IME639A: Analytics In Transport and Telecom

Term Project Report

INVENTORY ROUTING PROBLEM

Group 9:

Abhay Singh	*200013*
Dusane Atharv Sachin	*200356*
Aman Jain	*190105*
Moin Ahmed	*190510*

INTRODUCTION

Inventory routing problem (IRP) is a challenge that includes the integration and coordination of two supply chain management components: inventory management and transportation. This typically involves delivering to clients based on their needs from a warehouse or plant in order to provide the commodity on a regular basis. The major goal of this problem is to minimise the corresponding costs (fixed and variable costs) while meeting the delivery deadline. In general, IRP can be classified according to the planning horizon, whether it is single or multi-period, and whether the demand is predictable or stochastic. Many meta-heuristic approaches, such as genetic algorithms, tabu search, and branch-and-cut algorithms, have been adjusted to suit the issues and lead to optimal or near optimal solutions in the literature. We will first provide several literatures on IRP, with a focus on recent literatures.

THE PROBLEM AND PROPOSED SOLUTION

We consider a one to many network where a fleet of homogeneous vehicles transports multi products from a warehouse or depot to a set of geographically dispersed customers in a finite planning horizon. The following assumptions are made in this model.

- The fleet of homogenous vehicles with limited capacity is available at the warehouse.
- Customers can be served by more than one vehicle (split delivery is allowed).
- Each customer requests a distinct product and the demand for the product is known in advance but may vary between different periods.
- The holding cost per unit item per unit time is incurred at the customer sites. The holding cost does not vary throughout the planning horizon.
- The demand must be met on time and backordering or backlogging is not allowed.

The problem is modelled as mixed integer programming and the following notation is used in the model:

INDICES

$\tau = \{1,2, \ldots, T\}$ period index

$W = \{0\}$ warehouse depot

$S = \{1,2,...,N\}$ a set of customers where customer *i* demands product *i* only

PARAMETERS

C vehicles capacity (assume to be equal for all the vehicles).

F fixed vehicle cost per trip (assumed to be the same for all periods)

V travel cost per unit distance

M size of the vehicle fleet and it is assumed to be λ (unlimited)

c_{ij} travel distance between customer i and j where $c_{ij} = c_{ji}$ and the triangle inequality, $\dot{c}_{ik} + c_{kj} \geq c_{ij}$ holds for any i, j, and k with i ≠ j, i ≠ k, k ≠ j.

hi inventory carrying cost at the customer for product i per unit product per unit time

d_{it} demand of customer i in period t

VARIABLES

a_{it} delivery quantity to customer i in period t

I_{it} inventory level of product i at the customer i at the end of period **t**

q_{ijt} quantity transported through the directed arc (ij) in period t

x_{ijt} number of times that the directed arc ñ.ộ is visited by vehicles in period t

The model for our inventory routing problem is given below

$$Z = \min \underbrace{\sum_{t\in\tau}\sum_{i\in S} h_i I_{it}}_{\text{I}} + \underbrace{V\left(\sum_{t\in\tau}\sum_{j\in S}\sum_{i\in S\cup W} c_{ij}x_{ijt} + \sum_{t\in\tau}\sum_{i\in S} c_{i0}x_{i0t}\right)}_{\text{II}} + \underbrace{F\sum_{t\in\tau}\sum_{i\in S} x_{0it}}_{\text{III}} \quad (1)$$

subject to

$$I_{it} = I_{i,t-1} + a_{it} - d_{it}, \forall i \in S,\ \forall t \in \tau \quad (2)$$

$$\sum_{\substack{j\in S\cup W\\ i\neq j}} q_{ijt} + a_{it} = \sum_{\substack{j\in S\cup W\\ i\neq j}} q_{jit},\ \forall i \in S,\ \forall t \in \tau \quad (3)$$

$$\sum_{i\in S} q_{0it} = \sum_{i\in S} a_{it},\ \forall t \in \tau \quad (4)$$

$$\sum_{\substack{i\in S\cup W\\ i\neq j}} x_{ijt} = \sum_{\substack{i\in S\cup W\\ i\neq j}} x_{jit},\ \forall j \in W,\ \forall t \in \tau \quad (5)$$

$$I_{it} \geq 0, \forall i \in S,\ \forall t \in \tau \quad (6)$$

$$a_{it} \geq 0,\ \forall i \in S, \forall t \in \tau \quad (7)$$

$$q_{ijt} \geq 0,\ \forall i \in S \cup W,\ \forall j \in S,\ j \neq i, \forall t \in \tau \quad (8)$$

$$q_{ijt} \leq C x_{ijt}, \forall i \in S \cup W, \forall j \in S,\ i \neq j,\ \forall t \in \tau \quad (9)$$

$$x_{ijt} \in \{0,1\},\ \forall i, j \in S,\ \forall t \in \tau \quad (10)$$

$$x_{0jt} \geq 0, \text{ and integer},\ \forall j \in S,\ \forall t \in \tau \quad (11)$$

