IME639A: ANALYTICS IN TRANSPORT AND TELECOM

Project Report

Site-Dependent Multi-Trip Periodic Vehicle Routing Problem

Problem statement:
http://people.brunel.ac.uk/~mastjjb/jeb/orlib/sdmtpvrpinfo.html

Project Members:

Akshit Singh Chauhan (200090)
Manish Mayank (190482)
Ronit Mittal (200820)
Sahil Singh (200838)

Course Instructor:

Dr. Faiz Hamid

Acknowledgement

It gives us a great sense of pleasure to present the project report on Site-Dependent Multi-Trip Periodic Vehicle Routing Problem. I owe special debt of gratitude to professor Dr. Faiz Hamid, Faculty Department of Industrial and Management Engineering, Indian Institute of Technology, Kanpur for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day.

Keywords

periodic vehicle routing problem; multiple use of vehicles; accessibility restrictions; site-dependencies; multi-trip; tabu search;

Abstract

In this project report, we consider a periodic vehicle routing problem that includes, in addition to the classical constraints, the possibility of a vehicle doing more than one route per day, as long as the maximum daily operation time for the vehicle is not exceeded. In addition some constraints relating to accessibility of the vehicles to the customers, in the sense that not every vehicle can visit every customer, must be observed. We refer to the problem awe consider as the site-dependent multi-trip periodic vehicle routing problem. Later in the report, we have discussed an heuristic approach called as tabu search algorithm.

Introduction

A basic problem that arises in distribution systems is the well-known vehicle routing problem (VRP). Given a set of customers with known geographical locations and demands that must be satisfied by a homogeneous fleet of vehicles based at a central depot, the problem consists of designing a set of routes, one per vehicle, that minimizes the total distance travelled. Each route starts and finishes at the depot and each customer is visited exactly once. The total demand on a route must not exceed the capacity of the vehicle and the route may have limited length or duration.

Background

In some distribution systems, planning is not on a daily basis, rather there is a periodicity in deliveries so that customers must be served from 1 to *t* times during a period of *t* days. When a customer has a delivery frequency smaller than the number of days in the period, the deliveries to that customer must be done using one of the feasible patterns of delivery days defined for that customer. This problem is known as the periodic vehicle routing problem (PVRP) and seeks to define the arrangement of the customer's deliveries in a given period at the same time as a set of routes, one per vehicle, for every day of the period are designed minimizing the total distance travelled. The service level required by every customer must be met, the total demand on every route in every day of the period must not exceed the vehicle capacity and the length or duration of the routes may be limited.

In some distribution systems not all vehicles can visit all customers. In other words, there exists a compatibility relationship between the customers and the vehicles. This problem is known in the literature as the site-dependent vehicle routing problem (SDVRP) and involves a fleet of heterogeneous vehicles. In such problems, in addition to the classic constraints for the VRP, customer/vehicle compatibility (accessibility) restrictions must be respected.

The SDVRP can be viewed as a special case of the PVRP by relating vehicle types in the SDVRP to days of the period in the PVRP. Both problems can be characterized as multilevel routing problems or generalized assignment problems: at the first level, an allowable vehicle type (or pattern of delivery days) is selected for each customer and at the second level, a capacity and time constrained VRP is solved for each type of vehicle (or day of the period). Taking advantage of this characteristic, solved the SDVRP applying the same algorithm used to solve the PVRP ().

The standard VRP definition implicitly assumes that each vehicle is used only once over the planning period. However, once the vehicle routes have been designed, it may be possible to assign several of them to the same vehicle such that the vehicle can feasibly operate them over the planning period. The problem of designing routes with multiple vehicle trips is known in the literature as the vehicle routing problem with multiple use of vehicles (VRPM) due to or the multi-trip vehicle routing problem. (MTVRP)

In this report, we consider the VRP that includes the characteristics of the former ones, that is, a PVRP with multiple vehicle use and accessibility restrictions. The problem will be called the site-dependent multi-trip periodic vehicle routing problem (denoted by the abbreviation SDMTPVRP).

