Journal of the Operations Research Society of China (2022) 10:659–683
https://doi.org/10.1007/s40305-021-00390-5

Inventory Policy and Heuristic for Long-Term Multi-product Perishable Inventory Routing Problem with Static Demand

Xi-Yi Chen[1] · **Jian-Bo Yang**[1] · **Dong-Ling Xu**[1]

Received: 28 July 2021 / Revised: 10 December 2021 / Accepted: 14 December 2021 /
Published online: 2 July 2022

Abstract

This work considers a long-term Perishable Inventory Routing Problem with multiple products, static demand, and single vehicle, in the setting of Vendor Managed Inventory. By analyzing the optimal solutions of long-term cases that can be solved in Python+Gurobi within 2 h, we capture some patterns of optimal solutions. Utilizing these patterns, experiments show that under certain conditions, the mathematical models of multi-product problems could be simplified to single-product problems, which have the same optimal solutions while taking less time to solve. Managerial insights were generated that for products with static demand in the long term, delivery should be arranged at the store level rather than at the product level. Products in the same store should have the same delivery pattern, no matter how different the unit holding costs are. By further analyzing the optimal solutions of the simplified models, we find that optimal value will stabilize in the long term, and the optimal solution is very close to the solution point where total inventory holding cost and transportation cost are close. Based on these findings, we have developed a heuristic that always provides optimal or close-to-optimal solutions with far less computational time, compared with Python+Gurobi.

Keywords VMI · PIRP · Supply chain · Perishable products

This paper is dedicated to the late Professor Duan Li in commemoration of his contributions to optimization, financial engineering, and risk management.

The work is supported by the European Union's Horizon 2020 Research and Innovation Programme RISE (No. 823759) (REMESH).

✉ Xi-Yi Chen
xiyi.chen@manchester.ac.uk

Jian-Bo Yang
jian-bo.yang@manchester.ac.uk

Dong-Ling Xu
ling.xu@manchester.ac.uk

[1] Alliance Manchetser Business School, The University of Manchester, Manchester, UK

Mathematics Subject Classification 90B05 · 90B06

1 Introduction

Inventory Routing Problems (IRPs) are complex logistic problems in which two hard problems are integrated into a unified framework: Inventory Management and Vehicle Routing [1].

IRPs usually arise from a setting of vendor-managed-inventory (VMI) system, in contrast to retailer-managed-inventory (RMI) system. As the name indicates, in the system of RMI, retailers decide independently when and how much to order. As a consequence, without exchanging information properly, the retailers' decision strongly limits the performance of a whole supply chain, in terms of transportation cost and inventory-related cost [2]. In a VMI system, a supplier, having the full information of inventory and demand of various retailers, is the central decision-maker, instead of the retailers [3]. The cost reduction and collaborative benefits of VMI have been proved by many scholars and retail chains including Wal-Mart [4,5].

In general, IRPs model the delivery policy of a supplier to a set of retail stores over a short- or long-term planning period with a single product coming from a single plant or facility, referred to as the vendor. Two main decisions are made during this process: 1. how much to replenish to each store to minimize waste and holding cost; 2. which routes to choose to minimize transportation cost [6,7].

The structure of this work is as follows. In Sect. 2, we briefly explain the literature and our contributions. In Sect. 3, we introduce the problem, notations, and mathematical models. In Sect. 4, we present our instances and some analysis of the optimal solutions. Based on the analysis in Sect. 4, Sect. 5 elaborates the development of an equivalent simplified model and experiment results of 2 models. This is followed by detailed description of our heuristic in Sect. 6. The computational results are presented in Sect. 7, including comparing solutions and computation time of our heuristic and commercial solver Python+Gurobi. In the final section, Sect. 8, we conclude our findings.

2 Literature Review

IRP has a history of about 30 years. It was first introduced by [8] by integrating decisions on inventory, vehicle scheduling for the delivery of chemical products. More comprehensive and detailed reviews can be found in [9] and [10], which are very good reviews of IRP.

Perishable Inventory Routing Problems (PIRP) was first motivated by a healthcare application [11,12] due to the human blood perishability. Later on, the applicable range gradually extends to more general context where the perishable items could be food or medical drugs.

Generally speaking, the work on PIRP is scarce [13]. Several variants have been proposed. Various topics includes economic issue, environmental issue [14] etc. Among these topics, the economic issue continues to attract most attention, which is minimizing cost. Environmental issue is also popular, which considers carbon emission. According to model assumptions, PIRPs can be further divided into models of deter-

ministic or stochastic demand [15,16], trans-shipments (delivery between stores) [17] or direct delivery (delivery from depot to stores only), homogeneous vehicle or heterogeneous vehicle (can involve different capacity, unit transportation cost and speed etc) [18], multiple [16] or single products. Since it is not possible to list all of the papers, we cite some recent papers. For more detailed review, interested readers are referred to reference [19], which is a comprehensive review.

After years of development, more complicated models have been formulated. However, these models differ in small details. Most of them are formulated as or found to be equivalent to mixed-integer linear program (MILP). The methods adopted in those models are usually based on exact methods such as Branch-and-Cut [1,20]) or metaheuristic such as Genetic Algorithm (GA) [21,22], Simulated Annealing (SA) [22], Tabu-Search [23] etc. Some also rely on commercial solver such as Cplex [14,16], Gurobi [15] to solve their models. When it comes to assumptions, majority of problems are single-product PIRP with homogeneous fleet, deterministic demand and finite shelf life.

PIRP with multi-product is a variant that is even less studied in PIRP because of its complexity brought by the various products and inventory deterioration due to different limited shelf-lives [12,16].

Reference [9] reports that instances of a single-product IRP with deterministic demands can rarely be solved to optimality when the number of stores exceeds 30. Considering product perish-ability makes the intrinsic-complicated IRP even more complicated as the combinatorial complexity grows exponentially with the increase of both product variety and store numbers [24].

However, the need for research in PIRP with multiple products is not neglectable. First of all, waste of perishable products requires more attention. Perishable products constitute over 52% of the sales revenue of grocery retail chains [25]. Nevertheless, it is reported that food lost after harvest and at the transport, storage, and processing stages stands at 13.8% globally, which amounts to over 400 billion USD per year [26]. In 2019, the profit margin of food retail barely reaches 1% after-tax [27]. Secondly, the solution and method available for single-product PIRP are not suitable for real-life retail chains [28]. A small retail chain could dispatch thousands of (perishable and dry) products from a central warehouse to hundreds of stores.

Due to the complexity of real-world problems and the NP-hard nature of PIRP, demand arises for more efficient algorithms to find optimal or near optimal solutions. Perishable inventory is difficult to manage. However, analysis of optimal replenishment policies remains a topic of great interest in the area of pure inventory management. For example, references [15] and [29] explores optimal base-stock level, while references [30] and [31] develop approximation algorithms for perishable inventory systems. Due to the complex nature of algorithms for finding the solutions, papers of PIRP tend to focus less on analysis but more on modelling and computation. Our work differs from previous PIRP papers in the way that we obtain our close-to-optimal solutions by analyzing the relationship between parameter values and optimal solutions.

In our problem, the relationship is found through both extensive experiments and analyzing the trade-off of the 3 costs in our objective function, which are transportation cost, inventory holding cost and waste cost. To the best of our knowledge, our work is the first to study the long-term multi-product PIRP with static demand and provide

solutions based on parameter-solution relationships. Since our problem has never been studied before, we base our model on the famous single-product single-period Miller-Tucker-Zemlin (MTZ) formulation of Travelling Salesman Problem (TSP) [32], and modify the model to our need.

This work contributes to the literature in the following 4 ways: (a). We study the PIRP of multiple perishable products with stable demand in the long term; (b). We build a single-product PIRP model that provides the same solution and objective function value as the multi-product PIRP model; (c). We find that the optimal solutions, including the quantity to deliver to each store in each period, and the transportation routes, could be found near the solution point whose total inventory holding cost and transportation cost are close to each other; and (d). We develop a heuristic that can provide the optimal or close-to-optimal solution for larger-scale problems with far less computing time than Gurobi+Python.

3 Problem Description

The problem studied here is a long-term Inventory Routing Problem with multiple perishable products, single-vehicle, and static demand. In the following subsections we introduce our assumptions, notations and then mathematical models.

