Solving Vehicle Routing Problem by using Brute Force Algorithm

by Mosharaf Repon

Submission date: 23-May-2022 10:53PM (UTC+0600)
Submission ID: 1842633980
File name: Repon_Thesis_Report_Updated_Plag.docx (1.37M)
Word count: 8366
Character count: 42528

Solving Vehicle Routing Problem by using Brute Force Algorithm

ABSTRACT

The Vehicle Routing Problem (VRP) is described followed by an outline of solution approaches. Different variations of the basic VRP are examined that involve the consideration of additional constraints or other changes in the structure of the appropriate model. An introduction is provided to Brute Force issues that are relevant to vehicle routing and scheduling including discussion of the environmental objectives that should be considered. Particular consideration is given to VRP models that relate to environmental issues including the time- dependent VRP, the transportation of hazardous materials and dynamic VRP models. Finally some conclusions are drawn about further research needs in this area and the relation to road distance and travel timing

Chapter 1

Introduction

1.1 Introduction

The Vehicle Routing Problem (VRP) concerns the determination of routes and schedules for a fleet of vehicles to satisfy the strain of a collection of shoppers. the fundamental Vehicle Routing Problem (VRP) are often described within the following way. We are given a group of homogeneous vehicles each of capacity, located at a central depot and a collection of shoppers with known locations and demands to be satisfied by deliveries from the central depot. Each vehicle route must start and end at the central depot and also the total customer demand satisfied by deliveries on each route must not exceed the vehicle capacity. the target is to see a group of routes for the vehicles that may minimize the overall cost. the entire cost is sometimes proportional to the full distance traveled if the amount of vehicles is fixed and will also include a further term proportional to the quantity of vehicles used if the amount of routes may vary.

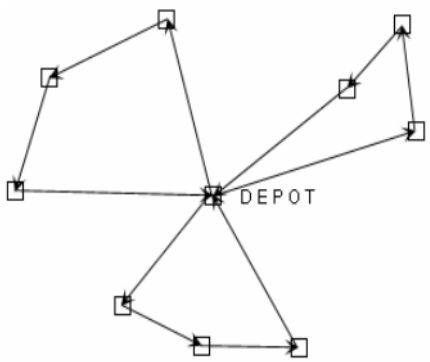


Figure 1.1 : The Vehicle Routing Problem Scheme

The VRP and plenty of of its variants are well studied within the literature since its introduction by Danzig and Ramser in 1959 [1]. Its exact solution is difficult to work out for large-scale problems because it could be a member of the category of NP-Hard problems, where no algorithms have yet been devised where the time to search out a guaranteed optimal solution that's proportional to the scale of the matter (in this case, the amount of customers). Specialized algorithms are able to consistently find optimal solutions for cases with up to about 50 customers; larger problems are solved to optimality in some cases, but often at the expense of considerable computing time.

In practice, other variations and extra constraints that has to be taken into consideration usually make the vehicle routing problem even harder to unravel to optimality. such a big amount of solution procedures are supported heuristic algorithms that are designed to supply good feasible solutions within a suitable computing time, but without a guarantee of optimality. There are several books and survey articles that are written that summaries different approaches and supply references to the massive number of journal articles that are written on this subject. There are many other research works about the classical VRP; we may cite some exact methods for this problem. Most papers assume that the prices and times of traveling between the depot and therefore the customers and between customers are known and glued. they're either given or calculated employing a shortest path algorithm on the graph or network representing the locations.

Distance or time constraints for each route

In some formulations, a constraint on the most distance which will be traveled by any vehicle before it completes its route is imposed. In other cases, the constraint concerns the utmost time taken for every route which can relate to a legal constraint on the working time of the motive force.

1.2 Time Window constraints

These constraints restrict the days at which a customer is out there to receive a delivery. These time window constraints are usually expressed in terms of a amount or intervals within which service must begin for every customer. If a vehicle arrives at a customer before the beginning of a time window, then it must normally wait until the beginning of the time window before service can commence. Other formulations may treat the time windows as soft constraints where early or late service could also be allowed but at the expense of a penalty within the objective function. Some exact and approximate methods are tailored especially to resolve the VRPTW. One example is an optimization algorithm developed for the VRPTW.

1.3 VRP with pickup and delivery

This situation concerns a collection of customer orders where one commodity is to be picked up from one location and delivered to a second location. The vehicle routes, starting and finishing at a central depot, must be constructed in order that the route visits the pick-up location for an order before the corresponding delivery location and therefore the load on each vehicle never exceeds the capacity, Q. We recall here some research addressed to the present style of VRP.

1.5 VRP with non-homogeneous vehicle fleet

In practice, many vehicle routing problems are addressed by a non-homogeneous vehicle fleet where vehicles of various capacities are also used. The vehicles may additionally have differences in other operating characteristics like the common speed of travel or the service time at a customer.

1.6 Period VRP

This problem is defined because the problem of finding routes for all days of a given total day period to attenuate total costs. These varieties of problems are called allocation/routing problems. The allocation part consists of the assignment of shoppers to days of the amount, while the routing part governs for the daily planning. Some articles coping with this variant of the VRSP are available within the literature.