Formulation

- **Notations**

$n \longrightarrow$ Total number of customers
$t \longrightarrow$ Number of the days of period
$m \longrightarrow$ Number of available vehicles
$w \longrightarrow$ Maximum number of daily routes per vehicle
$C \longrightarrow$ Set of available delivery-day patterns
$P \longrightarrow$ Set of available vehicle types
$K \longrightarrow$ Set of feasible vehicles
$C_i \in C \longrightarrow$ Set of feasible delivery-day patterns for customer i
$P_i \in P \longrightarrow$ Set of feasible vehicle types for customer i
$K_i \in K \longrightarrow$ Set of feasible vehicles for customer i
$q_i \longrightarrow$ Demand per service for customer i
$Q_k \longrightarrow$ Maximum load capacity of vehicle k
$D_l \longrightarrow$ Maximum duration of a route for vehicle k
$c_k \longrightarrow$ Operation cost per unit of distance travelled for vehicle k
$d_{ij} \longrightarrow$ Travel distance between customer i and customer j
$t_{ij} \longrightarrow$ Travel time between customer i and customer j
$\phi_i \longrightarrow$ Unload time for customer i
$\mu_i \longrightarrow$ Load time at the depot for customer i

- **Decision Variables**

The SDMTPVRP is defined on a multigraph $G = (V, A)$, where $V = v_0, v_1, ..., v_n$ is the set of vertices and $A = \{(v_i, v_j)^{k,r,l}\}$ is the set of arcs. Vertex v_0 represents the depot and the vertices $V - \{v_0\}$ correspond to the customers. Each arc is associated with a cost depending on the arc length and the vehicle that goes through it.

Here,
$k \longrightarrow$ vehicle
$r \longrightarrow$ route of the vehicle
$l \longrightarrow$ delivery-day

- **Objective function**

Making use of the notation above, the set K_i of every feasible vehicle customer i is created with every vehicle from the set K whose type belongs to the given set P_i of the feasible vehicle types for customer i. The task of generating these sets is accomplished in a starting step. Since the depot does not have any demand, we can assume $q_0 = \mu_0 = \phi_0 = 0$; $K_0 = K$ is also assumed. The objective function can be written in thne following way:

$$\text{Minimize} \quad \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k \in K_i \cap K_j} \sum_{r=1}^{w} \sum_{l=1}^{t} d_{ij} c_k x_{ij}^{krl}$$

- **Constraints**

Constraint to ensure that one feasible delivery pattern is assigned to each customer.

$$\sum_{h \in C_i} y_{ih} = 1 \qquad (i = 1, ..., n) \tag{1}$$

Constraint to ensure that each customer is visited only on the days corresponding to the assigned pattern.

$$\sum_{j=0}^{n} \sum_{k \in K_i \cap K_j} \sum_{r=1}^{w} x_{ij}^{krl} - \sum_{h \in C_i} a_{hl} y_{ih} = 0 \qquad (i = 1, ..., n;\ l = 1, ..., t) \tag{2}$$

Constraint to make sure that when a vehicle arrives at a customer on a given day it also leaves that customer on the same day on the same vehicle following the same route.

$$\sum_{i=0}^{n} x_{ie}^{krl} - \sum_{j=0}^{n} x_{ej}^{krl} = 0 \qquad (e = 1, ..., n;\ k \in K_e;\ r = 1, ..., w;\ l = 1, ..., t) \tag{3}$$

Constraint to ensure that if some vehicle leaves the depot then it does so only once per route.

$$\sum_{j=0}^{n} x_{0j}^{krl} \leq 1 \qquad (k = 1, ..., m;\ r = 1, ..., w;\ l = 1, ..., t) \tag{4}$$

Constraint to make sure that is the load on vehicle is not exceeding the maximum limit.

$$\sum_{i=0}^{n} \sum_{j=0}^{n} q_i x_{ij}^{krl} \leq Q_k \qquad (k = 1, ..., m;\ r = 1, ..., w;\ l = 1, ..., t) \tag{5}$$

Constraint to take the time care of the maximum operational time of vehicles.

