



Article

Iterated Local Search Approach to a Single-Product, Multiple-Source, Inventory-Routing Problem

Federico Alonso-Pecina ^{1,*}, Irma Yazmín Hernández-Báez ² and Roberto Enrique López-Díaz ²
and Martín H. Cruz-Rosales ¹

¹ Faculty of Accounting, Administration & Informatics, Universidad Autónoma del Estado de Morelos, Cuernavaca 62209, Mexico; mcr@uaem.mx

² Academic Direction of Engineering Information Technologies, Universidad Politécnica del Estado de Morelos, Jiutepec 62550, Mexico; ihernandez@upemor.edu.mx (I.Y.H.-B.); rlopezd@upemor.edu.mx (R.E.L.-D.)

* Correspondence: federico.alonso@uaem.mx

Abstract: We address an inventory-routing problem that arises in a liquid oxygen-producing company. Decisions must be made for the efficient transport of the product from sources to destinations by means of a heterogeneous fleet of trucks. This combinatorial problem has been stated as a constrained minimization one, whose objective function is the quotient of the operating cost divided by the total amount of delivered product. The operating cost comes from the distances traveled, the drivers' salary, and the drivers' overnight accommodation. The constraints include time windows for drivers and destinations, inventory safety levels, lower bounds for the quantity of product delivered to destinations, and maximum driving times. To approximate the optimal solution of this challenging problem, we developed a heuristic algorithm that first finds a feasible solution, and then iteratively improves it by combining the Metropolis criterion with local search. Our results are competitive with the best proposals in the literature.

Keywords: combinatorial optimization; iterated local search; heuristic; inventory routing problem; local search; simulated annealing

MSC: 90C27; 90C59; 90C90



Citation: Alonso-Pecina, F.; Hernández-Báez, I.Y.; López-Díaz, R.E.; Cruz-Rosales, M.H. Iterated Local Search Approach to a Single-Product, Multiple-Source, Inventory-Routing Problem. *Mathematics* **2024**, *12*, 991. <https://doi.org/10.3390/math12070991>

Academic Editors: Mohammad Shokouhifar, Frank Werner, Laith Abualigah and Seyedali Mirjalili

Received: 29 February 2024

Revised: 24 March 2024

Accepted: 25 March 2024

Published: 27 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In this paper we address the inventory routing problem (IRP) proposed in the ROADEF/EURO Challenge 2016 [1], which belongs to the general class of so-called inventory-routing problems. In the IRP, a schedule of truck drivers along a planning horizon is sought so as to deliver liquid oxygen (single product) to customers, minimizing the logistic ratio, and subject to the full satisfaction of operating constraints. The logistic ratio, a performance index for fluid transportation industries, is defined as the quotient of the operating cost and the amount of delivered product. The relevant costs come from the distance traveled by trucks, the drivers' salary, and the overnight accommodation for drivers (referred to as layover). Thus, in the IRP it is required to determine optimal itineraries for drivers (who, when, where), as well as the quantity of product to deliver at stops. For some customers, it is mandatory to keep their local inventory levels above prescribed safety stocks, while for others (known as 'call-in') special orders specify the amount and time for the delivery.

To approximate the optimal solution of the IRP, we propose here an Iterated Local Search (ILS) [2] approach consisting of two phases: the first one finds from scratch a feasible solution, the second iteratively improves it by performing a Metropolis cycle and a local search procedure employing various neighborhoods.

The rest of the paper is organized as follows: In Section 2 we succinctly comment on related work. A detailed description of the problem is provided in Section 3. Then,

Sections 4 and 5 are devoted to describe the first and the second phase of our approach, respectively. The corresponding computational experiments are presented in Section 6, followed by our conclusions in Section 7.

2. Related Work

Originally proposed by Bell et al. [3] in 1983, IRP has been the subject of much research, motivating a rich and varied literature. For a discussion on the most relevant articles published on the subject until 2013, we highly recommend the book by Coelho et al. [4]. Lenstra and Rinooy Kan [5] proved that, in general, IRP belongs to the NP-hard class, so exact solution methods are scarce, such as the branch-and-cut algorithm of Coelho and Laporte [6], who addressed the case with multiple trucks and products.

In regard to early heuristic approaches, it is worth mentioning the one by Dror and Levy [7] in two steps: a solution is first found by dealing with a Vehicle Routing Problem, then a route improvement procedure is applied.

Recently, a plethora of modern heuristic approaches have been proposed to solve different versions of the IRP; here some are mentioned briefly. The one by Adulyasak, Cordeau, and Jans [8] consists of an adaptive large neighborhood search to find initial solutions, followed by branch-and-cut algorithms. Cordeau et al. [9] developed a three-phase heuristic for a multiple-product problem, including replenishment plans, delivery sequence, and routing. In Shao et al. [10], a solution was heuristically obtained, based on a rolling time algorithm plus a greedy randomized adaptive search procedure, and neighborhood search.