Heuristic Algorithm

Ant Colony Optimisation (ACO) is inspired by the nature behaviour of ants finding the shortest path between their colony and a source of food. The information collected by ants during the searching process is stored in pheromone trails. The higher density of pheromones on an arc leads to attracting more ants to the arc. Therefore, an appropriate formulation associated with the model for updating pheromones is very crucial.
The procedure for ACO can be divided into three main steps: the route construction, a local pheromone-update rule and a global pheromone-update rule. These steps are described in detail in the following subsections and Figure 1 outlines the algorithm and the following definitions are required:

τ_0 the amount of pheromone deposited (an initial pheromone value assigned to all arcs).

Q a random number generated from a uniform distribution over the interval (0, 1).

q_0 a predefined real number where $0 \leq q_0 \leq 1$.

τ_{ij} refer to the pheromone value allocated on arc (i, j).

η_{ij} $1/c_{ij}$ where c_{ij} is the length of arc (i,j).

α,β the parameters to control the influence of the pheromone value allocated on arc (i, j) and the desirability of arc (i, j) respectively.

Ω_i all the arcs connected to unvisited node j (unvisited customer) such that the ants in node i passing through arc (i, j) will not violate any constraint.

Note the value of τ_0, the initial value of pheromones for each arc is obtained from the total distance of the initial solution. Starting from the depot (warehouse) each ant utilizes equation (12) to select the next customer to be visited. Ants tend to be attracted to the arc which consists of higher density of pheromones. From equation (12), if q is less than the predefined parameter q_0 then the next arc chosen is the arc with the highest attraction. Otherwise, the next arc is chosen using the biassed Roulette Method with the state transition probability p_{ij} given by equation (14).

$$j = \begin{cases} \max_{j \in \Omega_i} \{Att_{i,j}\} & if \quad q \le q_0 \\ p_{ij} & otherwise \end{cases} \tag{12}$$

$$\text{where } Att_{ij} = (\tau_{ij})^{\alpha} (\eta_{ij})^{\beta} \tag{13}$$

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha} (\eta_{ij})^{\beta}}{\sum_{k \in \Omega_i} (\tau_{ik})^{\alpha} (\eta_{ik})^{\beta}} & \forall j \in \Omega_i \\ 0 & \forall j \notin \Omega_i \end{cases} \tag{14}$$

Local updating is used to reduce the amount of pheromone on all the visited arcs in order to simulate the natural evaporation of pheromone and it is intended to avoid a very strong arc being chosen by all the ants. After a predefined number of ants, m had completed their solutions, the best among the built solutions is chosen and the pheromone on each arc is updated using equation (15).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + (\rho)\tau_0 \tag{15}$$

where ρ*(rho)* represents the rate of pheromone evaporation.

After a predefined number of iterations, N_GL, the ACO updates the pheromone allocation on the arcs to compose the current optimum route γ^{gl}.The global pheromone-updating rule resets the ant colony's situation to a better starting point and encourages the use of shorter routes. Moreover, it increases the probability that future routes use the arcs contained in the best solutions. The classical ACO takes into account the transportation cost only. Since the IRP tries to find a balance between the transportation and inventory cost, it is natural to incorporate the inventory holding cost in the formulation. The global update rule is as follows:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \frac{\rho}{J_{\gamma^{gl}}} \qquad (i,j) \in \gamma^{gl} \tag{16}$$

Where $J_{\gamma^{gl}}$ is the weight of the best solution found where it incorporates the inventory element as well as the variable transportation costs and is given by

$$J_{\gamma^{gl}} = \sum_{t \in \tau} \sum_{i \in S} h_i I_{it} + V \left(\sum_{t \in \tau} \sum_{j \in S} \sum_{i \in S \cup W} c_{ij} x_{ijt} + \sum_{t \in \tau} \sum_{i \in S} c_{i0} x_{i0t} \right) \tag{17}$$