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{r=1}^{w}(\mu_i + t_{ij} + \phi_i)x_{ij}^{krl} \leq D_k \qquad (k = 1, ..., m;\ l = 1, ..., t) \qquad (6)$$

Constraint for eliminating subtour constraints.

$$\sum_{i,v_i \in S}\ \sum_{j,v_j \in S} x_{ij}^{krl} \leq |S| - 1 \qquad (k = 1, ..., m;\ r = 1, ..., w;\ l = 1, ..., t; \forall S \subseteq V - \{0\};\ |S| \geq 2) \qquad (7)$$

Binary Constraints (12), (13), (14).

$$a_{lh} \in \{0, 1\} \qquad (l = 1, ..., t;\ \forall h \in C_i) \qquad (8)$$

$$x_{ij}^{krl} \in \{0, 1\} \qquad (i = 0, ..., n;\ j = 0, ..., n;\ k = 1, ..., m;\ r = 1, ..., w;\ l = 1, ..., t) \qquad (9)$$

$$y_{ih} \in \{0, 1\} \qquad (i = 0, ..., n;\ \forall h \in C_i) \qquad (10)$$

Heuristic algorithm for the Site-Dependent Multi-Trip Periodic Vehicle Routing Problem

○ Tabu Search Algorithm

Our tabu search algorithm for the SDMTPVRP is based on that developed by for the PVRP and solves a generalization of that problem. Note here that the algorithm itself shares many features with the earlier tabu search algorithm developed for the VRP by . Although the structure of the iterative process within our algorithm is similar to that in , there are key differences with respect to:

- the definition of the attributes of a solution;
- the construction of the neighbourhood of a solution;
- the evaluation of the objective function and
- the construction of the initial solution.

The tabu search algorithm uses the GENIUS heuristic, developed by for the insertion and removal of customers from routes. We first present our heuristic to generate the initial solution from which start the search. This heuristic shares some features with that presented in for the same task, but in this case several steps have been added in order to cope with the accessibility constraints and multiple vehicle trips. We then explain the general functioning of the search procedure, followed by a complete description of our tabu search algorithm.

The algorithm starts with an initial solution s^0, is initialized with the parameter vector $(\delta,\gamma,\theta,\eta,p)$, and returns the best solution found s^*, if any. We assume that $c(s^*)=\infty$ if no feasible solution has yet been found. The steps followed by the algorithm are:

1. If s^0 is feasible, set $s^* := s^0$. Set $\alpha := 1$ and $\beta := 1$.
2. For each attribute (i,k,r,l):
    1. Set $\rho_{ikrl} := 0$ and $\tau_{ikrl} := 0$.
    2. If $(i,k,r,l) \in B(s^0)$ and s^0 is feasible, set $\sigma_{ikrl} := c(s^0)$; otherwise, set $\sigma_{ikrl} := \infty$.
3. Set $s := s^0$.
4. For $\lambda = 1,\ldots,\eta$,
    1. Set $M(s) := \emptyset$.
    2. For each $\bar{s} \in N(s)$, set $M(s) := M(s) \cup \bar{s}$ if there exists $(i,k,r,l) \in B(\bar{s}) \setminus B(s)$ such that:
        - $\tau_{ikrl} < \lambda$; or
        - $\bar{s}$ is feasible and $c(\bar{s}) < \sigma_{ikrl}$.
    3. For each $\bar{s} \in M(s)$,
        - If $f(\bar{s}) \geqslant f(s)$, set
        - Otherwise, set $g(\bar{s}) := f(\bar{s})$.
    4. Identify the solution $s' \in M(s)$ such that $g(s') = \min\{g(\bar{s}), \bar{s} \in M(s)\}$. We now move from the current solution s to this neighbouring solution s' by suitable adjustment of the routes associated with s.

5. Let R represent the set of routes associated with the solution s and:
    - For each attribute $(i,k,r,l) \in B(s) \backslash B(s')$,
        - ° Adjust R by removing customer i from route r of vehicle k on day l using the GENIUS heuristic.
        - ° Set $\tau_{ikrl} := \lambda + \theta$.
    - For each attribute $(i,k,r,l) \in B(s') \backslash B(s)$,
        - ° Adjust R by inserting customer i into route r of vehicle k on day l using the GENIUS heuristic.
        - ° Set $\rho_{ikrl} := \rho_{ikrl} + 1$.
6. Let s represent the solution associated with R. If s is feasible:
    - Set $s^* := s$ if $c(s) < c(s^*)$.
    - For each $(i,k,r,l) \in B(s)$, set $\sigma_{ikrl} := \min\{\sigma_{ikrl}, c(s)\}$.
7. If $q(s)=0$, set $\alpha := \alpha/(1+\delta)$; otherwise set $\alpha := \alpha(1+\delta)$
8. If $d(s)=0$, set $\beta := \beta/(1+\delta)$; otherwise set $\beta := \beta(1+\delta)$

