Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 304981, 11 pages
http://dx.doi.org/10.1155/2015/304981

Research Article

Three-Phase Methodology Incorporating Scatter Search for Integrated Production, Inventory, and Distribution Routing Problem

Noor Hasnah Moin and Titi Yuliana

Institute of Mathematical Sciences, University of Malaya, 50603 Kuala Lumpur, Malaysia

Correspondence should be addressed to Noor Hasnah Moin; noor_hasnah@um.edu.my

Received 21 January 2015; Revised 2 June 2015; Accepted 8 June 2015

Academic Editor: Matteo Gaeta

This paper proposes the use of scatter search metaheuristic to solve an integrated production, inventory, and distribution routing problem. The problem is based on a single production plant that produces a single product that is delivered to N geographically dispersed customers by a set of homogenous fleet of vehicles. The objective is to construct a production plan and delivery schedule to minimize the total costs and ensuring each customer's demand is met over the planning horizon. We assumed that excess production can be stored at the plant or at customer's sites within some limits, but stockouts due to backordering or backlogging are not allowed. Further testing on a set of benchmark problems to assess the effectiveness of our method is also carried out. We compare our results to the existing metaheuristic algorithms proposed in the literature.

1. Introduction

Two substantial components to improve the timeliness and consistency of delivery are integrated logistics management and product availability. Integrating production and distribution decisions can have a significant impact on setup, holding, and delivery costs. In general, the problem of coordinating production and transportation is called the production-inventory-distribution routing problem (PIDRP) [1]. The research on PIDRP involves some different areas such as vehicle routing, production, and inventory [2]. PIDRP is most similar to the inventory routing problem and periodic routing problem, because it requires multiple visits to each customer's sites over the planning horizon.

Chandra and Fisher [3] studied about a single plant, multicustomers in a multiple period by comparing two different methods to investigate the value of coordinating production and distribution. The problem is to minimize the total cost of production, transportation, and inventory. Two different alternative solution approaches are presented to manage this operation, one in which the production scheduling and vehicle routing problems are solved separately and another in which they are coordinated within a single model. The computational results reported a reduction in total operating cost from coordination ranged from 3% to 20%.

Lei et al. [1] presented multifacility, heterogeneous fleet version of the PIDRP that was motivated by a chemical manufacturer with international customers. The problem is to coordinate the production, inventory, and transportation schedules to minimize the total cost over the planning horizon. They proposed a two-phase methodology, by solving a restricted version of the problem by eliminating the routing constraints in phase one and proposed a routing heuristic based on an extended optimal partitioning procedure in phase two to transform the less-than transporter load assignments obtained in phase one into more efficient delivery schedules.

Boudia et al. [4] proposed integer linear model to solve the PIDRP but it failed to solve the large instances, and then a Greedy Randomized Adaptive Search Procedure (GRASP) is developed to tackle the production and distribution decisions simultaneously. Another two improved versions using either a reactive mechanism or a path-relinking process embedded

in GRASP are also developed and the results are better than the GRASP alone.

Boudia and Prins [5] presented an alternative method to solve the PIDRP. They solved the problem with memetic algorithm population management (MA|PM) and compared with a two-phase heuristics and GRASP. Computational testing showed that MA|PM can tackle large instances and gave better results compared to two-phase approach and GRASP.

Bard and Nananukul [2, 6] developed a mixed integer programming (MIP) model aimed at minimizing production, inventory, and delivery costs. The objective of the problem is to minimize the total cost over the planning horizon without incurring any stockouts at the customer sites. The problem includes a production plant, multiple customers with time varying demand, a finite planning horizon, and a fleet of homogenous vehicles. They developed a three-phase methodology centered on tabu search and using allocation model to find a good initial solution. They also combined the features of reactive tabu search algorithm and branch-and-price algorithm by taking efficiency of the tabu search heuristic and the precision of the branch-and-price algorithm. Nananukul [7] extended the idea by improving the clustering of the customers by creating adaptive core clusters in the reactive tabu search algorithms which are used in the clustering process instead of the original data points thus enabling the algorithm to be efficient. Unfortunately the detailed computational results were not given.

Armentano et al. [8] presented two tabu search variants for PIDRP. The first variant involves construction of a short-term memory and integrating a path relinking procedure, while another one incorporates a longer term memory and integrate the first variant. The algorithms are tested on generated instances and on instances taken from Boudia et al. [4] which involved a single product. Computational results showed that the two variants of tabu search yield good tradeoffs between solution quality and computational time and successfully outperformed Boudia and Prins [5] and Bard and Nananukul [2, 9] in all instances.

Adulyasak et al. [10] improves upon the results of Armentano et al. [8] by proposing adaptive large neighborhood search heuristic to take care of binary variables representing the setup and routing variables whilst the continuous variables associated with inventory, production, and quantities delivered are handled by solving a network flow problems. The results outperformed all other know heuristics for PIDRP.

Two inventory replenishment policies, order-up-to level and maximum level were considered in Adulyasak et al. [11] for inventory routing problem and PIDRP. By using adaptive large neighborhood search to obtain the initial solutions and the branch-and-cut algorithm were proposed to solve the different formulations. The authors managed to solve to optimality relatively small instances, 50 customers, three periods, and three vehicle on parallel computers.

Torabi et al. [12] proposed a two-step solution approach to solve an integrated multisites production planning, procurement, and distribution plans. The first step restricts the vehicle routings into direct shipment and solved the full model as mixed integer problem. Scatter search algorithm is employed in the last step by solving the associated consolidation problem in order to improve the solutions. Computational testings showed that this method gives the comparable or improved solution compared to the best solution for original model for 8 out of 10 cases and failed to get a better solution for 2 other problems. However the algorithms were not performed on benchmark instances.

We refer the readers to a review of formulations and solution algorithms in PIDRP by Adulyasak et al. [13] for the state of the art.