3.1 Assumptions and Notations

The mathematical models are developed based on the following assumptions:

1. One warehouse supplies multiple perishable products to multiple stores with an unlimited supply. For the same product, different stores have different demands.
2. The unit holding costs are different for different products but are the same for the same product across stores. Different products have a different maximum shelf life, which is the same for the same product across stores.
3. FIFO policy: In each store, inventory is consumed in a first-in-first-out (FIFO) manner, which means the oldest inventory will be consumed first then the fresher ones. FIFO policy is one of the most effective policies to reduce waste and increase revenue [25,33], which is widely adopted by supermarkets.
4. Symmetric distance matrix: The distance from any store i to store j equals the distance from store j to store i.
5. For any route, a truck always departs from the warehouse and returns to the warehouse at the end of a period. Each store can receive 1 delivery only in a period.
6. Single vehicle and static demand: In real life, 1 truck driver is usually responsible for a certain area and this area tends not to change or change very slowly in terms of population size and lifestyle. This also implies that, without intervention of special events, the demand of product in each store tends to be stable in the long term, especially for daily necessities like food.
7. No demand shortage is allowed in each period for each product at each store.
8. The vehicle has infinite capacity. There are 3 reasons: (a). Perishable products require frequent delivery and small delivery quantity to control waste; (b). The constraint that truck returns to depot at the end of the period (usually a day) limits

the number of stores visited; (c). The vehicle capacity is much larger than the total delivery quantity per route.

The notations used in this paper are presented in Table 1.

3.2 Mathematical Formulations of Multi-product PIRP with Static Demand and FIFO

$$\min \left[\mathrm{uT} \times \sum_{i=1}^{I} \sum_{j=1}^{J\&j!=i} T_{ij} \times \sum_{t=1}^{T} x_{ijt} + \sum_{i=1}^{I} \sum_{p=1}^{P} \mathrm{uH}_p \times \sum_{t=1}^{T} \left(\sum_{lp=2}^{L_p+1} \mathrm{Inv}_{ipt,lp} - D_{ip}/2 \right) + \sum_{i=1}^{I} \sum_{p=1}^{P} W_p \times \sum_{t=1}^{T} \mathrm{Inv}_{ipt,1} \right] /T, \tag{1}$$

$$\sum_{j=1}^{J\&j!=i} x_{ijt} \leqslant 1, \forall i \in I, \forall t \in T, \tag{2}$$

Table 1 Problem Notations

	Indices
i, j	Notation of node, $i, j \in \{1, 2, \cdots, I\}$. $I = J$. Node 1 ($i = 1$) is the depot and I is the total number of stores and depot.
p	Notation of product, $p \in \{1, 2, \cdots, P\}$. P is the total number of products.
lp	Notation of remaining shelf life of product p, $lp \in \{1, 2, 3 \cdots, L_p + 1\}$. Product with $lp = 1$ will be discarded at the beginning of the period. L_p is the shelf life of product p.
t	Notation of a period, normally in the unit of day, in a planning horizon, $t \in \{1, 2, \cdots, T\}$ and T is the total number of periods in the planning horizon.
	Parameters
uT	Unit transportation cost, unit: pound per km.
TC	Truck Capacity of the vehicle, a very large number.
T_{ij}	Transportation distance between store i and store j, in which store 1 is the depot, unit: km.
uH_p	Unit inventory holding cost of product p, unit: pound per piece per period.
W_p	Unit wastage cost of product p unit: pound per piece per period.
D_{ip}	The demand of product p in store i, unit: piece.
M	A large number used in if-or constraints.
	Decision variables
x_{ijt}	Binary: 1, when truck departs from store i to store j at period t; 0, otherwise.
Q_{ipt}	Non-negative real: quantity of product p to deliver to store i in period t.
u_{it}	Real: quantity left in the vehicle after visiting store i in period t.
$\mathrm{Inv}_{ipt,lp}$	Non-negative real: inventory level of product p at store i and with remaining shelf life lp, at the beginning of period t. $\mathrm{Inv}_{ipt,1}$ is the inventory that should be discarded at the beginning of each period; Inv_{ipt,L_p+1} is the freshest inventory received at the beginning of each period.

$$\sum_{j=1}^{J\&j!=i} x_{jit} = \sum_{j=1}^{J\&j!=i} x_{ijt}, \forall i \in I, \forall t \in T, \tag{3}$$

$$Q_{ipt} \geqslant 0, \forall i \in I, \forall p \in P, \forall t \in T, \tag{4}$$

$$\mathrm{Inv}_{ipt,L_p+1} = Q_{ipt}, \forall i \in I, \forall p \in P, \forall t \in T, \tag{5}$$

$$\sum_{lp=2}^{L_p+1} \mathrm{Inv}_{ipt,lp} \geqslant D_{ip}, \forall i \in I, \forall p \in P, \forall t \in T, \tag{6}$$

$$\sum_{p=1}^{P} Q_{jpt} \leqslant M \times \sum_{i=1}^{I\&i!=j} x_{ijt}, \forall j \in J, \forall t \in T, \tag{7}$$

$$u_{it} - u_{jt} \geqslant \sum_{p=1}^{P} Q_{jpt} - \mathrm{TC} \times (1 - x_{ijt}), \forall i \in [2, I], \forall j \in [2, J],$$
$$\forall i! = j, \forall t \in T, \tag{8}$$

$$\mathrm{Inv}_{ip,1,lp} = 0, \forall i \in I, \forall p \in P, \forall lp \in [1, L_p], \tag{9}$$

$$\mathrm{Inv}_{ip,t+1,1} = \max(\mathrm{Inv}_{ipt,2} - D_{ip}, 0), \forall i \in I, \forall p \in P, \forall t \in T, \tag{10}$$

$$\mathrm{Inv}_{ip,t+1,n} = \max\left(\sum_{lp=2}^{n+1} \mathrm{Inv}_{ipt,lp} - D_{ip}, 0\right) - \sum_{lp=1}^{n-1} \mathrm{Inv}_{ip,t+1,lp},$$
$$\forall i \in I, \forall p \in P, \forall n \in [2, L_p], \forall t \in T, \tag{11}$$

$$x_{ijt} \in \{0, 1\}, \forall i \in I, \forall j \in J, \forall i! = j, \forall t \in T. \tag{12}$$

The objective function (1) minimizes the total cost, which equals the transportation cost, plus inventory holding cost, plus perishable cost. Constraint (2) means the number of trucks that leave any node should not exceed 1. Constraint (3) means the number of trucks that enter any node should be equal to the number of trucks that leave the same node. Constraint (4) means the quantity delivered should not be negative for all product p, store i in all period t. Constraint (5) means the new delivery received at the beginning of any period t is freshest among all inventory, having the longest remaining shelf life. Constraint (6) means the quantity of product p that delivers to store i in period t should at least satisfy the demand of this period, for all products in all stores. $\mathrm{Inv}_{ipt,1}$ is not included in the on-hand inventory because they reach the end of their shelf life and are discarded at the beginning of the period. Constraint (7) means, if the total quantity delivered to store j in period t is 0, then store j will not be visited in period t; else the number of trucks that goes to store j should be 1. Constraint (8) is the sub-tour elimination and capacity constraints, based on the known MTZ constraints and modified to our multi-period PIRP background. The MTZ constraints have a polynomial cardinality, were first proposed for the single-period TSP [32] and subsequently extended in reference [34]. When $x_{ijt} = 0$, the constraint is not binding; whereas $x_{ijt} = 1$, the constraint imposes $u_{it} \geqslant u_{jt} + \sum_{p=1}^{P} Q_{jpt}$, which means the load left in the vehicle at store i after visiting should be no smaller than the load left in the vehicle at store j after visiting plus the quantity needed in store j. This

constraint has been used extensively to model the basic Vehicle Routing Problem (VRP) and Capacitated Vehicle Routing Problem (CVRP). Constraint (9) means the initial inventory at the beginning of period $t = 1$ is 0 for all products and all stores. Constraint (10) means that according to FIFO policy, the oldest inventory, which is $lp = 2$, will be used to first satisfy the demand. The remnant will perish in the next period. Constraint (11) is similar to (10), meaning that the oldest inventory with shorter remaining shelf life will be used first, then the fresher batch. Constraint (12) enforces x_{ijt} to be binary.

4 Solution Method

The model is coded in Python 3.8, and Python + Gurobi is used to compute the optimal solution. All computations are executed on a machine equipped with an Intel i7 with CPU 1.80 GHz and 8 GB of RAM, with the Windows 10 operating system.

4.1 Instances Generation

Randomly generated instances are used to explore the relationship between optimal solutions and the value of parameters. Instances vary in terms of the number and locations of stores, length of planning horizon, unit holding cost, unit transportation cost, and product shelf life. The details of our test-bed parameters is provided in Table 2.

This study aims to explore the optimal solution in the long run, which is $T = \infty$. However, due to the NP-hard nature of IRP [1], the problem is not solvable when T is infinity. As a result, optimal solutions with limited T are observed at this stage.

All the other parameters are introduced earlier in Sect. 3.1 except node coordinate. Node coordinates are the locations of stores and the depot. We randomly generate the locations first, and then calculate Euclidean distance between node i and node j based on $T_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Their locations are also used to draw the optimal delivery routes per period for each instance.

