the same driver must be separated by a minimum rest duration (in minutes). Specifically, upon completing a shift, a driver can begin the next shift only after a minimum rest time has elapsed.

• $C_{2.2}$ *Maximum driving time.* The cumulative (i.e., total) driving time of a driver in a shift cannot be longer than the driver's maximum allotted driving time. If there is a layover in the shift, the cumulative driving time is split into two parts by the layover, a before-layover part and an after-layover part. Each part cannot be longer than the maximum driving time. Note that a layover occurs once the cumulative driving time reaches the maximum driving time limit, even if the trailer does not happen to be at a node.

• $C_{2.3}$ *Time windows of drivers.* For each driver, a set of availability intervals (time windows) is given, which limits the driver's working time. A time window in the IRP-Challenge2016 is indicated by its starting time and ending time (in minutes). The interval for each shift must lie in one of its driver's time windows.

3. Constraints related to trailers:

• $C_{3.1}$ *Usage conflict of trailers.* For any two shifts s_1 and s_2 using the same trailer, either s_1 ends before the

start of s_2 or s_2 ends before the start of s_1. In other words, different shifts of the same trailer cannot overlap in time.

• $C_{3.2}$ *Trailer–driver compatibility.* There is a compatibility relation between trailers and drivers—that is, every driver can only drive a subset of trailers. For each shift, the assigned trailer must be one of the trailers that can be driven by the driver.

• $C_{3.3}$ *Capacity of trailers.* For each trailer, the capacity and the amount of product (in kilograms) at the beginning of the horizon are given. The initial quantity in a trailer for a shift is the end quantity of its previous shift. The product quantity in each trailer cannot be negative or exceed the trailer's capacity.

4. Constraints related to shifts:

• $C_{4.1}$ *Fixed operation time.* A fixed loading and a fixed delivery time (in minutes) take place at a source and at a customer, respectively. Consequently, the departure time for a node is equal to the arrival time plus the operation time. The operation time only depends on the node where the operation occurs and has no relation to the operation quantity.

• $C_{4.2}$ *Internodes duration in a shift.* For any two consecutive nodes n_1 and n_2 in a shift, the interval between the arrival time of n_2 and the departure time of n_1 must be equal to or greater than the travel time from n_1 to n_2 plus the layover time (if there is a layover between these two nodes). Beside the necessary travel time and possible layover time, a trailer can wait at the gate of a source or a customer. That is, an arbitrarily long idle time can precede an operation if all the constraints are satisfied.

• $C_{4.3}$ *Node–trailer compatibility.* There is a compatibility relation between operation nodes (sources and customers) and trailers, where every operation node can only be served by a subset of trailers. For each shift, the assigned trailer must be compatible with every operation node.

• $C_{4.4}$ *Time windows of customers.* For each customer, a set of opening time windows is given. Delivery must be performed within one of the customer's opening time windows.

• $C_{4.5}$ *Operation quantity restriction.* The operation quantity cannot be zero. When a trailer arrives at a customer, some amount of product must be delivered. Likewise, when a trailer arrives at a source, a certain amount of product must be loaded.

• $C_{4.6}$ *Delivery quantity restriction.* For each VMI customer, the minimum amount of product (in kilograms) that should be delivered in a single operation is given. The operation quantity of any delivery to a VMI customer cannot be less than the minimum amount or greater than the capacity of the customer's tank.

5. Constraints related to VMI customers:

• $C_{5.1}$ *Capacity of VMI customers.* The hourly forecasted consumption amount of product (in kilograms) for every VMI customer during the schedule horizon is known. The amount of product in the customer's tank at the beginning of the horizon and the capacity of the tank are also given. A solution to the problem must guarantee that the product quantity in each VMI customer's tank in every hour throughout the horizon cannot be negative or greater than its capacity. The tank quantity in each hour equals the tank quantity in the previous hour plus the quantity of product delivered by various trailers in the current hour, minus the amount consumed in the current hour.

• $C_{5.2}$ *Run-out avoidance.* A safety inventory level is given for each VMI customer. The quantity in the customer's tank should not be less than the safety level at each hour of the schedule horizon.

6. Constraints related to call-in customers:

• $C_{6.1}$ *Call-in orders satisfaction.* For each call-in customer, a set of orders is given. An order consists of an ordered quantity of product (in kilograms), a minimum ratio ([0%, 100%]) of the ordered quantity to be delivered, and a time window for delivery. The time windows for different orders of a given customer never overlap in time. Orders of each call-in customer must be satisfied by one or more deliveries. The cumulative delivery quantity has to be above a certain threshold and should not exceed the ordered quantity.

• $C_{6.2}$ *Call-in order delivery time restriction.* Each delivery to a call-in customer should be related to one of its orders, which means that the start time of the delivery should be within the time window corresponding to an order of the given customer.

A solution of the IRP-Challenge2016 problem is feasible if it satisfies all the above constraints. Given a feasible solution, the objective is the logistic ratio, defined as the total cost of the shifts divided by the total quantity delivered in the shifts. An optimal solution of the IRP-Challenge2016 problem is a feasible solution with the minimum objective value. The cost of a shift consists of the corresponding distribution costs, including distance cost, time cost, and layover cost. The distance cost for a shift is the product of the cost per distance of the assigned trailer and the total distance of the shift. The time cost for a shift is the product of the cost per working time of the assigned driver and the total working time. The layover cost of a shift is equal to the layover cost of the assigned driver if there is a layover in the shift, and is zero otherwise. For the complete definition of the problem, please refer to the official website of the ROADEF/EURO Challenge at http://www.roadef.org/challenge/2016/en/sujet.php.

3. Related Works

Combining the inventory-management problem and the vehicle-routing problem, the IRP-Challenge2016 takes into account both the VMI and the call-in

customers, which renders the resulting problem even more challenging. As described in Coelho, Cordeau, and Laporte (2013), numerous models for the IRP have been proposed in the literature, considering specific conditions or problem applications. The most common criteria that distinguish between different variants of the IRP are horizon length, demand pattern, and fleet characteristics. With respect to horizon length, Federgruen and Zipkin (1984) and Chien, Balakrishnan, and Wong (1989) dealt with the single-period version in the early years of the IRP research. More recently, the multiperiod extension has been widely studied in Kleywegt, Nori, and Savelsbergh (2004); Moin, Salhi, and Aziz (2011); Coelho and Laporte (2013a); and Adulyasak, Cordeau, and Jans (2013). For the demand pattern, Coelho, Cordeau, and Laporte (2014) introduced heuristics for both the dynamic and the stochastic IRP, where customer demands are gradually revealed over time. On the other hand, most of the literature in the last decade and a half has been focused on static, nonstochastic demand. Various approaches for solving the deterministic IRP can be found in Bertazzi, Paletta, and Speranza (2002); Cousineau-Ouimet (2002); Campbell and Savelsbergh (2004); Guerrero et al. (2013); Coelho and Laporte (2013a); Mjirda et al. (2014); Cordeau et al. (2015); and Desaulniers, Rakke, and Coelho (2016). Concerning different fleet characteristics, there are mainly two types of variants featuring homogeneous and heterogeneous fleets, respectively. For homogeneous fleet variants, Archetti et al. (2007) and Solyali and Süral (2011) studied the single-vehicle IRP that can be regarded as a special case of a homogeneous fleet variant, whereas Coelho and Laporte (2013a) and Guerrero et al. (2013) focused on the general homogeneous fleet variants. Meanwhile, many researchers have studied the heterogeneous fleet versions, including Abdelmaguid, Dessouky, and Ordóñez (2009); Benoist et al. (2011); Coelho and Laporte (2013b); and Hewitt et al. (2013).

Over the last decade and a half, both exact and metaheuristic optimization approaches have been proposed for tackling the IRP. Exact methods for the IRP include branch-and-cut algorithms proposed in Archetti et al. (2007); Coelho and Laporte (2013b); Desaulniers, Rakke, and Coelho (2016); Van Anholt et al. (2016); and Avella, Boccia, and Wolsey (2018), as well as dynamic programming introduced in Kleywegt, Nori, and Savelsbergh (2004). Aside from exact algorithms, metaheuristics including tabu search, variable neighborhood search, and genetic algorithms have been investigated in Cousineau-Ouimet (2002); Liu and Lee (2011); Liu and Chen (2012); Mjirda et al. (2014); and Park, Yoo, and Park (2016).

To exploit the combined power of different schemes, many hybrid algorithms have also been developed. One of the most popular hybrid approaches is the so-called matheuristic, which incorporates mathematical programming models for solving subproblems with heuristic or metaheuristic search. There are three main types of matheuristics for routing problems, as classified in Archetti and Speranza (2014)—the decomposition approaches, the improvement heuristics, and the branch-and-price/column-generation-based methods. In a decomposition approach, the original problem is divided into subproblems, some of which are solved with mathematical programming. In an improvement heuristic, a mathematical model can be used to generate an initial solution, to complement a partial solution, or to improve a complete solution by fixing some part of it. A branch-and-price/column-generation-based (B&P/CG) method might employ a heuristic to solve the restricted master problem, to guide the branching, to determine the bounds, or to improve the feasibility of relaxed solutions obtained by B&P/CG.

Over the last decades, matheuristic algorithms have been widely used to solve the IRP. Guerrero et al. (2013) integrated mixed-integer programming with local search to tackle the problem. In addition, other techniques such as decomposition techniques have been applied to complex models in order to reduce the problem complexity. A common decomposition scheme adopted by Bard et al. (1998), Campbell and Savelsbergh (2004), and Cordeau et al. (2015) is to decide on the approximate times and delivery quantities for each customer, prior to determining the actual routes. Although the partial solution produced in the first phase can be of poor quality and misleading due to the nature of an optimal complete solution, it nevertheless often brings good performance gains by providing an advanced start for the second phase.

Matheuristics have also been frequently employed for different problems presented in the ROADEF/EURO Challenge. For example, Mansi et al. (2012) used it to solve the disruption-management problem proposed by Amadeus in 2009. Anghinolfi et al. (2012) employed it to solve the energy-management problem proposed by EDF in 2010 and reported better average results than the winning algorithm. Lopes et al. (2015) used it to solve the machine-reassignment problem proposed by Google in 2012.

Apart from the general algorithm frameworks, various replenishment policies have been introduced in the literature to simplify the problem model, such as (a) Direct Shipping in Ramalhinho Dias Lourenço and Ribeiro (2003); (b) Order Up-to Level in Bertazzi, Paletta, and Speranza (2002), Ramalhinho Dias Lourenço and Ribeiro (2003), Coelho, Cordeau, and Laporte (2012b), and Archetti et al. (2012); (c) Zero Inventory Ordering Policy in Ramalhinho Dias Lourenço and Ribeiro (2003); (d) Fixed Partition in Ramalhinho Dias Lourenço and Ribeiro (2003) and

Zhao, Wang, and Lai (2007); and (e) Power-of-Two in Zhao, Chen, and Zang (2008). These policies can significantly reduce the solution space, at the cost of overlooking some solutions.

To understand the complexity of the challenge, a comprehensive comparison between different versions of the IRP models that are related to the challenge topic is given in Table 1. Column Entity presents the corresponding entity involved in each feature, including Driver, Trailer, Site, and Product. Column Type categorizes the features from another perspective into objectives (denoted with O), time-related constraints (T), space-related constraints (S), and resource-related constraints (R). Column Feature lists the key attributes to distinguish between different versions of the IRP. Finally, the following five columns give the representative IRP versions for comparison. C16 reports the characteristics of the

Table 1. Comparison Between the IRP Versions That Are Related to IRP-Challenge2016

Entity	Type	Feature	C16	BGJ	CCL	ABL/ABH	DRC
	O	Routing cost	*		✓	✓	✓
	O	Delivery quantity	*				
	O	Logistic ratio	✓	✓			
	O	Holding cost			✓	✓	✓
Driver	R	Multiple drivers	✓	✓			
	RT	Maximum driving time	✓	✓			
	RT	Minimum intershifts duration	✓	✓			
	RT	Layover	✓	✓			
	RT	Multiple driver time windows	✓	✓			
	R	Driver consistency			✓		
	R	Driver partial consistency			✓		
Trailer	R	Multiple trailers	✓	✓	✓		✓
	R	Floating driver–trailer pairs	✓	✓			
	RT	Multiple shifts in a period per vehicle	✓	✓			
	R	Vehicle capacity	✓	✓	✓	✓	✓
	R	Heterogeneous fleet	✓	✓	✓		✓
	RT	Vehicle filling rate	*		✓		
Site	R	Deterministic periodic demand	✓	✓	✓	✓	✓
	R	Deterministic periodic production			✓	✓	✓
	T	Nontrivial travel time within a period	✓	✓			
	ST	Multiple visits to a customer in a period	✓	✓			
	T	Multiple periods	✓	✓	✓	✓	✓
	S	Single base	✓	✓	✓	✓	✓
	S	Multiple suppliers	✓	✓			
	S	Distinct base and supplier	✓	✓			
	S	VMI customers	✓	✓	✓	✓	✓
	S	Call-in customers	✓	✓			
	ST	Multiple customer time windows	✓	✓			
	S	Customer capacity	✓	✓	✓	✓	✓
	R	Supplier capacity			✓		✓
	ST	Visit spacing	*		✓		
Product	R	Single product	✓	✓	✓	✓	✓
	R	Maximum-level policy	✓	✓	✓	✓	✓
	R	Order-up-to policy	*		✓	✓	
	R	Quantity consistency	*		✓		

Notes. Column Entity presents the corresponding entity involved in each feature, including Driver, Trailer, Site, and Product. Column Type categorizes the features from another perspective into objectives (O), time-related constraints (T), space-related constraints (S), and resource-related constraints (R). Column Feature lists the key attributes to distinguish between different versions of the IRP. Finally, the remaining five columns give the representative IRP versions for comparison. C16 reports the characteristics of the IRP-Challenge2016. BGJ represents the problem solved by the randomized local search proposed in Benoist et al. (2011). CCL stands for the variant solved by the adaptive large neighborhood search proposed in Coelho, Cordeau, and Laporte (2012a). ABL and ABH are the versions solved by the branch-and-cut and the hybrid approach for inventory routing proposed in Archetti et al. (2007, 2012). DRC represents the IRP solved by the branch-price-and-cut method proposed in Desaulniers, Rakke, and Coelho (2016). A ✓ mark indicates that the solver is able to handle the feature, and a * mark in the second column indicates that our algorithm can solve the IRP with the given feature after minor modifications, without a significant deterioration in performance.

