Chiang Mai J. Sci. 2018; 45(2) : 1145-1160
http://epg.science.cmu.ac.th/ejournal/
Contributed Paper

MatHeuristic Approach for Production-Inventory-Distribution Routing Problem

Dicky Lim Teik Kyee and Noor Hasnah Moin*
Institute of Mathematical Sciences, Faculty of Science, University of Malaya, 50603 Kuala Lumpur, Malaysia.
* Author for correspondence; e-mail: noor_hasnah@um.edu.my

Received: 23 June 2015
Accepted: 26 December 2016

ABSTRACT

In this paper, the integrated Production, Inventory and Distribution Routing Problem (PIDRP) is modelled as a one-to-many distribution system, in which a single warehouse or production facility is responsible for restocking geographically dispersed customers whose demands are deterministic and time-varying. The demand can be satisfied either from inventory held at the customer sites or from daily production. A fleet of homogeneous capacitated vehicles for making the deliveries is also considered. Capacity constraints for the inventory are given for each customer and the demand must be fulfilled on time. We propose a two-phase approach within a MatHeuristic framework. Phase I solves a mixed integer programming model which includes all the constraints in the original model except the routing constraints. In phase 2, we propose a variable neighborhood search procedure as the metaheuristics for solving the problem. We carried out a statistical analysis and the findings showed that our results are significantly superior to those from the Greedy Randomized Adaptative Search Procedure (GRASP) in all instances. We also managed to improve 23 out of 30 instances when compared to the Memetic Algorithm with Population Management (MA|PM). The superiority of our algorithm is reemphasized when tested on larger instances with the results showing significantly improved solutions by 100% and 90% respectively when compared to GRASP and MA|PM.

Keywords: production-inventory-distribution routing problem, mixed integer programming, variable neighborhood search

1. INTRODUCTION

In an increasingly business environment, many companies face problems with their inventory and distribution management. After years of focusing on the reduction of costs in production and operation, they are beginning to investigate distribution activities as the last component for cost reduction. At the planning level, the goal is to coordinate production, inventory, and delivery (to meet customers' demands) so all the corresponding costs are minimized. However, although the literature on supply chain management is extensive, the benefits and challenges of coordinated decisions making within supply

chain scheduling models have not been intensively studied. In general, the problem of optimally coordinating production, inventory and transportation is called the production-inventory-distribution routing problem (PIDRP). This problem is known to be NP-hard [1] and PIDRP sometimes known as a production routing problem [2].

The PIDRP usually arises in the retail industry where customers or outlets rely on a central supplier or manufacturer to provide them with a given commodity on a regular basis. The integration of production, inventory, and distribution increases the complexity of the problem. It is characterized by a combination of a capacitated lot-sizing problem and a capacitated, multi-period vehicle routing problem (VRP). A manufacturer must develop minimum cost schedules for production and distribution for a single product that are sufficient to meet all customer demands over the planning horizon. The PIDRP is relevant in Vendor Managed Inventory, where the supplier (manufacturer) monitors the inventory at the retailers and coordinates efficient utilization of resources to replenish the retailers.

2. LITERATURE REVIEW

The PIDRP is different from the traditional VRPs because it requires multiple customer visits to satisfy demand spread out over an extended period of time. It is more related to the inventory routing problem, IRP [3-6] and the periodic routing problem, PRP [7-9]. The PIDRP is studied under the following assumptions: The production plants can be either single or multiple and they can produce a single or multiple products. The inventory policy commonly employed is that of the maximum or order-up-to level or a combination of both. The distribution is carried out by a fleet of homogeneous or heterogeneous vehicles with limited capacity. For detailed classification we refer the reader to [2].

Among the early papers to discuss the PIDRP are those of Chandra [10] and Chandra and Fisher [11], where the authors showed that the benefit of coordinating the three components amount to in a 3-20% cost savings compared to sequentially solving the problems. Lei et al [12] studied a multi-facility PIDRP with a heterogeneous fleet that was motivated by a chemical manufacturer with international customers. The authors proposed a two-phase solution. In phase 1, the model is solved as a mixed integer programming (MIP) problem subject to all the constraints in the original model with the condition that the routings are restricted to direct shipment between facilities and customer sites. The potential inefficiency of direct shipment is then solved in phase 2, where a heuristic procedure is applied to solve an associated consolidation problem that is formulated as a capacitated transportation problem with additional constraints. Testing showed that the approach gave good solutions to instances with up to 50 customer sites over 2 to 4 periods.

An alternative approach using the Greedy Randomized Adaptative Search Procedure (GRASP) is proposed by Boudia et al. [13]. The PIDRP considered comprises a single production facility that produces a single product. The originality of the approach lies in considering production and routing concurrently instead of resorting to the classical two-phase approach still widely used in practice. The two principal difficulties in the GRASP design were in randomizing the construction of a trial solution without excessively altering the solution quality and in developing a local search to modify both the production plan and the sets of trips for each period. The GRASP embedded a reactive tabu search algorithm in the

improvement strategy for solving the PIDRP. Similarly, Bard and Nananukul [14] developed a reactive tabu search algorithm for solving the PIDRP. An essential component of their methodology was the use of an MIP allocation model to find good feasible solutions that were used as starting points for the tabu search. The neighborhood consisted of swap and transfer moves. Path relinking was also used in the post-processing phase to seek out marginal cost reductions.

Armentano et al. [15] extended the ideas in [13, 14] to include multiple products. The authors presented two heuristic approaches that allow trajectories with feasible and infeasible solutions. The first approach is a tabu search with short memory that uses a compound move. At each iteration involved a shift of an amount of an item delivered in a given period to every preceding and succeeding period, the determination of a new route, and the formulation of a new production plan over the time horizon. The second approach makes use of path relinking that is integrated with tabu search, such that every local minimum in a tabu search is linked with the farthest solution of a pool of elite solutions. Both approaches were tested on a set of small and large generated instances with multiple items. The algorithms were also tested on a set of single item instances [13] and they outperformed the memetic algorithm suggested by Boudia and Prins [16] and the reactive tabu search developed by Bard and Nananukul [14].