Table 2 Test-bed parameter values

Parameters	Value
Number of stores: I	[0,70]
Number of Periods: T	[1,10]
Shelf life of products: L_p	Random [1,20]
Number of products: P	[1,200]
Demand: D	Random [1,40]
Node coordinate: (x_i, y_i)	Both Random [0, 200]
Unit inventory holding cost: uH	0.01, 0.1, 0.2, 10, 100
Unit transportation cost: uT	0.1
Initial inventory level: Inv_0	0

4.2 Analysis of Results and Features of Quantity in Optimal Solution

Due to the NP-Hard nature of the problem and the limited computational power of the machine, it is found that some instances could not be solved to optimal within an acceptable time, taking up to more than 15 h, such as problem scale of 1 product, 7 customers and 10 periods, or 4 products, 7 customers and 8 periods, or 1 product, 20 customers and 2 periods.

In our experiments, we let the model run for maximum 2 h. The cases that can be solved within 2 h are referred to as 'solvable' cases for short in this paper, while the rest are referred to as 'not solvable' cases for short. However, in the early stage of the experiments, to test the limit of Python+Gurobi and observe optimal solution patterns, we let some cases to run for more than 10 h.

In this section, we try different combinations of parameters and list computational time and objective function value of 18 solvable cases. The cases that take longer than 2 h are marked with **. The rest of the larger-scale cases are not listed, for the optimal solution could not be attained, due to the program terminated after 2 h or memory running out.

Generally, the cases with longer planning horizon and more customers takes more time to solve. More specifically, cases with customer number exceeding 20 or planning horizon longer than 8 periods needs to be paired with other parameters of small values, otherwise they are less likely to be solved within 2 h.

In the process of analyzing optimal solutions, we notice the delivery interval plays a very important role. It is the lever that influences the trade-off of 3 costs (transportation, inventory holding and waste costs) in our objective function. When the optimal delivery interval for each store is found, the optimal delivery quantity is found. The rest of the job is scheduling the routes to optimal. Denote delivery interval of store i as Int_i, total delivery quantity to store i as Q_i, the optimal delivery quantity at the beginning of any delivery interval has the following 3 features:

Feature 1 $\text{Int}_i \leqslant \min(L_p)$: Delivery intervals should not be longer than the shortest shelf-life of all products in store. The perishable nature of products requires that the delivery interval must be smaller than product shelf life, which is $\text{Int}_i \leqslant L_p, \forall i \in I, \forall p \in P$. This is equivalent to $\text{Int}_i \leqslant \min(L_p), \forall i \in I$. The smallest shelf-life in a store limits its delivery interval.

Explanation If the delivery interval is longer than product shelf-life, constraint (6), which allows no demand shortage, will surely be violated. Let us assume the shelf-life of a product is 3 days. If the store is replenished every 4 days, we will face 1-day stock-out. No matter how much inventory we have at the beginning of the interval, that inventory will drop to 0 at the end of shelf-life period. If not replenished in time, the rest of the time, stores face stock-out situations.

Feature 2 $Q_i = \text{Int}_i \times \sum_{p=1}^{P} D_{ip}$: Under the condition of Feature 1, when the initial inventory is 0 for all products in all stores as in our problem, optimal total quantity delivered to each store at the beginning of any delivery interval should always cover exactly the total demand during this interval; All products in the same store have the same delivery pattern, which means when optimized, situations will not happen that in the same store, some

products receive 0 while other products receive positive number of delivery quantity.

Explanation The objective function consists of 3 costs, in which delivery quantity only influences 2 of them, inventory holding and waste costs. Transportation cost depends solely on delivery patterns, not on delivery quantity. Inventory holding cost depends on the inventory level at the beginning and the demand, which is represented as $\sum_{lp=2}^{L_p+1} \text{Inv}_{ipt,lp} - D_{ip}/2$. Suppose new delivery is received at the beginning of period t, the inventory level at the end of the interval period is $\sum_{lp=2}^{L_p+1} \text{Inv}_{ipt,lp} - \text{Int}_i \times D_{ip}$. Since no shortage is allowed in our problem, inventory level can not be negative and we have $\sum_{lp=2}^{L_p+1} \text{Inv}_{ipt,lp} \geqslant \text{Int}_i \times D_{ip}$. Combined with constraint (5), we know that, during any interval period Int_i, initial inventory should always satisfy the following: $\sum_{lp=2}^{L_p} \text{Inv}_{ip,lp} + Q_{ip} \geqslant \text{Int}_i \times D_{ip}$.

Objective function being minimizing cost, we will have:

$$\sum_{lp=2}^{L_p} \text{Inv}_{ip,lp} + Q_{ip} = \text{Int}_i \times D_{ip}, \forall i \in I, \forall p \in P. \tag{13}$$

In equation (13), $\sum_{lp=2}^{L_p} \text{Inv}_{ip,lp}$ is the total inventory left at the end of previous period, and also the total inventory at the beginning of the current interval period before receiving new delivery. If not planned well, this part of inventory will be positive, which means store ordered too much for last interval, $Q_{ip} \leqslant \text{Int}_i \times D_{ip}$; nevertheless, when planned well enough, initial inventory at the beginning of any interval before receiving new delivery will be 0 for all products in all stores, we will have:

$$Q_{ip} = \text{Int}_i \times D_{ip}, \forall i \in I, \forall p \in P. \tag{14}$$

Note that the value of Int_i can be different for different stores. In the same store, when Int_i is given, equation (14) holds for all products. At the end, the optimal total quantity delivered to store i: $Q_i = \text{Int}_i \times \sum_{p=1}^{P} D_{ip}$.

An example of how different delivery quantities influence inventory level is provided in Fig. 1. In this example, demand is 2 pieces per period, delivery interval is 3 periods. We can see that inventory level of the yellow line is clearly not optimal. Store holds 2 more pieces than needed in first 3 periods. If quantity is optimized for all periods, we would have the blue line. Thus, no matter how delivery intervals change, when inventory is optimized, we will have no excess inventory, no shortage and no waste.

Feature 3 $Q_i \leqslant \min(L_p) \times \sum_{p=1}^{P} D_{ip}$: When the initial inventory is 0 for all products in all stores, the optimal delivery quantity for any product in any store should not exceed the demand in the minimum shelf life period of all products in the store.

Explanation Feature 3 is simply the combination of Features 1 and 2 when delivery quantity of previous period is optimized, or when initial inventory is 0. However, when

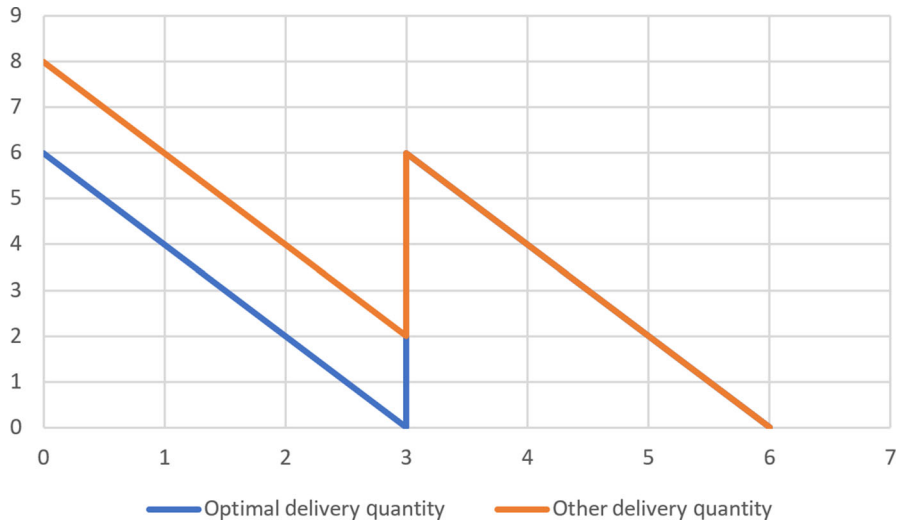


Fig. 1 Inventory level with different delivery quantity

we are unsure of the initial inventory level, we should have

$$\sum_{p=1}^{P}\sum_{lp=2}^{L_p} \mathrm{Inv}_{ip,lp} + Q_i \leqslant \min(L_p) \times \sum_{p=1}^{P} D_{ip}, \forall i \in I, \forall p \in P. \tag{15}$$

Inequality (15) makes sure that stores do not order too much. Combined with no shortage constraint (6), inequality (15) forces stores to order at least once every minimum shelf-life period.

The above 3 features show that optimal solutions of multi-product PIRP could be presented at the store level. In other words, notations related to products can be discarded and multi-product problem can be simplified to single-product problem.