In particular, the PIDRP considered in this paper is very similar to that of Bard and Nananukul [2, 6] which involves a production plant, multiple customers with time varying demand, a finite planning horizon, and a fleet of homogenous vehicles that delivers the product from the production plant to the customers' sites. Excess production can be stored either at the plant or at the customer sites within some limits, but inventory cannot be transferred between sites and stockouts are not permitted. The objective is to minimize a combination of setup, holding, and routing costs both at the production facility and at customers, without incurring any stockouts at the customer sites. We also propose a three-phase methodology and our algorithm differs from Bard and Nananukul [2, 6] in Phase 2 and Phase 3. In Phase 2, we employ the Giant Tour procedure [14], sweep, and savings algorithms which have been shown to be efficient routing procedures and Phase 3 introduces the scatter search algorithm, embedding the new inventory updating mechanism as the improvement methodology.

The remainder of this paper is organized as follows. In Section 2, we present the description of PIDRP and its mathematical formulation. In Section 3, the three-phase scatter search method we employed to solve the PIDRP is discussed in detail and followed by the presentation of computational experiments and results in Section 4. Finally, conclusions are drawn in Section 5.

2. Problem Description and Mathematical Formulation

We consider a production, inventory, and distribution routing problem similar to the one proposed by Bard and Nananukul [2, 9]. It consists of a single production plant that produces a single product and distributes it to a set of N customers with nonnegative demand d_{it} in period t where $i = 1, 2, \ldots, N$ and $t = 1, \ldots, \tau$ and limited number of items p_t can be produced in period t and a limited number of inventories I_t^P can be stored by incurring unit holding cost h^P at production plant. A fleet of homogeneous vehicles with capacity Q delivers the items to the customer's sites, and each vehicle can make at most one trip per period. A limited amount of inventory I_{it}^C can be stored at customers' sites with unit holding cost of h_i^C, and each customer can only be visited at most once per period.

Furthermore, it is assumed that at the end of planning horizon all inventories (both at the production facility and at customer's site) are required to be zero. The objective

is to construct a production plan and delivery schedule which minimizes production, inventory, and distribution costs while fulfilling each customer's demand requirement. Let c_0 denote the production plant (depot) and $\{c_1, c_2, \ldots, c_N\}$ be a set of customers. w_{it} is a decision variable denoting the amount to be delivered to customer i in period t. The traveling distance (cost) from customer c_i to customer c_j $(i, j = 1, 2, \ldots, N)$ is denoted by c_{ij}. x_{ijt} equals 1 if there is a route from customer i to customer j $(i, j = 1, 2, \ldots, N)$ and 0 otherwise. y_{it} represents the total quantity on a vehicle before delivering to customer i in period t.

The maximum inventory level for production plant is denoted by $I^P_{\max}$ and for customers' sites is denoted by $I^C_{\max,i}$. In this study, the initial inventories at customers' sites are assumed to be zero.

The integrated production, inventory, and distribution routing problem (PIDRP) can be formulated as follows [2]:

$$\Phi_{IP} = \min \sum_{t\in T}\sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt} + \sum_{t\in T} f_t z_t + \sum_{t\in T_0\setminus\{\tau\}} h^P I^P_t + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h^C_i I^C_{it} \tag{1}$$

subject to

$$I^P_t = I^P_{t-1} + p_t - \sum_{i\in N} w_{it}, \quad \forall t \in T_0, \tag{2}$$

$$I^C_{it} = I^C_{i,t-1} + w_{it} - d_{it}, \quad \forall i \in N,\ t \in T, \tag{3}$$

$$\sum_{i\in N} w_{it} \le I^P_{t-1}, \quad \forall i \in N,\ t \in T, \tag{4}$$

$$p_t \le C z_t, \quad \forall t \in T_0 \setminus \{\tau\}, \tag{5}$$

$$p_0 \ge \sum_{i\in N} \left(d_{i1} - I^C_{i0}\right), \tag{6}$$

$$\sum_{\substack{j\in N_0\\ j\ne i}} x_{ijt} \le 1, \quad \forall i \in N,\ t \in T, \tag{7}$$

$$\sum_{\substack{i\in N_0\\ i\ne j}} x_{ijt} = \sum_{\substack{i\in N_0\\ i\ne j}} x_{jit}, \quad \forall j \in N,\ t \in T, \tag{8}$$

$$\sum_{j\in N} x_{0jt} \le \theta, \quad \forall t \in T, \tag{9}$$

$$y_{jt} \le y_{it} - w_{it} + D^{\max}_t \left(1 - x_{ijt}\right), \quad \forall i \in N,\ j \in N_0,\ t \in T, \tag{10}$$

$$w_{it} \le D^{\max}_{it} \sum_{j\in N_0} x_{ijt}, \quad \forall i \in N,\ t \in T, \tag{11}$$

$$\begin{aligned} &0 \le I^P_t \le I^P_{\max},\\ &0 \le I^C_{it} \le I^C_{\max,i}; \quad \forall i \in N,\ t \in T \setminus \{\tau\}; \end{aligned} \tag{12}$$

$$\begin{aligned} &I^P_\tau = I^C_{i\tau} = 0, \quad \forall i \in N,\\ &x_{ijt} \in \{0, 1\},\\ &z_t \in \{0, 1\},\\ &0 \le y_{it} \le Q,\\ &p_t \ge 0,\\ &w_{it} \ge 0 \text{ and integer}, \quad \forall i \ne j \in N_0,\ t \in T, \end{aligned} \tag{13}$$

where $D^{\max}_{it} = \min\{Q, \sum_{l=t}^{\tau} d_{il}\}$ and $D^{\max}_t = \min\{Q, \sum_{i\in N}\sum_{l=t}^{\tau} d_{il}\}$.