Due to the complexity of the PIDRP, only a few researchers have proposed exact algorithms. Amongst them are Bard and Nananukul [1] who combined a heuristic within the exact branch and price framework. The approach exploits the efficiency of heuristics and the precision of branch and price. The authors devised a new branching strategy to accommodate the unique degeneracy characteristics of the master problem and the algorithms were tested on instances with up to 50 customers and 8 time periods. Adulyasak et al. [17] proposed a branch-and-cut algorithm for both the PIDRP and IRP for the maximum level (ML) and order-up-to level (OU) inventory replenishment policies. The algorithms were tested on PIDRP and IRP instances with up to 35 customers, three periods, and three vehicles and the algorithms were extended to parallel implementation to be able to solve larger instances. The largest instance for the PIDRP on a multicore machine is 35 customers, six periods, and three vehicles.

We refer the reader to the review paper by Sarmento and Nagi [18] and the recent review of Adulyasak et al. [2], which give a comprehensive state of the art of the PIDRP.

The contribution of this paper is in the design of a MatHeuristic algorithm to solve PIDRP problems where the first phase exploits the allocation model that solves an MIP problem by using approximated routes. The output of this model determines the delivery quantity and the inventory for each customer and these are incorporated into the modified variable neighborhood search adopted in the second phase. The modified variable neighborhood search incorporates the tabu search mechanism where it forbids the move that has recently been made in order to avoid being trapped in local minima. The model is solved by using the Concert Technology of CPLEX 12.5 Optimizers with Microsoft Visual C++ 2010. The results show that the MatHeuristic algorithm is better than the GRASP in all instances and very competitive when compared to MA|PM.

The paper is organized as follows: Section 3 reviews the description of PIDRP and its mathematical formulation. In Section

4, the two phase methodology we propose to use in solving the problem is discussed in detail. In Section 5, we present the computational results and the performance of our algorithm are compared with GRASP and MA|PM. Finally, we present our conclusion in Section 6.

3. PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

We consider a production, inventory and distribution routing problem (PIDRP) similar to the one proposed by [19]. The problem consists of a single production facility that produces a single product and distributes it to a set of n customers with time varying and non-negative demands d_{it} in each period and can be stored as the inventory by incurring unit holding cost at the production facility as well as the customer sites. If production takes place at the facility in period t, then a setup cost f is incurred. In constructing delivery schedules, each customer can be visited at most once per period (split delivery is not allowed) and each of the K homogeneous vehicles can make at most one trip per period. In this study, the initial inventories at the production facility and customer sites are assumed to be zero.

It is further assumed that at the end of the planning horizon all inventories (both at the production facility and customer sites) are zero. It is required that all deliveries take place at the beginning of the period and arrive on time to satisfy demand for at least that day. The objective is to construct a production plan and delivery schedule that minimizes production, inventory at the production facility and customer sites as well as distribution costs while fulfilling customer demand on time. The number of vehicles is given and the total delivery quantity must not exceed vehicle capacity.

The following notations are used in the development of the mathematical formulation.

Indices

i, j indices for customers, where 0 denotes the depot

t index for periods

N set of customers; $N_0 = N \cup \{0\}$ and $|N| = n$

T set of periods in the planning horizon; $T_0 = T \cup \{0\}$ and $|T| = \tau$

Parameters

d_{it} demand of customer i in period t

K number of vehicles

Q vehicle capacity

C_{ij} travel distance between customer i and customer j where $C_{ij} = C_{ji}$ and the triangle inequality, $C_{ik} + C_{kj} \geq C_{ij}$, holds for any i, j, k with i, j, k all distinct

C production capacity

f fixed production setup cost

h^P unit production holding cost

I^P_{max} maximum inventory level at the production facility

h^C_i unit inventory holding cost at customer site i

$I^C_{max,i}$ maximum inventory level at the customer site i

Decision Variables

x_{ijt} 1 if customer i immediately precedes customer j on a delivery route in period t ; 0 otherwise

y_{it} load on a vehicle immediately before making a delivery to customer i in period t

w_{it} amount delivered to customer i in period t

p_t production quantity in period t

I^P_t inventory at the production facility at the end of period t

z_t 1 if there is production in period t ; 0 otherwise

I^C_{it} inventory at customer site i at the end of period t

The PIDRP can be formulated as follows (model P):

$$\phi_{IP} = \min \sum_{t\in T} \sum_{i\in N_0} \sum_{j\in N_0} c_{ij} x_{ijt} + \sum_{t\in T} f z_t + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\in T\setminus\{\tau\}} \sum_{i\in N} h_i^C I_{it}^C \quad (1)$$

subject to:

$$I_t^P = I_{t-1}^P + p_t - \sum_{i\in N} w_{it}, \ \forall t \in T_0 \quad (2)$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it}, \ \forall i \in N, t \in T \quad (3)$$

$$\sum_{i\in N} w_{it} \le I_{t-1}^P, \ \forall i \in N, t \in T \quad (4)$$

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

$$p_0 \ge \sum_{i\in N} (d_{i1} - I_{i0}^C) \quad (6)$$

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

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

$$\sum_{j\in N} x_{0jt} \le K, \ \forall t \in T \quad (9)$$

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

$$w_{it} \le G_{it}^{max} \sum_{j\in N_0} x_{ijt}, \ \forall i \in N, t \in T \quad (11)$$

$$0 \le I_t^P \le I_{max}^P \quad 0 \le I_{it}^C \le I_{max,i}^C, \ \forall i \in N, t \in T\setminus\{\tau\}, I_\tau^P = I_{i\tau}^P = 0, \forall i \in N \quad (12)$$

$$x_{ijt} \in \{0, 1\}, z_t \in \{0, 1\}, 0 \le y_{it} \le Q, p_t \ge 0, w_{it} \ge 0 \text{ and integer}, \forall i \ne j \in N_0, t \in T \quad (13)$$