1.7 Real Time VRP

As conventional static vehicle routing problems are NP - hard, it's not always possible to seek out optimal solutions to problems of realistic sizes during a reasonable amount of computation time. this suggests that the dynamic vehicle routing problem also belongs to the category of NP-hard problems, since a static VRP should be solved anytime a brand new immediate request is received. This is a very time window constrained routing problem will usually be rather more difficult than in a very non- time constrained problem.

1.8 Related Works

Title: The Vehicle Routing Problem: State of the Art Classification and Review

This paper classifies 277 VRP articles published between 2009 and mid-2015 in line with an adapted taxonomy supported Eksioglu et al. (2009). The resulting classification table (in electronic Appendix) enables future researchers to seek out relevant literature by eliminating or selecting characteristics within the taxonomy, leaving only articles tailored to their interests. Additionally, the classification allowed to research which characteristics and combinations of characteristics are preferred. Results represented that the vehicle routing and developing efficient solution methods to unravel these problems. Future research could therefore specialize in even 'richer' problems by simultaneously considering multiple real-life characteristics, and developing efficient solution methods to unravel these problems.

On the opposite hand, many researchers still propose highly problem-tailored solution methods which don't seem to be directly applicable to other problem variants. In our classification, only some articles propose more general algorithms which might solve multiple variants. The future improvement of those kinds of solution approaches seems very much worthwhile.

Limitations:

- In this paper only vehicle routing problem has described with real life characteristics. There has no discussion about shortest path.
- Besides, in this paper there has no solution about how to travel more distance in less times.
- Visiting all nodes is not main target of this paper.

Title: The vehicle routing problem for perishable goods: A systematic review

Vehicle Routing Problem (VRP) is that the problem for locating optimal routes within the distribution system, and this problem needs attention because it can improve distribution performance. one among the issues in VRP is that the Vehicle Routing Problem for Perishable Goods. This issue refers vehicle routing problem are rapidly losing quality and acceptance. This paper have discussed to review the Vehicle Routing Problem for Perishable Goods.

A systematic review has been administered on fifty-nine published papers from 2001 to March 2020. The optimization methods during this review were classified into single and multi-objective problems. additionally, the review was also conducted supported the target function. The results show the met heuristic algorithm to be a preferred optimization method for solving single and multi-objective problems. Further, the results also show that minimizing the entire cost distribution to be the target function that was often employed by researchers. At last, in future researches are suggested to develop a more comprehensive discover.

Limitations:

- This paper aims to provide only an in-depth analysis of single and multi-objective problems in the VRPfPG.
- Besides, in this paper there has no solution about how to travel more distance in less times.
- This paper cannot provide route visiting solutions.

Chapter 2
Classes of Vehicle Routing Problems

The objective of this section is to introduce the core problem studied during this thesis, the pickup and delivery problem with time windows. so as to try to to so four simpler variants of the matter are first introduced. this offers an introduction to a number of the core problems within the field of auto routing and surveys the relevant literature. The four preliminary problems studied are the voice problem, the m-traveling salesmen problem, the capacitated vehicle routing problem and therefore the vehicle routing problem with time windows. The section on the pickup and delivery problem with time windows will be found. Each section first introduces the matter in words then gives a mathematical definition of the matter. Finally literature pointers are given and up to date advances are discussed. The introduction of the matter and mathematical models are often understood with a basic knowledge of research, while the literature discussion will be technical every now and then and requires a deeper understanding of research and specifically of solution method paradigms.

2.1 The Travelling Salesman Problem

One of the best, but still NP-hard, routing problems is maybe the interpreter problem (TSP). within the TSP one is given a group of cities and how of measuring the space between each city. One should find the shortest tour that visits all cities exactly once and returns back to the starting node. In Figure 2.1 an example of a TSP instance is shown to the left and to the correct the optimal solution is shown when Euclidean distances are wont to measure the space between two cities. the matter comes in numerous flavors betting on what properties the distances satisfy. If the spaces satisfy that the distance from city i to city j is that the same because the distance from city j to city i for all cities i and j, the matter is claimed to be symmetric. If this property doesn't hold then the matter is alleged to be asymmetric. an issue is claimed to be Euclidean if the cities are located in R d and also the distance between two cities is that the Euclidean distance. the matter are often formulated as a mathematical model within the following way. Let G = (V, A) be an entire, directed graph where V = is that the set of nodes/cities and A is that the set of arcs. to every arc (i, j) ∈ A could be a assigned a distance or cost cij . We define binary decision variable xij that's set to 1 if and provided that arc (i, j) is employed within the solution. the matter will be formulated..