For better illustration of the above 3 features, optimal delivery quantity and routes of 2 cases both with 7 stores, 4 products, and 6-period planning horizon (I7P4T6) are presented in this section. Table 3 shows product-related parameter values such as demand in each store, shelf life and unit holding cost. The locations of depot and stores are provided in Table 4.

In these 2 cases, the parameter values are carefully controlled. We specifically make unit holding cost the same for different products in case1, and case2 with each product having a very different unit holding cost. All the other data are the same between the two cases.

The decision variable Q_{ipt}, optimal quantity delivered to each store in each period is presented below in Table 5. For case1, $I1$ and $I5$ are replenished every period, while the rest of the stores are replenished every 2 periods. The optimal solutions of the remaining 4 periods are repetition of solutions for $t = 1$ and $t = 2$: Solution ($t = 1$) = Solution ($t = 3$) = Solution ($t = 5$), Solution ($t = 2$) = Solution ($t = 4$) =

Table 3 Products demand, shelf life and unit holding cost of 2 cases

		$P1$	$P2$	$P3$	$P4$
D_{ip}	$I1$	2	5	8	7
	$I2$	10	10	8	9
	$I3$	8	4	5	4
	$I4$	3	5	3	4
	$I5$	9	4	9	8
	$I6$	5	9	7	4
	$I7$	4	10	2	3
Shelf life L		4	3	7	10
uH_p of Case1		0.15	0.15	0.15	0.15
uH_p of Case2		0.01	0.1	0.2	100

Table 4 Nodes (stores) position coordinate of 2 cases

Coordinates	Nodes							
	Depot	$I1$	$I2$	$I3$	$I4$	$I5$	$I6$	$I7$
X	62.5	82	91	12	92	63	9	28
Y	58.9	66	3	85	94	68	76	75

Table 5 Optimal quantity of 2 cases

(a) Case1 optimal quantity

Stores	$t=1$				$t=2$			
	$P1$	$P2$	$P3$	$P4$	$P1$	$P2$	$P3$	$P4$
$I1$	2	5	8	7	2	5	8	7
$I2$	20	20	16	18	0	0	0	0
$I3$	16	8	10	8	0	0	0	0
$I4$	6	10	6	8	0	0	0	0
$I5$	9	4	9	8	9	4	9	8
$I6$	10	18	14	8	0	0	0	0
$I7$	8	20	4	6	0	0	0	0

(b) Case2 optimal quantity

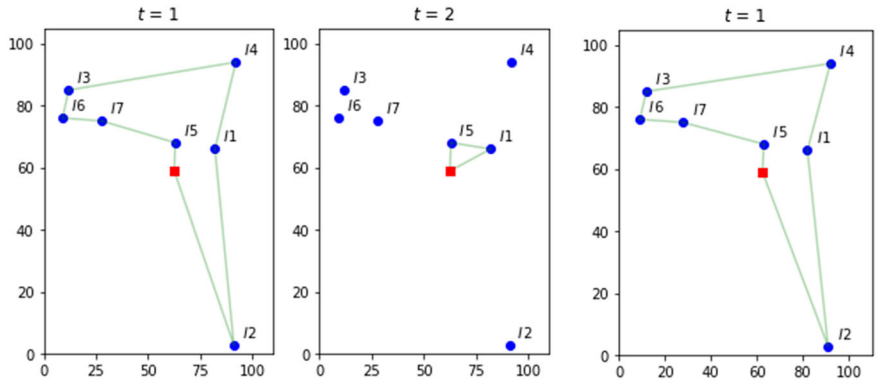
Stores	$t=1$			
	$P1$	$P2$	$P3$	$P4$
$I1$	2	5	8	7
$I2$	10	10	8	9
$I3$	8	4	5	4
$I4$	3	5	3	4
$I5$	9	4	9	8
$I6$	5	9	7	4
$I7$	4	10	2	3

Solution ($t = 6$). While for case 2, all stores are replenished every 1 period. Optimal solutions of the remaining 5 periods are repetition of those of '$t = 1$'.

The optimal routes of the 2 cases are presented in Fig. 2 graphically. The red square is the location of the depot, and the blue squares are the locations of stores.

5 Simplified Model Equivalent

Concluding from Sect. 4.2, the 3 features imply that optimal solution of multi-product model could be presented without product-related indices, which inspire us to simplify the multi-product model to single-product model. To further verify the 3 features of optimal solutions found and utilize them to our benefit, a simplified model was formulated by dropping product and shelf-life-related indices. The updated



(a) Case1 optimal routes (b) Case2 optimal routes

Fig. 2 Optimal routes of 2 cases

parameters and simplified models are presented in the following subsections and in Table 6.

Comparing optimal solutions and objective function values, experiments show that the 2 models are equivalent, while solving the simplified model is less time-consuming.

5.1 Updated Indices and Parameters

There are 4 main differences between the updated parameter table and the original one after dropping the product-related indices p and its corresponding shelf life lp. The detailed update is presented in Table 6.

1. D_i replacing $\sum_{p=1}^{P} D_{ip}$: New demand is considered at the store level rather than product level, adding up the demand for different products in the same store.

Table 6 Updated indices and parameters

	Updated Parameters
D_i	$\sum_{p=1}^{P} D_{ip}$, total demand in store i in period t, unit: piece
uH_i	$(\sum_{p=1}^{P} D_{ip} \times \text{uH}_p)/(\sum_{p=1}^{P} D_{ip})$, average unit holding cost in store i, unit: pound/piece
L	$\min(L_p)$, unit: periods.
	Deleted Parameters
W_p	Unit wastage cost per product piece, unit: pound/piece
	Updated Decision Variable
Q_{it}	Non-negative real: quantity to deliver to store i in period t, replacing $\sum_{p=1}^{P} Q_{ipt}$
Inv_{it}	Non-negative real: inventory level at store i at the beginning of period t, replacing $\sum_{p=1}^{P} \sum_{lp=1}^{L_p} \text{Inv}_{ipt,lp}$

2. uH_i added, uH_p removed: New unit holding cost is also presented at the store level rather than product level, with unit of new demand having the same unit holding cost.
3. L replacing $\min(L_p)$: Shelf life per product is replaced by minimal shelf life of products. For example, if the old shelf lives for 4 products are 3, 4, 15, 20 respectively, then the new shelf life is 3.
4. Inventory age is no longer tracked.

5.2 Simplified Models

$$\min \left[\mathrm{uT} \times \sum_{i=1}^{I} \sum_{j=1}^{J\&j!=i} T_{ij} \times \sum_{t=1}^{T} x_{ijt} + \sum_{i=1}^{I} \mathrm{uH}_i \times \sum_{t=1}^{T} (\mathrm{Inv}_{it} - D_i/2) \right] /T, \quad (16)$$

s.t.

$$\sum_{j=1}^{J\&j!=i} x_{ijt} \leqslant 1, \forall i \in I, \forall t \in T, \quad (17)$$

$$\sum_{j=1}^{J\&j!=i} x_{jit} = \sum_{j=1}^{J\&j!=i} x_{ijt}, \forall i \in I, \forall t \in T, \quad (18)$$

$$\mathrm{Inv}_{i,1} = 0, \forall i \in I, \quad (19)$$

$$Q_{it} \geqslant 0, \forall i \in I, \forall t \in T, \quad (20)$$

$$\mathrm{Inv}_{it} + Q_{it} \geqslant D_i, \forall i \in I, \forall t \in T, \quad (21)$$

$$\mathrm{Inv}_{it} + Q_{it} \leqslant D_i * L, \forall i \in I, \forall t \in T, \quad (22)$$

$$\mathrm{Inv}_{i,t+1} = \mathrm{Inv}_{it} + Q_{it} - D_i, \forall i \in I, \forall t \in T, \quad (23)$$

$$Q_{jt} \leqslant M \times \sum_{i=1}^{I\&i!=j} x_{ijt}, \forall j \in J, \forall t \in T, \quad (24)$$

$$u_{it} - u_{jt} \geqslant Q_{jt} - \mathrm{TC} \times (1 - x_{ijt}), \forall i \in [2, I], \forall \in [2, J], \forall i! = j, \forall t \in T, \quad (25)$$

$$x_{ijt} \in \{0, 1\}, \forall i \in I, \forall j \in J, \forall i! = j, \forall t \in T. \quad (26)$$

The main body of objective function and constraints are the same. There are 3 major differences between the simplified model and the original one:

1. Waste cost is eliminated from the objective function because the waste will always be 0. This is achieved by adding constraint (22) when minimizing the objective function and applying FIFO policy.