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

The objective function (1) expresses the minimization of the total cost of transportation, production setup and inventory at the production facility and at the customer sites. Constraints (2) and (3) represent the inventory flow balance equations for the production facility and the customers respectively. The total amount available for delivery on day t is limited by the amount in inventory at the production facility in period t - 1, as indicated in (4). The inequality (5) limits production in period t to the capacity of the production facility, and (6) allows production in period 0. The bound in (7) ensures that if customer i is serviced in period t, then it must have a successor on its route, and the route continuity is enforced by (8). The inequality in (9) limits the number of vehicles departing from the production facility in period t to the number K of available vehicles and (10) keeps track of the load on the vehicles. The value of G_t^{max} is specified to be as small as possible while ensuring that (10) is feasible. The amount delivered to each customer is limited by the parameter G_{it}^{max} in (11). The bounds for the variables are specified in (12) and (13).

4. SOLUTION METHODOLOGY

We propose a two phase approach to solve the PIDRP model. The allocation model, a simplified version of model P is solved in phase 1 to determine the production capacity, amount delivered to customers and inventories at both the production facility and customer sites. In phase 2, the delivery routes for each period are constructed based on the customer allocations obtained from phase 1 by using the giant tour and Dijkstra's algorithm and a complete initial solution is obtained. The variable neighborhood search (VNS) is developed to improve the initial solution.

4.1 Initial Solution

In the allocation model, the routing variables, x_{ijt} in the objective function and the associated constraints (7)-(11) are removed from the formulation. The routing variables are replaced an approximated routes and

an additional constraint that defines an aggregated vehicle capacity constraint is introduced.

The formulation without routing components requires some additional notations: f_{it}^{C} represents the fixed cost of making a delivery to customer i on day t, e_{it}^{C} represents the variable cost of delivering one item to customer i on day t, and z_{it}^{C} takes the value 1 if a delivery is made to customer i on day t and 0 otherwise.

As in Nananukul [19], we divide the problem into two cases. For instances with $n^2\tau \leq 500$, the routing costs on any period are approximated by the cost of a round trip between the depot and customer i (round trip value = $2c_{i0}$), and we use a surrogate cost term $\Sigma_{it} f_{it}^{C} z_{it}^{C}$ with $f_{it}^{C} = 2c_{i0}$ for all i and t with e_{it}^{C} is set to zero. For instances with $n^2\tau > 500$, we set $z_{it}^{C} = f_{it}^{C} = 0$ for all i and t, and the variable cost term $\Sigma_{it} e_{it}^{C} w_{it}$ is used to replace the routing component, where e_{it}^{C} is approximated by the ratio of the cost of making a delivery to customer i directly from the depot and the total demand of customer i in period t (i.e., $e_{it}^{C} = 2c_{0i}/d$). Since all instances we considered are $n^2\tau > 500$, we set the variables $z_{it}^{C} = f_{it}^{C} = 0$ and the allocation model is simplified as follows (Model P1).

$$\phi_{IP} = \min \sum_{t\in T}\sum_{i\in N} e_{it}^{C} w_{it} + \sum_{t\in T} f z_t + \sum_{t\in T_0\setminus\{\tau\}} h^{P} I_t^{P} + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h_i^{C} I_{it}^{C} \quad (1a)$$

with additional new constraint:

$$\sum_{i\in N} w_{it} \leq 0.8QK, \ \forall t \in T \quad (14)$$

The primary difference between the models P and P1 is that the routing variables (x_{ijt}) and related constraints (7) - (11) in the full model P have been removed in P1 and an additional constraint (14) has been added to limit the total amount that can be delivered in period t to a fixed percentage of the total transportation capacity. This provides a hedge against the need for split deliveries. By limiting the maximum load to 80% of the total vehicle capacity, the authors in [19] showed that the model always produces feasible solutions. Note that the routing component in the objective function (1a) has been replaced by $e_{it}^{C} w_{it}$.

4.2 Variable Neighborhood Search

The variable neighborhood search (VNS) was initially proposed by Mladenovic and Hansen [20] for solving combinatorial and global optimization problems. It works on the principle that different neighborhoods generate different search topologies and it is applied within local search. The exploration of the search space is carried out by the local search, which allows the algorithm to jump from one neighborhood to another. This allows the algorithm to escape from the local optimum.

Let us denote a finite set of pre-selected neighborhood structures by N_k, where $k = 1, ..., k_{max}$, with k_{max} referring to the maximum number of neighborhood used, and $N_k(x)$ the set of solutions in the k th neighborhood of x. The stopping condition may be the maximum number of iterations, maximum number of iterations between two improvements or the maximum CPU time allowed. There are three phases in the main VNS: *Shaking*, *Local Search*, and *Move or Not*. The basic VNS heuristic comprises the steps given in Figure 1. The flow chart of the VNS is outlined in Figure 2.

Initialization. Step 0: Define a set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search and a set of local searches $\mathcal{R}_l$, $l = 1, \ldots, l_{max}$; generate an initial solution x;
choose a stopping condition;
Repeat the following steps until the stopping condition is met:
Step 1: Set $k \leftarrow 1$;
Step 2: Until $k = k_{max}$, repeat the following steps:
(a) **Shaking**. Generate a point x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$)
(b) **Local Search**. Apply some local search method with x' as initial solution;
denote by x'' , the local optimum;
(c) **Move or not**. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and go back to (1); otherwise, set $k \leftarrow k + 1$.

Figure 1. Steps of the basic VNS.