Etebaria and Dabirib [11] analyzed a hybrid heuristic method involving five steps: initialization, demand generation, demand adjustment, inventory routing, and neighborhood search, all embedded in a simulated annealing framework. Shaabani and Kamalabadi [12] considered a case of multi-retailer perishable products, proposing a population-based simulated annealing algorithm. In the paper by Cheng, Wang, and Zhang [13], a series of mixed integer nonlinear programming models was constructed, together with linearization methods; then, a hybrid genetic algorithm based on allocation first and routing second was employed to find near-optimal solutions of these models.

Among the latest investigations those undertaken by C. Archetti, M.G. Speranza, and collaborators are outstanding; we briefly mention three of them. In [14], two policies were compared: Retailer-Managed Inventory (RMI) and Vendor-Managed Inventory (VMI). In the first policy, the customer controls the time and the quantities to be delivered; in the second policy the supplier monitors the customer's product levels, and makes decisions on the time and amount to deliver so as to avoid stock-outs. From computational experiments on 120 instances, they found that the second policy yielded the lowest costs.

Contrary to most previous studies, in [15] the considered objective function is the minimization of the logistic ratio, defined as the quotient of the total cost and the total quantity of product delivered. An advantage of using this objective function is that, in general, the customer inventory levels do not tend to be too low along the planning horizon.

Ref. [16] deals with the case when bounds are added to the amount to load or unload product. The proposed solution method comprises three phases: generation of initial solution, tabu search, and final improvement, obtaining good results in most of the considered 880 benchmark instances.

In the literature, there are few papers working in the same IRP tackled here. One of them, ref. [17], developed a branch-cut-and-price algorithm was developed. They used a surrogate relaxation in the constraints. They were able to obtain a feasible solution in the most of the instances proposed in [1]. In [18], a matheuristic was developed. They combined mathematical programming with a local search. They divided the problem into two phases: one to design the routes and other to decide the quantity to deliver and fit the route in the time windows. In [19], a fixed sequence subproblem was identified and tackled with a model to check if, with a sequence of visits given, there exists a feasible plan to deliver the liquid. Given a feasible sequence, this model optimizes the timing and quantity

in the visits. They used greedy heuristics and a mathematical model to design a sequence using column generation. Their method is efficient for small instances, and even improved the best results known in some instances. In [20], a hyperheuristic was developed to the IRP. The hyperheuristic was used to choose a low-level heuristic in every step of the search. They used hidden Markov chain to choose the heuristic. They obtained the best results in the ROADEF Challenge of 2016.

3. The Problem: Nomenclature and Notation

Throughout the planning horizon, a heterogeneous truck fleet, driven by an also heterogeneous staff of drivers, is used to transport liquid oxygen (the product) from a set of sources (production sites) to a set of destinations (customers). The depot, the sources, and the destinations are all interconnected by a road network within a geographical region.

In the IRP discussed here, it is required to determine optimal itineraries for drivers and trucks through the road network, as well as the time and quantity of product delivery at the destinations, all subject to a set of logistical and contractual constraints. For example, although for most customers, hereinafter referred to as ‘regular’, local inventory levels should be maintained above the safety stocks along the planning horizon, for some customers—called ‘special’—it is a must-do for orders specifying quantity and time for deliveries.

In this section we describe the IRP as a combinatorial optimization problem, making explicit its objective function and constraints, as well as the required nomenclature and notation.

3.1. The Drivers

Let A be the set of drivers, all initially located in the depot d . The following data are given to each driver $a \in A$:

- A non-empty set $W(a)$ of availability time windows;
- A cost $C_1(a)$ per each time unit on duty;
- A lower bound $\kappa(a)$ for resting time between any two itineraries;
- An upper bound $\psi(a)$ for uninterrupted driving time.

3.2. The Trucks

Let Θ denote the set of trucks, all initially located at the depot d . For each truck $\sigma \in \Theta$, let $C_2(\sigma)$ be the cost per traveled distance unit, and $w(\sigma)$ its volume capacity. Also, the binary constant $\rho_{a\sigma}$ is equal to one if and only if driver $a \in A$ is allowed to drive truck σ .

3.3. The Product Sources

The production sites are referred to here as *sources*, their set is denoted O , and ℓ_j is the loading time at source $j \in O$. Also, the binary constant $m_{\sigma j}$ is equal to one if and only if truck $\sigma \in \Theta$ is allowed to load product at source $j \in O$.

3.4. The Destinations

Let $D = R \cup S$ be the set of destinations, where R corresponds to the set of ‘regular’ customers, and S corresponds to the set of ‘special’ customers. For each destination $d \in R$ the following information is provided:

- Deliveries can only occur within a given set of time windows.
- v_d is the quantity of product in the tank of destination d at time zero.
- The consumption forecast along the planning horizon in an hourly basis.
- The unloading time ℓ_d , and the safety stock \underline{v}_d .
- The lower bound γ_d for the amount to deliver at each visit of any truck.
- A binary constant $u_{\sigma d}$, which is equal to one if and only if truck $\sigma \in \Theta$ is allowed to unload product at destination $d \in R$.

For each destination $d \in S$ a set of specific orders is given, indicating lower and upper bounds on the quantities to be delivered, as well as the corresponding time windows.