Constraint (22) is originated from inequality (15). It erases the part of waste caused by over-ordering. As explained in Sect. 4.2, combined with no shortage constraint (21), this constraint limits the delivery interval to be shorter than or equal to minimum shelf-life of all products in store.

The objective function of minimizing costs helps to control the inventory level to be just enough between any delivery interval, leaving no excess inventory for next delivery interval, as explained in Sect. 4.2, Feature 2.

By adding constraint (22) when minimizing the objective function, the waste can already be thoroughly eliminated. Objective function helps to minimize the inventory level during an interval, and constraint (22) helps to control the length of intervals. FIFO policy is not necessary anymore in our problem setting. Nevertheless, adopting FIFO in real life is still a good practice. It requires the inventory with shortest remaining shelf-life to be consumed first, which reduces possible waste due to mismanagement. Note that in constraint (22), the total initial inventory in store Inv_{it} equals $\sum_{p=1}^{P}\sum_{lp=1}^{L_p}\text{Inv}_{ip,lp}$, while in inequality (15) it is $\sum_{p=1}^{P}\sum_{lp=2}^{L_p}\text{Inv}_{ip,lp}$. They are equal because in optimal solutions, $\text{Inv}_{ipt,1}=0$.

2. Since there is no waste, there is no need to track inventory age in our mathematical model. Supermarkets can still track inventory age in real life. In mathematical model, whether tracking inventory age or not does not affect our optimal solutions. Inventory-age-related constraints in the original model are all deleted. The original inventory dynamic constraints (10) and (11) are now replaced with constraint (23), which means the total inventory at beginning of next period before receiving new delivery equals total inventory at the beginning of this period, plus new delivery, minus demand.
3. As discussed in Sect. 4.2, product-related parameters and variables are all deleted. The multi-product model is now simplified to single-product model.

5.3 Computational Results and Conclusions of 2 models

Demonstrated by experiments, the 2 models are equivalent in terms of solutions and objective function value. The simplified model generally takes less time for Python+Gurobi to solve. We list detailed computational time and objective function value in Table 7. The 'Obj' in the table is short for 'Objective Function Value'. The cases of the original model that take longer than 2 h to solve are marked with **.

After the above verification, some conclusions can be drawn and managerial insights can be derived. Since inventory holding cost is usually proportionate to product price, the more expensive the product, the higher the unit holding cost. This is to say, for perishable products in the same store with stable demand and unlimited supply, if they could be delivered in the same cold-chain vehicle, then:

1. There is no need to hoard cheap products. The reason is that if other conditions remain unchanged, for any store, the transportation cost occurs and is fixed as long as delivery happens. Hoarding 1 product while trucks still deliver other products to the same store only increases inventory holding cost. The better way is to break down the bulk of large orders and fit them in the delivery schedule of other products.
2. There is no need to arrange frequent delivery for expensive products to reduce holding cost. The reason is similar to the previous one. The transportation of delivery to each store does not depend on the quantity, but on the distance only. Increasing delivery frequency for only 1 product while frequency for other products

Table 7 Computational time in seconds (S) and OBJ value of 2 models

I	P	T	Original model		Simplified model	
			Time/s	Obj	Time/s	Obj
5	100	8	7	392.5	1	392.5
5	200	5	4	400.2	0.4	400.2
7	4	5	250	300.2	150	300.2
7	4	6	1 320	243.5	250	243.5
9	4	4	53	167.2	8	167.2
9	4	4	1 048	356.7	89	356.7
10	50	6	123	347.2	6	347.2
10	150	4	459	687.9	8	687.9
15**	8	4	11 779	189.7	3 894	189.7
15	50	3	748	225.2	411	300.3
19	4	2	2 280	141.5	5 847	141.5
19	6	2	1 789	178.1	890	178.1
20	6	4	50	328.7	17	328.7
20	50	4	69	357.9	14	357.9
25	8	2	193	132.9	239	132.9
25	20	2	71	130.8	14	130.8
30	8	2	557	138.9	366	138.9
30	10	2	2 954	101.6	1 394	101.6

remains the same is not optimal. Inventory holding cost of other products can be further reduced by increasing the delivery frequency and reduce the quantity of delivery each time, which also evens the truck utilization rate for each delivery.

3. There is no need to track inventory age and consider product-age-related policies, such as FIFO policy, since all products can be sold and waste is eliminated.

In conclusion, when the delivery frequency for 1 product changes, the delivery frequency of other products should be adjusted accordingly. The best strategy is to sync the delivery for all products, provided the assumptions of this problem.

6 Heuristic for Long Term PIRP

Though the simplified model consumes less time than the original model by reducing some constraints and variables, long-term models are still not solvable, even for small-scale problems. In the earlier experiments, we know that the long-term delivery pattern is repeating the solutions of its shortest cycle (denoted as T^*).

The shortest cycle T^* has the following 3 features:

Feature 1 All conditions are the same for all cycles including initial inventory. The end inventory of current cycle is the initial inventory for next cycle, and optimal end inventory is always 0 (no left over). Therefore, in each cycle, the initial inventory and end inventory are 0 for all products in all stores.

Feature 2 The periodic objective function value is the smallest.

Feature 3 T^* is the smallest value among all Ts that satisfy Feature 1.

The shortest cycle T^* is difficult to find and unknown. One way to find this T^* is to compare objective function values of all solvable problems for each case, gradually increasing T from 1 period. Usually to observe an accurate T^*, T should be at least $2 \times T^*$, because we need to be sure that we have found the smallest periodic objective function value.

This method is quite time-consuming because unfortunately, not all T^*s are small numbers. If intervals for all stores are the same, then T^* equals to this interval; else, T^* is the least common multiple of all stores replenish intervals. For example, considering 2 stores, the optimal intervals are 3 and 5 periods. The end inventory of 2 stores returns to 0 at the end of each 3 and 5 periods. The smallest period in which Feature 1 can be satisfied is period 15. The situation complexity can only increase with the number of stores. As we can imagine, even for some small cases, the optimal long-term solutions are not attainable, due to computational complexity.

To solve long-term problems, we propose a heuristic based on the relationships observed between parameters values and optimal solutions computed via Python+Gurobi. For small-scale problems for which the optimal solution is computable by using Python+Gurobi, the heuristic can provide optimal or close-to-optimal solutions with less computational time. For medium to large scale problems for which only 1-period optimal solution is computable, the heuristic can provide better solutions with reasonably less computational time. For larger-scale problems that 1-period problem could not be solved to optimal within 2 h, the heuristic is still able to provide good solutions.

6.1 Heuristic for Long-Term PIRP

For the long-term PIRP with static demand and single vehicle, due to those special assumptions, when the T is large enough, the periodic inventory and transportation cost will converge to a value, and the optimal solutions of multiple periods will be a repetition of solutions of a smaller T^*. Take the I7P4T6 case1 we saw in Sect. 4 for example, the optimal solutions are the repetition of the first 2 periods', which is $T^* = 2$. That is because the delivery intervals of different stores are 1 or 2 periods, $T^* = 2$ is the smallest common multiple of 1 and 2. The heuristic approaches optimal solution by finding this T^*, which comprises the optimal delivery interval of each store.

The overview of heuristic is shown in Fig. 3, which displays rough idea. As mentioned earlier, during the experiments, we notice delivery interval is the lever in the

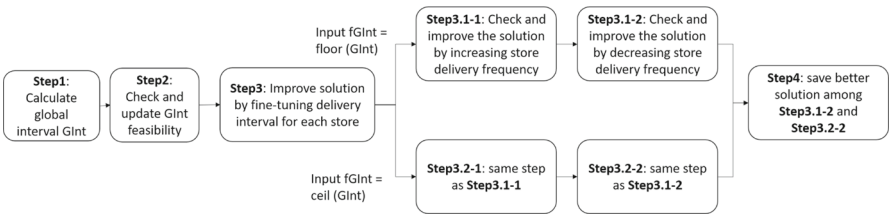


Fig. 3 Main idea of steps in heuristics

trade-off of our PIRP model. The detailed steps are presented below, in which 2 steps are the cores: (a) how to calculate close-to-optimal global interval; (b) how to adjust interval for each store based on the global interval to approach to the optimal solution.

Step1 Calculate Global Delivery Interval, denoted as GInt, assuming all the customers have the same delivery pattern and before considering product perishability constraint. The same delivery pattern means, in any period t, either all stores are replenished, or no store is replenished at all. In other words, having the same delivery pattern means having both the same delivery interval and the same 1st delivery date. For example, if GInt $= 3$, all stores are replenished every 3 periods and on the same day.