Steps 1–3 constitute the initialization process of the search procedure while Step 4 constitutes the iterative process. Within this iterative process, Steps (a)–(d) involve the construction of a subset of the neighbourhood of the current solution from which a good neighbouring solution to move to is selected. Step (e) corresponds to the removal and insertion of customers in appropriate routes guided by the attribute set differences between the current solution and its chosen neighbour. Note here that, because of the way the neighbourhood is defined, any neighbour automatically satisfies the constraints relating to delivery-day patterns. At Step (f) if a feasible solution has been reached attribute aspiration levels are updated, as is the best solution found. At Steps (g) and (h) the penalizing factors for the over-capacity and over-time are also updated depending on the new current solution. Note that we apply adaptive penalties here (unlike some previous work, eg which used constant penalty values).

An interesting feature of our algorithm is that, considering practical constraints, it allows the possibility of including, in a very easy way, constraints such as:

- prevent a specific vehicle visiting a specific customer, even if the vehicle belongs to a feasible vehicle type for that customer;
- force a specific vehicle to visit a specific customer.

  This is because the algorithm uses the set K_i of feasible vehicles for each customer i (created from the set P_i of feasible vehicle types), so the constraints referred to above can be dealt with by amending K_i appropriately.

  Note here that the above algorithm reduces to that in when both $|P|$, the number of different vehicle types, and w, the number of routes per vehicle are one. Nevertheless, it should be stressed here that our algorithm is capable of dealing with a much more general problem than the algorithm of , namely any problem with $|P|>1$ and/or $w>1$.

Clarke and Wright Savings Algorithm

Clarke and Wright Savings Algorithm is also a possible heuristic algorithm for solving the Site-Dependent Multi-Trip Periodic Vehicle Routing Problem.

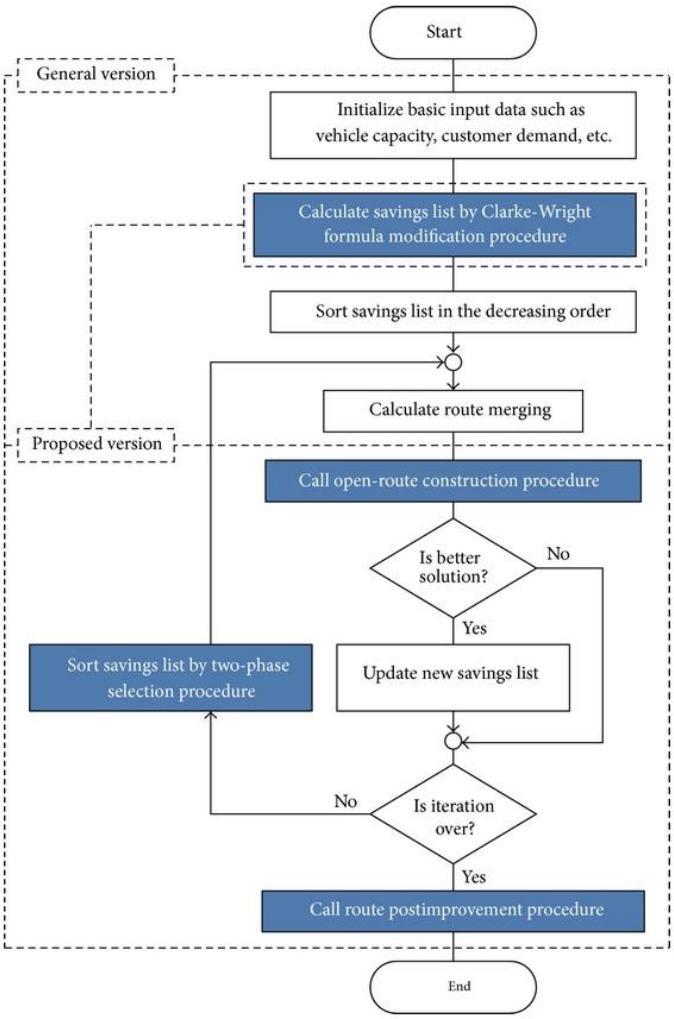
The algorithm begins by computing the savings for all pairs of customers, where the savings for a pair of customers i and j is defined as the cost reduction that would result from combining their deliveries into a single route, as opposed to delivering them separately. Here is how the cost savings are computed:

$$s(i,j) = c(0,i) + c(0,j) - tt * c(i,j)$$

Here, c(i, j) is the cost of traveling from customer i to customer j, c(0, i) is the cost of traveling from the depot to customer i, and tt is the proportionality constant for time. The savings are sorted in descending order, and the algorithm proceeds to combine the customers into routes, starting with the highest savings.