IRP-Challenge2016. BGJ represents the problem solved by the randomized local search proposed in Benoist et al. (2011). CCL stands for the variant solved by the adaptive large neighborhood search proposed in Coelho, Cordeau, and Laporte (2012a). ABL and ABH are the versions solved by the branch-and-cut and the hybrid approach for inventory routing proposed in Archetti et al. (2007, 2012). DRC represents the IRP solved by the branch-price-and-cut method proposed in Desaulniers, Rakke, and Coelho (2016). A ✓ mark indicates that the solver is able to handle the feature, and a * mark in the second column indicates that our algorithm can solve the IRP with the given feature after minor modifications, without a significant deterioration in performance.

We can observe from Table 1 that the resolution techniques tend to use local search-based metaheuristics when more features are considered. On the other hand, they rely more on exact methods like mathematical programming or tree search, when less features are required. Furthermore, it is very common to use hybrid algorithms, especially matheuristics, to solve problems that integrate both continuous and discrete optimization. One of the key features that distinguishes the IRP-Challenge2016 from the classic IRP versions is that traveling takes nontrivial time, where the number of nodes visited in a period is limited, while there could be shifts that cover multiple periods. Aside from the problem proposed in Benoist et al. (2011), which was also motivated by Air Liquide, none of the other versions support this feature. Another interesting point is that some features, such as quantity consistency and visit spacing presented in Coelho, Cordeau, and Laporte (2012a), are only used as acceleration strategies in our algorithm, as described later in Section 5.3.4. It is thus easy to extend our matheuristic to cover such characteristics.

Table 2. Sets in IRP-Challenge2016

Set	Description
B	Set of bases. Bases are both the starting and the ending nodes of shifts.
S	Set of sources where a trailer can be loaded with product
V	Set of VMI customers
C	Set of call-in customers
N	Set of all nodes, $N = B \cup S \cup V \cup C$
N^*	Set of operation nodes, $N^* = S \cup V \cup C$
D	Set of drivers
T	Set of trailers
R	Set of orders from all the call-in customers
M	Set of minutes over the schedule horizon, $M = \{0, \ldots, \lvert M\rvert - 1\}$
H	Set of hours over the schedule horizon, $H = \{0, \ldots, \lvert H\rvert - 1\}$
W_d^D	Set of available time windows of driver $d \in D$
W_v^N	Set of opening time windows of customer $v \in V \cup C$

4. Mathematical Formulation

To describe the mathematical formulation of the IRP-Challenge2016, Tables 2 and 3, respectively, present the notations of sets and constants. We define decision variables and auxiliary variables in Tables 4 and 5, respectively.

Because of the complexity of the IRP-Challenge2016 problem, we adopt several preprocessing rules to automatically satisfy some of the constraints and to prune some unnecessary decision variables that will not affect the optimality of the solution.

1. For constraint $C_{2.1}$ (*minimum intershifts duration*), the decision variable $y_{dtnmm'}$ for a base that does not satisfy $m' - m \geq M_d^I$ is removed (except for the cases where m is the first minute of the horizon or m' is the last minute of the horizon).

Table 3. Constants in IRP-Challenge2016

Constant	Description
M^H	Number of minutes in an hour, $M^H = 60$
$E_{nn'}^D$	Travel distance from node n to node n'
$E_{nn'}^T$	Travel time from node n to node n'
E_n^S	Operation time at operation node n
A_{tn}	$A_{tn} = 1$ if and only if trailer t and operation node n are compatible; 0 otherwise
A_{dt}	$A_{dt} = 1$ if and only if driver d can drive trailer t; 0 otherwise
M_d^D	Maximum driving time of driver d in a shift
M_d^L	Layover time of driver d
M_d^I	Minimum rest time of driver d between two consecutive shifts
Q_t^C	Capacity of trailer t
Q_t^B	Product quantity in trailer t at the beginning of the horizon
I_v^C	Tank capacity of VMI customer v
I_v^B	Product quantity in the tank of VMI customer v at the beginning of the horizon
I_v^S	Safety level of the inventory at VMI customer v
I_v^M	Minimum amount of product that should be delivered to VMI customer v in a single operation
U_{vh}	Cumulative consumption of VMI customer v from the first hour 0 to the hour h
A_v^L	$A_v^L = 1$ if and only if customer v is a layover customer; 0 otherwise
W_{nw}^B, W_{nw}^E	First and last minute of time window w of customer n
W_{dw}^B, W_{dw}^E	First and last minute of time window w of driver d
R_r^U	Quantity of order r, which indicates the maximum quantity to deliver for r
R_r^L	Minimum quantity to deliver for order r, which is equal to the product of the ordered quantity and the minimum ratio
R_r^C	Call-in customer of order r
R_r^B, R_r^E	First and last minute of the time window corresponding to order r
F_d^W	Cost per working time unit of driver d
F_d^L	Layover cost of driver d
F_t^D	Cost per distance unit of trailer t

Table 4. Decision Variables for IRP-Challenge2016

Variable	Domain	Description
$x_{dtnmn'm'}$	{0, 1}	Traveling edge, $x_{dtnmn'm'} = 1$ for driver d ($\forall d \in D$) driving trailer t ($\forall t \in T$), departing from node n ($\forall n \in N$) at the minute m ($\forall m \in M$) and arriving at node n' ($\forall n' \in N$) at the minute m' ($\forall m' \in M, m' \geq m$); 0 otherwise
$y_{dtnmm'}$	{0, 1}	Holding edge, $y_{dtnmm'} = 1$ for driver d ($\forall d \in D$) driving trailer t ($\forall t \in T$), operating (if node n is an operation node) and staying at node n ($\forall n \in N$) from the minute m ($\forall m \in M$) to the minute m' ($\forall m' \in M, m' \geq m$); 0 otherwise
$z_{dtnmm'}$	{0, 1}	Layover edge, $z_{dtnmm'} = 1$ for driver d ($\forall d \in D$) driving trailer t ($\forall t \in T$), taking a layover from the minute m ($\forall m \in M$) to the minute m' ($\forall m' \in M, m' \geq m$) after departing from node n ($\forall n \in N$); 0 otherwise
o_{dtvm}	$[0, Q_t^C]$	Delivery quantity to customer v ($\forall v \in V \cup C$). The delivery is performed by driver d ($\forall d \in D$) and trailer t ($\forall t \in T$) and begins at the minute m ($\forall m \in M$). Note that we use positive values to indicate delivered quantity.
o_{dtsm}	$[-Q_t^C, 0]$	Load quantity at source s ($\forall s \in S$). The load is performed by driver d ($\forall d \in D$) and trailer t ($\forall t \in T$) and begins at the minute m ($\forall m \in M$). Note that we use negative values to indicate loaded quantity.

2. For constraint $C_{2.3}$ (*time windows of drivers*), decision variables $x_{dtnmn'm'}$, $y_{dtnmm'}$, $z_{dtnmm'}$, o_{dtvm}, and o_{dtsm}, whose related times cannot fit in any time window of driver d are removed.

3. For constraint $C_{3.2}$ (*trailer–driver compatibility*), decision variables $x_{dtnmn'm'}$, $y_{dtnmm'}$, $z_{dtnmm'}$, o_{dtvm}, and o_{dtsm} whose driver d and trailer t are incompatible are removed.

4. The operation time at one operation node is fixed. Hence, for constraint $C_{4.1}$ (*fixed operation time*), the decision variable $y_{dtnmm'}$ for an operation node that does not satisfy $m' - m \geq E_n^S$ is removed.

5. The travel time between any two nodes is fixed. Hence, for constraint $C_{4.2}$ (*internodes duration in a shift*), a decision variable $x_{dtnmn'm'}$ that does not satisfy $m' - m = E_{nn'}^T$ is removed. This rule also ensures that layovers never occur at traveling edges to correctly handle them in layover edge-related formulas.

6. For constraint $C_{4.3}$ (*node–trailer compatibility*), decision variables $x_{dtnmn'm'}$, $y_{dtnmm'}$, $z_{dtnmm'}$, o_{dtvm}, and o_{dtsm} are removed if the corresponding trailer t and operation node n (n', v, or s) are incompatible.

7. For constraint $C_{4.4}$ (*time windows of customers*), decision variables $x_{dtnmn'm'}$, $y_{dtnmm'}$, and o_{dtvm} whose corresponding times do not fit in any time window of customer n (n' or v) are removed.

8. For constraint $C_{6.2}$ (*call-in order delivery time restriction*), decision variables $x_{dtnmn'm'}$, $y_{dtnmm'}$, and o_{dtvm} whose corresponding times do not fit in any time window of call-in customer n (n' or v) are removed.

9. The layover time of each driver is fixed. Hence, a decision variable $z_{dtnmm'}$ that does not satisfy $m' - m = M_d^L$ is removed.

10. As each shift starts and ends at a base, decision variables $y_{dtn0m'}$ and $y_{dtnm(|M|-1)}$ for the corresponding operation nodes are removed.

With the above preprocessing, we can describe the mathematical model of the IRP-Challenge2016 as follows, where P represents a large (penalty) value.

$$\text{Minimize} \quad \frac{c^D + c^T + c^L}{q}, \tag{1}$$

subject to

$$\begin{aligned}
&\sum_{n \in N} \sum_{m^- = m}^{m' - M_d^L} z_{dtnm^-(m^- + M_d^L)} \\
&\quad \leq P \cdot \left(1 - \sum_{m^- = 0}^{m} y_{dtbm^- m}\right) + P \cdot \left(1 - \sum_{m^+ = m'}^{|M|-1} y_{dtbm'm^+}\right) \\
&\quad + P \cdot \sum_{m^- = m}^{m'} \sum_{m^+ = m^-}^{m'} y_{dtbm^- m^+} \\
&\quad + \min\left\{1, \sum_{v \in V \cup C} A_v^L \cdot \sum_{m^- = m}^{m' - E_v^S} \sum_{m^+ = m^- + E_v^S}^{m'} y_{dtvm^- m^+}\right\} \\
&\qquad \forall d \in D, t \in T, m \in M, m' \in M,
\end{aligned} \tag{2}$$

Table 5. Auxiliary Variables for IRP-Challenge2016

Variable	Domain	Description
c^D	$[0, +\infty)$	Total distance cost
c^T	$[0, +\infty)$	Total time cost
c^L	$[0, +\infty)$	Total layover cost
q	$[0, +\infty)$	Total delivered quantity

$$\sum_{n,n'\in N} E^T_{nn'} \cdot \sum_{m^-=m}^{m'-E^T_{nn'}} x_{dtnm^-n'(m^-+E^T_{nn'})} \le M^D_d + M^D_d \cdot \sum_{n\in N}\sum_{m^-=m}^{m'-M^L_d} z_{dtnm^-(m^-+M^L_d)} + P\cdot\left(1-\sum_{m^-=0}^{m} y_{dtbm^-m}\right) + P\cdot\left(1-\sum_{m^+=m'}^{|M|-1} y_{dtbm'm^+}\right) + P\cdot\sum_{m^-=m}^{m'}\sum_{m^+=m^-}^{m'} y_{dtbm^-m^+} \quad \forall d\in D, t\in T, m\in M, m'\in M, \tag{3}$$

$$\sum_{n,n'\in N} E^T_{nn'} \cdot \sum_{m^-=m}^{m^*-E^T_{nn'}} x_{dtnm^-n'(m^-+E^T_{nn'})} \le M^D_d + P\cdot\left(1-z_{dtn^*m^*(m^*+M^L_d)}\right) + P\cdot\left(1-\sum_{m^-=0}^{m} y_{dtbm^-m}\right) + P\cdot\left(1-\sum_{m^+=m'}^{|M|-1} y_{dtbm'm^+}\right) + P\cdot\sum_{m^-=m}^{m'}\sum_{m^+=m^-}^{m'} y_{dtbm^-m^+} \quad \forall d\in D, t\in T, m\in M, m'\in M, n^*\in N, m\le m^*\le m', \tag{4}$$

$$\sum_{n,n'\in N} E^T_{nn'} \cdot \sum_{m^-=m^*}^{m'-E^T_{nn'}} x_{dtnm^-n'(m^-+E^T_{nn'})} \le M^D_d + \sum_{n\in N} E^T_{n^*n}\cdot x_{dtn^*m^*n(m^*+E^T_{n^*n})} + P\cdot\left(1-z_{dtn^*m^*(m^*+M^L_d)}\right) + P\cdot\left(1-\sum_{m^-=0}^{m} y_{dtbm^-m}\right) + P\cdot\left(1-\sum_{m^+=m'}^{|M|-1} y_{dtbm'm^+}\right) + P\cdot\sum_{m^-=m}^{m'}\sum_{m^+=m^-}^{m'} y_{dtbm^-m^+} \quad \forall d\in D, t\in T, m\in M, m'\in M, n^*\in N, m\le m^*\le m', \tag{5}$$

$$\sum_{t\in T}\sum_{n\in N}\sum_{n'\in N}\sum_{m^-=max\{0,m-E^T_{nn'}\}}^{min\{m,|M|-1-E^T_{nn'}\}} x_{dtnm^-n'(m^-+E^T_{nn'})} + \sum_{t\in T}\sum_{n\in N}\sum_{m^-=max\{0,m-M^L_d\}}^{min\{m,|M|-1-M^L_d\}} z_{dtnm^-(m^-+M^L_d)} + \sum_{t\in T}\sum_{n\in N^*}\sum_{m^-=0}^{m}\sum_{m^+=m}^{|M|-1} y_{dtnm^-m^+} \le 1 \quad \forall d\in D, m\in M, \tag{6}$$

$$\sum_{d\in D}\sum_{n\in N}\sum_{n'\in N}\sum_{m^-=max\{0,m-E^T_{nn'}\}}^{min\{m,|M|-1-E^T_{nn'}\}} x_{dtnm^-n'(m^-+E^T_{nn'})} + \sum_{d\in D}\sum_{n\in N}\sum_{m^-=max\{0,m-M^L_d\}}^{min\{m,|M|-1-M^L_d\}} z_{dtnm^-(m^-+M^L_d)} + \sum_{d\in D}\sum_{n\in N^*}\sum_{m^-=0}^{m}\sum_{m^+=m}^{|M|-1} y_{dtnm^-m^+} \le 1 \quad \forall t\in T, m\in M, \tag{7}$$