- **Step1-1** Solve TSP for all customers using Python +Gurobi, which is calculating the shortest distance visiting all customers in 1 route. Denote the optimal objective function value as TC, and save the optimal routes. The TSP mathematical model is presented in the following Sect. 6.2.
- **Step1-2** Calculate optimal global interval before considering product perishability: $\text{GInt} = \sqrt{\frac{2\times \text{TC}\times \text{uT}}{\sum_{i=1}^{I} \text{uH}_i \times D_i}}$. The objective function could be written as follows, minimizing periodic transportation cost and inventory holding cost:

$$\min \left(\frac{\text{TC} \times \text{uT}}{\text{GInt}} + \frac{\sum_{i=1}^{I} (\text{uH}_i \times D_i) \times \text{GInt}}{2}. \right) \tag{27}$$

GInt is the only variable. The optimal solution is found where the gradient concerning GInt is 0, which is $-\frac{\text{TC}\times \text{uT}}{\text{GInt}^2} + \sum_{i=1}^{I} (\text{uH}_i \times D_i)/2 = 0$. In the end, we have $\text{GInt} = \sqrt{\frac{2\times \text{TC}\times \text{uT}}{\sum_{i=1}^{I} \text{uH}_i \times D_i}}$. Go to Step2.

Step2 Since the interval GInt can be non-integer, or greater than shelf life (infeasible), we then check if the interval (GInt) is feasible and integer.

- **Step2-1** If GInt $\geqslant L$, then GInt $= L$, go to Step3-0; else if GInt $\leqslant 1$, then GInt $= 1$, go to Step3-2; else if $1 <$ GInt $< L$, go to Step3-0.

Step3 Assume x' is x rounded up to its nearest larger integer and x_* is x rounded down to its nearest smaller integer. For example, if GInt $= 3.5$, then $\text{GInt}_* = 3$, $\text{GInt}' = 4$. In the following steps, based on the 2 nearest integers (GInt_*, GInt') of GInt, we fine-tune the intervals for each store and check if the solution can be improved.
If GInt is not an integer, after Step3, we will have 2 delivery schedules, one is based on global interval GInt_* (e.g., 3), another is based on global interval GInt' (e.g., 4). These 2 delivery schedules are different. In final Step4, we pick the schedule with the smaller cost. However, if GInt is an integer itself (e.g., 2), we will have 1 schedule, whose global interval is itself (e.g., 2).

- **Step3-0** If $(\frac{\text{GInt}}{2})_* = (\frac{\text{GInt}}{2})'$, then GInt is an even number and no smaller than 2, go to Step3-2; else, go to Step3-1.

– **Step3-1** Let fGInt $=$ GInt$_*$. If fGInt $\geqslant 2$, go to Step3-1.1; else, go to Step3-1.2.

Step3-1.1 Check if total cost could be improved, when shortening delivery interval for each store. For each store i, we insert 1 delivery in the middle of the global delivery interval fGInt, without changing intervals of other stores. Let hfGInt $=$ fGInt$/2$. After inserting, the original delivery interval is split into 2 intervals, which are hfGInt$_*$ and hfGInt$'$. For example, if fGInt $= 3$, then hfGInt$_* = 1$, hfGInt$' = 2$. If fGInt $= 4$, the 2 new intervals are both 2.

$$\text{Hsave}_i = [(\text{fGInt})^2 - (\text{hfGInt}_*)^2 - (\text{hfGInt}')^2] \times \text{uH}_i \times D_i/2, \quad (28)$$

$$\text{Tincre}_i = 2 \times T_{0i} \times \text{uT}. \quad (29)$$

As introduced in Sect. 4.2, in any interval with 0 initial inventory, the optimal inventory at the beginning of a period should equal to the total demand in this interval. Equation (28) calculates the difference in fGInt periods. Before insertion of one extra delivery, optimal original initial inventory is fGInt $\times D_i$, which can last for fGInt periods. After the insertion, we have hfGInt$_* \times D_i$ pieces of product for store i, which lasts for hfGInt$_*$ periods and hfGInt$' \times D_i$ which lasts hfGInt$'$. The saved holding cost Hsave$_\text{i}$ is the difference between original holding cost (fGInt)$^2 \times \text{uH}_i \times D_i/2$ and new holding cost after the insertion [(hfGInt$_*$)^2 + (hfGInt$'$)^2] $\times$ uH$_i \times D_i/2$. Equation (29) calculates the increased transportation cost Tincre$_i$, which is the cost of vehicle leaving depot to store i and returning.

If Hsave$_i >$ Tincre$_i$, insert one more delivery for store i. If the number of stores is greater than 1, solve TSP for these stores. Update total cost and delivery interval for all i. Go to Step3-1.2.

The reasons for inserting new delivery in the middle of the original interval for all stores are: (a) to maximize the reduction on inventory holding cost for each single store; (b) to further reduce transportation cost since delivery of multiple stores can be arranged on the same day in 1 route. In this way, the total cost will be minimized.

Step3-1.2 If $2 \times$ fGInt $> L$, go to Step3-2; else, which is if $2 \times$ fGInt $\leqslant L$, check if the total cost could be improved for each store by increasing the delivery interval. For each store i, we increase the interval to be twice the global interval, the new interval newInt$t = 2 \times$ fGInt. For example, if $L = 7$, fGInt $= 3$, then newInt$t = 6$; if $L = 5$, fGInt $= 3$, then Step3-1.2 is skipped, go straight to Step3-2.

$$\text{Tsave}_i = (T_{ni} + T_{im} - T_{nm}) \times \text{uT}, \quad (30)$$

$$\text{Hincre}_i = (\text{newInt}^2 - 2 \times (\text{fGInt})^2) \times \text{uH}_i \times D_i/2. \quad (31)$$

For each store i, denote the nodes before and after in the TSP routes, as n and m. Before increasing the interval, store i is visited twice in newInt periods with the same store visiting order, store n to store i to store m. After increasing, store i is visited once in newInt periods. The 2nd visit is cancelled and the vehicle

visits store m directly after leaving store n. Equation (30) calculates the saved transportation cost Tsave_i caused by the cancellation of the 2nd visit in newInt periods. Equation (30) is a fast and easy estimation of saved transportation cost. Equation (31) calculates the increased holding cost Hincre_i, which is the difference between 2 costs, $2 \times (\text{fGInt})^2 \times \text{uH}_i \times D_i/2$ (original interval) and $\text{newInt}^2 \times \text{uH}_i \times D_i/2$ (after interval increase). If $\text{Tsave}_i > \text{Hincre}_i$, increase the delivery interval for store i. Please note that the Tsave_i is an estimation of saved transportation cost by simply removing node i from the routes and linking the nodes before and after i. To increase the accuracy, we keep a record of these store i, update the route by deleting these stores from the original one, calculate $T\text{save}_i$ for the rest of the stores, and compare them with Hincre_i again. We repeat the steps until no more stores could be improved.

If the number of stores is greater than 1, solve TSP for these stores. Update delivery interval for all i and total cost. Go to Step3-2.

– **Step3-2** Let $\text{fGInt} = \text{GInt}'$, go to Step3-1.1, and then go to Step4.

Step4 If $\text{GInt}_* = \text{GInt}'$, save final solutions of Step3-2; otherwise, compare the final solutions obtained from Step3-1 and Step3-2, and select the delivery schedule with smaller objective function value. Now the delivery intervals are known, denote interval for store i as Int_i, and the quantity that should be delivered is: $\text{Int}_i \times D_i$.

6.2 Mathematical Formulation for TSP

Here we present the model of typical TSP. The one we adopt is the famous MTZ formulation [34], in which the TSP is formulated as an integer linear program. The assumptions of the two models in the earlier sections still hold, and the data generation rule also applies.

The TSP was first formulated in 1930. After all these years, many variants emerge, extending from the seminal model. Interested readers could refer to [35] for more details. This area is very mature with many efficient algorithms.

During our exploration to PIRP solution optimality, it is found that optimal solution of TSP, or the accuracy of transportation cost estimation, attaches great importance. Thus, despite of its NP-hard nature, in our heuristic, we still choose to solve TSP to optimal using Python+Gurobi. With the quality of TSP routes guaranteed, we are able to obtain optimal solution for some cases.

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J\&j!=i} T_{ij} \times x_{ij}, \tag{32}$$

s.t.

$$\sum_{j=1}^{J\&j!=i} x_{ij} = 1, \forall i \in I, \tag{33}$$

$$\sum_{j=1}^{J\&j!=i} x_{ji} = 1, \forall i \in I, \tag{34}$$

$$u_i - u_j + 1 \leqslant I * (1 - x_{ij}), \forall i \in [2, I], j \in [2, J], i! = j, \tag{35}$$

$$1 \leqslant u_i \leqslant I - 1, \forall i \in [2, I], \tag{36}$$

$$x_{ij} \in \{0, 1\}, \forall i \in I, \forall j \in J, \forall i! = j. \tag{37}$$

The objective function (32) minimizes the total traveling distance. Constraint (33) means the number of trucks that leave any node should be 1. Constraint (34) means the number of trucks that enter any node should be 1. Constraint (35) and (36) are for sub-tour elimination. Constraint (37) enforces integrality.