Each customer is initially assigned to a distinct route, and the algorithm then merges routes that share customers until no further improvements are possible. The algorithm ends once no additional routes can be merged.

This heuristic algorithm is a good starting point for locating high-quality solutions, and it can be improved with additional local search techniques, such as 2-opt or tabu search.



The Clarke and Wright Savings Algorithm is a heuristic algorithm for solving the Vehicle Routing Problem (VRP) that was introduced by Clarke and Wright in 1964. The algorithm starts with an initial solution that is usually generated by a simple construction heuristic, and then improves the solution iteratively by combining routes that share common customers. The algorithm has been shown to provide good solutions for many VRP instances, and is relatively simple to implement.

In the context of the Site-Dependent Multi-Trip Periodic Vehicle Routing Problem, the Clarke and Wright Savings Algorithm can be used to generate feasible routes that satisfy the delivery-day patterns and accessibility restrictions. The algorithm proceeds as follows:

1. **Compute the savings for all pairs of customers**

   For each pair of customers i and j, the algorithm computes the savings s(i, j) as the reduction in cost that would be achieved by combining their deliveries into a single route, compared to delivering them separately.

   The savings are computed as: $s(i, j) = c(0, i) + c(0, j) - tt * c(i, j)$

   Here, c(0, i) is the cost of traveling from the depot to customer i, c(0, j) is the cost of traveling from the depot to customer j, and c(i, j) is the cost of traveling from customer I to customer j. The travel costs are computed using the proportionality constant tt, which converts distance units into time units.

2. **Sort the savings in descending order**

   The savings are sorted in descending order, so that pairs of customers with the highest savings are considered first. The algorithm proceeds to combine customers into routes starting with the highest savings, until no further improvements can be made.

3. **Combine routes that share common customers**

   Initially, each customer is assigned to a separate route. The algorithm proceeds to merge routes that share common customers, until no further improvements can be made. The merging process proceeds as follows:
   For each pair of routes R1 and R2 that share a customer i, compute the savings s(i, R1, R2) as the reduction in cost that would be achieved by merging the routes at customer i, compared to keeping them separate.

   The savings are computed as: $s(i, R1, R2) = c(0, i) + c(i, 0) - tt * c(R1) - tt * c(R2)$ Here, c(R1) is the cost of the route R1, and c(R2) is the cost of the route R2.

   Sort the savings s(i, R1, R2) in descending order, and consider the pairs of routes with the highest savings first. If merging the routes at customer i would not violate any delivery-day patterns or accessibility

restrictions, and would not exceed the maximum load or duration constraints for any vehicle, then merge the routes at customer i. Repeat the merging process until no further routes can be merged.

4. **Output the final solution**

   The algorithm outputs the final solution, which consists of a set of routes that satisfy the delivery-day patterns and accessibility restrictions.
   Additional local search procedures, such as 2-opt or tabu search, can be used to improve the solution obtained by the Clarke and Wright Savings Algorithm. These procedures can be used to iteratively swap pairs of customers between routes in order to further reduce the total travel cost.

Code Files

We have written two code files for this problem statement.

1. Code 1: Using Python library pulp
   This uses python library pulp and computes the optimization problem.

2. Code 2: Using Cplex in Python
   This uses python library pulp and computes the optimization problem.

References:

http://people.brunel.ac.uk/~mastjjb/jeb/jeb.html#vehicle

https://link.springer.com/article/10.1057/palgrave.jors.2602405

https://journals.sagepub.com/doi/10.1177/0734242X18807001

https://www.jstor.org/stable/pdf/20202157.pdf?refreqid=excelsior%3Aa7a0d7b6741ca8a8e1514d6f989babe0&ab_segments=&origin=&initiator=

https://onlinelibrary.wiley.com/doi/full/10.1002/net.22028

https://www.researchgate.net/publication/349669857_A_solution_approach_for_multi-trip_vehicle_routing_problems_with_time_windows_fleet_sizing_and_depot_location/link/609e7467299bf14769983728/download