$$0 \le Q^B_t - \sum_{d\in D}\sum_{v\in V}\sum_{m'=0}^{m} o_{dtvm'} - \sum_{d\in D}\sum_{s\in S}\sum_{m'=0}^{m} o_{dtsm'} \le Q^C_t \quad \forall t\in T, m\in M, \tag{8}$$

$$\sum_{n\in N^*}\sum_{m^-=m}^{m'-M^L_d-E^S_n}\sum_{m^+=m^-+M^L_d+E^S_n}^{m'} y_{dtnm^-m^+} \le P\cdot\left(1-\sum_{m^-=0}^{m} y_{dtbm^-m}\right) + P\cdot\left(1-\sum_{m^+=m'}^{|M|-1} y_{dtbm'm^+}\right) + P\cdot\sum_{m^-=m}^{m'}\sum_{m^+=m^-}^{m'} y_{dtbm^-m^+} + \min\left\{1, \sum_{v\in V\cup C} A^L_v\cdot\sum_{m^-=m}^{m'-E^S_v}\sum_{m^+=m+E^S_v}^{m'} y_{dtvm^-m^+}\right\} \quad \forall d\in D, t\in T, m\in M, m'\in M, \tag{9}$$

$$o_{dtsm} \ge -Q^C_t\cdot\sum_{m'\in M} y_{dtsmm'} \quad \forall d\in D, t\in T, s\in S, m\in M, \tag{10}$$

$$I^M_v\cdot\sum_{m'\in M} y_{dtvmm'} \le o_{dtvm} \le I^C_v\cdot\sum_{m'\in M} y_{dtvmm'} \quad \forall d\in D, t\in T, v\in V, m\in M, \tag{11}$$

$$R^L_r \le \sum_{d\in D}\sum_{t\in T}\sum_{m=R^B_r}^{R^E_r} o_{dtR^C_r m} \le R^U_r \quad \forall r\in R, \tag{12}$$

$$I^S_v \le I^B_v - U_{vh} + \sum_{d\in D}\sum_{t\in T}\sum_{m=0}^{M^H\cdot(h+1)-1} o_{dtvm} \le I^C_v \quad \forall v\in V, h\in H, \tag{13}$$

$$\sum_{n'\in N}\sum_{m'\in M} x_{dtn'm'nm} = \sum_{m'\in M} y_{dtnmm'} \quad \forall d\in D, t\in T, n\in N, m\in M-\{0\}, \tag{14}$$

$$\sum_{m'\in M} z_{dtnm'm} = \sum_{n'\in N}\sum_{m'\in M} x_{dtnmn'm'} \quad \forall d\in D, t\in T, n\in N, m\in M, \tag{15}$$

$$\sum_{m'\in M} y_{dtnm'm} = \sum_{n'\in N}\sum_{m'\in M} x_{dtnmn'm'} + \sum_{m'\in M} z_{dtnmm'} \quad \forall d\in D, t\in T, n\in N, m\in M-\{|M|-1\}, \tag{16}$$

$$\sum_{m\in M} y_{dtb0m} = 1 \quad \forall d\in D, t\in T, b\in B, \tag{17}$$

$$\sum_{m\in M} y_{dtbm(|M|-1)} = 1 \quad \forall d\in D, t\in T, b\in B, \tag{18}$$

$$c^D = \sum_{t\in T} F^D_t\cdot\sum_{n,n'\in N} E^D_{nn'}\cdot\sum_{d\in D}\sum_{m=0}^{|M|-1-E^T_{nn'}} x_{dtnmn'(m+E^T_{nn'})}, \tag{19}$$

$$c^T = \sum_{d\in D} F^W_d\cdot\sum_{t\in T}\sum_{b\in B}\left(|M|-1-\sum_{m,m'\in M}(m'-m)\cdot y_{dtbmm'}\right), \tag{20}$$

$$c^L = \sum_{d\in D} F^L_d\cdot\sum_{t\in T}\sum_{n\in N}\sum_{m=0}^{|M|-1-M^L_d} z_{dtnm(m+M^L_d)}, \tag{21}$$

$$q = \sum_{d\in D}\sum_{t\in T}\sum_{v\in V\cup C}\sum_{m\in M} o_{dtvm}. \tag{22}$$

Constraints (2) impose layover amount restriction ($C_{1.1}$), ensuring that there is no more than one layover per shift if there are layover customers visited on its route. These only make sense when minutes m and m' are the start and the end time of a shift, respectively—that is, there is a holding edge at the base ending at minute m, a holding edge at the base starting at minute m', and no holding edge at the base between m and m'. Constraints (3), (4), and (5) limit the maximum driving time ($C_{2.2}$), where m and m' denote the start and the end time of a shift as mentioned above, n^* denotes the last visited node before a layover, and m^* denotes the departure time from node n^*. Constraints (3) are defined for each shift, ensuring that the total driving time of a shift should not exceed the driver's maximum driving time M_d^D if there is no layover in the shift, and not be longer than $2 \cdot M_d^D$ otherwise. Constraints (4) and (5) guarantee, respectively, that the cumulative driving time before arriving at n^* and after departing from the succeeding node of n^* will never exceed the driver's maximal driving time. Constraints (6) ensure that the working time and the layover time of a driver never overlap, implying that for each driver at each minute, the total number of visited traveling, holding, and layover edges covering the minute should not exceed one. Constraints (7) restrict usage conflict of trailers ($C_{3.1}$). They are defined for each trailer at each minute, implying that the total number of the three types of edges should not be greater than one. Constraints (8) restrict the capacity of trailers ($C_{3.3}$), indicating that the initial quantity in the trailer minus the total operation quantity in the whole horizon should always be nonnegative and never exceed its tank capacity. Constraints (9) impose the layover amount restriction ($C_{1.1}$) and the internodes duration in a shift ($C_{4.2}$). These ensure that there can be no holding edge whose interval is longer than the setup time plus the layover duration, unless there are layover customers and an overlong holding edge signifies a layover. Constraints (10) and (11) ensure the operation quantity restriction ($C_{4.5}$) and the delivery quantity restriction ($C_{4.6}$). Constraints (10) guarantee that the loaded quantity is equal to zero at unvisited sources and can never exceed the tank capacity of the trailer. Constraints (11) ensure that the quantity delivered to a VMI customer should never be less than the minimal delivery quantity or greater than the customer's tank capacity if the customer is visited. Constraints (12) satisfy call-in orders ($C_{6.1}$), requiring that the total quantity delivered to a call-in customer within the order's time window should be between the minimum and the maximum required quantity. Constraints (13) limit the capacity of VMI customers ($C_{5.1}$) and the run-out avoidance ($C_{5.2}$). These constraints are applied to each VMI customer, implying that the remaining quantity at the end of each hour is equal to the quantity in the customer's tank at the beginning minus the forecasted cumulative consumption until the current hour plus the total delivered quantity until the last minute of the hour and should never be greater than its capacity or less than its safety level. Constraints (14), (15), (16), (17), and (18) are defined for each pair of driver and trailer to guarantee the route connectivity of a shift, while ensuring that each shift starts and ends at a base. Constraints (14) are defined for each node n and each minute m except the first minute, implying that the total number of traveling edges ending at node n at minute m is equal to the total number of visited holding edges at n starting from m. Constraints (15) are defined for each node n and each minute m, meaning that the total number of layover edges at node n ending at minute m is equal to the total number of visited holding edges starting from n at m. Constraints (16) are defined for each node n and each minute m, except the last minute, restricting that the total number of holding edges at node n ending at minute m is equal to the total number of visited traveling edges and layover edges, which start from n at m. Constraints (17) and (18) guarantee that each route for a shift is a cycle that starts from and ends at the base. The total number of holding edges that start from the beginning of the horizon at the base should be one, while there only exists one holding edge at the base in the last minute of the horizon. Finally, constraints (19), (20), and (21), respectively, define the total distance cost, total time cost, and total layover cost, whereas constraint (22) defines the total delivered quantity. Our formulation yields a mixed-integer fractional programming model that can be converted to the standard mixed-integer linear programming (MILP) model via the Charnes–Cooper transformation proposed in Charnes and Cooper (1962) or the parametric approach described in Dinkelbach (1962) and Bajalinov (2013).

5. Solution Method

5.1. Main Framework

We present a matheuristic algorithm for IRP-Challenge2016 that integrates a multineighborhood search-based metaheuristic with mathematical programming. The main framework of the proposed matheuristic is given in Algorithm 1, where s^* is returned as the best solution. The algorithm starts by generating an initial feasible solution with a constructive heuristic algorithm (line 1), followed by the matheuristic optimization procedure (lines 4–11) to improve the solution. At each iteration of the matheuristic optimization, a neighborhood is selected based on a probability, which is adjusted in an adaptive way according to the historical performance of the neighborhood move (lines 5 and 6). The selected

neighborhood is then explored, and the best neighboring solution is returned to replace the current solution s (line 7). The best solution s^* is updated if an improved solution is found (lines 8–10). When a specified termination condition is met, the algorithm terminates and returns the best solution s^*.

Algorithm 1 (The Matheuristic Algorithm for IRP-Challenge2016)

Input: Instance I
Output: Best solution s^* found
1: $s \leftarrow$ InitialSolutionGeneration(I) // s denotes the current solution
2: $s^* \leftarrow s$ // s^* denotes the best solution found
3: $s' \leftarrow \emptyset$ // s' denotes the previous solution
4: **while** termination condition is not met **do**
5: $\mathcal{N}_i \leftarrow$ NeighborhoodSelection(s, s', s^*) // select a neighborhood with a probability
6: $s' \leftarrow s$
7: $s \leftarrow$ MatheuristicSearch(I, s, $\mathcal{N}_i$) // see Algorithm 2
8: **if** s is better than s^* **then**
9: $s^* \leftarrow s$
10: **end if**
11: **end while**

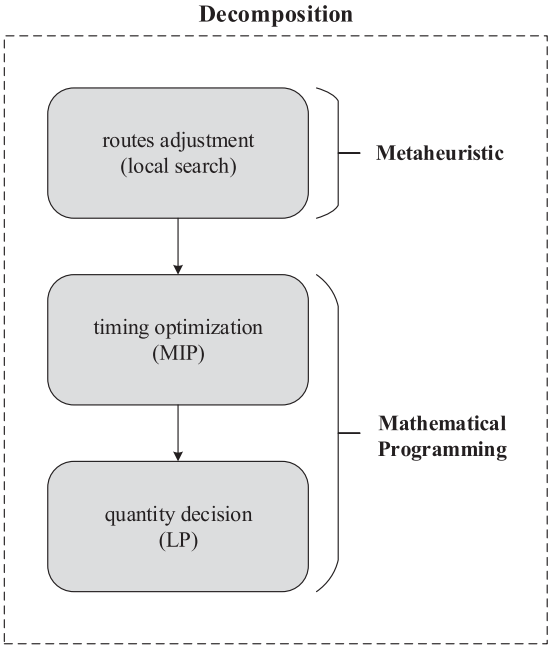
5.2. Initial Solution Generation

We propose a constructive heuristic to generate an initial feasible solution. The algorithm obtains a solution by chronologically constructing shifts one after another. For the construction of each shift, the algorithm randomly selects a driver–trailer pair that is available and compatible. It then moves the selected pair to one of the reachable operation nodes from the current node and determines the operation quantity repeatedly. A node is reachable if the driver is able to visit it and return to the base before reaching the end of its time window or before exceeding the maximum driving time. The earlier a reachable customer runs out of stock, the greater the chance of visiting it in the current shift. More precisely, we sort all customers by their start time of shortage and select one of the top τ^k customers as the kth one to visit in the shift, where τ is a parameter that controls the randomness of the construction. If the gap between the start times of shortage at two customers is less than μ, the customer with a smaller distance cost, smaller time cost, and fewer compatible trailers is preferred. Similarly, the driver–trailer pair tends to visit a reachable source if there is not enough quantity left in the trailer, and the probability of refilling it is equal to $\sqrt{\eta \cdot \rho / Q_t^C}$, where ρ is the remaining quantity in trailer t's tank and η is a parameter to control the aggressiveness of refilling. The algorithm makes use of the Order Up-to Level policy as described in Coelho, Cordeau, and Laporte (2013) to determine the quantity to be loaded or delivered. When there is no reachable operation node, the trailer returns to the base, and the construction of the shift ends. Finally, the process of generating the initial solution stops if there is no available pair to construct a new shift. In addition, we use the concept of economic level, which forbids small-quantity delivery. If the quantity remaining in a customer's tank is above the economic level, the operation for this customer will be postponed or cancelled. In our algorithm, the economic level of a customer v is set according to the capacity of its tank and the visiting trailer t, which is equal to $\min\{0.3 \cdot I_v^C, 0.4 \cdot Q_t^C\}$. As it is a greedy algorithm with a large number of random choices, this process is not guaranteed to produce a feasible solution at every single run. The procedure is thus restarted until a feasible solution is found. Fortunately, it is able to generate feasible solutions within a few attempts and a reasonable time for most benchmark instances. The effectiveness of this constructive heuristic will be discussed later in Section 6.3.

5.3. Matheuristic Optimization

As described above, after an initial feasible solution is generated, we launch the matheuristic optimization procedure to improve the solution in an iterative manner. As the IRP-Challenge2016 combines inventory management and vehicle routing, we need to decide routes, times, and operation quantities of a complete solution. As different types of constraints and fine-grained time discretization make the model described in Section 4 too large to simultaneously consider all these factors, we adopt a decomposition scheme to optimize all the factors in a hierarchical way. Specifically, the original problem is decomposed into one master problem (routes adjustment) and two subproblems (timing optimization and quantity decision), as shown in Figure 1. As previously indicated, we employ MIP and LP as slave methods and embed them within a multineighborhood local search procedure. To explore as many routes as possible, we introduce six neighborhoods to adjust routes of the current solution. When searching the selected neighborhood (see Algorithm 2), we first adjust routes of the current solution to obtain the set of the neighboring solutions $\mathcal{N}_i(s)$ (line 1). To enhance the search efficiency of the algorithm, we adopt several acceleration strategies for the local search, as described in Section 5.3.4. Specifically, we perform neighborhood sampling and reduction on $\mathcal{N}_i(s)$, which ignores some unpromising neighboring solutions, thus reducing the size of the neighborhood (line 2). We then optimize the timing for each neighboring solution by solving an MIP model (line 3) and rank the solutions by incremental value of the total cost in an increasing order (line 4). Finally, we search the ordered solutions, decide operation quantities by solving an LP

Figure 1. Decomposition Scheme for the IRP-Challenge2016



model, and find the best neighborhood move (line 5). If there is no feasible solution in the neighborhood of the current feasible solution, nothing is recorded in the output s', which is regarded as the worst solution in line 8 of Algorithm 1. The search then proceeds with the evaluation of the next neighborhood.