The objective function comprises the transportation costs, production setup costs, holding costs at the warehouse and holding costs at the customer sites. Equations (2) and (3) represent the inventory flow balance equations for production facility and customers, respectively. Equation (5) limits production on period t to the capacity of the factory, and (6) allows production in period 0. The total amount available for delivery on period t is limited by the amount of inventories at the factory on period $t-1$ as formulated in (4). Equation (11) limits the amount delivered to each customer and (7) ensures that if customer i is serviced on period t, then it must have a successor on its route, while route continuity is enforced by (8). Equation (9) limits the number of vehicles that leaves the factory at period t, and (10) keeps track of the load on the vehicles. We note that Adulyasak et al. [10] use slightly different approach in the formulation.

In this study, extending the idea from Bard and Nananukul [6] we propose a three-phase methodology to solve the PIDRP. Our algorithm differs from Bard and Nananukul [6] in Phase 2 and Phase 3. Starting from Phase 1 that solves the allocation model which is the simplified version (relaxed) of the model to determine the amount to be delivered to each customer in each period, Phase 2 routes the customers using the Giant Tour procedure [14], sweep, and savings algorithms to determine the delivery routes for each period. In phase 3, we develop scatter search method by creating composite decision rules and surrogate constraints to improve the initial solutions. We also incorporate the inventory updating based on the forward and backward transfer.

We identify the initial solution in Phase 1 by solving the allocation model as a mixed integer programming to get a set of feasible allocations. The routing variables and routing constraints (7)–(10) are removed and aggregated vehicle capacity constraints are introduced to the allocation model. Since we already deleted the routing constraints in the allocation model, we need an alternative representation for the cost term to determine the approximated cost which is needed to make a delivery to customer i in period t.

f_{it}^C represents the fixed cost of making delivery to customer i on period t, e_{it}^C denotes the variable cost of delivering one item to customer i in period t, and z_{it}^C is valued to 1 if delivery is made to customer i in period t and 0 otherwise.

As in Bard and Nananukul [2], we divide the problem into two cases, for problem instances with $N^2T \leq 500$ and for instances with $N^2T > 500$. Since all the instances we considered are $N^2T > 500$, we introduce the variable cost term $\sum_{it} e_{it}^C w_{it}$, where e_{it}^C is approximated by the cost of making a delivery to customer i directly from the depot divided by the total demand of customer i in period t.

The allocation model of the PIDRP is formulated as follows:

$$\Phi_{IP} = \min \sum_{t\in T} f_t z_t + \sum_{t\in T}\sum_{i\in N} e_{it}^C w_{it} + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^C I_{it}^C \tag{1a}$$

Additional new constraints are as follows:

$$\sum_{i\in N} w_{it} \leq 0.8Q\theta, \quad \forall t \in T, \tag{14}$$

$$w_{it} \leq D_{it}^{\max} z_{it}^C, \quad \forall i \in N,\ t \in T. \tag{15}$$

Allocation model modifies the objective function in the full model and replaces the routing variables with additional parameter to represent the costs (the second term in (1a)). The term $\sum_{it} e_{it}^C w_{it}$ is the approximated transportation cost replacing the term $\sum_{t\in T}\sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt}$, the actual distance (transportation) cost. Allocation model also uses the same constraints but eliminating the routing constraints (constraints (7)–(9)) and adding additional two constraints. Equation (14) limits the total amount that can be delivered on period t to a fixed percentage of the total transportation capacity. The parameter testing by Bard and Nananukul [2] showed that the percentage value of 80% always yielded feasible solutions. Additional variable z_{it}^C is included in constraint (15) to keep track of whether customer i receives a delivery in period t. Constraint (15) was a modification of constraint (11) by replacing the routing variables x_{ijt} with z_{it}^C.

3. Scatter Search

The scatter search metaheuristic has successfully been applied to a widespread variety of vehicle routing problems. Corberán et al. [15] proposed a scatter search to solve a real-life problem with multiple objectives. Two different heuristics were used to construct the initial trial solutions in the scatter search. Two simple exchange procedures, insertion and swap, are used to improve the solutions. The combination method is based on a voting scheme. The algorithms tested on real data show that scatter search can solve the practical problem efficiently. A more recent application of scatter search is by Mota et al. [16], who presented a scatter search for vehicle routing problem with split demands. Local search is adopted as the improvement method. Four kinds of critical clients are defined to produce new solutions. The algorithm was tested on a set of benchmark instances and it was found that scatter search algorithm always produces the least number of vehicles compared the tabu search developed by the authors.

Scatter search is an evolutionary metaheuristic that operates on a set of solutions, which the scatter search literature refers to as the reference set (*Refset*). The evolution of the *Refset* is achieved by way of combining reference solutions to yield trial solutions with combination of attributes not present in the previous set of solutions. The Refset is a collection of "good" solutions found during the search, where the meaning of "good" is not limited to quality as measured by the objective function value. For instance, a solution may be good because it provides diversity with respect to other solutions in the reference set. In fact, some implementations of scatter search divide the *Refset* into two subsets, consisting, respectively, of solution quality and diversity. Scatter search was first introduced by Glover [17]. Glover made a template in a version customized for nonlinear optimization problems with continuous variables. Laguna and Marti [18] published the first book on scatter search, containing introductory tutorials and advanced techniques such as the use of memory and path relinking.

The scatter search terminology that is used in this paper is similar to Laguna and Marti [18]. The algorithm is made up of several distinct steps: A diversification generator, an improvement method, a reference set update, a subset generation method which operates on the reference set in order to produce a subset of its solutions as a basis for creating combined solutions, and a solution combination method which transforms a given subset of solutions produced by the subset generation method into one or more combined solutions vectors.

The scatter search procedure stops when a termination criterion—either the maximum number of iterations, *Max-Iter*, is reached, or the reference set does not change, or improvement does not warrant further iterations.