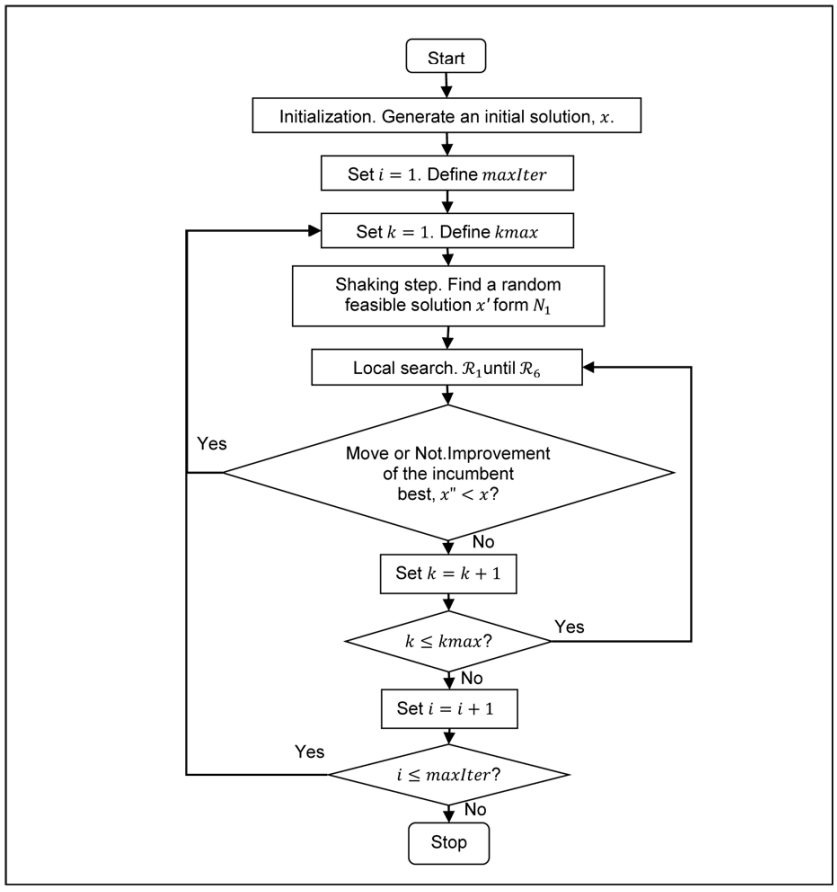

Figure 2. Flow chart of the VNS.

4.2.1 Step 0: initialization

The initial solution is obtained in two steps: (a) Construct a giant tour using the sweep algorithm as described in [21] and (b) Find the corresponding optimal fleet size by constructing a cost network and subsequently applying Dijkstra's algorithm, which provides an initial feasible solution that contains routes.

The giant tour, by definition includes all the customers obtained in phase 1. We define a tour $G = (V, A)$ with $V = \{v_1, v_2, ..., v_n\}$, the set of nodes representing all the customers' positions in the tour, and $A = \{(v_i, v_j): v_i, v_j \in V\}$ consisting of the arcs which

maintain the order of the customers' visitation sequence, together with a distance cost c_{ij}. Define a path starting from the depot to the closest customer, and repeat this step at each node $v_i \in V$ where i = 1, 2, 3, ..., n with n denoting the total number of customers to be served in the current period. In order to apply Dijkstra's algorithm, we first construct a cost network considering customer data, capacity constraint, distance constraint, and unit variable cost and fixed cost for a vehicle. For illustration, consider 12 customers making up the following giant tour σ = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) with customer demand d = (4, 2, 2, 2, 1,4, 2, 2, 2, 1, 4, 2). Assume that there is only one type of vehicle, with a maximum capacity of 10 units. Let c_{ij} be the distance between node i and node j. Let F denote the fixed vehicle cost and v the unit variable cost. Note that in our study, F and v are assumed to be 0 and 1, respectively.

We construct the cost network by considering the cost from the depot, denoted by 0, to customer 1 and from this customer to the depot (return journey) as the cost of arcs 0-1. This is expressed as $C_{01} = F + v(2c_{01})$. If the total demand of both customers 1 and 2 does not violate the capacity constraint of the vehicle, we proceed by considering the cost of the arcs 0-2 as $C_{02} = F + v(c_{01} + c_{12} + c_{20})$. We continue with this cost construction until the vehicle is full, and then we start using the next vehicle. Figure 3 shows that the vehicle can only visit customers 1, 2, 3 and 4. The process is continued until there are no more arcs connecting the last customer in the giant tour. In general, the cost of arc ij can be stated as

$$C_{ij} = F + v(c_{0,\,i+1} + \sum_{k=i+1}^{j-1} c_{k,\,k+1} + c_{j,\,0}) \qquad (15)$$

After the cost network has been constructed, Dijkstra's algorithm is then applied to obtain the initial feasible solution. This procedure is repeated for each period considered. After an initial feasible solution is found, set $x_{best} \leftarrow x$ and proceed to Step 1.

The stopping criteria may differ from one program to another. Most algorithms adopt the maximum number of iterations as the stopping condition, as in our case. Other criteria can also be used such as maximum running time or cpu time allowed, or number of iterations between two improvements.

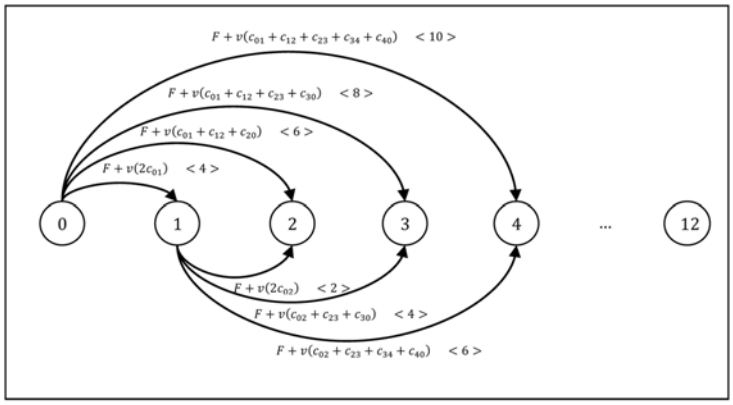

Figure 3. Construction of the cost network.