Algorithm 2 (Matheuristic Search Procedure for Neighborhood $\mathcal{N}_i$)

Input: Current solution s and neighborhood $\mathcal{N}_i$
Output: Best neighboring solution s'

1: $\mathcal{N}_i(s) \leftarrow$ adjust the routes of s // $\mathcal{N}_i(s)$ denotes the set of neighboring solutions
2: $\tilde{\mathcal{N}}_i(s) \leftarrow$ perform neighborhood sampling and reduction for $\mathcal{N}_i(s)$
3: Solve timing optimization subproblem for each solution in $\tilde{\mathcal{N}}_i(s)$
4: $L \leftarrow$ rank the solutions in $\tilde{\mathcal{N}}_i(s)$ by the incremental value of the resulting total cost in an increasing order // L denotes the list of the ordered solutions
5: $s' \leftarrow$ search the list L, solve quantity decision subproblem and find the best neighboring solution

5.3.1. Routes Adjustment. In order to adjust the routes of a solution in various ways, we introduce six neighborhoods that are based on the following: adding an operation ($\mathcal{N}_1$), removing an operation ($\mathcal{N}_2$), replacing an operation ($\mathcal{N}_3$), swapping two operations within a shift ($\mathcal{N}_4$), moving an operation from one shift to another ($\mathcal{N}_5$), and adding a shift ($\mathcal{N}_6$).

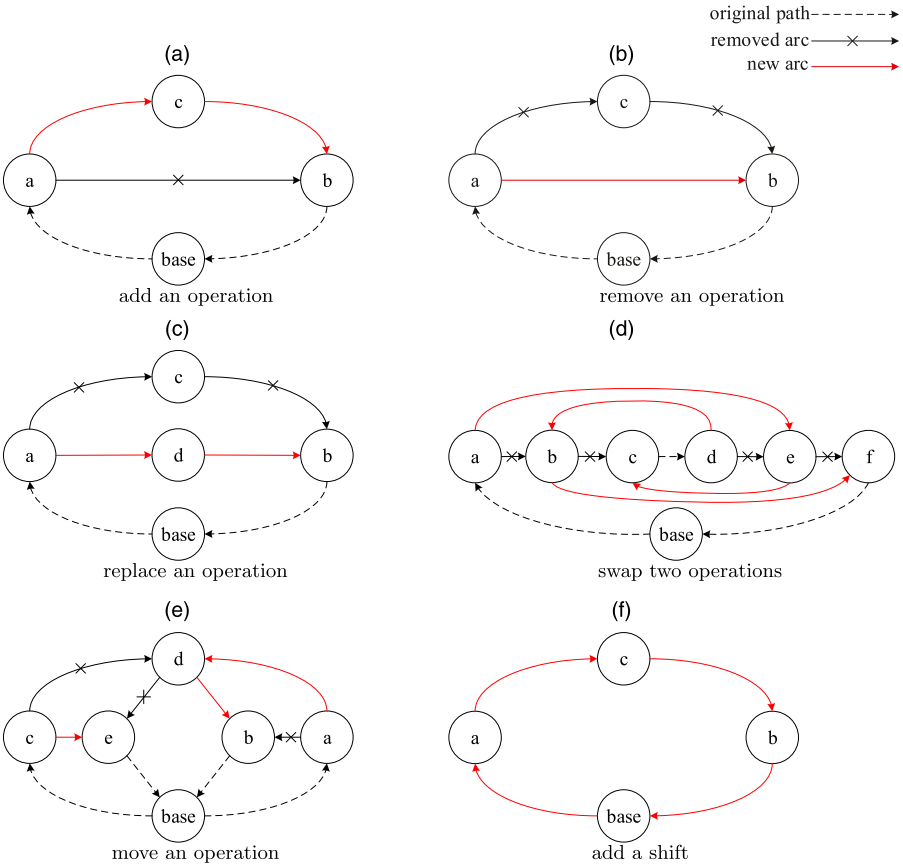
- $\mathcal{N}_1$ inserts a new operation between two existing operations of a shift, before the first operation of a shift, or after the last operation of a shift. Figure 2(a) presents an example of $\mathcal{N}_1$ where a new node c, which has not been visited in the shift, is inserted between nodes a and b.
- $\mathcal{N}_2$ removes an existing operation, as opposed to moving $\mathcal{N}_1$. Figure 2(b) shows an example of $\mathcal{N}_2$ where node c is removed from the shift.
- $\mathcal{N}_3$ changes the visiting node of an existing operation. It can be regarded as removing an operation and adding a new operation visiting a different node in the same order in the shift. Figure 2(c) gives an example of $\mathcal{N}_3$, where a visiting node c between nodes a and b is replaced by a new node d.
- $\mathcal{N}_4$ consists of exchanging a pair of operations belonging to different nodes within a shift. Figure 2(d) provides an example of $\mathcal{N}_4$, where nodes b and e are swapped within a shift.
- $\mathcal{N}_5$ moves an existing operation from one shift to another. It can be regarded as removing an operation in a shift, and adding a new operation visiting the same node in another shift. Figure 2(e) shows an example of $\mathcal{N}_5$ where the operation belonging to node d is moved.
- $\mathcal{N}_6$ constructs a new shift. It is conducted by performing several $\mathcal{N}_1$ moves, one by one, where a new operation is always inserted after the last operation of the shift that is being constructed. Figure 2(f) presents an example of $\mathcal{N}_6$ where a new shift visiting nodes a, c and b is sequentially constructed.

As exploring all the neighborhoods is time-consuming, only one neighborhood is selected and evaluated at each iteration of the local search. The selection of a neighborhood is based on a roulette-wheel principle and an adaptive adjustment mechanism. More precisely, we let $\Omega = \{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4, \mathcal{N}_5, \mathcal{N}_6\}$ denote the set of all the neighborhoods and assign a selection weight ω_i to each neighborhood $\mathcal{N}_i \in \Omega$. At each iteration of the local search, neighborhood $\mathcal{N}_i$ is chosen with a probability p_i calculated as follows:

$$p_i = \frac{\omega_i}{\sum_{j=1}^{|\Omega|} \omega_j}. \tag{23}$$

The selection weights are adjusted to control the choice of the neighborhood for search space exploration. Initially, the weight of each neighborhood is set to the same value $\bar{\omega}$, giving each neighborhood an equal selection probability. As the search progresses, the weight is dynamically adjusted to control how often a particular neighborhood is selected during the search, based on the historical performance of each neighborhood. At each iteration *iter* of the local search, the best neighboring solution s is obtained by

Figure 2. (Color online) Illustration of Neighborhood Moves



Notes. (a) Add an operation. (b) Remove an operation. (c) Replace an operation. (d) Swap two operations. (e) Move an operation. (f) Add a shift.

exploring the selected neighborhood $\mathcal{N}_i$ of the current solution s'. Let $\Delta_i = f(s) - f(s')$ denote the incremental value of the objective obtained for the selected neighborhood $\mathcal{N}_i$ at iteration *iter*. We record the historical incremental value of the objective Δ_i^*, which is initially set to 0 for each neighborhood. When Δ_i is obtained at a certain iteration, Δ_i^* is updated by $\Delta_i^* = \delta_0 \cdot \Delta_i^* + (1 - \delta_0) \cdot \Delta_i$. If there is no feasible neighboring solution in $\mathcal{N}_i$, then Δ_i does not exist, and Δ_i^* is updated by $\Delta_i^* = \delta_0 \cdot \Delta_i^* + \delta_1 \cdot |\Delta_i^*|$. We next adjust the weight of the selected neighborhood $\mathcal{N}_i$ by comparing Δ_i^* with the median of the historical incremental values of the objective over all the neighborhoods as follows:

$$\omega_i = \begin{cases} \omega_i + \lambda \cdot (\beta - \omega_i), & \text{if } \Delta_i^* \text{is smaller than} \\ & \text{the median value} \\ \omega_i + \lambda \cdot (\alpha - \omega_i), & \text{otherwise,} \end{cases} \quad (24)$$

where α and β $(0 < \alpha < \beta)$ are the lower and the upper bounds of the selection weight, respectively. The value of the weight for each neighborhood always lies in the interval $[\alpha, \beta]$, where $\lambda \in [0, 1]$ is the adjustment factor that controls the convergence speed of ω_i toward the bounds. The weights of the neighborhoods that are not selected at the current iteration remain unchanged. One observes that the value of ω_i will increase if Δ_i^* is better than the median value over all the neighborhoods. The aim of the adaptive weight-adjustment mechanism is to encourage the selection of a neighborhood whose contribution to the improvement of the solution is greater than the median level of all the neighborhoods.

5.3.2. Timing Optimization. Once the routes of a solution are adjusted by a neighborhood move, timing-related decision variables are optimized, which include start times of shifts and operations. As a change made by a neighborhood move is local, and only the routes of one or two shifts are adjusted at each iteration, there is no need to optimize the start time of all the shifts and operations. Thus, we propose an MIP model that only considers one affected shift and optimizes the start time of this shift and its operations.

Given the route of a shift, the notations for the timing optimization model are presented in Table 6.

Table 6. Notations for the Timing Optimization Model

Notations	Description
r	$r = \langle r_0, r_1, \ldots, r_g, r_{g+1}\rangle$ is the route of the shift under consideration, where r_0 and r_{g+1} represent the base, $r_k (k = 1, \ldots, g)$ denotes the kth visited operation node on the route, and g refers to the number of operation nodes.
d	Driver assigned to the shift under consideration
et	Earliest start time of the shift under consideration
lt	Latest end time of the shift under consideration

We let $r = \langle r_0, r_1, \ldots, r_g, r_{g+1}\rangle$ denote the route of a shift that is adjusted by a neighborhood move and required to optimize the timing-related decision variables. The earliest start time and the latest end time of a shift can be calculated by considering the minimum intershifts duration ($C_{2.1}$), the time windows of drivers ($C_{2.3}$), and the usage conflict of trailers ($C_{3.1}$) under the condition that other shifts are fixed and not considered in the model. Table 7 gives the decision variables for the timing optimization model. As described above, each customer has a set of opening time windows, which limit the time when delivery to the customer can occur. Each call-in customer has a set of order time windows, which, combined with opening time windows, confine the delivery times to the call-in customers. To handle the time windows consistently for both VMI and call-in customers, we identify the intersections of opening time windows and order time windows for each call-in customer. We then use the intersections of time windows to limit delivery time for call-in customers, while using opening time windows for VMI customers. The timing optimization model is as follows.

Minimize
$$u_{g+1} - u_0, \tag{25}$$

subject to

$$\sum_{k=1}^{g+1} z_k \le \min\left\{1, \sum_{k=1}^{g} A^L_{r_k}\right\}, \tag{26}$$

$$\sum_{k=0}^{g} E^T_{r_k r_{k+1}} \le \sum_{k=1}^{g+1} z_k \cdot M^D_d + M^D_d, \tag{27}$$

$$z_k \le \left(\left(\sum_{i=0}^{k-2} E^T_{r_i r_{i+1}} \le M^D_d\right) \wedge \left(\sum_{i=k}^{g} E^T_{r_i r_{i+1}} \le M^D_d\right)\right) \quad \forall k \in \{1, \ldots, g+1\}, \tag{28}$$

$$E^T_{r_{k-1}r_k} + E^S_{r_{k-1}} + z_k \cdot M^L_d \le u_k - u_{k-1} \le E^T_{r_{k-1}r_k} + E^S_{r_{k-1}} + (1 - z_k) \cdot \left(M^L_d - 1\right) + z_k \cdot |M| \quad \forall k \in \{1, \ldots, g+1\}, \tag{29}$$

$$\sum_{w \in W^N_{r_k}} y_{kw} = 1 \quad \forall k \in \{k | k \in \{1, \ldots, g+1\}, r_k \in V \cup C\}, \tag{30}$$

$$y_{kw} \cdot W^B_{nw} \le u_k \le W^E_{nw} + (1 - y_{kw}) \cdot |M| \quad \forall k \in \{k | k \in \{1, \ldots, g+1\}, r_k \in V \cup C\}, w \in W^N_{r_k}, \tag{31}$$

$$u_0 \ge et, \tag{32}$$

$$u_{g+1} \le lt. \tag{33}$$

To make the shift as compact as possible, objective (25) minimizes the time span of the shift. Constraint (26) guarantees layover amount restriction ($C_{1.1}$), ensuring that the total amount of layovers is not more than one when there exists at least one delivery to a layover customer, and is zero otherwise. Constraints (27) and (28) impose maximum driving time ($C_{2.2}$). Constraints (27) ensure that the total driving time for a given route should not exceed the given maximal driving time, where the maximal driving time is doubled if there exists a layover in the shift. Constraints (28) are stated for each node on the route, to guarantee that if there is a layover before arriving at a given node, both the cumulative driving time before reaching its preceding node and after leaving it do not exceed the maximum driving time of the driver. Constraints (29) present the relationship among layover time ($C_{1.1}$), fixed operation time ($C_{4.1}$), and internodes duration in a shift ($C_{4.2}$) by restricting the difference value between the start time of two successive operations. These constraints are defined for each operation on the route, imposing that $(z_k = 0) \Leftrightarrow (E^T + E^S \le u_k - u_{k-1} < E^T + E^S + M^L)$ and $(z_k = 1) \Leftrightarrow (E^T + E^S + M^L \le u_k - u_{k-1})$. That is to say, there must be a

Table 7. Decision Variables for the Timing Optimization Model

Variable	Domain	Description
u_k	M	If $k = 0$, u_0 represents the start time of the shift under consideration; if $k = 1, \ldots, g$, u_k represents the start time of the operation that happens at node r_k; and if $k = g + 1$, u_{g+1} represents the end time of the shift.
z_k	{0, 1}	$z_k = 1, k = 1, \ldots, g+1$ if and only if there is a layover after leaving from node r_{k-1} and before arriving at node r_k; 0 otherwise.
y_{kw}	{0, 1}	$y_{kw} = 1, k = 1, \ldots, g, r_k \in V \cup C$, if and only if the operation that happens at customer r_k is performed within the time window w of the customer; 0 otherwise.

layover if and only if the waiting time is longer than the layover time, and vice versa. Constraints (30) and (31) impose time windows of customers ($C_{4.4}$) and call-in order time restrictions ($C_{6.2}$). Constraints (30) are defined for each delivery operation, indicating that the operation performed at each node must lie in a time window of the corresponding node, while constraints (31) determine the time window of the operation for each node. Constraints (32) and (33) restrict the start time and the end time of the shift, respectively.