A layover enables a driver to travel for an extended duration, covering a larger area. A number of destinations, called *layover destinations*, cannot be served in a single day because they are too far from the depot. When serving those destinations, it is possible to include into a shift, a resting time φ (called layover time) in any location (not necessarily a layover destination), incurring a constant cost C_3 . No shift can include two or more layovers.

3.5. The Objective Function

The objective function of the IRP is to minimize the Logistic Ratio, namely, the quotient of the total cost (which comes from the drivers on duty, the traveled distances, and the layovers) and the quantity of product delivered over the planning horizon. The cost related to the driver's salary is proportional to the duration of the shifts, which includes the driving time, the idle time, and the loading/unloading times. On the other hand, the cost due to the traveled distances is mainly related to the fuel consumption of trucks, and each layover has a fixed cost, covering the hotel fare.

3.6. The Model

Let $V := O \cup D \cup \{\mathbf{d}\}$, where \mathbf{d} denotes the depot, O the set of sources, and $D = R \cup S$ the set of destinations; R and S being, respectively, the set of 'regular' and 'special' customers. The elements of V are indexed $0, 1, \dots, n$, and referred to as *locations*. Also, $L \subset D$ is the set of layover destinations.

The length of the planning horizon is denoted \mathcal{L} .

The time for a truck to cover the distance between locations $i, j \in V$ in either direction is denoted $h(i, j)$. Recall ℓ_j is the required time to load (or unload) product at location j . A *tour* is a quadruple $\lambda = (u, t, t', x)$ where

- u is a finite sequence of locations $u = (u_0, u_1, \dots, u_n)$, $n \geq 2$, where either
 - $u_0 = u_n = \mathbf{d}$ (u starts and ends at the depot) and $u_j \in V$, for $j = 1, \dots, n-1$ (the depot is not an intermediate location in u), or;
 - $u_0 = \mathbf{d}$ (u starts at the depot) and $u_j \in V$, for $j = 1, \dots, n$ (the depot is only at the beginning of the sequence), or;
 - $u_n = \mathbf{d}$ (u ends at the depot) and $u_j \in V$, for $j = 0, \dots, n-1$ (the depot is only at the end of the sequence).
- $t = (t_0, t_1, \dots, t_n)$ and $t' = (t'_0, t'_1, \dots, t'_n)$ contain timing information on u as follows. The departure time from u_0 is $t'_0 \geq 0$, the arrival time to u_n is $t_n \leq \mathcal{L}$, and for $j = 1, \dots, n-1$, the time of arrival to (resp. departure from) location u_j is t_j (resp. $t'_j = t_j + \ell_j$), such that the interval $[t_j, t'_j]$ is completely contained in a time window of location u_j ; also $t_j \geq t'_{j-1} + h(u_{j-1}, u_j)$, for $j = 1, \dots, n$. We consider t_0 and t'_n as being immaterial.
- $x = (x_0, x_1, \dots, x_n)$, where $x_j \geq 0$ ($j = 1, \dots, n-1$) is the amount of unloaded product at location u_j . Clearly, if $u_j \in O \cup \{\mathbf{d}\}$ then $x_j = 0$.

In the sequel Ω denotes the set of tours. For each tour $\lambda = (u, t, t', x) \in \Omega$, sets $O(\lambda) := \{j \mid 0 \leq j \leq n, u_j \in O\}$ and $D(\lambda) := \{j \mid 0 \leq j \leq n, u_j \in D\}$ correspond, respectively, to the visited sources and destinations, $\theta(\lambda)$ is the total traveled distance, $\chi(\lambda) := \sum_{j=1}^{n-1} x_j$ is the amount of the delivered product, and $\underline{\delta}(\lambda) := t_0$ and $\bar{\delta}(\lambda) := t'_n$ are its time limits.

Now, recall A and Θ are the sets of drivers and trucks, respectively, and define a *route* as a triple $(a, \sigma, \lambda) \in A \times \Theta \times \Omega$, where $\lambda = (u, t, t', x)$ and $u = (u_0, u_1, \dots, u_n)$, simultaneously satisfying

- $\rho_{a\sigma} = 1$ [driver $a \in A$ is allowed to drive truck $\sigma \in \Theta$];
- $\xi_{\sigma u_j} = 1$ for all $j \in O(\lambda)$ [truck σ is allowed to load product at every source in tour λ];

- $u_{\sigma u_j} = 1$ for all $j \in D(\lambda)$ [truck σ is allowed to unload product at every destination in tour λ];
- $t'_n - t_0 \leq \psi(a)$ [driver a does not exceed his/her maximum allowed driving time $\psi(a)$ in tour λ];
- The period of time $[t_0, t'_n]$, denoted $\epsilon(a, \sigma, \lambda)$, is entirely contained in a time window of driver a .