4.2.2 Step 2(a): shaking

In this step, a solution x' is picked randomly from the k^{th} neighborhood of the current solution x. This will ensure that the solution is not far from the current best solution. We consider four moves: Forward transfer, backward transfer, swap and transfer for the VNS. The algorithm of the shaking step is shown in Figure 4.

```
Set num=1
While (num<=k) do    //the number of changes depends on the value of k
{
        Randomly generate the value of r where 0<r<1. Define p1, p2 and p3
                if (r<=p1)     // the value of p1 represents the chosen probability
                    Apply forward transfer
                elseif (p1<r<=p2)
                    Apply backward transfer
                elseif(p2<r<=p3)
                    Apply swap
                else
                     Apply transfer
                endif
                num=num+1
}
```

Figure 4. The algorithm of the shaking step.

4.2.2.1 Neighborhood structure

We consider four neighborhood structures for each N_k: Forward and backward transfers, swap and transfer. The aim of the forward transfer is to reduce the inventory holding cost without drastically increasing the transportation cost. In the backward transfer, preference is given to the customers with lower holding costs in order to determine whether the transportation and the inventory holding cost can be further consolidated. Examples of a forward and a backward transfer are illustrated in Figures 5 and 6, respectively. In these examples we assume that 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 cost per unit for each customer is $h_1 = 12$, $h_2 = 9$, $h_3 = 6$, $h_4 = 3$ and $h_5 = 6$, the vehicle capacity is 10 and w_{it} and I_{it} are the pick-up quantity and the inventory, respectively and the routings are separated by zeros

Forward Transfer

Figure 5 illustrates a forward transfer where the selection of period and customer for the transfer is biased towards customers with high holding costs. In this example, we select customer 1 in period 1. Note that we limit the transfer to at most 2 periods. This is to ensure that the increase in the routing cost is not excessively high. The demand 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 $I_{12} = 4$, and so the resulting holding costs 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 the periods 2 and 3, so we apply forward transfer by inserting customer 1 into period 3 since insertion into period 2 would result in the violation of the vehicle capacity constraint. The savings after the transfer is 240.3398-153.4129= 86.9269.

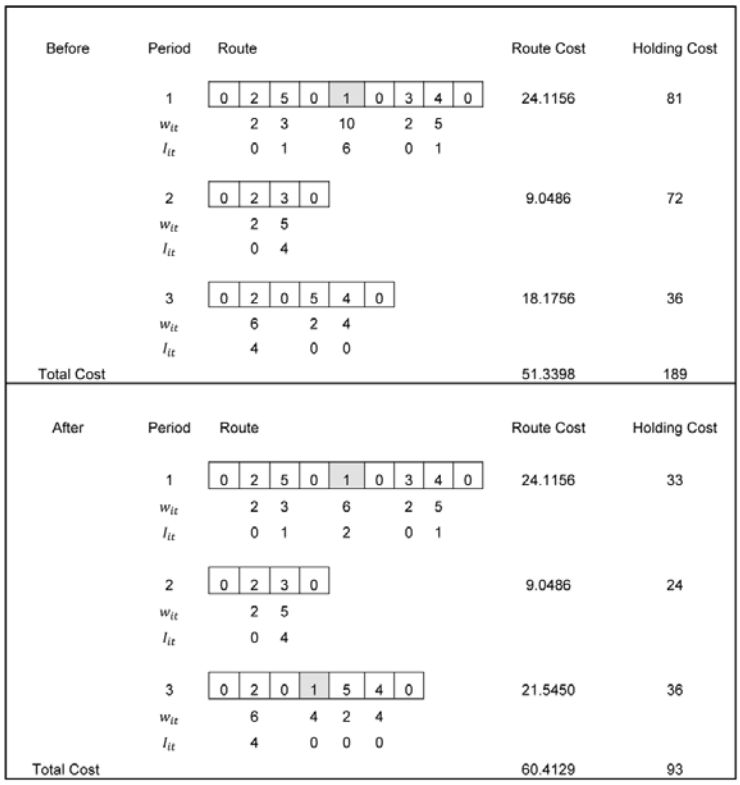

Figure 5. Example of a forward transfer.

Backward Transfer

The selection of period and customer for the transfer is favorable towards the lower holding cost in the backward transfer shown in Figure 6. In this example, we select customer 4 in period 5. The savings is effected by a decrease in the transportation cost at the expense of a slight increase in the inventory cost.

Note that the transfer of customer 4 with 2 units of delivery quantity in period 5 to period 4 would result in an additional increase of inventory cost of 6 since $h_4 = 3$. Since customer 4 is visited in period 4, so the new delivery amount of 6 units does not violate the vehicle capacity constraint. Customer 4 is eliminated in period 5 and the overall savings after the transfer is 64.2431 -58.5812= 5.6619.

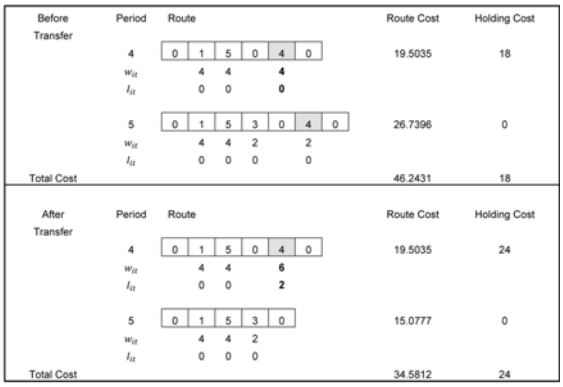

Figure 6. Example of a backward transfer.