The time complexity of this subproblem is still unknown, although there are some similar problems reported in the literature. Archetti and Savelsbergh (2009) introduced a Truckload Trip Scheduling Problem and proposed an $O(n^3)$-time algorithm for solving it, where n is the number of nodes to visit. However, only one possible opening time window is considered for each node. Benoist et al. (2011) solved a Shift Scheduling Problem with a linear time-space greedy algorithm, which cannot guarantee feasibility or optimality. Another difference between the above problems and the proposed timing optimization model is that both the Truckload Trip Scheduling Problem and the Shift Scheduling Problem require that either the beginning or the ending time of a shift is fixed. There is a naive pseudo-polynomial time algorithm for the timing optimization model, which expands a node into a series of node–time pairs $\langle n, t\rangle$ and sets the costs to $t' - t$ for edges from $\langle n, t\rangle$ to $\langle n', t'\rangle$, whereas this cost is zero if n or n' is the base node. Furthermore, an edge from $\langle n, t\rangle$ to $\langle n', t'\rangle$ exists only if n and n' are two adjacent nodes on the route of the shift, and t lies in the time windows of n, so do t' and n'. Then, the problem is converted into finding a shortest path from the base at the earliest start time of the shift to the base at the latest end time of the shift.

5.3.3. Quantity Decision. Once the routes of a solution are adjusted and the time-related decision variables are optimized, the total cost of the solution is fixed. To complete the solution, we need to decide the quantities of operations so as to maximize the total delivered quantity such that the logistic ratio of the solution is minimized. Thus, we propose an LP model to determine the operation quantity for each operation in a solution. Because the routes and times have been fixed at this stage, the only constraints to be considered are the quantity-related ones, such as capacity of trailer ($C_{3.3}$), run-out avoidance ($C_{5.2}$), and call-in orders satisfaction ($C_{6.1}$).

The notations for the quantity decision model are presented in Table 8, where $op = \langle op_0, \ldots, op_l\rangle$ denotes the list of all the operations in a solution, which are ranked in chronological order according to their start times. Table 9 gives the decision variables for the quantity decision model. For convenience and consistency, we adopt a uniform symbol x_k to denote the operation quantity and use positive and negative values to represent delivery and load, respectively.

Table 8. Notations for the Quantity Decision Model

Notations	Description
op	$op = \langle op_0, \ldots, op_l\rangle$ is the list of all the operations ranked in chronological order, where l refers to the number of operations.
O_k^N	O_k^N represents the node where the operation op_k happens.
O_k^T	O_k^T represents the trailer performing the operation op_k.
A_k^C	$A_k^C = 1$ denotes that the operation op_k is a delivery to a customer, and $A_k^C = 0$ indicates that the operation op_k is a load from a source.
A_{kt}^T	$A_{kt}^T = 1$ if and only if the operation op_k is performed by trailer t; 0 otherwise.
A_{kn}^N	$A_{kn}^N = 1$ if and only if the operation op_k happens at node n; 0 otherwise.
m_k	$m_k, k = 0, \ldots, l$ represents the start time (in minutes) of the operation op_k.
h_k	$h_k, k = 0, \ldots, l$ represents the hour to which m_k belongs.
K^F	$K^F = \{0\} \cup \{k \mid h_k \neq h_{k-1}, k = 1, \ldots, l\}$ is the index set of the first operations in each hour during the horizon.
K^L	$K^L = \{k \mid h_k \neq h_{k+1}, k = 0, \ldots, l-1\} \cup \{l\}$ is the index set of the last operations in each hour during the horizon.
A_{nh}^H	$A_{nh}^H = 1$ if and only if there exists an operation op_k that happens at the node n in the hour h, that is, $O_k^N = n$ and $h_k = h$; 0 otherwise.

Note. $op = \langle op_0, \ldots, op_l\rangle$ denotes the list of all the operations in a solution, which are ranked in chronological order according to their start times.

A trailer should be filled with products each time it visits a source. To check the capacity of trailers ($C_{3.3}$), the carried product quantities in the trailers should be compared with their capacities after each loading or delivery operation. However, a trailer's product quantity will not increase between two load operations, and, hence, we only need to verify that the trailer's quantity is nonnegative after the last delivery operation and before a load operation. Similarly, the product quantity in a VMI customer's tank in each hour during the horizon will not increase before a delivery operation. Therefore, we only need to guarantee that the customer's quantity in the hour before a delivery cannot be less than the safety level and to assure that the quantity in the hour when a delivery occurs

Table 9. Decision Variables for the Quantity Decision Model

Variable	Domain	Description
x_k	$[(A_k^C - 1) \cdot Q_{O_k^T}^C, A_k^C \cdot Q_{O_k^T}^C]$	Operation quantity; positive values represent delivery, and negative values represent load.

does not exceed the tank's capacity. We also check the inventory level in the last hour of the horizon. By performing this preprocessing, a large number of constraints can be eliminated, which makes the model much easier to solve.

The quantity decision model is as follows.

$$\text{Maximize} \quad \sum_{k=0}^{l} A_k^C \cdot x_k, \tag{34}$$

subject to

$$Q_t^B - \sum_{k=0}^{i-1} A_{kt}^T \cdot x_k \geq 0 \quad \forall (t,i) \in \Big\{(t,i) | t \in T, i \in \{0,\ldots,l\}, A_{it}^T = 1, A_i^C = 0\Big\}, \tag{35}$$

$$Q_t^B - \sum_{k=0}^{l} A_{kt}^T \cdot x_k \geq 0 \quad \forall t \in T, \tag{36}$$

$$Q_t^B - \sum_{k=0}^{i} A_{kt}^T \cdot x_k = Q_t^C \quad \forall (t,i) \in \Big\{(t,i) | t \in T, i \in \{0,\ldots,l\}, A_{it}^T = 1, A_i^C = 0\Big\}, \tag{37}$$

$$I_{O_k^N}^M \leq x_k \leq I_{O_k^N}^C \quad \forall k \in \Big\{k | k \in \{0,\ldots,l\}, O_k^N \in V\Big\}, \tag{38}$$

$$R_r^L \leq \sum_{k \in K'} x_k \leq R_r^U \quad \forall r \in R, K' = \Big\{k | O_k^N = R_r^C, m_k \in [R_r^B, R_r^E]\Big\}, \tag{39}$$

$$I_v^B + \sum_{k=0}^{i-1} A_{kv}^N \cdot x_k - U_{v(h_i-1)} \geq I_v^S \quad \forall (v,i) \in \Big\{(v,i) | v \in V, i \in K^F, A_{vh_i}^H = 1\Big\}, \tag{40}$$

$$I_v^B + \sum_{k=0}^{i} A_{kv}^N \cdot x_k - U_{vh_i} \leq I_v^C \quad \forall (v,i) \in \Big\{(v,i) | v \in V, i \in K^L, A_{vh_i}^H = 1\Big\}, \tag{41}$$

$$I_v^S \leq I_v^B + \sum_{k=0}^{l} A_{kv}^N \cdot x_k - U_{v(|H|-1)} \leq I_v^C \quad \forall v \in V. \tag{42}$$

Objective (34) maximizes the total delivery quantity of all the operations. Constraints (35), (36), and (37) restrict the capacity of trailers ($C_{3.3}$). Constraints (35) are defined for each loading operation, implying that the total quantity delivered minus the total quantity loaded in a trailer before the current load operation should not exceed the product quantity in the trailer at the beginning of the horizon. Constraints (36) ensure nonnegative quantity in the trailer after the last operation, and constraints (37) ensure that the quantity loaded from a source should be equal to the capacity of the trailer minus its current quantity. Constraints (38) impose a delivery-quantity restriction ($C_{4.6}$), limiting the quantity delivered to each VMI customer to the range from the minimum delivery amount to the capacity of the customer's tank. Constraints (39) satisfy the call-in orders ($C_{6.1}$), imposing that for each order, the sum of deliveries in the order's time window at the call-in customer is not less than the minimum quantity and does not exceed the maximum quantity. Constraints (40), (41), and (42) guarantee the capacity of VMI customers ($C_{5.1}$) and run-out avoidance ($C_{5.2}$). Constraints (40) are stated for each VMI customer hourly, so as to ensure that the amount of the residual in the customer's tank before the first delivery operation does not fall below the safety level. Constraints (41) imply that for each VMI customer, the quantity in the customer's tank after the last delivery operation conducted in each hour should not exceed the capacity. Finally, constraints (42) guarantee that in the whole horizon, the quantity in each VMI customer's tank should not be less than the safety level or exceed the capacity.

5.3.4. Acceleration Strategies. To enhance the search efficiency of the multineighborhood search procedure, we incorporate several acceleration strategies, which include neighborhood reduction, neighborhood sampling, and neighborhood estimation. In neighborhood reduction, neighborhood moves that have little potential to improve the solution are not considered, whereas neighborhood sampling ensures that only a part of the entire neighborhood is evaluated at each iteration. In neighborhood estimation, more attention is paid to the promising neighborhood moves.

Local search typically evaluates numerous candidate solutions to perform a single move at each iteration. Evaluation of too many moves can have a negative impact on the search performance. This effect is amplified in our situation, as, once the routes of a solution are adjusted by a neighborhood move, we need to solve two subproblems to evaluate the neighboring solution. For efficiency, our matheuristic algorithm ignores the neighborhood moves that have little potential to improve on the objective value by considering the traveling distance and operation frequency in the exploration of the neighborhood. When adjusting the routes by a neighborhood move, we only visit certain nodes on the adjusted route. A node should not be too far away from its preceding and succeeding nodes with respect to the traveling distance—that is, a driver can only drive to the top $|N| \cdot \gamma$ nearest nodes from its current node. Furthermore, the operation frequency comes into play by ignoring a neighborhood move if too many operations occur at a node within a short time. Specifically, an empirical threshold interval between operations is calculated, as shown in Algorithm 3.

Table 10. The Features of Benchmark Instances

Instance	$\|H\|$	$\|D\|$	$\|T\|$	$\|N\|$	$\|B\|$	$\|S\|$	$\|V\|$	$\|C\|$
2.12	240	13	15	326	1	1	301	23
2.13	240	5	5	55	1	1	53	0
2.14	840	5	5	55	1	1	53	0
2.15	240	4	3	136	1	1	131	3
2.16	240	7	4	186	1	1	183	1
2.17	840	4	3	136	1	1	131	3
2.18	840	4	3	136	1	1	131	3
2.19	840	5	5	55	1	1	53	0
2.20	840	7	4	186	1	1	183	1
2.21	840	7	4	186	1	1	183	1
2.22	504	13	15	326	1	1	301	23
2.23	504	13	15	326	1	1	301	23
2.24	240	5	6	35	1	2	23	9
2.25	840	5	6	35	1	2	23	9
2.26	840	5	6	35	1	2	23	9
2.27	240	13	15	326	1	1	301	23
2.28	240	7	4	186	1	1	183	1
2.29	840	4	3	136	1	1	131	3
2.30	504	13	15	326	1	1	301	23
2.31	504	13	15	326	1	1	301	23

Algorithm 3 (Calculation of Estimated Operation Interval Threshold)

Input: VMI customer v
Output: estimated operation interval e

1: $q^- \leftarrow U_{v(|H|-1)} - I_v^B + I_v^S$ // minimal total delivery quantity
2: $q^+ \leftarrow U_{v(|H|-1)} - I_v^B + I_v^C$ // maximal total delivery quantity
3: $q \leftarrow (q^- + q^+)/2$ // estimated total delivery quantity
4: $m \leftarrow \max_{t \in T}\{Q_t^C\}$ // maximal trailer capacity
5: $c \leftarrow 1 + q/\min\{0.4 \cdot I_v^C, 0.4 \cdot m\}$ // estimated operation count
6: $e \leftarrow |H|/c$

Even with the foregoing neighborhood reduction method, the size of some neighborhoods can still be very large. Therefore, we apply a neighborhood sampling mechanism, whose effectiveness has been verified on other large-size challenging optimization problems, such as Wang, Lü, and Ye (2016). In detail, only a part of the entire neighborhood is explored at each iteration, while each neighboring solution has an equal chance to be selected for evaluation.

In addition to neighborhood reduction and sampling, we introduce a strategy to help the search focus on promising neighborhood moves. The total cost of a solution has a greater effect on the objective value than the total delivered quantity, to the extent that the objective value is unlikely to be improved if the total cost is worsened by the neighborhood move. Consequently, we apply a more aggressive reduction strategy, which treats the partial objective as an estimate of the full objective to speed up the neighborhood evaluation, as in Lü and Hao (2010). As described above, the total cost of a neighboring solution is fixed after the routes are adjusted and the timing optimization model is solved. At each iteration of the local search, we first adjust the routes and solve the timing optimization model for all neighboring solutions to be evaluated. Then, the neighboring solutions are ranked by their incremental values of the total cost in increasing order and divided into segments. The algorithm examines segments of the ordered neighboring solutions one after another by solving the quantity decision model for all solutions in it and finding the best one. Once the best neighboring solution in a certain segment is better than the current solution, the search stops and returns the improved solution. Otherwise, all neighboring solutions in the sequence are evaluated, and the best one is returned.