The routes can be further improved by adding some strategies in the procedure. Here, we adopted the route improvement strategies that focus on the split customers and they comprise of a transfer to the selected vehicle or a swap between different vehicles. Starting from the last vehicle, the split customer is identified and we try to merge to the current selected vehicle if the respective vehicle capacity is not violated. If this fails, then the swap with the other customers from the preceding vehicle or to the current selected vehicle that results in the least transportation cost is carried out. If none of the swap provides an

improvement in the objective value than the solution built by ACO, the route remains unchanged. The process continues until all vehicles in every period have been examined. The aim of this method is to eliminate the split customers (merge as many as possible) if the merged results the improved in the objective value. We also introduce the 2 – *opt* [10] heuristic as an intra route optimization procedure. The purpose of this strategy is to test on all possible pairwise exchange within a vehicle to see if an overall improvement in the objective function can be attained. The heuristic calculates the distances for all pair wise permutations and compared to those distance obtained by ACO. If any of these solutions is found to improve the objective function, then it replaces the current solution.
After a predefined number of iterations, N_DEM, the inventory level is updated. The followings are the definitions introduced for the procedure of updating inventory level:

N_moveData maximum number of moves to be allowed for each data
N_moveTime the maximum number of move to be allowed per time ***temp_move*** the current number of moves
sum_move the current accumulative moves that have been done ***cur_move***
Number of moves generated by a random number which is not more than N_moveTime per time.

The inventory updating process is done only if the sum_move is less than *N_moveData*. Otherwise, the existing inventory level of the current best solution is selected to continue the routing.

Algorithm for updating inventory level:

Step 1: Check the availability of customers on all periods. If none of the period consists of available customers, go to Step 9. Otherwise, go to Step 2.

Step 2: Randomly select the period, **p**, with the condition where there is at least one available customer. Go to Step 3.

Step 3: Randomly select the number of moves
(cannot exceed N_moveTime), cur_move:

If (cur_move + sum_move) <= N_moveData
real_move = cur_move

else

real_move = N_moveData – sum_move

Set temp_move = 0

Go to Step 4.

Step 4: Select an available customer from period **p**, who will give the least inventory cost. Move all the quantity delivery on period **p** to period p-1.

temp_move++

Go to Step 5.

Step 5: Update the availability of the customer on period **p**.

If (temp_move < real_move)

Go to Step 6

else

Go to Step 7

Step 6: Check if there are any available customers on period **p**.
If yes, go to Step 4. Otherwise, go to Step 7.

Step 7:
sum_move += temp_move
Go to Step 8.

Step 8: Update the inventory level and inventory cost for each customer on each period.

Step 9: Select the set of inventory levels that had been built for the current best solution to continue with the routing.

Results:

In a Jupyter notebook, we put the aforementioned algorithm to use.
The outcomes of this investigation are contrasted with the lower bound (LB) and upper bound (UB) produced by resolving the formulation described in the technique section. The algorithm is evaluated using combinations with 12, 20, 50, and 100 consumers as well as 5, 10, 14 and 21 various period combinations. Each customer's coordinates are generated at random within a 100x100 area. For the 20 customer instances, each customer's coordinates are made up of their existing 12 customer instances plus an additional 8 randomly generated coordinates. The 50 and 100 customer instances are created using the same process. While the demand for each customer is produced at random between 0 and 50, the holding cost for each customer ranges from 0 to 10.

For all the instances we obtain the lower bound and the best integer solution. The parameters are set as follows:
$\alpha = 1$
$\beta = 5$
$q0 = 0.9$
$\varrho = 0.1$
$\tau = 1/Lnn$