The scatter search algorithm can be formally stated as shown in Algorithm 1.

Updating the Inventory Level (Step 7). In the inventory updating we propose two types of moves: a forward and backward transfer. The aim of the forward transfer is to reduce the inventory holding cost without increasing drastically the transportation cost. In the backward transfer the preference is given to the suppliers with the lower holding cost in order to determine whether the transportation and the inventory holding cost can be further consolidated. Examples of the forward and backward transfers are illustrated in Figures 1 and 2, respectively.

Backward Transfer. Figure 1 shows an example of backward transfer where the routings are separated by zeros and w_{it} and I_{it} are the pick-up quantity and the inventory, respectively. Assuming the coordinates of the 5 customers are $c_1(-2, 0)$, $c_2(1, 1)$, $c_3(4, 2)$, $c_4(3, 5)$, and $c_5(1, -2)$ and the depot is located at $D(0, 0)$ the holding costs per unit for each customer are $h_1 = 12$, $h_2 = 9$, $h_3 = 6$, $h_4 = 3$, and $h_5 = 6$ and the vehicle capacity is 10.

Step 1. Generate Solutions – Generate trial solutions using Giant Tour Procedure, sweep algorithm and savings algorithm
Step 2. Improve the Solutions – Apply 2-*opt* to improve solutions generated in Step 1
Step 3. Build the reference set – put |Refset| = $b_1 + b_2$ solutions in the reference set
Step 4. Initialize best solutions – make *BestSol* as the best solution in the current *Refset*,
$i = 1$
While ($i < MaxIter$) do
{
While (new solutions in *Refset*) do
{
Step 5. Generate subset – generate 2 pairs of solution (2 element subset) of the *Refset*.
Step 6. Combine solutions – generate new combined solutions from pairs of 2 element subsets in Step 5.
Step 7. Improve the solutions – update inventory level, and for the affected periods (transfer from and transfer to periods) apply within the same period 1-0 and 1-1 exchange, and 2-*opt*.
Step 8. Update reference set – update the reference set by maintaining the number of solutions inside the *Refset* by replacing the existing solutions with the better combined solutions.
Step 9. Update the best – update the *BestSol* in the *Refset*
$i = i + 1$
}
}

Algorithm 1

Before transfer	Period	Route							Route cost	Holding cost
	4	0	1	5	0	4	0		19.5035	18
	w_{it}		4	4		**4**				
	I_{it}		0	0		**0**				
	5	0	1	5	3	0	4	0	26.7396	0
	w_{it}		4	4	2		2			
	I_{it}		0	0	0		0			
Total cost									46.2431	18
After transfer	Period	Route							Route cost	Holding cost
	4	0	1	5	0	4	0		19.5035	24
	w_{it}		4	4		**6**				
	I_{it}		0	0		**2**				
	5	0	1	5	3	0			15.0777	0
	w_{it}		4	4	2					
	I_{it}		0	0	0					
Total cost									34.5812	24

Figure 1: Example of a backward transfer.

The selection of period and customer to be transferred is favorable to the lower holding cost. In this example, we select customer 4 in period 5. The saving is found by increasing the inventory cost and decrease in routing.

Initial routings are for periods 4 and 5, with the route cost 18.771591 and 24.36395 and 0 holding cost. According to the inventory updating mechanism, we transfer customer 4 in the period 5 to period 4. As the same customer is visited in period 4, we aggregate the amount to be transferred with the existing delivery quantity and we note that the resulting aggregated amount does not violate the capacity constraint. After the transfer of delivery amount by 2 units, we have a holding cost 6 with $h_4 = 3$. This will reduce the distance in period 5. The overall savings after the transfer is 64.2431 – 58.5812 = 5.6619.

Forward Transfer. Figure 2 shows a forward transfer and the objective is to reduce the holding cost whilst achieving a balance between the inventory and the transportation costs.

The selection of period and supplier to be transferred is biased towards customers with high holding cost. In this example, we select customer in period 1. Note that we limit the transfer to at most 2 periods only. This is to ensure that the increase in the routing cost is not exceedingly high.

The demands for customer 1 in periods 1, 2, and 3 are $d_{11} = 4$, $d_{12} = 2$, and $d_{13} = 4$. From the figure, $I_{11} = 6$ and $I_{12} = 4$,

Before	Period	Route									Route cost	Holding cost
	1	0	2	5	0	1	0	3	4	0	24.1156	81
	w_{it}		2	3		10		2	5			
	I_{it}		0	1		6		0	1			
	2	0	2	3	0						9.0486	72
	w_{it}		2	5								
	I_{it}		0	4								
	3	0	2	0	5	4	0				18.1756	36
	w_{it}		6		2	4						
	I_{it}		4		0	0						
Total cost											51.3398	189

After	Period	Route									Route cost	Holding cost
	1	0	2	5	0	1	0	3	4	0	24.1156	33
	w_{it}		2	3		6		2	5			
	I_{it}		0	1		2		0	1			
	2	0	2	3	0						9.0486	24
	w_{it}		2	5								
	I_{it}		0	4								
	3	0	2	0	1	5	4	0			21.5450	36
	w_{it}		6		4	2	4					
	I_{it}		4		0	0	0					
Total cost											60.4129	93

Figure 2: Example of a forward transfer.

the resultant holding cost for periods 1, 2, and 3 are 81, 24, and 36, respectively, and the total cost, including the routing cost for all 3 periods, is 240.3398. Customer 1 is not visited in periods 2 and 3, so we apply forward transfer by inserting customer 1 to period 3 according to the best insertion. Note that inserting customer 1 in period 2 results in the violation of vehicle capacity constraint. The saving after the transfer is $240.3398 - 153.4129 = 86.9269$.