6. Computational Results and Analysis

6.1. Benchmark Instances

The benchmark data set of the IRP-Challenge2016 consists of two sets of 20 instances used in the finals of ROADEF/EURO Challenge 2016, where sets B and X contain 15 and 5 instances, respectively. Each instance is characterized by timespan, drivers, trailers, and nodes. The schedule horizon of different instances ranges from 10 to 35 days, while the number of nodes ranges from 35 to 326. For most instances, the

Table 11. Setting of Parameters

Parameter	Description	Value	Definition
τ	Customer selection range expansion rate in construction	1.5	Section 5.2
η	Refilling/delivering selection coefficient	2	Section 5.2
δ_0	Adjustment factor of historical incremental values of the objective	0.5	Section 5.3.1
δ_1	Adjustment factor of historical incremental values of the objective	0.6	Section 5.3.1
α	Lower bound of selection weight	512	Section 5.3.1
β	Upper bound of selection weight	$\|\Omega\| \cdot \alpha$	Section 5.3.1
$\bar{\omega}$	Initial value of selection weight	$(\alpha + \beta)/2$	Section 5.3.1
λ	Adjustment factor of selection weight	0.5	Section 5.3.1
γ	Threshold ratio of distance rank for traveling between nodes	10%	Section 5.3.4
ϕ	Segment size of neighborhood moves	10	Section 6.6

Table 12. Normalized Scores of the Finalists of the ROADEF/EURO Challenge

Rank	Team ID	Teams	Country	Score B	Score B+X
1	S17	Ahmed Kheiri	United Kingdom	0.00	0.00
2	S15	Simon Crevals et al.	Netherlands	4.12	5.56
3	**S24**	**Zhouxing Su et al.**	**China**	**4.81**	**6.62**
4	S12	Aldair Álvarez et al.	Brazil	8.83	13.32
5	J9	Tamara Jovanovic et al.	France	9.40	14.05
6	S13	Yun He et al.	France	10.78	15.03
7	S9	Nabil Absi et al.	France	13.64	18.64
8	S23	Federico Alonzo-Pecina et al.	Mexico	8.02	—
9	S25	Yang Wang et al.	China	8.35	—

Notes. The normalized score is defined at http://challenge.roadef.org/2016/en/qualification.php. Boldface type indicates the authors of the present work.

number of drivers and trailers does not exceed 10. Table 10 shows the size of instances, including the numbers of hours, drivers, trailers, nodes, bases, sources, VMI, and call-in customers. Set B consists of the instances 2.12–2.26, while set X consists of the instances 2.27–2.31.

6.2. Experimental Protocol

Our matheuristic algorithm is programmed in C++ and compiled by using Visual C++ 2015. All computational experiments are carried out on Windows Server 2012 x64 with Intel Xeon E5-2698v3 (2.30GHz) and 192 GB of RAM. The MIP and LP models are solved by Gurobi 6.5.2 x64. The program uses 4 physical cores (8 logical threads) in each run. The main parameters of the matheuristic algorithm are presented in Table 11. A description of each parameter is given in the column Definition.

6.3. Computational Results

Table 12 presents the final outcomes of the ROADEF/EURO Challenge 2016, where all the algorithms were tested by the committee and ranked according to the normalized scores. Given a team t, for each instance i,

Table 13. Computational Results Under 30 Minutes Runtime Limit

		Construction			Matheuristic				
Instance	f_{win}	Average	No. attempt	Time (s)	Best	Average	σ	Time (s)	Iteration
2.12	0.010024	0.027173	72	3	0.013304	0.014080	0.000440	1,789	1,135
2.13	0.028875	0.075641	1	<1	0.034262	0.038856	0.001846	1,158	4,889
2.14	0.034971	0.096260	1	<1	0.044646	0.047195	0.001625	1,675	1,920
2.15	0.024993	0.068700	7	<1	0.033868	0.037221	0.001384	1,347	3,155
2.16	0.011783	0.032020	1	<1	0.016728	0.017296	0.000310	1,773	1,747
2.17	0.032130	0.061916	3,209	35	0.042233	0.045794	0.001257	1,729	1,051
2.18	0.031882	0.061213	3,145	38	0.043116	0.045480	0.001089	1,770	1,040
2.19	0.034022	0.100847	1	<1	0.043950	0.046859	0.001481	1,691	1,831
2.20	0.017486	0.034924	2	2	0.020864	0.021349	0.000392	1,796	843
2.21	0.016806	0.035522	1	2	0.020436	0.021520	0.000553	1,795	821
2.22	0.012667	0.027137	774	61	0.016229	0.017626	0.000468	1,784	630
2.23	0.012603	0.026796	3,451	272	0.016825	0.017357	0.000424	1,796	605
2.24	0.011219	0.027138	1	<1	0.013705	0.014151	0.000239	1,697	4,323
2.25	0.011451	0.036199	5	<1	0.014485	0.014804	0.000199	1,672	2,158
2.26	0.011281	0.041057	1	1	0.013974	0.014553	0.000283	1,736	2,119
2.27	0.010042	0.026587	14	7	0.013078	0.014129	0.000466	1,783	1,135
2.28	0.011799	0.031399	3	<1	0.016180	0.017226	0.000369	1,641	1,776
2.29	0.030760	0.060626	7,323	79	0.042713	0.045430	0.001327	1,757	1,045
2.30	0.012633	0.026965	2,120	170	0.016964	0.017554	0.000362	1,794	593
2.31	0.012965	0.026572	1,748	151	0.016284	0.017412	0.000514	1,788	578

Notes. Column f_{win} gives the best objective values obtained in the finals of the ROADEF/EURO Challenge 2016 across all the competing teams. On the right-hand side, columns Best and Average, respectively, present the best and the average objective values obtained by the matheuristic algorithm. Column σ gives the standard deviation over multiple runs, while columns Time and Iteration, respectively, report the average computing time and the number of iterations required to obtain the best solutions with our matheuristic algorithm. Note that the reported time is less than the maximum time limit if the best solution is reached before the maximum time limit has elapsed. In addition, columns Average, No. Attempt, and Time on the left-hand side, respectively, present the average objective value, number of attempts, and run time for the first feasible solution found by the constructive heuristic algorithm.

Table 14. Computational Results Under 3 Hours Runtime Limit

		Matheuristic				
Instance	f_{win}	Best	Average	σ	Time (s)	Iteration
2.12	0.010024	0.012582	0.013073	0.000426	1,0782	5,514
2.13	0.028875	0.033883	0.035785	0.001315	5,078	26,278
2.14	0.034971	0.041784	0.043862	0.001953	9,918	10,036
2.15	0.024993	0.031674	0.034354	0.001731	6,119	18,530
2.16	0.011783	0.016852	0.017084	0.000193	9,612	9,441
2.17	0.032130	0.043120	0.044007	0.000756	7,731	5,779
2.18	0.031882	0.042272	0.043314	0.000983	9,330	5,864
2.19	0.034022	0.041906	0.044288	0.001907	7,509	9,515
2.20	0.017486	0.018517	0.019157	0.000486	10,512	3,988
2.21	0.016806	0.018640	0.019110	0.000351	10,569	3,794
2.22	0.012667	0.015004	0.015733	0.000642	10,782	3,562
2.23	0.012603	0.014206	0.014953	0.000470	10,261	3,238
2.24	0.011219	0.013212	0.013676	0.000291	10,779	23,100
2.25	0.011451	0.013851	0.014068	0.000163	8,695	11,273
2.26	0.011281	0.013551	0.013832	0.000175	10,099	11,191
2.27	0.010042	0.012415	0.013025	0.000376	8,727	5,665
2.28	0.011799	0.015979	0.016940	0.000731	9,385	9,035
2.29	0.030760	0.040114	0.042603	0.001858	10,063	5,551
2.30	0.012633	0.014895	0.015603	0.000559	10,301	3,417
2.31	0.012965	0.013838	0.014834	0.000760	10,762	3,425

Notes. Column f_{win} gives the best objective values obtained in the finals of the ROADEF/EURO Challenge 2016 across all the competing teams. Columns Best and Average, respectively, present the best and the average objective values obtained by the matheuristic algorithm. Column σ gives the standard deviation over multiple runs, while columns Time and Iteration, respectively, report the average computing time and the number of iterations required to obtain the best solutions with our matheuristic algorithm. Note that the reported time is less than the maximum time limit if the best solution is reached before the maximum time limit has elapsed.

let o_{it} be the objective value for instance i, and let b_i be the best objective value for instance i obtained by all the participants. The normalized score of team t for instance i is $1 - \exp((b_i - o_{it})/b_i)$, while the total score is the sum of the normalized scores across all the instances. As seen in Table 12, the preliminary version of our matheuristic algorithm obtained the third place in the finals.

The winner of the challenge presented a sequence-based selection hyper-heuristic utilizing a hidden Markov model to control over 20 predefined low-level heuristics (operators) under an iterative framework. Compared with this work, the number of neighborhoods considered in the proposed algorithm is much smaller, and our neighborhood selection strategy is not as widely used as that of hyper-heuristics. To the best of our knowledge, these constitute the major disadvantages of our algorithm compared with the winner's algorithm.

To assess the effectiveness of our new algorithm, we conduct extensive tests on the complete problem benchmark and compare our outcomes with the best results reported in the finals of the ROADEF/EURO Challenge 2016. We run two sets of experiments. In the first experiment, we adopt the standard 30-minute runtime limit used in the competition and perform 30 independent runs for each instance. In the second experiment, we relax the runtime limit to 3 hours and perform 10 independent runs for each instance. Tables 13 and 14, respectively, show the computational results under these two runtime limits. Column f_{win} gives the best objective values obtained in the finals of the ROADEF/EURO Challenge 2016 across all the competing teams. On the right-hand side, columns Best and Average, respectively, present the best and the average objective values obtained by the matheuristic algorithm. Column σ gives the standard deviation over multiple runs, while columns Time and Iteration, respectively, report the average computing time and the number of iterations required to obtain the best solutions with our matheuristic algorithm. Note that the reported time is less than the maximum time limit if the best solution is reached before the maximum time limit has elapsed. In addition, columns Average, No. Attempt, and Time on the left-hand side of Table 13, respectively, present the average objective value, number of attempts, and run time for the first feasible solution found by the constructive heuristic algorithm.

Under the standard runtime limit, Table 13 discloses that the matheuristic algorithm produces high-quality solutions that are comparable to the best results

reported in the challenge. Furthermore, the proposed algorithm exhibits a stable performance with a standard deviation σ that is less than 0.002 for all the instances. Note that the normalized score of this updated algorithm based on the challenge rule is 5.10 versus 6.62 in the finals of the challenge. One observes that the gap between the initial solution obtained by the constructive heuristic and the best final solution is over 50% for all the instances. Notice that this gap is greater than 100% on 9 instances, implying that the proposed matheuristic is essential for improving the solution quality. For some instances—for example, 2.23, 2.29, and 2.30—the generation of the initial solution takes thousands of attempts and hundreds of seconds, due to multiple failures in finding a feasible solution. This suggests that there is still room for improvement by means of an improved algorithm for solution initialization, which could generate better starting points and save time for the optimization phase.

When the runtime limit is relaxed to 3 hours (10,800 seconds), Table 14 discloses better average logistic ratios for all the instances and further improvement of the best results for 18 out of the 20 instances, which highlights the search potential of our algorithm. The normalized score under this relaxed runtime condition goes down to 3.90. Despite the solution quality being improved, the best solutions are found after 10,000 seconds in half of the cases, which means that the optimization procedure did not converge within the extended runtime limit. One possible way to improve efficiency is to accelerate its components, including timing optimization and quantity decision, by means of faster exact/greedy algorithms to solve the subproblems or with parallel neighborhood evaluation techniques.

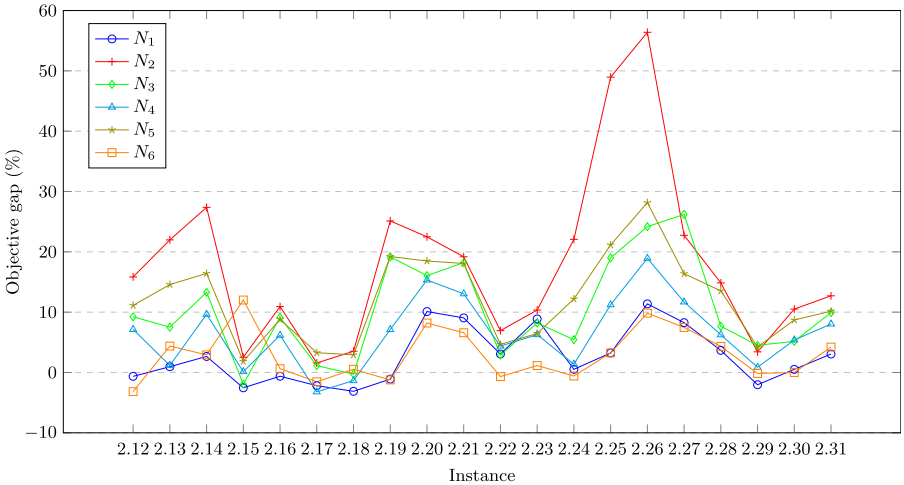
The executable code of our algorithm and the best solutions found so far by our solver are available at https://github.com/HUST-Smart/ROADEF2016-IRP-Results for future comparisons.

6.4. Importance of Different Neighborhood Structures

In our matheuristic algorithm, we propose six neighborhood structures to adjust routes with the multi-neighborhood search procedure, including adding an operation ($\mathcal{N}_1$), removing an operation ($\mathcal{N}_2$), replacing an operation ($\mathcal{N}_3$), swapping two operations within a shift ($\mathcal{N}_4$), moving an operation from one shift to another ($\mathcal{N}_5$), and adding a shift ($\mathcal{N}_6$). To ensure that these neighborhoods play a meaningful role, we conduct experiments to analyze the performance of the matheuristic algorithm in different cases. For each neighborhood, we test a less general version of our algorithm that excludes a given neighborhood and compare the average results with those obtained by the full version of the algorithm. For this purpose, we first perform 30 independent runs per instance for each version that excludes a specific neighborhood and for the complete algorithm and then calculate the average objective gap between the solutions obtained by the considered version and the complete algorithm. The resulting gaps for each version (labeled as N_1, N_2, N_3, N_4, N_5, and N_6) and for each instance are plotted in Figure 3. Positive gaps imply a deteriorating performance with respect to the original algorithm, whereas negative gaps imply an improved performance with respect to the complete version.