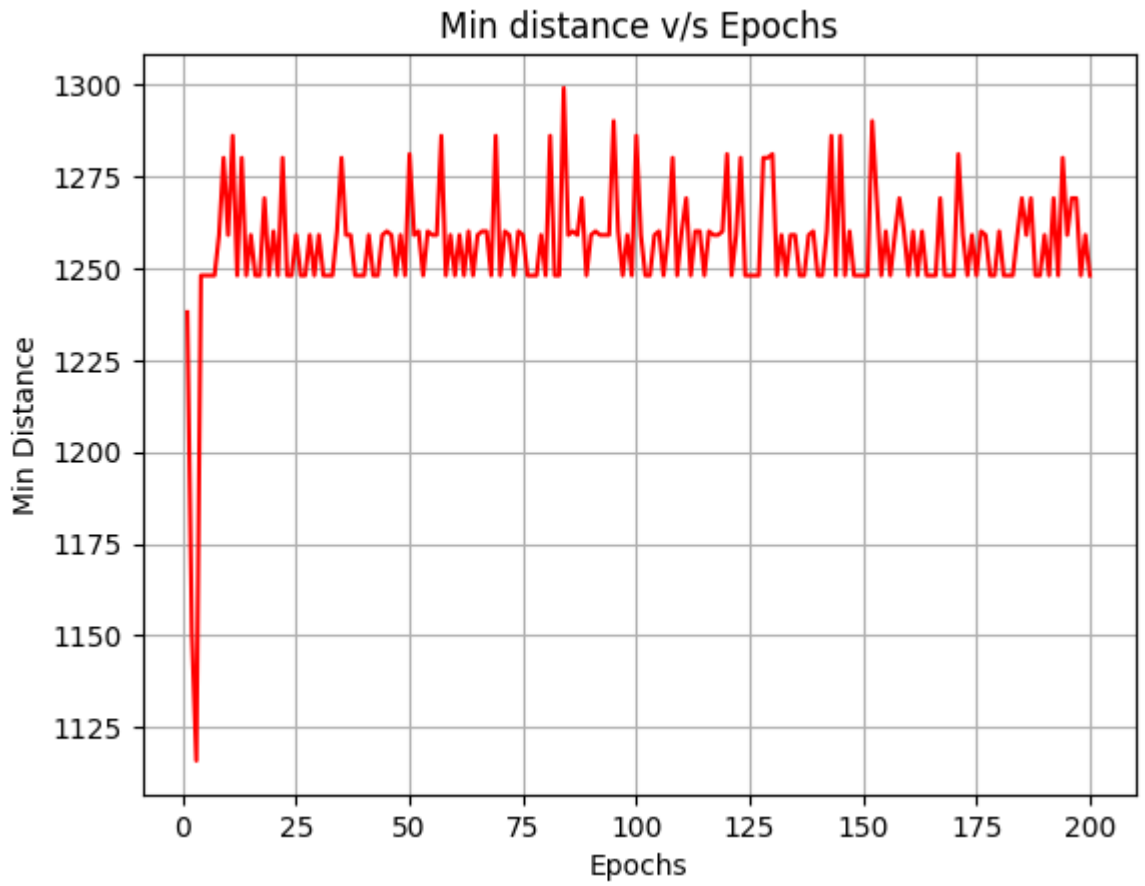
is the total distance obtained from the nearest neighbour algorithm.

The algorithm is run for 5000 iterations and each of the iteration consists of 25 ants to build a solution. *N_moveData* is determined by

*{T*N/12}*, while *N_moveTime* is set to be equal to 3.

For each data set, 10 runs were completed. derived from CPLEX, the best overall costs, the number of cars, the CPU time, the lower bound, and the higher bound were optimised. For all of the solutions, we found that the gap—defined as the difference between the lower bound and the upper limit divided by the lower bound—is larger than 10%. As the number of consumers and the number of periods rise, so does this ratio. As a result, it is challenging to defend the lower bound's quality, which CPLEX obtained. This can be because the upper limit is not very strong or the lower bound is very flimsy.

The total costs of 100 customers, as well as 50 customers with 14 and 21 periods, are lower than the upper bound, as is to be noted. For 100 customers, 50 customers with 14 and 21 periods, and the upper bound, the algorithm is able to produce superior results. Additionally, there is a less than 2% difference between the best expenses of 50 consumers with 5 and 10 period. For the small and medium instances, the algorithm's output showed a 10 percent or so difference between the optimal costs and the upper bound. Overall, it appears that the technique works better with larger instances. Here is a plot illustrating the minimum distance against epochs.



Conclusion:

In the supply chain management process, inventory and transportation integration is crucial. This paper outlines the model's formulation, which consists of an IRP with multiple items

and multiple time periods, as well as the creation of a modified ACO that incorporates the IRP's inventory component. All of the routes are improved by using transfer/swap and two-opt. Overall findings from this study indicate that, when compared to small and medium examples, the algorithm performs well in the larger instances. In the future, stronger route optimization techniques can be integrated into the algorithm to get better results.

References:

1. *M. Dorigo, "Learning and natural algorithms", Ph.D. dissertation, Politecnico di Milano, 1992.*
2. *N. H., Moin, S. Salhi and N. A. B. Aziz, Int J Prod Econ, 133, 334-343 (2011).*
3. *M. Dorigo and C. Blum, Theor Comput Sci, 344 (2-3), 243 – 278 (2005)*
4. *M. Dorigo and G. D. Caro, IEEE C Evol Computat, 2, 1470-1477 (1999).*