Swap

The swap involves an exchange of delivery quantities between two customers i_1 in period t_1 with quantity $\overline{w}_{i_1t_1}$ and i_2 in period t_2 with quantity $\overline{w}_{i_2t_2}$, where t_2 is the first period after t_1 such that $\overline{w}_{i_2t_2} > 0$. For customer i_1, the move considers the maximum portion of $\overline{w}_{i_1t_1}$ that can be reassigned to period t_2 in exchange for $\overline{w}_{i_2t_2}$ without causing a shortage in period t_1. If customer i_1 was not scheduled for a delivery in period t_2, then insertion into one of the K routes is initiated. In general, a swap produces a change in both the inventory holding costs and transportation costs in periods t_1 and t_2.

Transfer

The transfer mechanism is similar to the backward transfer but periods t_1 and t_2, for the transfer must be separated by at least another period (i.e. $t_1 - t_2 \geq 2$). The transfer examines each customer i one at a time and tries to reassign the delivery quantity w_{it1} scheduled for t_1 to the latest period, call it t_2, preceding t_1 in which a delivery is scheduled for at least one other customer that is, $t_2 = max\{t : t < t_1, \exists \overline{w}_{it} > 0 \text{ for some } i \in N\}$.

We also incorporate the concept of tabu search, which forbids the movement of a customer for a few iterations if the customer is chosen in the transfer or swap. In all the four moves, only moves that result in feasible solutions are allowed, so it is necessary to check for violations of the production constraints and the inventory bounds at the plant and the customer sites, as well as the vehicle capacity constraint.

4.2.3 Step 2(b): local search

In our study, the local search consists of six refinement procedures adopted from Imran and Salhi [22]. The order of the refinement procedures is as follows: the 1-insertion inter-route as R_1, the 2-opt inter-route as R_2, the 2-opt intra route as R_3, the swap intra route as R_4, the 1-insertion intra-route as R_5, and finally the 2-insertion intra-route as R_6. The process starts by generating a random feasible solution x' from N_1, to be used as a temporary solution. The multi-level approach then starts by finding the best solution x'' using R_1. If x'' is better than x', then $x' = x''$ and the search returns to R_1, otherwise the next refinement procedure, R_2 is applied. This process is repeated until R_6 cannot produce a better solution.

4.2.4 Step 2(c): move or not

If the solution x'' obtained by the multi-level approach is better than the incumbent best solution x, then we set $x = x''$ and the search returns to N_1. However if x'' worse or the same as x, we generate x' from the next neighborhood, say $N_{k+1}(x)$, and return to step 2(b). The process is repeated until the search reaches $N_{k\max}$.

5. COMPUTATIONAL RESULTS

All the algorithms are written in Microsoft Visual C++ 2010 and performed on a 3.1 GHz processor with 8GB of RAM. The codes for the allocation model were implemented as mixed integer programming in Concert technology of Microsoft Visual Studio 2010 linked to the CPLEX 12.5 libraries. CPU times were obtained using the time function in C++.

We generate 14 instances comprising 12, 20, 50 and 100 customers with 5, 10, 14 and 21 periods. The locations of the customers are generated randomly on a 100×100 Euclidean grid. The locations for the 20 customers are extended from those for the 12 customers instance by adding 8 new randomly generated customers. Similarly, the locations for 50 customers are extended from those for the 20 customers by generating

randomly 30 additional customers and an additional 50 customers' locations are generated randomly for the case of 100 customers instance.

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

Table 1. Results for our data sets.

Case	CPLEX		MatHeuristic	
	Best Integer	Time (s)	Obj	Time (s)
S12T5	4416.85	929.41	3966.91	5.554
S12T10	8674.45	1212.55	7732.44	13.608
S12T14	11814.85	3600.35	11051.5	21.006
S20T5	6882.51	1194.95	6223.26	20.14
S20T10	14820.74	1401.64	12532.4	53.163
S20T14	19448.52	897.93	17235.6	78.411
S20T21	26705.26	1989.7	25487	166.499
S50T5	12006.72	2881.62	14469.6	140.226
S50T10	31335.94	713.58	29586	548.575
S50T14	45082.01	830.42	42238.5	1044.29
S50T21	117350.35	1056.38	62914.4	1579.11
S100T5	36017.92	3352.85	27756.5	1066.84
S100T10	-	-	56601.2	3781.4
S100T14	-	-	80843.7	7763.24

Table 1 illustrates the results for randomly generated data sets. We let the CPLEX run for a maximum of 7200 seconds (3 hours). The results show that our algorithm is better except for S50T5. Note that for instances S100T10 and S100T14, CPLEX was unable to produce any significant results.

The algorithm is tested on a benchmark data set provided by Boudia et al. [13] consisting of 30 instances of a 50-customer problem with a 20-period planning horizon and holding cost $h^P = 1$, $h_i^C = 0$ for all $i \in N$. These instances were randomly generated on a 100×100 Euclidean grid. For each customer i, demand was uniformly distributed between 0 and the storage capacity $I_{max;i}^C$. The vehicle capacity, $Q_{50} = 8000$ and the number of vehicles $K_{50} = 5$. Other parameters such as production capacity C, fixed production setup cost f and maximum inventory level I_{max}^P are set at 50000, 50000 and 100000, respectively.

Table 2 presents the total costs and the corresponding computational times for our algorithm, GRASP [13] and MA|PM [16]. Both GRASP and MA|PM were implemented in the Pascal-like programming language Delphi 7 and tested on a 2.8 GHz PC under Windows XP ([13] and [16]). Columns 2 and 4 give the best solutions for both GRASP and MA|PM. Our best solutions are superior in all instances when compared to GRASP and have improved 23 out of 30 solutions when compared to MA|PM.

Table 2. Results for 50-customer 20-period instance.