Let Δ denote the set of possible routes, which we classify as round, forward, and backward. A *round route* starts and ends at the depot, a *forward route* starts at the depot and ends at some other location, a *backward route* starts at some source or destination and ends at the depot. A *shift* is either a round route or a pair of routes—one forward, one backward—such that

- They share a driver and truck, and;
- A layover destination is visited in either route, and;
- The forward route ends at the same location where the backward route starts, allowing enough resting time for the driver, namely, no less than φ .

Every couple of shifts are *compatible* unless:

- They share driver or truck and they intersect in time, or
- they share driver, say driver a , and between them there is no $\kappa(a)$ time for rest, or
- they share some destination, and their visiting time overlaps.

At time zero: (1) an unlimited amount of product is available in each source, (2) all drivers and trucks are available at the depot, with trucks holding some amount ≥ 0 of product in their tank, and (3) each regular destination has some amount of product above their safety stock. Moreover, the (not necessarily constant) consumption rate along the planning horizon is known for each regular destination, and for each special destination a set of specific orders is provided, each consisting of a lower and an upper bound on the quantity to be delivered as well as a corresponding time window. A *program* is a non-empty set of pairwise compatible shifts satisfying the following constraints:

- Each truck σ loads product to maximum capacity $w(\sigma)$ at every visited source.
- The quantity unloaded by a truck at destination $d \in R$ is always $\geq \gamma(d)$, and less than or equal to what it carries.
- Along the study horizon, the inventory level at each destination $d \in R$ is kept between safety stock $\underline{v}(d)$ and local capacity $\bar{v}(d)$.
- The quantities unloaded from trucks at each destination $d \in S$ satisfy the lower and upper bounds established in the corresponding specific orders.

Thus, the aim of the IRP is to determine a program P minimizing the logistic ratio $z = (C_d + C_f + C_l) / Q$, where Q is the total amount of oxygen (the delivered product), and costs C_d , C_f , C_l , arise from the in-service drivers, the distance traveled by trucks, and the carried-out layovers, respectively. As

$$\begin{aligned} C_d &= \sum_{a \in A} C_1(a) \sum_{(a, \sigma, \lambda) \in P} \epsilon(a, \sigma, \lambda), \\ C_f &= \sum_{\sigma \in T} C_2(\sigma) \sum_{(a, \sigma, \lambda) \in P} \theta(\lambda), \\ C_l &= C_3 \cdot |F|, \text{ and} \\ Q &= \sum_{(a, \sigma, \lambda) \in P} \chi(\lambda), \end{aligned}$$

we obtain

$$z = \frac{\sum_{a \in A} C_1(a) \sum_{(a, \sigma, \lambda) \in P} \epsilon(a, \sigma, \lambda) + \sum_{\sigma \in T} C_2(\sigma) \sum_{(a, \sigma, \lambda) \in P} \theta(\lambda) + C_3 \cdot |F|}{\sum_{(a, \sigma, \lambda) \in P} \chi(\lambda)}.$$

4. Finding a Feasible Solution

For any truck $\sigma \in \Theta$, we recall $w(\sigma)$ is the maximum capacity of its tank. Thus, an upper bound on the amount that truck σ can deliver in any visit to any destination $d \in R$, is $\zeta_d \leftarrow \min\{w(\sigma), \bar{v}(d) - \underline{v}(d)\}$, where $\bar{v}(d)$ and $\underline{v}(d)$ denote, respectively, the storage capacity and safety stock at destination d . Our strategy to find an initial feasible solution consists in successively selecting—in a rotating manner—an element $\mu \in \{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. Then, for each chosen μ a set of routes is produced such that every time destination d is served by truck σ no less than $\mu \times \zeta_d$ units are delivered.

In the Algorithm 1, μ is the parameter passed to the algorithm that tries to find a feasible solution, with all its deliveries of at least $\mu \times \zeta_i$ units. Although in our experiments, we set $MAX \leftarrow 10,000$, for most times the algorithm obtains a feasible solution in very few iterations. The following algorithm iteratively chooses a driver–truck combination, and tries to construct a route.

Algorithm 1: Finding a feasible solution

```

(1)  $\mu \leftarrow \{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$ ;
(2)  $ban \leftarrow false; i \leftarrow 0$ ;
(3) While  $i \leq MAX$  and  $sol == \emptyset$  do
(4)    $k \leftarrow (i \bmod 9)$ ;
(5)    $sol \leftarrow Feasible\_Solution(\mu[k])$ ;
(6)    $i \leftarrow i + 1$ ;
(7) EndWhile
(8) return  $sol$ ;

```

At the beginning of the Algorithm 2, many variables are initialized (line 1), including the list of routes (the empty set at the beginning), the time when the trucks are available (all trucks are available at time zero), the time when the drivers are available (when their first time window starts), etc.