7 Computational Results

Ideally, when solving the problem using Python+Gurobi, the longer the T the more accurate the long-term optimal solution. T should be at least $2 \times T^*$ so that the shortest repeating cycle T^* can reveal itself. With a longer T, by analyzing the optimal solution obtained from $t = 1$ to $t = T$, the shortest solution cycle T^* and the optimal solution pattern are clearer. However, due to the NP-hard nature of the problem, even for a small-scale problem, only problems with limited periods are solvable.

The generation of cases in this section follows the same rule as in previous Sect. 4.1. The detailed numerical results, objective value, and computational time for 45 randomly generated cases are presented in Table 8.

The 'T' column in Table 8 is the maximum period that Python+Gurobi could solve within 2 h. For all cases, we start with $T = 1$, and then gradually increase the T, until the limit of Python+Gurobi is reached. That is, if the largest T is 6 for a case, we solve the model to optimal for 6 times with T varying from 1 to 6, and save optimal solutions and objectives function value for each T. In the early stage of the experiments, to test the limit of Python+Gurobi and observe optimal solution patterns, we let some cases run for more than 10 h. These cases are marked with ** in the 'Dataset' column.

The 'Obj' in the table is short for 'Objective Function Value'. For cases with T greater than 1, the 'Obj' listed in Table 8 is the optimal periodic cost after comparing all solvable Ts. The PT* is the period where the optimal solution is found. The HT* is the global interval calculated using heuristic. For example, for the 1st case (I7Set1) in Table 8, the largest T computable is 6, the best solution is found when $T = 3$ and $T = 6$. The corresponding 'Obj' listed is the 'Obj' value of $T = 3$ and $T = 6$. Using heuristic, the global interval is also 3 periods. For some cases, $T = 0$. It means these cases with $T = 1$ could not be solved within 2 h, and the computational time and objective function are represented as '-'.

The results show that, even for the cases with the same number of stores, computational time can vary a lot as parameter values vary. For each data-set, we generate demand and maximum product shelf life randomly. It is found that cases with larger demand value and longer shelf life usually take more time. Nevertheless, generally,

Table 8 Comparison between Python+Gurobi and Heuristics

Dataset	T	Python+Gurobi			Heuristic		
		PT*	Time/s	Obj	HT*	Time/s	Obj
I7Set1	6	3,6	3 642	26.3	3	0.3	26.3
I7Set2**	5	2,4	43 996	37.3	2	0.2	37.3
I7Set3	8	4,8	14 536	29.9	4	105	29.9
I7Set4	9	6	154	24.9	6	0.3	24.9
I7Set5	8	8	2 854	22.5	8	0.05	22.5
I10Set1	7	6	5 007	46.3	3	0.4	46.8
I10Set2	6	2,4,6	4 175	32.2	3	0.3	32.8
I10Set3	6	5	335	20.8	5	0.4	20.8
I10Set4	3	3	4 389	53.6	2	0.3	54.9
I10Set5	7	6	78	27.7	6	0.5	27.7
I20Set1	2	2	1 145	73.9	2	4	73.9
I20Set2	2	2	1 394	50.3	4	7	50.2
I20Set3	2	2	531	46.4	3	133	45.9
I20Set4	2	2	5 936	71.7	2	215	71.7
I20Set5	2	2	3	50.8	4	1	50.7
I30Set1	1	1	548	105.7	2	297	101.4
I30Set2	2	2	986	91.3	2	9	91.7
I30Set3	2	2	89	84,3	2	2	84.7
I30Set4	3	2	2 501	58.9	4	6	58.5
I30Set5	2	2	200	69.2	2	17	69.2
I40Set1	2	2	433	75.1	4	263	74.3
I40Set2	1	1	8 180	85.9	2	4	74.2
I40Set3	2	2	378	80.2	2	25	80.2
I40Set4	1	1	60	97.5	2	6	78.9
I40Set5	1	1	30	95.3	2	24	77.9
I50Set1	2	2	3 389	81.9	2	53	81.9
I50Set2**	1	1	20 365	105.5	4	112	78.7
I50Set3	2	2	177	80.7	4	81	79.7
I50Set4	0	–	–	–	4	57	78.7
I50Set5	2	2	901	129.0	2	43	130.1
I55Set1	1	1	124	111.5	2	24	97.5
I55Set2	1	1	879	110.8	2	167	93.2
I55Set3	1	1	4 164	115.2	2	83	101.5
I55Set4	1	1	2 903	112.2	2	1 972	97.3
I55Set5	1	1	2 718	115.7	2	346	102.7
I60Set1	1	1	607	149.5	2	160	148.9
I60Set2	1	1	655	133.7	2	118	132.7
I60Set3	0	–	–	–	2	5 331	127.3
I60Set4	1	1	47	118.2	2	32	103.1

Table 8 continued

Dataset	T	Python+Gurobi			Heuristic		
		PT*	Time/s	Obj	HT*	Time/s	Obj
I60Set5	0	–	–	–	2	277	109.2
I70Set1	0	–	–	–	2	1 070	166.8
I70Set2	1	1	739	165.4	2	1 279	163.7
I70Set3	0	–	–	–	2	4 986	142.3
I70Set4	0	–	–	–	2	297	163.3
I70Set5	0	–	–	–	2	7 015	131.9

as the number of stores increases, the maximum planning horizon solvable shortens, and the number of 1-period problems that could be solved within 2 h decreases.
In conclusion, as the experiment shows, the heuristic is effective and efficient in 2 perspectives:

1. Solution quality: for all cases, as long as TSP is solvable using Gurobi, the heuristic could provide the same optimal solutions for some cases and close-to-optimal solutions for the rest.
2. Computational time: for almost all cases, the heuristic consumes less time, which lays a good base for more complicated problems, such as stochastic demand, multiple vehicles.

8 Conclusions and Findings

In this work, we have studied the long-term PIRP with multiple products, single-vehicle, and static demand. By analyzing the optimal solutions, we have proposed 2 ways of modeling, namely single product model and multiple product models, and experiments have showed that they are equivalent in terms of optimal solutions and objective function values, while the former takes less time to solve.

By verifying that the two models are equivalents, we could gain some managerial insights: for products with regular and stable demand in the long term, delivery should be arranged at the store level rather than at the product level. The latter means the truck carries 1 or limited kinds of product per delivery, while the former requires the delivery of all products to be in sync. This indicates that for any store, there is no need to hoard products with cheap inventory holding cost to reduce transportation cost; and no need to arrange a special delivery for products with expensive holding cost to reduce holding cost. The advantage of syncing delivery for all products in this problem setting could be concluded as stable supply and even vehicle utilization rate. One should also note that for products with unstable demand, the above managerial insights might not apply.

Due to the NP-hard nature of PIRP, to obtain good solutions for larger-scale problems and save computational time for small-scale problems, we have also developed a heuristic. Comparing the solution and computational time between heuristic and com-

mercial solver Gurobi coded on Python, it is demonstrated by experiments that the heuristic can provide optimal or close-to-optimal solutions for all solvable cases, with far less time. Though the heuristic is for single-vehicle problems, it could be easily implemented in multi-vehicle problems, by assigning each vehicle to its long-term serving zone and customers.

To balance the solution quality and speed of calculation, we mix the exact method and the analytical method. Exact method used for TSP assures the solution quality and approximation in our selection process reduces excessive time on exhausting all combinations.

The efficiency and quality of our heuristic is supported by extensive experiments. Nevertheless, we acknowledge some limitations of our study. Currently, our heuristic is tailored for the PIRP presented in this work, which is with single vehicle, static demand and single objective function. Our heuristic and some of the managerial insights are applicable to limited scenarios only. For example, when the number of vehicles increases or the demand is no longer static, the optimal solution pattern can be more difficult or impossible to capture. Besides, in our heuristic, the transportation cost is calculated to optimal by using Python+Gurobi, which can be time-consuming.

In the future, we may consider replacing the Python+Gurobi with other methods, which can increase the speed of calculation. One possible research direction is incorporating more realistic elements to our model such as multi-objective, multi-vehicle and stochastic demand. Another direction is to study the influence of different product issuing policy (other than FIFO) to the performance of our PIRP system.

Author Contributions Xi-Yi Chen conducted the main research. Jian-Bo Yang and Dong-Ling Xu co-supervised this work. All 3 authors discussed the results of experiments, reviewed, and approved the final manuscript.

Conflict of interest The authors declare that they have no conflict of interest.