3.1. Diversification Generation Method. Diversification generation method generates a set of diverse solutions which is denoted as P, and the size of the population of solutions is represented by *Psize* (i.e., $|P| = Psize$). Few different algorithms are applied within the diversification generation method to obtain initial solutions. This method produces feasible solutions that can be used as trial solutions for the scatter search procedure. The solutions are created using sweep algorithm [19], savings algorithm [20], and Giant Tour Procedure [14]. The Giant Tour Procedure starts by constructing cost network, which considers delivery amounts and vehicle capacity constraints. The construction starts by calculating the cost from the depot 0 to the customer c_i and from customer c_i going back to the depot as the cost of the arc $(0, c_i)$. If the combined deliveries of customers c_i and c_j are less than the vehicle's capacity, then the arc of (c_i, c_j) is added into the existing cost network. The construction is continued until the vehicle could not accommodate the new customer, and we start with a new vehicle. The process is continued until there are no more arcs connecting to the last customer in the giant tour. The giant tour is then partitioned using Djikstra's algorithm to form routes.

In order to improve the solutions obtained from sweep and saving algorithms, we applied Giant Tour Procedure following the order of the routes generated by sweep and savings algorithms. The procedure is able to improve on the solution by collapsing a few vehicles in each period.

3.2. Improvement Method. The solutions generated by the diversification generation method and combination method (see Section 3.5) are subjected to the improvement method. We apply 1-0 exchange, 1-1 interchange, 2-*opt** as interroute procedure, and 2-*opt* as intraroute in the improvement method to improve the solutions.

(1) 1-0 Exchange (insertion). This exchange method removes a customer form one route and inserts it into another route. Select a customer $c_i, i \in \{1, \ldots, N)$, from the route R_k in the period $t, t \in \{1, \ldots, \tau\}$, and insert it into another route R_l within the same period. The customer is inserted into another route if it does not violate any constraints and reduces the routing cost.

(2) 1-1 Interchange (swap). This method starts with removing two customers from their initial routes within the same period. The customer $c_i, i \in \{1, \ldots, N)$, from route R_k is exchanged with $c_j, j \in \{1, \ldots, N)$, from route R_l.

Table 1: Results from randomly generated data sets.

Case	CPLEX				SS			
	Lower bound	Best integer	Time (sec)	Veh. number	Obj.	Veh. number	Time (sec)	Gap
S12T5	4382.7182	4416.85	929.41	19	4430.96	19	21.47	0.32
S12T10	8330.1596	8674.45	1212.55	36	8746.18	36	35.79	0.83
S12T14	10694.11	11814.85	3600.35	51	11429.24	53	57.32	−3.26
S20T5	6617.0717	6882.51	1194.95	28	6813.3	29	83.8	−1.01
S20T10	13473.909	14280.74	1401.64	58	13993.89	60	126.02	−2.01
S20T14	18278.376	19448.52	897.93	80	19026.64	82	153.66	−2.17
S20T21	25369.66	26705.26	1989.7	118	26044.53	120	217.07	−2.47
S50T5	9840.5676	12006.72	2881.62	52	9957.22	56	315.05	−17.07
S50T10	11199.561	31335.94	713.58	114	22522.12	117	366.8	−28.13
S50T14	13750.901	45082.01	830.42	161	28480.12	164	437.96	−36.83
S50T21	9920.4403	117350.35	1056.38	239	39404.71	243	525.48	−66.42
S100T5	4442.154	36017.92	3352.85	—	19338.39	128	4855.77	−46.31
S100T10	—	—	—	—	39462.74	261	10934.05	—
S100T14	—	—	—	—	52185	365	13905.83	—

The exchange is applied if it could produce a shorter length of the routing cost or reduce the number of vehicles.

*(3) 2-opt**. 2-*opt**algorithm is similar to the 2-*opt*, but, instead of deleting edges within the same route, it deletes two different edges on the different routes and then reconnects them by considering the lower routing cost.

(4) 2-opt. This method selects two different edges within the same route and then deletes and reconnects them to other edges. The move is accepted if the resulting routing cost is lower than the previous cost. The process is continued until no further improvement is found.

3.3. Reference Set Update Method. *Refset* denotes the reference set and its size is denoted by b (i.e., $|Refset| = b$). Solutions are included in the reference set by a measure of quality (objective value) or diversity. Solutions are denoted by x^j, ($j \in 1, \ldots b$), and it is assumed that x^1 is the best solution and x^b is the worst solution according to the objective value. The reference set consists of two subsets, in which the subset of high quality solutions denoted by $Refset_1$ contains the b_1 best solutions, and the subset of diverse solutions denoted by $Refset_2$ contains the b_2 diverse solutions; hence $b = b_1 + b_2$.

The initial reference set consists of b_1 best solutions that belong to P and a number of b_2 solutions from P that maximize the minimum distance to *Refset*. The distance between the two solutions is calculated by adding the number of noncommon arcs of each solution before the combination. We adopt Russell and Chiang [21] for updating the *Refset* by composing the new reference set of $(b_1 + b_2)/2$ best solutions from the original reference set, and the remaining solutions are created from newly generated (combined) solutions. The solutions in the reference set are not changed until all combinations of solutions, generated by the combination method, are performed as prescribed by the subset generation method. This indicates the use of a static update of the reference set, where the set of solutions generated by the combination method is denoted as *Pool* and the subset of solutions chosen from *Refset* is denoted by *S*.

3.4. Subset Generation Method. Subset generation method is the foundation for constructing new solutions in scatter search. The subsets are built based on the reference set. This method generates subsets of the solutions in the reference set which are combined in the solution combination method. In our implementation, we restrict the method to select representative subsets by using Subset Type-1 [22], which consists of all 2-element subsets comprising of two different solutions which is explained further in the next section.