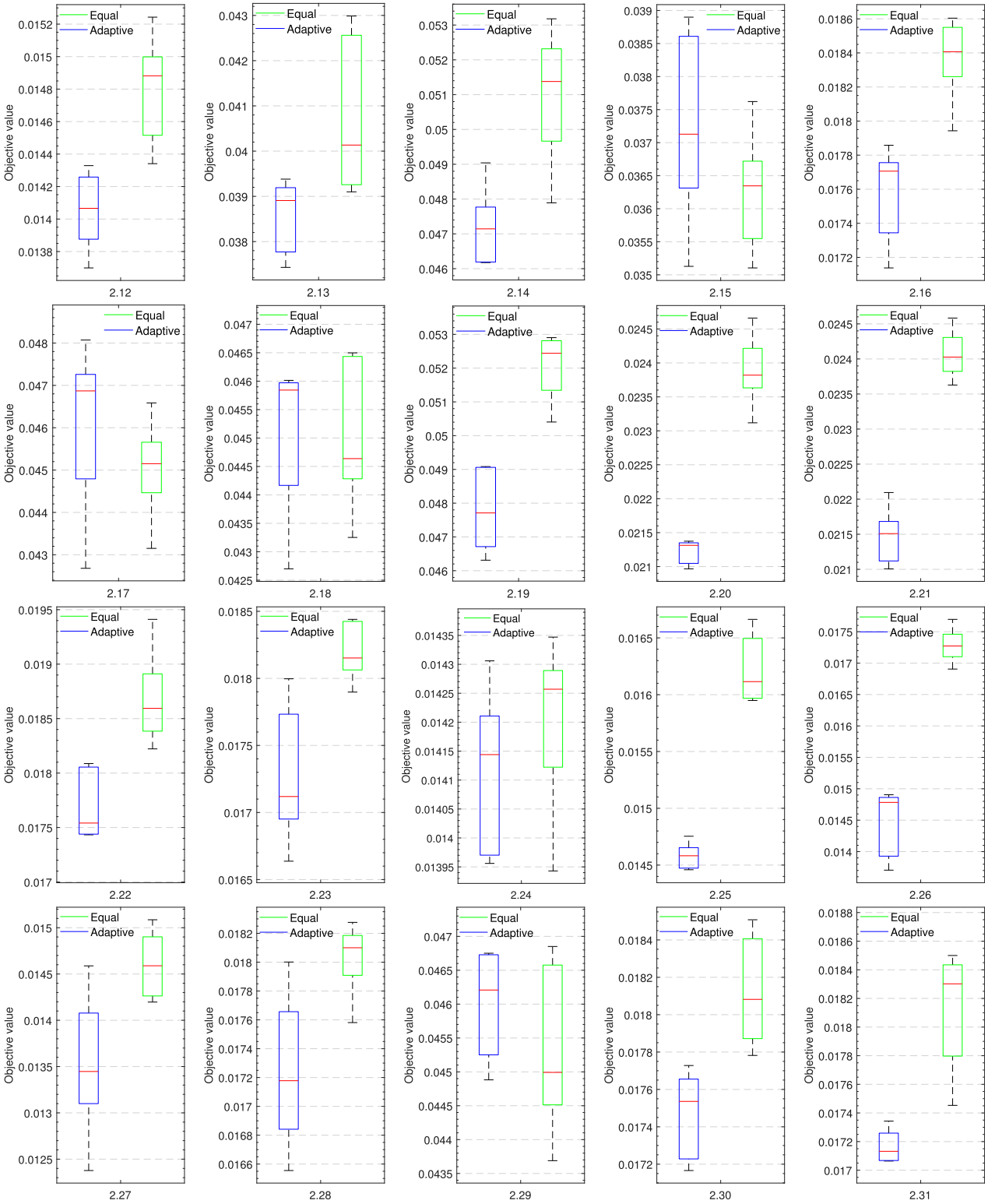
From Figure 3, we observe that the combined use of all six neighborhoods exhibits the best performance in general. Specifically, the complete algorithm reports better results for all the instances compared with the algorithms that exclude N_2 or N_5, for 18 instances compared with the algorithms that exclude N_3 or N_4, for 14 instances compared with the algorithm that

Figure 3. (Color online) Gaps Between the Average Objectives Obtained with the Full Version and the Versions Without Each Given Neighborhood



excludes N_6, and for 13 instances compared with the algorithm that excludes N_1. For those instances where the complete version is outperformed, there is no obvious benefit to exclude the related neighborhood. This experiment justifies the effectiveness of the proposed neighborhoods.

Figure 4. (Color online) Effectiveness of the Adaptive Neighborhood Selection Strategy



6.5. Effectiveness of the Adaptive Neighborhood Selection Strategy

As indicated above, we adopt an adaptive neighborhood selection strategy where a neighborhood is selected based on a probability that is adaptively adjusted to reflect historical performance. To determine whether this adaptive probability adjustment plays a meaningful role, we implement another selection strategy that gives each neighborhood an equal chance of being selected and compare the performance of these two versions. Figure 4 shows the outcome of executing 30 independent runs for each problem instance and for each selection strategy in a box-and-whisker plot.

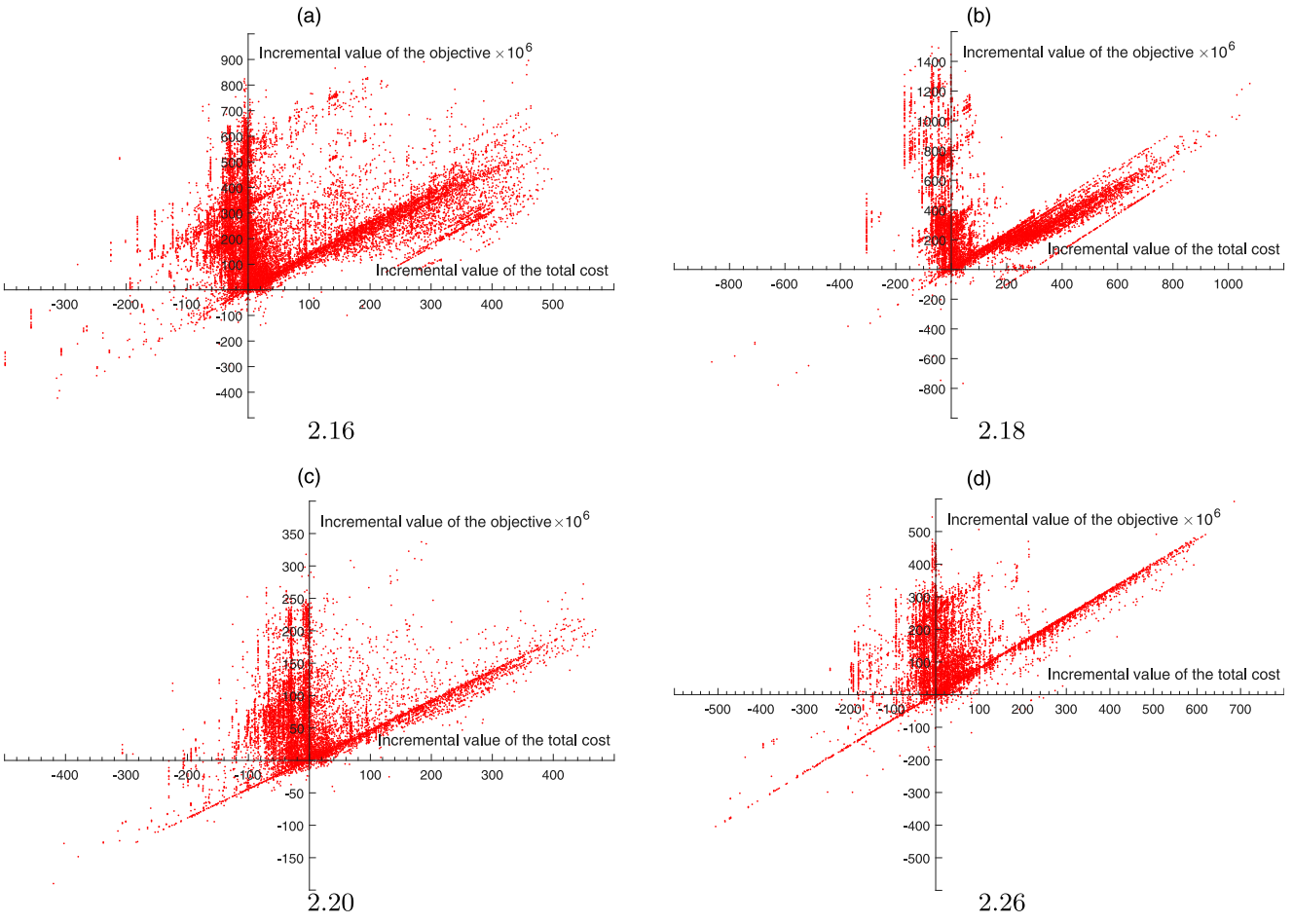
This figure demonstrates that the adaptive selection strategy is clearly superior to the equal probability version. The adaptive selection strategy obtains better overall results for 15 out of the 20 instances, while obtaining slightly worse or comparable results for the remaining 5 instances (2.15, 2.17, 2.18, 2.24, and 2.29). For 9 problem instances, the worst result of the adaptive selection strategy is better than the best result of the equal-chance selection strategy (2.16, 2.19, 2.20, 2.21, 2,22, 2.25, 2.26, 2.30, and 2.31), which highlights the effectiveness of the adaptive neighborhood selection strategy.

6.6. Significance of Neighborhood Estimation

To enhance the search efficiency, we adopt a neighborhood estimation based on the correlation between the incremental value of the total cost and the objective caused by the neighborhood move. We investigate the merit of this acceleration strategy by examining its performance on four typical instances (2.16, 2.18, 2.20, and 2.26). Note that similar results can be observed on other instances. For each neighborhood move considered by the local search procedure, we record the relation between its total cost and its objective value to observe if these values are correlated.

Figure 5 presents the distributions of these two values for all neighborhood solutions generated by the local search procedure and discloses that when the objective improves (i.e., the incremental value of

Figure 5. (Color online) Correlation Between the Incremental Value of the Total Cost and the Objective

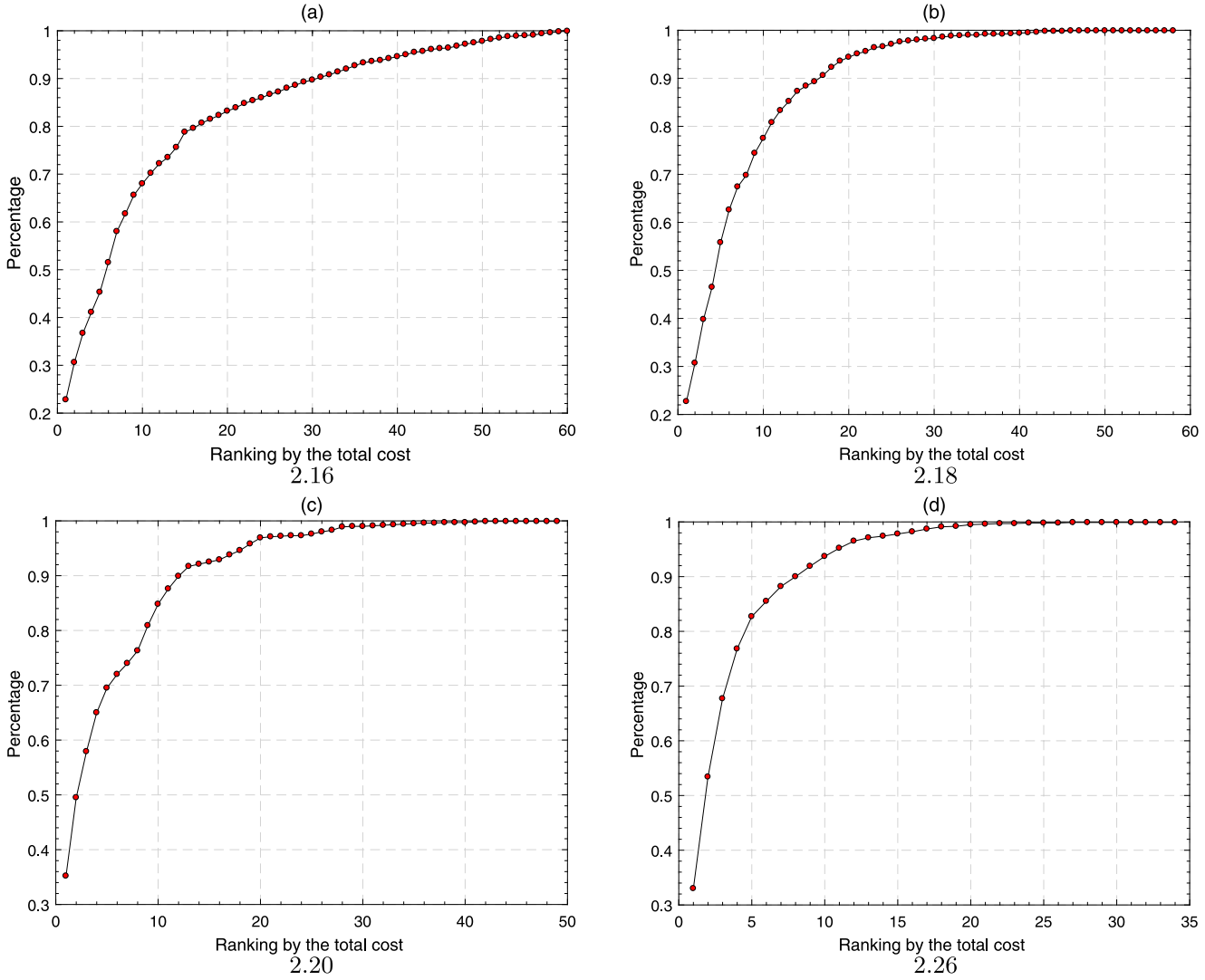


the objective is negative), the incremental value of the total cost is usually negative too. In other words, a better neighboring solution usually corresponds to an improved total cost. This phenomenon suggests that we should focus on the neighborhood moves with a good incremental value of the total cost instead of the logistic ratio. Thus, we can estimate the quality of the neighborhood moves by considering the total cost alone.

To further demonstrate this fact, at each iteration of the local search, we sort the neighborhood moves in an increasing order of their incremental values of the total cost and record the ranking of *elite* moves, considering the three best moves in the neighborhood. Figure 6 shows the statistical results of this experiment. A point on the curve such as (7, 0.7) means that the ranking of 70% of the *elite* moves at each iteration is less than 7.