Algorithm 2: $Feasible_Solution(\mu)$

```

(1)  $Initialize\_variables()$ ;
(2)  $order\_destinations()$ ;
(3)  $d \leftarrow choose\_driver()$ ;
(4) While  $d \neq \emptyset$  do
(5)    $time\_slot_d \leftarrow Get\_next\_time\_slot(d)$ ;
(6)    $\sigma \leftarrow choose\_truck(d)$ ;
(7)    $t_{ini} \leftarrow calculate\_time\_initial()$ ;
(8)    $L[d][time\_slot_d] \leftarrow calculate\_route(\mu)$ ;
(9)    $Update\_destinations(L[d][time\_slot_d])$ ;
(10)   $order\_destinations()$ ;
(11)   $d \leftarrow choose\_driver()$ ;
(12) EndWhile
(13) If  $Feasible\_Solution(L)$  then
(14)    $sol \leftarrow L$ ;
(15) else
(16)    $sol \leftarrow \emptyset$ ;
(17) endif
(18) return  $sol$ ;

```

For $c \in R$, let \bar{v}_c be the time when safety level \underline{v}_c is attained—easily computed from the initial inventory ν_c , and the forecasted consumption rate for c —and denote $\pi_{c,d}$ ($\pi_{c,d} \leq \bar{v}_c$)

the latest time in which driver $d \in A$ can arrive to c with the product to deliver, allowing delivery within one time window of c , and one time window of d .

For every destination $i \in D$ we calculate tim_i as the time when the product in the destination tank attains its safety level or, in case of a special destination, the time of their next order. However, if tim_i does not fit in a time window of i , then tim_i is decreased until it fits in the preceding time window, leaving enough time to unload, and provided that the driver is allowed to serve it. Then, we order the set $\{tim_i \mid i \in D\}$ in a non-decreasing order by means of the function *order_destinations()* (lines 2 and 10 of the algorithm).

Now, the algorithm chooses the earliest available driver d , and in case of ties one is randomly chosen. Hereupon, the earliest available truck σ that d is allowed to be used is chosen (lines 3 and 11). We denote tc as the earliest time when the couple (d, σ) is available. The function returns the empty set if there is no driver that can work in the time interval or when some destination needs a service to maintain their tank level above the minimum allowed before drivers are able to serve them (the special orders are also considered).

For the selected driver, we obtain the number of the next route to calculate (line 5). For driver d , if we have more than one possible truck for the driver, we choose the truck available at the earliest time; in case of ties, one is randomly chosen (line 6). In line 7 we obtain the time t_{ini} when the combination of driver d and truck σ can operate.

To determine the route we proceed as follows: We obtain the first destination i_1 that needs to be served. If time t_{ini} is lower than the first destination i that needs to be served with a difference of at least $\psi + \kappa_d + 60$ minutes, then the first destination chosen is the first one available in the time period when it can be served by σ . Otherwise, the first destination is chosen according to the order of the urgent list of the destinations.

The list $L[d][time_slot_d]$ (or l_δ) is the order in which the destinations assigned to driver d are served, which are initialized with the empty set. For the remaining destinations we proceed as follows: the algorithm tries to assign the next destination in the ordered list that can be visited by the truck, if it succeeds, it tries the next destination that can be visited in the list; if it succeeds, the algorithm assigns the destination to the driver, otherwise it continues with the next destination until it can no longer add any destinations (seeks through the list looking for candidates). Every time we try to aggregate a new destination, we try all the possibilities in the list.

Once we add one destination to the route, we compute the maximum time by which the destination needs to be served again, in order to maintain feasibility, so when we try to add a new destination, this maximum time cannot be reduced (the amount that the truck served can be reduced). At each visit, the quantity of product delivered to a destination is the minimum between the product available in the truck and the capacity of the destination's tank; the minimum amount served to all destinations will be $\mu \times \zeta_i$.

At the beginning of each scheduling of a truck, if it has less than 50% of available product, it recharges in a source. If the truck is out of product after visiting a destination, and the driver has enough time, it goes to the nearest source to recharge. At the end of the route, if the driver has enough time, they try to refill the truck.

When the schedule of the combination driver-truck-time is assigned, a reordering of destinations is performed, considering new deliveries of the product. At the end of cycle, if the solution is feasible, it is returned, otherwise the empty set is returned.

With the algorithm described in this section, we were able to find feasible solutions in few seconds for all the instances.

5. Improving the Solution

The second stage makes intensive use of seven *neighborhoods* in order to explore the space solution. For $\xi = 1, \dots, 7$, neighborhood $N_\xi(Y)$ consists of all feasible solutions obtained from a feasible solution Y by:

- Swapping two destinations of some pair of routes, if $\xi = 1$.
- Removing an element from some route, if $\xi = 2$.

- Removing an element from some route, and adding this element to another route, if $\xi = 3$.
- Altering in some route the order in which the destinations are visited, within a prescribed time period, if $\xi = 4$.
- Adding to some route a new destination, if $\xi = 5$.
- Eliminating one route, if $\xi = 6$.
- Delaying the starting time of some route, if $\xi = 7$.

All the neighbours are generated keeping feasibility, so, infeasible neighbours are discarded.

In Algorithm 3, the following schema is used in to improve the solution.