3.5. Solution Combination Method. We adapt the solution combination method in Torabi et al. [12] that use Subset Type-1 to create new solutions and applied to all pairs of subsets generated by subset generation method. Each subset comprises of two different solutions that is combined to generate a now solution. This method is divided into two steps. Step 1 aims to combine only the common elements of the combined routes and Step 2 assigns the demand of the remaining customers.

Step 1. Let A be a solution with p routes and B a solution with q routes, where A_i is the ith route for solution A, $i = \{1, \ldots, p\}$, and B_k is the kth route for solution B. The solutions A and B are combined as follows:

(1.1) Build a matrix $A \times B$ where its components (A_i, B_k) have the number of common arcs between the route i of solution A and route k of solution B.

(1.2) Choose the components (A_i, B_k) which have the greater number of common arcs; if there is a tie, then select randomly.

(1.3) The combined route is performed by the routes' common elements (arcs).

(1.4) Each combined route is excluded from the list (delete the line and column referring to components of (A_i, B_k)).

Step 2. This step aims to fulfill the customers who have not been served by Step 1 and done through the insertion procedure. This procedure is done as follows:

(2.1) Pick customer $i, i \in (A_i, B_k)$, which is the farthest one from the depot and is going to be inserted.

(2.2) Based on the initial solutions A and B, all the routes in which customer i is inserted are verified and also all the edges in which $x_{ij} = 1$ (customer i is served before customer j) or $x_{ji} = 1$ (customer i is served after customer j).

(2.3) For each j belonging to one of the combined routes in Step 1, we calculate the cost for inserting customer i before customer j or the cost for inserting customer i after customer j.

(2.4) Choose each possible insertion with the minimum cost. If there is no feasible position to be inserted, construct a new route.

The procedure stops when it reaches the maximum iteration.

4. Results and Discussion

All the algorithms are written in Matlab 7.7 and performed on a 3.1 GHz processor with 8 GB of RAM. For the algorithm testing, we generate randomly 14 data sets to test the algorithm that comprises of 12, 20, 50, and 100 customers with 5, 10, 14, and 21 periods. The locations of the customers are generated randomly in a square of 100×100. The locations of the customers for the 20 customers are extended from the 12 customers instance by adding 8 new randomly generated customers. Similarly, the 50 customers instance is extended from the 20 customers by generating randomly 30 additional customers. The same procedure was applied for the 100 customers instance by generating randomly an additional 50 customers' locations.

All of the data sets have demands in every period, with exception for case 50 customers. The holding cost for each customer is generated within the range $[1, 10]$ and the demands are generated randomly within the range $[0, 50]$. The vehicle capacity is fixed to 100 and the depot is located at $(0, 0)$ for all data sets.

Table 1 shows the results, the number of vehicles, and the CPU time for scatter search metaheuristic. CPLEX 12.4 is allowed to run 10800 seconds (3 hours) but, in most cases, CPLEX terminates prematurely because of being out of memory. It is observed that CPLEX failed to get lower bounds for large sized problems. We note that the formulation presented in Section 2 is weak because of the imposed number of vehicles and is able to solve to optimality for a very small problem [9]. However the CPLEX solutions are used as guideline in order to ensure that our solutions are correct. Table 1 illustrates the CPLEX solutions and the solutions obtained by our algorithm.

Table 2: Results for 50 customers instances.

Instance	MA|PM	RTS	SS			
	Obj.	Obj.	Obj.	(MA|PM)	RTS	Vehicles
1	378378	398795	**373172**	−1.38	−6.43	5
2	403913	373374	380129	−5.89	1.81	5
3	409573	353058	369355	−9.82	4.62	5
4	399220	361176	419256	5.02	16.08	5
5	422279	364819	410140	−2.87	12.42	5
6	407122	368082	372717	−8.45	1.26	5
7	414977	396963	**366617**	−11.65	−7.64	5
8	379744	370822	399269	5.14	7.67	5
9	407935	379379	416811	2.18	9.87	5
10	396258	370655	421940	6.48	13.84	5
11	402475	354025	382085	−5.07	7.93	5
12	358702	354981	365139	1.79	2.86	5
13	371030	365432	398346	7.36	9.01	5
14	406114	363404	386252	−4.89	6.29	5
15	373076	367659	406994	9.09	10.70	5
16	379404	360534	**360433**	−5.00	−0.03	5
17	406353	398442	400878	−1.35	0.61	5
18	401179	368533	**365400**	−8.92	−0.85	5
19	406893	377073	392953	−3.43	4.21	5
20	398508	372141	381826	−4.19	2.60	5
21	397112	374743	408826	2.95	9.10	5
22	358749	347329	355774	−0.83	2.43	5
23	407369	362619	398696	−2.13	9.95	5
24	369784	375022	**369717**	−0.02	−1.41	5
25	411556	374682	414254	0.66	10.56	5
26	408704	366167	**363157**	−11.14	−0.82	5
27	366197	375261	387595	5.84	3.29	5
28	401032	373155	416631	3.89	11.65	5
29	384282	379320	**371621**	−3.29	−2.03	5
30	369959	369223	**364819**	−1.39	−1.19	5

Furthermore, we test the performance of scatter search algorithm on the set of instances generated by Boudia et al. [4] comprising of three subsets of 30 instances each with 50, 100, and 200 customers over a planning horizon of 20 periods. Every subset of instances has a limited number of vehicles, with 5, 9, and 13 vehicles, respectively.

All of the data sets have demands in every period. The holding costs for each customer and the production plant are assigned at 1 for all instances, and the vehicle's capacity is fixed at 8000 for instances with 50 and 100 customers and 12000 for instances with 200 customers. The production capacities are fixed at 50000, 120000, and 240000 for instances with 50, 100, and 200 customers, respectively, and the storage capacity at the plant varies between one and a half and twice of the production capacity. The production setup cost is proportional to production capacity.