Instance	GRASP		MA|PM		MatHeuristic	
	Total Cost	Time (s)	Total Cost	Time (s)	Total Cost	Time (s)
1	440505	51.16	378378	170.35	374264	60.184
2	448695	88.42	403913	149.02	382679	53.696
3	419730	98.07	409573	135.72	374059	49.049
4	456398	56.13	399220	159.66	377022	69.877
5	434466	65.65	422279	193.34	372027	55.638
6	452564	90.73	407122	174.74	386968	64.324
7	436812	110.62	414977	173.98	377495	49.008
8	420935	77.34	379744	170.83	383376	29.003
9	434789	133.81	407935	158.09	383931	80.277
10	436221	121.12	396258	179.44	369095	33.691
11	433890	72.79	402475	178.25	367246	57.523
12	452705	74.49	358702	151.02	370920	52.134
13	440771	97.08	371030	193.33	372797	30.168
14	419412	79.90	406114	160.81	378460	55.876
15	453875	90.08	373076	173.41	385600	73.224
16	457310	90.51	379404	186.22	367923	55.196
17	455663	91.08	406353	177.52	389281	64.391
18	441685	88.53	401179	163.54	381641	59.171
19	418896	99.39	406893	186.93	373799	50.988
20	452183	91.25	398508	182.72	371942	59.563
21	409677	66.23	397112	188.36	367850	60.014
22	429116	83.17	358749	146.07	360679	33.793
23	443184	95.90	407369	188.49	375067	31.559
24	426113	101.40	369784	180.11	364765	34.79
25	462245	76.65	411556	192.38	384851	64.056
26	442029	75.03	408704	159.93	382984	36.3
27	444695	82.28	366197	173.13	365470	59.261
28	449894	104.55	401032	171.03	365761	60.314
29	461555	67.52	384282	198.35	389899	56.675
30	434006	88.39	369959	163.48	371576	52.623

STATISTICAL ANALYSIS

We performed nonparametric procedures, specifically the Friedman test, the Iman and Davenport and Friedman Aligned Rank tests to determine whether there is a significant difference between the algorithms. The choice of different tests is to mitigate the weakness of the Friedman test [23]. Here, the null hypothesis (H_0) for the test states that there is no significant difference between the behavior of GRASP, MA|PM and our algorithm.

The chi-square approximation of the Friedman test statistics (x_F^2) is calculated at the significance level of $\alpha = 0.05$ using equation (2) in [23]. The value obtained is $F_f = 50.4$ while the table critical chi-square value x_α^2 is $x_{0.05}^2 = 5.99$ with 2 degrees of

freedom. Since $\chi^2_F > \chi^2_{0.05}$, we reject the null hypothesis at $\alpha = 0.05$.

In view of the fact that the Friedman test is conservative, we utilize a less conservative test proposed by Iman and Davenport. By using equation (3) in [23], the Friedman value in (F_{ID}) Iman and Davenport, is computed using an F distribution with (k - 1) and (k - 1)(n - 1) degrees of freedom. The value of F_{ID} obtained is 152.3 and the critical value $F_{2,58} = 3.1504$. Since $F_{ID} > F_{2,58}$, we reject the null hypothesis at $\alpha = 0.05$.

In order to mitigate the weakness of the ranking scheme in the Friedman test, the Friedman Aligned Rank Test is adopted where the observation is aligned with respect to the problems as well as with respect to the algorithms. The Friedman Aligned Ranks test statistic is computed by using the equation (4) in [23]. We obtained the value of the test statistic $F_{AR} = 48.41$ and the critical value $\chi^2_{0.05} = 5.99$. Since $F_{AR} > \chi^2_{0.05}$, we reject the null hypothesis at significance level $\alpha = 0.05$.

Table 3 presents the results of all statistical tests. From the results, we can conclude that our algorithm is significantly different from GRASP and MA|PM .

Table 3. The results of the Friedman, Iman-Davenport and Friedman Aligned Rank tests ($\alpha = 0.05$).

Friedman Value	Critical Value in χ^2	Iman-Davenport Value	Critical Value in F distribution	Friedman Aligned Rank Value	Critical Value in χ^2
50.4	5.99	152.3	3.1504	48.41	5.99

We reject the null hypotheses in all the three tests. However, the main drawback of the Friedman, Iman-Davenport, and Friedman Aligned tests is that they can only detect significant differences over the whole multiple comparison, and are unable to establish proper comparisons between some of the algorithms considered. We carried out a post-hoc test using the Holm procedure to establish whether our algorithm (control algorithm MatHeuristic) is significantly better than GRASP and MA|PM. We chose Holm procedure because it is less conservative and more powerful than the Bonferroni method and is suitable for small sample sizes [23].

We obtained the p-value through the conversion of the rankings computed from the Friedman test by using normal approximation and compared it to those from GRASP and MA|PM with our algorithm as the control method. We then obtained the z-value by using equation (12) from [23]. For each z_i, the corresponding cumulative normal distribution values p_i can be obtained. Let p_1, p_2, ..., p_{k-1} be the ordered p-values (smallest to largest), so that $p_1 \leq p_2 \leq ... \leq p_{k-1}$, and let H_1, H_2, ..., H_{k-1} be the corresponding hypotheses. If p_i is below $\theta_i = \frac{\alpha}{k-i}$, the corresponding hypothesis H_i is rejected. In our study H_1 indicates that there is no significant difference between GRASP and the control algorithm whilst H_2 indicates that there is no significant difference between MA|PM and the control algorithm.

From the ranking results based on the Holm procedure displayed in Table 4, we can conclude that our algorithm is significantly better than both GRASP and MA|PM.

Table 4. Ranking results based on Holm Procedure for the compared algorithms.

i	Algorithms	z_i	p_i	θ	Hypothesis
0	MatHeuristic (control method)	-	-	-	-
1	GRASP	6.971	0	0.025	Rejected
2	MA|PM	2.324	0.0201	0.05	Rejected

Additional Data Set

We extended the problem to a larger data set consisting of 30 instances of 100 customers with a 20-period planning horizon (Boudia et al. [13]) with vehicle capacity Q_{100} = 8000 and the number of vehicles K_{100} = 9. Other parameters such as production capacity C, fixed production setup cost f and maximum inventory level I^p_{max} are set at 120000, 70000 and 400000 respectively. The results show that they are better in all instances (30) when compared to GRASP and 90% better (27 out of 30 instances) when compared to MA|PM.