References

[1] Coelho, L.C., Laporte, G.: The exact solution of several classes of inventory-routing problems. Comput. Oper. Res. **40**(2), 558–565 (2013). https://doi.org/10.1016/j.cor.2012.08.012

[2] Archetti, C., Speranza, M.G.: The inventory routing problem: the value of integration. Int. Trans. Oper. Res. **23**(3), 393–407 (2016). https://doi.org/10.1111/itor.12226

[3] Razmi, J., Hosseini Rad, R., Sangari, M.S.: Developing a two-echelon mathematical model for a vendor-managed inventory (VMI) system. Int. J. Adv. Manuf. Technol. **48**(5–8), 773–783 (2010). https://doi.org/10.1007/s00170-009-2301-7

[4] Waller, M., Johnson, M., Davis, T.: Vendor-managed inventory in the retail supply chain. J. Bus. Logist. **20**(1), 183 (1999)

[5] Xu, K., Dong, Y., Evers, P.T.: Towards better coordination of the supply chain. Transp. Res. Part E: Logist. Transp. Rev. **37**(1), 35–54 (2001). https://doi.org/10.1016/S1366-5545(00)00010-7
[6] Li, K., Chen, B., Sivakumar, A.I., Wu, Y.: An inventory-routing problem with the objective of travel time minimization. Eur. J. Oper. Res. **236**(3), 936–945 (2014). https://doi.org/10.1016/j.ejor.2013.07.034
[7] Madadi, A., Kurz, M.E., Ashayeri, J.: Multi-level inventory management decisions with transportation cost consideration. Transp. Res. Part E: Logist. Transp. Rev. **46**(5), 719–734 (2010)
[8] 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 (Providence, Rhode Island) **13**(6), 4–23 (1983). https://doi.org/10.1287/inte.13.6.4
[9] Coelho, L.C., Cordeau, J.F., Laporte, G.: Thirty years of inventory routing. Transp. Sci. **48**(1), 1–19 (2014). https://doi.org/10.1287/trsc.2013.0472
[10] Moin, N.H., Salhi, S.: Inventory routing problems: a logistical overview. J. Oper. Res. Soc. **58**(9), 1185–1194 (2007). https://doi.org/10.1057/palgrave.jors.2602264
[11] Federgruen, A., Prastacos, G., Zipkin, P.H.: Allocation and distribution model for perishable products. Oper. Res. **34**(1), 75–82 (1986). https://doi.org/10.1287/opre.34.1.75
[12] Le, T., Diabat, A., Richard, J.P., Yih, Y.: A column generation-based heuristic algorithm for an inventory routing problem with perishable goods. Optim. Lett. **7**(7), 1481–1502 (2013). https://doi.org/10.1007/s11590-012-0540-2
[13] Stephan, B., Panadero, J., Onggo, B.S., Panadero, J., Corlu, C.G., Juan, A.A.: Agri-food supply chains with stochastic demands: a multi-period inventory routing problem with perishable products. Sim. Modell. Pract. Theor. **97**(July), 19 (2019). https://doi.org/10.1016/j.simpat.2019.101970
[14] Soysal, M., Bloemhof-Ruwaard, J.M., Haijema, R., Van Der Vorst, J.J.G.A.J.: Modeling an Inventory Routing Problem for perishable products with environmental considerations and demand uncertainty. Int. J. Prod. Econ. **164**, 118–133 (2015). https://doi.org/10.1016/j.ijpe.2015.03.008
[15] Haijema, R.: Optimal ordering, issuance and disposal policies for inventory management of perishable products. Int. J. Prod. Econ. **157**(1), 158–169 (2014). https://doi.org/10.1016/j.ijpe.2014.06.014
[16] Soysal, M., Bloemhof-Ruwaard, J.M., Haijema, R., van der Vorst, J.J.G.A.J.: Modeling a green inventory routing problem for perishable products with horizontal collaboration. Comput. Oper. Res. **89**, 168–182 (2018). https://doi.org/10.1016/j.cor.2016.02.003
[17] Timajchi, A., Mirzapour Al-e-Hashem, S.M.J., Rekik, Y.: Inventory routing problem for hazardous and deteriorating items in the presence of accident risk with transshipment option. Int. J. Prod. Econ. **209**, 302–315 (2019). https://doi.org/10.1016/j.ijpe.2018.01.018
[18] Cheng, C., Yang, P., Qi, M., Rousseau, L.M.: Modeling a green inventory routing problem with a heterogeneous fleet. Transp. Res. Part E: Logist. Transp. Rev. **97**, 97–112 (2017). https://doi.org/10.1016/j.tre.2016.11.001
[19] Batero-Manso, D.F., Orjuela-Castro, J.A.: Inventory routing problem in perishable supply chains: a literature review. Ingeniería **23**(2), 117–143 (2018). https://doi.org/10.14483/23448393.12691
[20] Alvarez, A., Cordeau, J.F., Jans, R., Munari, P., Morabito, R.: Formulations, branch-and-cut and a hybrid heuristic algorithm for an inventory routing problem with perishable products. Eur. J. Oper. Res. **283**(2), 511–529 (2020). https://doi.org/10.1016/j.ejor.2019.11.015
[21] Azadeh, A., Elahi, S., Farahani, M.H., Nasirian, B.: A genetic algorithm-Taguchi based approach to inventory routing problem of a single perishable product with transshipment. Comput. Ind. Eng. **104**, 124–133 (2017). https://doi.org/10.1016/j.cie.2016.12.019
[22] Li, L., Yang, Y., Qin, G.: Optimization of integrated inventory routing problem for cold chain logistics considering carbon footprint and carbon regulations. Sustainability (Switzerland) **11**(17), 4628 (2019). https://doi.org/10.3390/su11174628
[23] Diabat, A., Abdallah, T., Le, T.: A hybrid tabu search based heuristic for the periodic distribution inventory problem with perishable goods. Ann. Oper. Res. **242**(2), 373–398 (2016). https://doi.org/10.1007/s10479-014-1640-4
[24] Vonolfen, S., Affenzeller, M., Beham, A., Lengauer, E., Wagner, S.: Simulation-based evolution of resupply and routing policies in rich vendor-managed inventory scenarios. Central Eur. J. Oper. Res. **21**(2), 379–400 (2013). https://doi.org/10.1007/s10100-011-0232-5
[25] Chao, X., Gong, X., Shi, C., Zhang, H.: Approximation algorithms for perishable inventory systems. Oper. Res. **63**(3), 585–601 (2015). https://doi.org/10.1287/opre.2015.1386

[26] FAO: SDG progress report 2020. In United Nations 10(3). http://www.fao.org/sdg-progress-report/en/ (2020)
[27] Food Market Institute: Supermarket Facts. Food Marketing Institute. http://www.fmi.org/research-resources/supermarket-facts (2019)
[28] Crama, Y., Rezaei, M., Savelsbergh, M., Van Woensel, T.: Stochastic inventory routing for perishable products. Transp. Sci. **52**(3), 526–546 (2018). https://doi.org/10.1287/trsc.2017.0799
[29] Zhang, H., Chao, X., Shi, C.: Perishable inventory systems: Convexity results for base-stock policies and learning algorithms under censored demand. Oper. Res. **66**(5), 1276–1286 (2018). https://doi.org/10.1287/opre.2018.1724
[30] Chao, X., Gong, X., Shi, C., Yang, C., Zhang, H., Zhou, S.X.: Approximation algorithms for capacitated perishable inventory systems with positive lead times. Manag. Sci. **64**(11), 5038–5061 (2018). https://doi.org/10.1287/mnsc.2017.2886
[31] Zhang, H., Shi, C., Chao, X.: Technical note - Approximation algorithms for perishable inventory systems with setup costs. Oper. Res. **64**(2), 432–440 (2016). https://doi.org/10.1287/opre.2016.1485
[32] Christofides, N., Mingozzi, A., Toth, P.: The Vehicle Routing Problem, pp. 315–338. Wiley, Chichester, UK (1979)
[33] Stanger, S.H.W., Wilding, R., Yates, N., Cotton, S.: What drives perishable inventory management performance? Lessons learnt from the UK blood supply chain. Supply Chain Manage. Int. J. **17**(2), 107–123 (2012). https://doi.org/10.1108/13598541211212861
[34] Miller, C.E., Zemlin, R.A., Tucker, A.W.: Integer programming formulation of traveling salesman problems. J. ACM (JACM) **7**(4), 326–329 (1960). https://doi.org/10.1145/321043.321046
[35] Gutin, G., Holloway, R., Punnen, A., Dordrecht, B., London Chapter, Fischetti, M., Toth, P. : The traveling salesman problem and its variations edited by kluwer academic publishers the generalized traveling salesman and orienteering problems Juan-Jos e Salazar-Gonzz alez (2006)