Table 3: Results for 100 customers instances.

Instance	MA|PM	RTS	SS			
	Obj.	Obj.	Obj.	(MA|PM)	RTS	Vehicles
1	714401	711671	714768	0.05	0.44	9
2	722047	694694	706531	−2.15	1.70	9
3	677598	683270	702056	3.61	2.75	9
4	710552	718252	715081	0.64	−0.44	9
5	733040	731260	**709207**	−3.25	−3.02	9
6	696146	744927	704383	1.18	−5.44	9
7	705322	695728	701951	−0.48	0.89	9
8	679210	706058	709180	4.41	0.44	9
9	699518	705035	721338	3.12	2.31	9
10	705778	696521	721714	2.26	3.62	9
11	709122	711895	**705973**	−0.44	−0.83	9
12	755726	703162	747220	−1.13	6.27	9
13	695466	721066	707786	1.77	−1.84	9
14	718260	698548	728132	1.37	4.24	9
15	736041	711506	729536	−0.88	2.53	9
16	715209	714873	**709663**	−0.78	−0.73	9
17	737832	702314	702391	−4.80	0.01	9
18	723413	720238	727677	0.59	1.03	9
19	720218	748734	728944	1.21	−2.64	9
20	724727	729099	738669	1.92	1.31	9
21	724328	738746	**712796**	−1.59	−3.51	9
22	701506	702849	717464	2.27	2.08	9
23	710033	712717	723209	1.86	1.47	9
24	734327	727741	**726290**	−1.09	−0.20	9
25	725446	725869	730997	0.77	0.71	9
26	718939	700719	722588	0.51	3.12	9
27	715068	686382	719054	0.56	4.76	9
28	685117	700980	695956	1.58	−0.72	9
29	722571	725030	**717728**	−0.67	−1.01	9
30	721850	698942	701024	−2.89	0.30	9

Table 4: Results for 200 customers instances.

	MA|PM	RTS	SS			
	Obj.	Obj.	Obj.	(MA|PM)	RTS	Vehicles
1	996151	1030684	1033864	3.79	0.31	13
2	978373	1010158	995600	1.76	−1.44	13
3	986147	1016681	**983290**	−0.29	−3.28	13
4	962937	1042854	1004710	4.34	−3.66	13
5	970638	1023680	1004463	3.48	−1.88	13
6	965646	1025262	967194	0.16	−5.66	13
7	980562	1038746	1011310	3.14	−2.64	13
8	1014809	1066068	**1001545**	−1.31	−6.05	13
9	967738	1018420	984520	1.73	−3.33	13
10	1093230	1035240	**1024832**	−6.26	−1.01	13
11	1008080	1037705	**981277**	−2.66	−5.44	13
12	998951	1035350	**995275**	−0.37	−3.87	13
13	984918	1063024	1028545	4.43	−3.24	13
14	964301	1024491	1002466	3.96	−2.15	13
15	981167	1026787	997539	1.67	−2.85	13
16	1017777	1033656	1028718	1.07	−0.48	13
17	1073640	1022250	1067403	−0.58	4.42	13
18	1003670	1063306	1030854	2.71	−3.05	13
19	997348	1065705	1053589	5.64	−1.14	13
20	981788	1027134	1001820	2.04	−2.46	13
21	974384	1044771	**965862**	−0.87	−7.55	13
22	1065780	1045790	**1001734**	−6.01	−4.21	13
23	1070520	1027042	**1014335**	−5.25	−1.24	13
24	978491	1045014	**973569**	−0.50	−6.84	13
25	1029327	1024239	**983691**	−4.43	−3.96	13
26	961728	1043128	988534	2.79	−5.23	13
27	1028006	1030753	**995478**	−3.16	−3.42	13
28	1011689	1032478	**994108**	−1.74	−3.72	13
29	1015741	1019371	1048934	3.27	2.90	13
30	985496	1027915	1024006	3.91	−0.38	13

Tables 2, 3, and 4 present the total costs and the number of vehicles for our algorithm compared to Memetic Algorithm/Population Management (MA|PM) algorithm [5] and Reactive Tabu Search (RTS) algorithm [2, 9] for instances with 50, 100, and 200 customers, respectively.

It is observed that scatter search algorithms outperform both the Memetic Algorithm/Population Management (MA|PM) algorithm of Boudia and Prins [5] and the Reactive Tabu Search (RTS) algorithm of Bard and Nananukul [2] in 28 instances. The results are highlighted in bold in Tables 2, 3, and 4. Individually our algorithm outperforms 44 of the instances compared to Memetic Algorithm/Population Management (MA|PM) algorithm [5] and 47 instances compared to Reactive Tabu Search (RTS) algorithm of Bard and Nananukul [2]. The highlighted objective functions indicate our algorithm outperforms both the memetic/population management and the reactive tabu search.

Table 5 shows the average total costs of the well know heuristics. All the results were taken from Adulyasak et al. [13]. It shows that SS is superior to GRASP [4] and comparable to MA-MP algorithms of Boudia and Prins (2007) and it also performs well for large instances when compared to the reactive tabu search algorithms of Bard and Nananukul [2]. However when compared to Armentano et al. [8] the difference between the results obtained by our algorithm is between 4% and 6% as worst off. Our algorithm outperforms only in one instance, instance 1 in 50-customer problem. When compared to the algorithms of Adulyasak et al. [10] our algorithm performs poorly and they are in between 10% and 12%.

5. Conclusion

In this paper, we present scatter search metaheuristic algorithms for a finite horizon, multiperiod, and multiproduct PIDRP with no-split deliveries and no-backlogging. We propose three-phase methodology that implements the allocation model in the first phase and construct the routes in Phase 2 for the vehicle routing problem. In Phase 3, we develop a scatter search algorithm by creating composite

Table 5: Average total costs obtained by different heuristics.