6. CONCLUSION

In this paper, we proposed a two-phase methodology to solve the production-inventory-distribution routing problem (PIDRP). The problem is decomposed into two parts: An allocation model to determine the amount to be delivered and the corresponding inventory, and heuristic variable neighborhood search algorithm. The problem comprises a single product and multiple periods over a finite planning horizon.

Phase 1 solves the mixed integer programming allocation model. Once the amount to be delivered and the inventories were determined, the routes were constructed using a giant tour procedure to find a feasible solution. The solution is then improved using the well known variable neighborhood search algorithm which embeds the forward, backward, swap and transfer mechanisms to perturb the solution. In addition, local search is performed using the 1-insertion inter-route, the 2-opt inter-route, the 2-opt intra route, the swap intra route, 1-insertion intra-route, and the 2-insertion intra-route.

We compared our algorithm on a set of randomly generated problems with holding cost $h_i > 0$ with CPLEX to show the superiority of our algorithm. Our results were also compared to those from GRASP and MA|PM on benchmark instances and statistical analysis showed that our proposed algorithm can obtain high quality solutions in a reasonable computational time and is significantly better when compared to either GRASP or MA|PM. We extended our algorithm to a larger problem with 100 customers and 20 periods and the solutions obtained are better in all instances (30) when compared to those from GRASP and 90% better (27 out of 30 instances) when compared to those from MA|PM.

ACKNOWLEDGEMENTS

This research received support from University of Malaya Research Grant UMRG RG116/10AFR. The first author would also like to acknowledge the support from the MyMaster programme under the Department of Higher Education, Ministry of Education. We would like to thank the anonymous referees for their invaluable comments which have greatly improved the readability of this paper.

REFERENCES

[1] Bard J.F. and Nananukul N., *Comput. Oper. Res.*, 2010; **37(12)**: 2202-2217. DOI 10.1016/j.cor.2010.03.010.

[2] Adulyasak Y., Cordeau J.F. and Jans R., *Comput. Oper. Res.*, 2015; **55**: 141-152. DOI 10.1016/j.cor.2014.01.011.

[3] Abdelmaguid T.F. and Dessouky M.M., *Int. J. Prod. Res.*, 2006; **44(21)**: 4445-4464. DOI 10.1080/00207540600597138.

[4] Bard J.F., Huang L., Jaillet P. and Dror M., *Transportation Sci.*, 1998; **32(2)**: 189-203. DOI 10.1287/trsc.32.2.189.

[5] Golden B.L., Assad A.A. and Dahl R., *Large Scale Syst.*, 1984; **7(2-3)**: 181-190. DOI 10.1287/opre.1030.0096

[6] Dror M. and Ball M., *Nav. Res. Log.*, 1987, **34(4)**: 891-905. DOI 10.1002/1520-6750(198712)34:63.0.CO;2-J.

[7] Gaudioso M. and Paletta G., *Transportation Sci.*, 1992; **26(2)**: 86-92. DOI 10.1287/trsc.26.2.86.

[8] Mourgaya M. and Vanderbeck F., *Eur. J. Oper. Res.*, 2007; **183(3)**: 1028-1041. DOI 10.1016/j.ejor.2006.02.030.

[9] Parthanadee P. and Logendran R., *IIE Trans.*, 2006; **38(11)**: 1009-1026. DOI 10.1080/07408170600575454.

[10] Chandra P., *J. Oper. Res. Soc.*, 1993; **44(7)**: 681-692. DOI 10.1038/sj/jors/0440705.

[11] Chandra P. and Fisher M.L., *Eur. J. Oper. Res.*, 1994; **72(3)**: 503-517. DOI 10.1016/0377-2217(94)90419-7.

[12] Lei L., Liu S., Ruszczynski A. and Park S., *IIE Trans.*, 2006; **38(11)**: 955-970. DOI 10.1007/s10951-008-0081-9.

[13] Boudia M., Louly M.A.O. and Prins C., *Comput. Oper. Res.*, 2007; **34(11)**: 3402-3419. DOI 10.1016/j.cor.2006.02.005.

[14] Bard J.F. and Nananukul N., *J. Sched.*, 2009; **12(3)**: 257-280. DOI 10.1007/s10951-008-0081-9.

[15] Armentano V.A., Shiguemoto A.L. and Løkketagen L., *Comput. Oper. Res.*, 2011; **38(8)**: 1199-1209. DOI 10.1016/j.cor.2010.10.026.

[16] Boudia M. and Prins C., *Eur. J. Oper. Res.*, 2009; **195**: 703-715. DOI 10.1016/j.ejor.2007.07.034.

[17] Adulyasak Y., Cordeau J.F. and Jans R., *INFORMS J. Comput.*, 2013; **26(1)**: 103-120. DOI 10.1287/ijoc.2013.0550.

[18] Sarmiento A.M. and Nagi R., *IIE Trans.*, 1999; **31(11)**: 1061-1074. DOI 10.1023/A:1007623508610.

[19] Nananukul N., *Lot-sizing and Inventory Routing for a Production-distribution Supply Chain*, PhD Dissertation, The University of Texas, Austin, USA, 2008.

[20] Mladenovic N. and Hansen P., *Comput. Oper. Res.*, 1997; **24**: 1097-1100. DOI 10.1016/S0305-0548(97)00031-2.

[21] Gillett B.E. and Miller L.R., *Oper. Res.*, 1974; **22**: 340-344. DOI 10.1287/opre.22.2.340.

[22] Imran A., Salhi S. and Wassan N.A., *Eur. J. Oper. Res.*, 2009; **197(2)**: 509-518. DOI 10.1016/j.ejor.2008.07.022.

[23] Derrac J., Garcia S., Molina D. and Herrera F., *Swarm. Evol. Comput.*, 2011; **1(1)**: 3-18. DOI 10.1016/j.swevo.2011.02.002