Algorithm 3: Iterated Local Search

```

(1)   $best\_sol \leftarrow sol \leftarrow Find\_Feasible\_Solution();$ 
(2)   $cont \leftarrow 0;$ 
(3)  While  $cont \leq MAXTRIES$  and  $t_{actual} \leq MAXTIME$  do
(4)     $sol \leftarrow SIMULATEDANNEALING(sol);$ 
(5)     $sol \leftarrow LOCALSEARCH(sol);$ 
(6)    If  $f(sol) < f(best\_sol)$  then
(7)       $best\_sol \leftarrow sol;$ 
(8)       $cont \leftarrow 0;$ 
(9)    else
(10)      $cont \leftarrow cont + 1;$ 
(11)   EndIf
(12) EndWhile
(13)  $sol \leftarrow DOUBLE(sol);$ 
(14)  $Print(best\_sol);$ 

```

In (1), an initial feasible solution is obtained with the procedure described in Section 4. Lines (3)–(12) contain the main loop of the algorithm. In (3), as long as the number of attempts does not reach MAXTRIES or the time limit is not reached, the algorithm will try to improve the solution. In (4)–(5), the procedures SimulatedAnnealing and LocalSearch are successively performed attempting to improve the initial solution. In (6)–(11), if the current best solution is improved, then the new solution is accepted, otherwise the counter $cont$ is increased. In (13), the procedure $DOUBLE(sol)$ randomly produces two neighbors, one after the other; the first one, say y , belongs to $\bigcup_{j=1,2,3,5,7} N_j(sol)$, the second one belongs to $\bigcup_{j=1,2,3,4,5,7} N_j(y)$. Finally, in (14) the best solution found is rendered together with its cost. In Algorithm 4, the Simulated Annealing procedure is described.

In line (1) of the Simulated Annealing procedure, the initial temperature is calculated as the sum of the number of drivers $|A|$, the number of destinations $|D|$, the number of hours in the planning horizon, and the number of trucks $|\Theta|$. Here, ι stands for the system temperature, and the parameters T_o, T_f, α, ν , denote, respectively, the initial and the final temperature, the cooling factor, and the internal cycle length. The procedure $NEIGHBOR(S)$ randomly produces a feasible solution by iteratively looking at neighborhoods $N_1(S), N_2(S), N_3(S), N_5(S)$, and $RAND$ delivers a uniformly distributed random number in the interval $(0, 1)$. The algorithm exits as soon as an improvement on the best solution in the internal cycle is found. The procedure $LOCALSEARCH(sol)$ considers one by one all elements of $\bigcup_{j=1}^{j=7} N_j(sol)$ rendering the first improving solution found. All these procedures will be executed until the time limit is reached.

Algorithm 4: SimulatedAnnealing(S)

```

(1)   $T_0 \leftarrow |D| + |C| + \text{units} + n_h$ 
(2)   $\iota \leftarrow T_0;$ 
     $S^* \leftarrow S; \text{ban} \leftarrow \text{true};$ 
(3)  While  $\iota \geq T_f$  and  $\text{ban}$  do
(4)     $k \leftarrow 0; \text{ban} \leftarrow \text{false}$ 
(5)    While  $k \leq \nu$  do
(6)       $S' \leftarrow \text{NEIGHBOR}(S);$ 
(7)       $\delta \leftarrow z(S') - z(S);$ 
(8)      If  $\text{RAND}(0,1) < e^{-\delta/\sigma}$  or  $(\delta < 0)$  then
(9)         $S \leftarrow S';$ 
(10)     EndIf
(11)      $k \leftarrow k + 1;$ 
(12)     If  $S < S^*$  then
(13)        $S^* \leftarrow S;$ 
     $k \leftarrow 0; \text{ban} \leftarrow \text{true};$ 
(14)   EndIf
(15)   EndWhile
(16)    $\iota \leftarrow \alpha \iota;$ 
(17) EndWhile

```

6. Results

Our algorithms were executed in a computer with the following features: Windows 10, Intel i64470 processor, 3.4 GHZ, 16 GB, and Visual Studio 2012 under C++ language.

To test the algorithm we used the 31 instances published in [1]. These are all the instances available to the specific IRP tackled in this paper. Tables 1 and 2 show general information for the 31 instances. Instances of Table 1 are smaller than those of Table 2, and correspond to a simplified version of the IRP: there is a single source, every driver is associated with only one truck, there are neither special customers nor layovers nor unloading lower bounds, and destinations have only one time window. Instances in Table 2 have all possible constraints and features of the problem.

Table 1. General data of simplified instances of the IRP.

Instance	Name	Number of Destinations	Number of Drivers	Number of Trucks	Number of Hours
1	V_1.1	12	2	2	720
2	V_1.2	12	2	2	720
3	V_1.3	53	1	1	240
4	V_1.4	64	2	2	240
5	V_1.5	54	2	2	240
6	V_1.6	54	2	2	840
7	V_1.7	99	6	6	240
8	V_1.8	99	6	6	82
9	V_1.9	99	6	6	840
10	V_1.10	89	3	3	240
11	V_1.11	89	3	3	840

Table 2. General data of complex instances of the IRP.