	GRASP[1]	MA-MP[2]	RTS[3]	TSPR[4]	ALNS[5]	SS	SS-GRASP	SS-MA-MP	SS-RTS	SS-TSPR	SS-ALNS
B1	443,264	393,263	393,662	361,704	346,878	387360	−14.43	−1.52	4.573	6.62	10.45
B2	791,839	714,627	712,294	685,898	636,962	716643	−10.49	0.28	0.61	4.29	11.12
B3	1070,026	1001,026	1034,923	951,638	876,761	1006302	−6.33	0.464	−2.84	5.43	12.87

[1]Boudia et al. [4].
[2]Boudia and Prins [5].
[3]Bard and Nananukul [2].
[4]Armentano et al. [8].
[5]Adulyasak et al. [10, 11].

decision rules and surrogate constraints to improve the initial solutions. Phase 1 is similar to Bard and Nananukul [2] but second phase uses the giant tour, savings, and sweep algorithms to construct the routes.

We observe that our algorithm is superior when compared to the memetic/population management of Boudia and Prins [5] for the 50 customers instances but performs quite poorly when compared to the reactive tabu search of Bard and Nananukul [2]. For large instances, the 200-customer problem, our algorithm is very much superior when compared to Bard and Nananukul [2], producing better results in 28 out of 30 instances and comparable to Boudia and Prins [5]. However our algorithm performs poorly when compared to Armentano et al. [8] and Adulyasak et al. [10]. Adulyasak et al. produced the best known results so far. For future research we need to further fine-tune our algorithm in order to be competitive with the current best known results.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work has been carried out under Fundamental Grant FRGS (F04/2010B) and University Of Malaya Research Grant UMRG-R116-10AFR. The authors would like to thank the two anonymous referees for their invaluable insights to improve the paper.

References

[1] L. Lei, S. Liu, A. Ruszczynski, and S. Park, "On the integrated production, inventory, and distribution routing problem," *IIE Transactions*, vol. 38, no. 11, pp. 955–970, 2006.

[2] J. F. Bard and N. Nananukul, "The integrated production-inventory-distribution-routing problem," *Journal of Scheduling*, vol. 12, no. 3, pp. 257–280, 2009.

[3] P. Chandra and M. L. Fisher, "Coordination of production and distribution planning," *European Journal of Operational Research*, vol. 72, no. 3, pp. 503–517, 1994.

[4] M. Boudia, M. A. O. Louly, and C. Prins, "A reactive GRASP and path relinking for a combined production-distribution problem," *Computers and Operations Research*, vol. 34, no. 11, pp. 3402–3419, 2007.

[5] M. Boudia and C. Prins, "A memetic algorithm with dynamic population management for an integrated production-distribution problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 703–715, 2009.

[6] J. F. Bard and N. Nananukul, "A branch-and-price algorithm for an integrated production and inventory routing problem," *Computers and Operations Research*, vol. 37, no. 12, pp. 2202–2217, 2010.

[7] N. Nananukul, "Clustering model and algorithm for production inventory and distribution problem," *Applied Mathematical Modelling* , vol. 37, no. 24, pp. 9846–9857, 2013.

[8] V. A. Armentano, A. L. Shiguemoto, and A. Løkketangen, "Tabu search with path relinking for an integrated production-distribution problem," *Computers & Operations Research*, vol. 38, no. 8, pp. 1199–1209, 2011.

[9] J. F. Bard and N. Nananukul, "Heuristics for a multiperiod inventory routing problem with production decisions," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 713–723, 2009.

[10] Y. Adulyasak, J.-F. Cordeau, and R. Jans, "Optimization-based adaptive large neighborhood search for the production routing problem," *Transportation Science*, vol. 48, no. 1, pp. 20–45, 2014.

[11] Y. Adulyasak, J.-F. Cordeau, and R. Jans, "Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems," *INFORMS Journal on Computing*, vol. 26, no. 1, pp. 103–120, 2014.

[12] S. A. Torabi, A. R. Pourghaderi, and S. Sekhavat, "A two step approach including scatter search algorithm for the integrated procurement, production and distribution planning," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '09)*, pp. 354–359, Hong Kong, China, December 2009.

[13] Y. Adulyasak, J.-F. Cordeau, and R. Jans, "The production routing problem: a review of formulations and solution algorithms," *Computers & Operations Research*, vol. 55, no. 1, pp. 141–152, 2015.

[14] A. Imran, S. Salhi, and N. A. Wassan, "A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem," *European Journal of Operational Research*, vol. 197, no. 2, pp. 509–518, 2009.

[15] A. Corberán, E. Fernández, M. Laguna, and R. Martí, "Heuristic solutions to the problem of routing school buses with multiple objectives," *Journal of the Operational Research Society*, vol. 53, no. 4, pp. 427–435, 2002.

[16] E. Mota, V. Campos, and A. Corberan, "A new metaheuristic for the vehicle routing problem with split demands," in *Evolutionary Computation in Combinatorial Optimization*, C. Cotta and J. Van Hemert, Eds., vol. 4446 of *Lecture Notes in Computer Science*, pp. 121–129, Springer, Berlin, Germany, 2007.

[17] F. Glover, "Heuristic for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, no. 1, pp. 156–166, 1977.

[18] M. Laguna and R. Marti, *Scatter Search: Methodology and Implementations in C*, Springer, New York, NY, USA, 2003.

[19] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations Research*, vol. 22, no. 2, pp. 340–349, 1974.

[20] G. Clarke and J. W. Wright, "Scheduling a number of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[21] R. A. Russell and W.-C. Chiang, "Scatter search for the vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 169, no. 2, pp. 606–622, 2006.

[22] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and Cybernetics*, vol. 29, no. 3, pp. 653–684, 2000.