Figure 6 shows that at least 70% of the *elite* moves are contained among those with a ranking of at most 10. As described above, the incremental value of the total cost incurred by a neighborhood move is fixed after the route is adjusted and the timing optimization model is solved. Once these two procedures are performed for all neighborhood moves, they are sorted by the incremental value of the total cost in an increasing order. Then, we can identify 70% of the *elite* moves by examining only the first 10 moves in the order at each iteration. According to this analysis, at each iteration of the local search, we divide the ordered neighborhood moves, or neighboring solutions, into several segments of equal size. If a single segment contains all the moves, it is equivalent to the classical local search based on the best improvement strategy. If there is only one move in each segment, it reduces to the well-known local search based on the

Figure 6. (Color online) Statistical Results of the Ranking of the Elite Neighborhood Moves



first improvement strategy. To make a balance between these two policies, we define a parameter ϕ (segment size) in order to control the trade-off between intensification and diversification, as shown in Table 11. Finally, at each iteration, we solve the quantity decision model for all the neighborhood moves in a given segment and select the best one. The ordered segments will be sequentially evaluated until an improved solution is obtained or until all the segments are evaluated.

6.7. Objective for the Timing Optimization Model

As the operation quantity is not specified when solving the timing optimization model, there is no straightforward objective for the model. Therefore, we must use an implicit objective function to evaluate the quality of different timing schedules.

As described above, we use $u_{g+1} - u_0$, which represents the time span of the shift, as the objective. In this configuration, the compact timing schedule is preferred, which minimizes the total wasted waiting time within the shift. To analyze the influence of different settings of the objective, we test two other objectives and compare them with the time span of the shift. The first configuration is $-u_0$ which prefers a "late beginning," where a shift is started as late as possible. This configuration can delay the start time of each operation and may be effective because later delivery implies that more quantity can be delivered. However, it also increases the risk of run-out. The second configuration is u_{g+1}, which prefers an "early ending," where a shift is finished as early as possible. Assuming the inventory levels of customers are close to the safety level, this objective might be helpful for avoiding shortage.

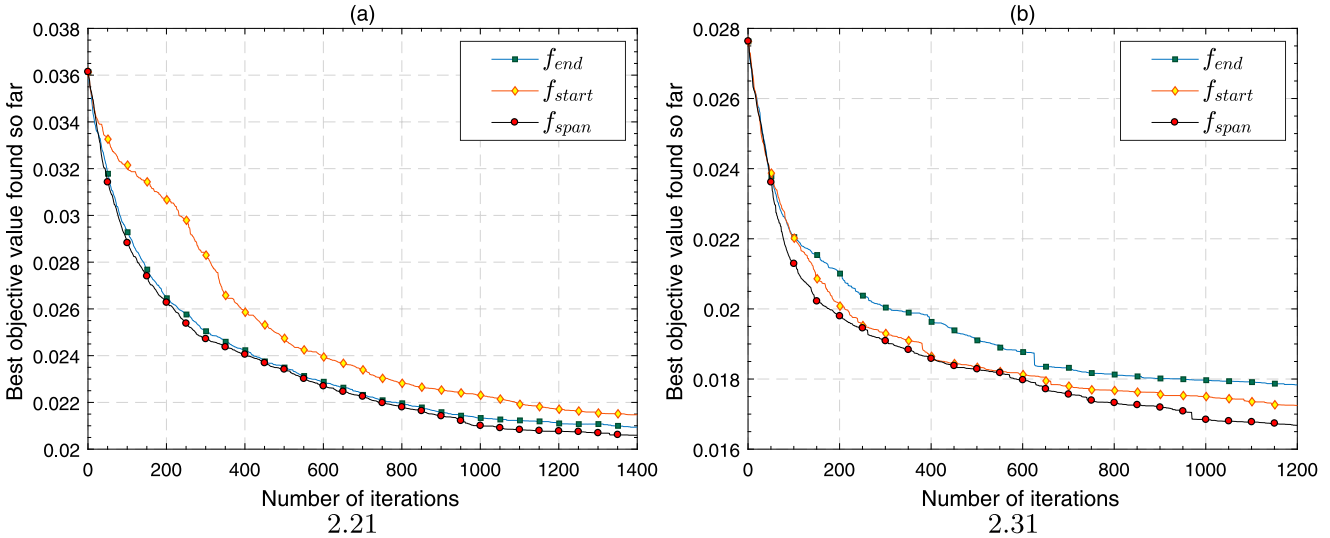
We select two typical instances and observe how the best objective value evolves with the search. Note that similar results can be observed on other instances. The comparison results are presented in Figure 7, using f_{start}, f_{end}, and f_{span} to denote the results obtained by the configurations $-u_0$, u_{g+1}, and $u_{g+1} - u_0$, respectively. For each instance and each configuration, we perform 30 independent runs.

Figure 7 shows that the overall performance of f_{span} outperforms other two configurations. A possible reason may be that $u_{g+1} - u_0$ provides a measure that identifies a better balance between $-u_0$ and u_{g+1}, thereby enabling the model to find a timing schedule that makes good use of the time resource and does not start the shift too early or finish the shift too late.

7. Conclusion

The integrated inventory-management and vehicle-routing problem (denoted as IRP-Challenge2016), originally proposed for the ROADEF/EURO Challenge 2016, aims to minimize the unit distribution cost while guaranteeing the quality of service to each customer. In this paper, we have presented a matheuristic algorithm for solving the IRP-Challenge2016 that integrates a local search-based metaheuristic with mathematical programming. The matheuristic adopts a new hierarchy of decomposition, which divides the problem into one master problem (routes adjustment) and two subproblems (timing optimization and quantity decision). The algorithm for the master problem uses a multineighborhood search metaheuristic to adjust the routes. The algorithms for the timing optimization and the quantity decision subproblems, respectively, introduce an MIP model and an LP model

Figure 7. Comparison Between Different Objectives for the Timing Optimization Model



2.21 2.31

that can be exactly solved by a mathematical solver such as Gurobi.

By testing on 20 benchmark instances from the ROADEF/EURO Challenge 2016, we have shown that the proposed matheuristic algorithm obtains outcomes comparable to the best results in the competition. Although our matheuristic algorithm obtained third place in the challenge, we show that there is still room for improvement, which may narrow the gap to the winner's algorithm. We anticipate that the exploitation of more complex neighborhoods and more advanced problem-specific heuristics will bring further benefits. The algorithm may also be improved by making use of information gathered from preprocessing, such as partitioning the nodes by location or consumption rate to obtain balanced clusters of customers, as in Cao and Glover (2010). The success of the matheuristic algorithm and the decomposition scheme on the IRP-Challenge2016 reaffirms the value of employing different strategies, establishing a balance between intensification and diversification, and incorporating problem-specific knowledge in designing algorithms for complex problems.

Acknowledgments

Zhipeng Lü is the corresponding author. The authors thank Dr. Fred Glover for his careful revision and valuable suggestions that led to a significant improvement of the paper. The authors thank Air Liquide and the committee of the ROADEF/EURO Challenge 2016 for providing this challenging problem and for the opportunity to test our algorithm on practical data. The authors also acknowledge Dr. Ahmed Kheiri for providing a brief description of his winning algorithm in the ROADEF/EURO Challenge 2016. Thanks are given to Liangwei Shen for his work on solution visualization, which helped the authors improve the algorithm. The authors also thank the editors and anonymous referees for their valuable comments that helped them to improve this paper.

References

Abdelmaguid TF, Dessouky MM, Ordóñez F (2009) Heuristic approaches for the inventory-routing problem with backlogging. *Comput. Industrial Engrg.* 56(4):1519–1534.

Adulyasak Y, Cordeau JF, Jans R (2013) Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS J. Comput.* 26(1):103–120.

Anghinolfi D, Gambardella LM, Montemanni R, Nattero C, Paolucci M, Toklu NE (2012) A matheuristic algorithm for a large-scale energy management problem. Lirkov I, Margenov S, Waśniewski J, eds. *Large-Scale Scientific Computing* (Springer, Berlin), 173–181.

Archetti C, Savelsbergh M (2009) The trip scheduling problem. *Transportation Sci.* 43(4):417–431.

Archetti C, Speranza MG (2014) A survey on matheuristics for routing problems. *EURO J. Comput. Optim.* 2(4):223–246.

Archetti C, Bertazzi L, Hertz A, Speranza MG (2012) A hybrid heuristic for an inventory routing problem. *INFORMS J. Comput.* 24(1):101–116.

Archetti C, Bertazzi L, Laporte G, Speranza MG (2007) A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Sci.* 41(3):382–391.

Avella P, Boccia M, Wolsey LA (2018) Single-period cutting planes for inventory routing problems. *Transportation Sci.* 52(3): 497–508.

Bajalinov EB (2013) *Linear-Fractional Programming: Theory, Methods, Applications and Software*, *Applied Optimization*, vol. 84 (Springer Science & Business Media, New York).

Bard JF, Huang L, Jaillet P, Dror M (1998) A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Sci.* 32(2):189–203.

Benoist T, Gardi F, Jeanjean A, Estellon B (2011) Randomized local search for real-life inventory routing. *Transportation Sci.* 45(3): 381–398.

Bertazzi L, Paletta G, Speranza MG (2002) Deterministic order-up-to level policies in an inventory routing problem. *Transportation Sci.* 36(1):119–132.

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

Cao B, Glover F (2010) Creating balanced and connected clusters to improve service delivery routes in logistics planning. *J. Systems Sci. Systems Engrg.* 19(4):453–480.

Charnes A, Cooper WW (1962) Programming with linear fractional functionals. *Naval Res. Logist.* 9(3-4):181–186.

Chien TW, Balakrishnan A, Wong RT (1989) An integrated inventory allocation and vehicle routing problem. *Transportation Sci.* 23(2): 67–76.

Coelho LC, Laporte G (2013a) A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *Internat. J. Production Res.* 51(23–24):7156–7169.

Coelho LC, Laporte G (2013b) The exact solution of several classes of inventory-routing problems. *Comput. Oper. Res.* 40(2):558–565.

Coelho LC, Cordeau JF, Laporte G (2012a) Consistency in multi-vehicle inventory-routing. *Transportaion Res. Part C: Emerging Tech.* 24:270–287.

Coelho LC, Cordeau JF, Laporte G (2012b) The inventory-routing problem with transshipment. *Comput. Oper. Res.* 39(11):2537–2548.

Coelho LC, Cordeau JF, Laporte G (2013) Thirty years of inventory routing. *Transportation Sci.* 48(1):1–19.

Coelho LC, Cordeau JF, Laporte G (2014) Heuristics for dynamic and stochastic inventory-routing. *Comput. Oper. Res.* 52(Part A): 55–67.

Cordeau JF, Laganà D, Musmanno R, Vocaturo F (2015) A decomposition-based heuristic for the multiple-product inventory-routing problem. *Comput. Oper. Res.* 55:153–166.

Cousineau-Ouimet K (2002) A tabu search heuristic for the inventory routing problem. *Proc. 37th Annual ORSNZ Conf.* (Operational Research Society of New Zealand, Auckland, New Zealand).

Desaulniers G, Rakke JG, Coelho LC (2016) A branch-price-and-cut algorithm for the inventory-routing problem. *Transportation Sci.* 50(3):1060–1076.

Dinkelbach W (1962) Die maximierung eines quotienten zweier linearer funktionen unter linearen nebenbedingungen. *Probab. Theory Related Fields* 1(2):141–145.

Federgruen A, Zipkin P (1984) A combined vehicle routing and inventory allocation problem. *Oper. Res.* 32(5):1019–1037.

Guerrero WJ, Prodhon C, Velasco N, Amaya CA (2013) Hybrid heuristic for the inventory location-routing problem with deterministic demand. *Internat. J. Production Econom.* 146(1):359–370.

Hewitt M, Nemhauser G, Savelsbergh M, Song JH (2013) A branch-and-price guided search approach to maritime inventory routing. *Comput. Oper. Res.* 40(5):1410–1419.

Kleywegt AJ, Nori VS, Savelsbergh MW (2004) Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Sci.* 38(1):42–70.

Liu SC, Chen AZ (2012) Variable neighborhood search for the inventory routing and scheduling problem in a supply chain. *Expert Systems Appl.* 39(4):4149–4159.

Liu SC, Lee WT (2011) A heuristic method for the inventory routing problem with time windows. *Expert Systems Appl.* 38(10): 13223–13231.

Lopes R, Morais VWC, Noronha TF, Souza VAA (2015) Heuristics and matheuristics for a real-life machine reassignment problem. *Internat. Trans. Oper. Res.* 22(1):77–95.

Lü Z, Hao JK (2010) Adaptive tabu search for course timetabling. *Eur. J. Oper. Res.* 200(1):235–244.

Mansi RD, Hanafi SD, Wilbaut C, Clautiaux FO (2012) Disruptions in the airline industry: Math-heuristics for re-assigning aircraft and passengers simultaneously. *Eur. J. Indust. Engrg.* 6(6): 690–712.

Mjirda A, Jarboui B, Macedo R, Hanafi S, Mladenović N (2014) A two phase variable neighborhood search for the multi-product inventory routing problem. *Comput. Oper. Res.* 52(Part B):291–299.

Moin NH, Salhi S, Aziz N (2011) An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *Internat. J. Production Econom.* 133(1):334–343.

Park YB, Yoo JS, Park HS (2016) A genetic algorithm for the vendor-managed inventory routing problem with lost sales. *Expert Systems Appl.* 53:149–159.

Qin L, Miao L, Ruan Q, Zhang Y (2014) A local search method for periodic inventory routing problem. *Expert Systems Appl.* 41(2): 765–778.

Ramalhinho-Lourenço H, Ribeiro R (2003) Inventory-routing model, for a multi-period problem with stochastic and deterministic demand. Technical report, Department of Economics and Business, Universitat Pompeu Fabra, Barcelona.

Solyali O, Süral H (2011) A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. *Transportation Sci.* 45(3):335–345.

Van Anholt RG, Coelho LC, Laporte G, Vis IF (2016) An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines. *Transportation Sci.* 50(3):1077–1091.

Wang Z, Lü Z, Ye T (2016) Multi-neighborhood local search optimization for machine reassignment problem. *Comput. Oper. Res.* 68:16–29.

Zhao QH, Chen S, Zang CX (2008) Model and algorithm for inventory/routing decision in a three-echelon logistics system. *Eur. J. Oper. Res.* 191(3):623–635.

Zhao QH, Wang SY, Lai KK (2007) A partition approach to the inventory/routing problem. *Eur. J. Oper. Res.* 177(2):786–802.