Instance	Name	Number of Destinations	Number of Drivers	Number of Trucks	Number of Sources	Number of Hours
12	V2.12	324	13	15	1	240
13	V2.13	53	5	5	1	240
14	V2.14	53	5	5	1	840
15	V2.15	134	4	3	1	240
16	V2.16.2	184	7	4	1	240
17	V2.17	134	4	3	1	840
18	V2.18	134	4	3	1	840
19	V2.19	53	5	5	1	840
20	V2.20.2	184	7	4	1	840
21	V2.21.2	184	7	4	1	840
22	V2.22	324	13	15	1	504
23	V2.23	324	13	15	1	504
24	V2.24	32	5	6	2	240
25	V2.25	32	5	6	2	840
26	V2.26	32	5	6	2	840
27	X1	324	13	15	1	240
28	X2	184	7	4	1	240
29	X3	134	4	3	1	840
30	X4	324	13	15	1	504
31	X5	324	13	15	1	504

The values of the best results obtained by running our algorithm 100 times are shown in Tables 3 and 4. For each simplified instance k , columns 2 and 3 of Table 3 indicate, respectively, the value of the best known solution, and the value of the best results obtained by running our algorithm 100 times are shown in Tables 3 and 4. For each simplified instance k , columns 2 and 3 of Table 3 indicate, respectively, the value of the best known solution obtained in [1], and the best value obtained by us. In the Table 4, HBCP is the Heuristic Branch-Cut-and-Price Algorithm developed in [17]. They used a computer with Intel(R) Core(TM) i7-5600U CPU @ 2.60 GHz and Cplex 12.6 to run their algorithm. HBCP uses different time limits to its execution: three, six, and twelve hours; these times are in function of the instance size. MA is the matheuristic algorithm published in [18]; this algorithm was programmed in C++ and compiled by Visual C++ 2015 the models used were solved using Gurobi 6.5.2 $\times 64$. MA was tested in two runs using a time limit of 1800 s and 3 h, respectively. MFPSR is the matheuristic with fixed sequence reoptimization of [19] it was coded in C++ and solved by CPLEX 12.7.0 with one single thread, and the running time of the algorithm in every instance was 1800 s. Finally, HSS is the Heuristic Sequence Selection designed by Kheiri [20]; he used 1800 s as a time limit too. Although there were more competitors in the ROADEF Challenge [1] these are the only papers published, and we included the best solutions known. Each run of our algorithm in Tables 3 and 4 took 1800 s.

Table 3. Results of our algorithm on simplified instances of the IRP.

Instance k	Best Known Value [1]	Value Yielded by ILS $q(k)$
1	0.027466	0.027619
2	0.027304	0.027536
3	0.013279	0.014919
4	0.015495	0.019625
5	0.011877	0.012021
6	0.012812	0.013306
7	0.012890	0.013860
8	0.007756	0.008169
9	0.015279	0.015418
10	0.018941	0.019625
11	0.028666	0.029427

Table 4. Results of our algorithm on complex instances of the IRP. All the algorithms ran for 1800 s.

Instance <i>k</i>	Best Solution of [1]	ILS	HBCP [17]	MA [18]	MFSR [19]	HSS [20]
12	0.010024	0.010776	0.016290	0.013304	0.010046	0.010266
13	0.028875	0.034784	0.034803	0.034262	0.029789	0.030768
14	0.034972	0.048244	0.047637	0.044646	0.036996	0.037582
15	0.024993	0.031317	0.034528	0.033868	0.025894	0.026608
16	0.011783	0.014439	0.016826	0.016728	0.012207	0.012420
17	0.032130	0.037611	0.049813	0.042233	0.031260	0.031538
18	0.031882	0.037524	0.047547	0.043116	0.032700	0.033018
19	0.034022	0.047624	0.041487	0.043950	0.035183	0.036018
20	0.017486	0.019584	0.026095	0.020864	0.018489	0.018656
21	0.016806	0.019955	0.025374	0.020436	0.017051	0.017210
22	0.012667	0.013655	0.018819	0.016229	0.012667	0.012992
23	0.012603	0.013488	—	0.016825	0.013003	0.013311
24	0.011219	0.015778	0.014175	0.013705	0.012523	0.013033
25	0.011451	0.016523	0.013194	0.014485	0.012137	0.012411
26	0.011281	0.016038	0.013580	0.013974	0.012691	0.012866
27	0.010042	0.010904	0.015974	0.013078	0.010010	0.010234
28	0.011799	0.014222	0.016886	0.016180	0.012214	0.012410
29	0.030760	0.038088	—	0.042713	0.031584	0.031905
30	0.012633	0.013597	0.018325	0.016964	0.012690	0.013015
31	0.012965	0.013334	0.018101	0.016284	0.013681	0.013994

In Table 4, we can observe that in general, the results of the ILS are competitive and obtains the third place among the algorithms proposed in the literature. Particularly, in the instances 31 (or X5), when we ran the algorithm for 3 h like [18], we obtained a result of **0.012687**, improving the best result known in the literature. The feasibility and value of the solutions obtained by ILS were verified by the checker provided by [1].

7. Conclusions

We have presented above an original approach to the IRP problem which consists of two main procedures. While the former is intended to produce an initial feasible solution with little computational effort, the second provides heuristics that iteratively improve the initial solution. Our approach is compared with other approaches to this challenging problem, obtaining competitive results. We improved the best result known for the instance X5. We observe that, in general, it is better to serve the destinations as few times as possible, each time with product quantities as big as possible. A future heuristic could take advantage of this observation.

Author Contributions: Conceptualization F.A.-P.; methodology, F.A.-P. and I.Y.H.-B.; validation, F.A.-P.; formal analysis, I.Y.H.-B. and R.E.L.-D.; resources, administrated by F.A.-P. and I.Y.H.-B.; writing, F.A.-P. and M.H.C.-R.; supervision, R.E.L.-D. and I.Y.H.-B.; project administration, M.H.C.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. ROADEF/EURO Challenge 2016: Inventory Routing Problem. Available online: <https://www.roadef.org/challenge/2016/en/index.php> (accessed on 22 March 2024).
2. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search. In *Handbook of Metaheuristics*; Springer: New York, NY, USA, 2003; pp. 320–352.
3. Bell, W.J.; Dalberto, L.M.; Fisher, M.L.; Greenfield, A.J.; Jaikumar, R.; Kedia, P.; Mack, R.G.; Prutzman, P.J. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* **1983**, *13*, 4–23. [CrossRef]
4. Coelho, L.C.; Cordeau, J.-F.; Laporte, G. Thirty years of inventory routing. *Transp. Sci.* **2013**, *48*, 1–19. [CrossRef]

5. Lenstra, J.K.; Rinnooy Kan, A.H.G. Complexity of vehicle routing and scheduling problems. *Networks* **1981**, *11*, 221–227. [\[CrossRef\]](#)
6. Coelho, L.C.; Laporte, G. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *Int. J. Prod. Res.* **2013**, *51*, 7156–7169. [\[CrossRef\]](#)
7. Dror, M.; Levy, L. A vehicle routing improvement algorithm comparison of a greedy and a matching implementation for inventory routing. *Comput. Oper. Res.* **1986**, *13*, 33–45. [\[CrossRef\]](#)
8. Adulyasak, Y.; Cordeau, J.-F.; Jans, R. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS J. Comput.* **2013**, *26*, 103–120. [\[CrossRef\]](#)
9. Cordeau, J.-F.; Lagana, D.; Musmanno, R.; Francesca, F. A decomposition-based heuristic for the multiple-product inventory-routing problem. *Comput. Oper. Res.* **2015**, *55*, 153–166. [\[CrossRef\]](#)
10. Shao, Y.; Furman, K.C.; Geol, V.; Hoda, S. A hybrid heuristic strategy for liquefied natural gas inventory routing. *Transp. Res.* **2015**, *C 53*, 151–171.
11. Etebari, F.; Dabiri, N. A hybrid heuristic for the inventory routing problem under dynamic regional pricing. *Comput. Chem. Eng.* **2016**, *95*, 231–239. [\[CrossRef\]](#)
12. Shaabani, H.; Kamalabadi, I.N. An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem. *Comput. Ind. Eng.* **2016**, *99*, 189–201. [\[CrossRef\]](#)
13. Cheng, C.; Qi, M.; Wang, X.; Zhang, Y. Multi-period inventory routing problem under carbon emission regulations. *Int. J. Prod. Econ.* **2016**, *182*, 263–275. [\[CrossRef\]](#)
14. Archetti, C.; Speranza, M.G. The inventory routing problem: The value of integration. *Int. Trans. Oper. Res.* **2016**, *23*, 393–407. [\[CrossRef\]](#)
15. Archetti, C.; Desaulniers, G.; Speranza, M.G. Minimizing the logistic ratio in the inventory routing problem. *EURO J. Transp. Logist.* **2017**, *6*, 289–306. [\[CrossRef\]](#)
16. Archetti, C.; Boland, N.; Speranza, M.G. A metaheuristic for the multivehicle inventory routing problem. *INFORMS J. Comput.* **2017**, *29*, 377–387. [\[CrossRef\]](#)
17. Absi, N.; Cattaruzza, D.; Feillet, D.; Ogier, M.; Semet, F. A heuristic branch-cut-and-price algorithm for the ROADEF/EURO challenge on Inventory Routing. *Transp. Sci.* **2020**, *54*, 313–329. [\[CrossRef\]](#)
18. Su, Z.; Lü, Z.; Wang, Z.; Qi, Y.; Benlic, U. A matheuristic algorithm for the inventory routing problem. *Transp. Sci.* **2020**, *54*, 330–354. [\[CrossRef\]](#)
19. He, Y.; Artigues, C.; Briand, C.; Jozefowiez, N.; Ngueveu, S.U. A matheuristic with fixed-sequence reoptimization for a real-life inventory routing problem. *Transp. Sci.* **2020**, *54*, 355–374. [\[CrossRef\]](#)
20. Kheiri, A. Heuristic sequence selection for inventory routing problem. *Transp. Sci.* **2020**, *54*, 302–312. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.