2.2 m-Travelling Salesman Problem

The m-traveling salesman problem (m-TSP) may be a generalization of the TSP that introduces quite one salesman. within the m-TSP we are given n cities, m salesmen and one depot or home base. All cities should be visited exactly once on one among m tours, starting and ending at the

depot. The tours aren't allowed to be empty. If distances satisfy Triangulum inequality, that's if d(i, k) ≤ d(i, j) + d(j, k) for all i, j and k then it's easy to work out that the gap of the shortest TSP tour on the n cities plus the depot always is a smaller amount than or capable the gap of the shortest m-TSP solution for any m. Any m-TSP with n cities may be formulated as a TSP with m+ n cities. The distances between depot nodes is then set to a sufficiently sizable amount while the distances between the depot nodes and ordinary nodes are copied from the mTSP. the massive distance between depot nodes ensures that no salesmen tours are empty. Notice that the resulting TSP doesn't obey constellation inequality In Figure 2.2 an example of an answer to an m − T SP with m = 3 and n = 38 is shown. the particular solution is shown within the top right a part of the figure. Observe that one salesman serves one city, the following salesman serves 4 cities and therefore the last salesman serves the remainder of the cities.

2.3 Capacitated Vehicle Routing Problem

In the capacitated vehicle routing problem (CVRP) a vocabulary different from the one employed in the TSP community is employed. The objects called cities within the TSP world are called customers within the CVRP world and also the salesmen are called vehicles. The common place to begin remains denoted the depot. within the CVRP we are given a depot, a collection of n customers, a collection of m vehicles and a distance measure as within the m-TSP, but within the CVRP every vehicle includes a capacity Q and each customer i ∈ encompasses a demand qi. The task within the CVRP is to construct vehicle routes specified all customers are served exactly once and such the capacities of the vehicles are obeyed. this could be done while minimizing the entire distance traveled. We now introduce a mathematical model for the matter. We use a group partitioning approach (or path based modeling) as this makes it easier to model the more complicated problems that are described below. A model like the one presented for the TSP (Section 2.1) is certainly possible; such a model will be found in Toth and Vigo [2002b]. Let G = (V, A) be a directed graph as before, let V = be the set of nodes within the graph where node 0 and n + 1 corresponds to the depot and node corresponds to customers. The depot has been split into two nodes to form modeling easier, node 0 corresponds to the beginning of the routes and n + 1 corresponds to the tip of the routes. From now on we'll call the distances for costs. A legal route r¯ must be an easy (that is, no node is visited twice) path from node 0 to node n + 1. we will write such a path r¯ = (v0, v1. . . , vh, vh+1) (2.6) where vi , i ∈ are the nodes visited on the route. We always have that the number of consumers visited on the route. The route should satisfy the capacity requirement.

2.4 The Vehicle Routing Problem with time windows

The vehicle routing problem with time windows (VRPTW) generalizes the CVRP by associating travel times tij with arcs (i, j) and repair times si and time windows [ai, bi] with customers i and depot i = 0. The vehicle should arrive before arriving customers. If it arrives before the beginning

of the time window, it's to attend until the time window opens before service at the customer can start. the matter are often modeled using the framework introduced. To make the process easy, we have to consider the depot as split into different nodes. The route $\bar{r} = (v0, v1 \ldots, vh, vh+1)$ should satisfy the subsequent criteria so as to be valid..

2.5 Pickup and delivery problem with time windows

The pickup and delivery problem with time windows generalizes the vehicle routing problem time window. within the PDPTW one now not delivers goods from a depot to the purchasers, instead the shoppers need goods to be transported from a pickup location to a delivery location. Each pickup-delivery pair is named missive of invitation. the matter is defined on a graph with 2n + 2 nodes, where n is that the number of requests. Each request i is related to node i and n + i, where i is that the pickup and n + i is that the delivery of the request. Node 0 and 2n + 1 represents the terminals where vehicles start point (0) and end point (2n + 1) in their path. A time window [ai , bi] is related to every node within the graph and a load di is related to every node $1 \leq i \leq 2n$. it's assumed that $d_{n+i} = -d_i$ for all $i = 1, \ldots, n$. even as for the VRPTW, travel times tij and costs cij are related to arcs (i, j) and repair times si are related to node i. it's assumed that every one vehicles are identical and have capacity Q.

The task within the PDPTW is to construct routes for the vehicles specified the pickup and delivery appreciate the identical request is served by the identical vehicle, that the pickup is served before the corresponding delivery and such time window and therefore the capacity constraints are obeyed. even as for the VRPTW it's common to either minimize route costs (cij) or minimize the quantity of vehicles necessary to serve all requests.

2.6 Brute Force for Pickup and delivery problem with time windows

Several metaheuristics are proposed for the PDPTW in recent years and a group of problem instances has appeared as a standard platform for Brute Force. The instances were constructed from the Solomon test set for the VRPTW and Gehring and Hombergers larger VRPTW instances. The instances were created by first solving the VRPTW instances with a VRPTW heuristic then pairing nodes that occur within the same route within the VRPTW solution to create an invitation. The requests are created specified the pickup is visited before the delivery on the VRPTW route. this manner of making PDPTW instances won't end in very realistic instances.

One can argue that the requests that are constructed are too “easy” because the pickup and delivery fit well together as they were served by the identical route within the VRPTW solution. It seems that the PDPTW instances created this manner are challenging for both heuristics and exact methods, especially the larger instances with 100 requests or more. They used a stimulating neighborhood supported the ejection chain idea: letter of invitation is off from its route r1 and

reinserted into another route r2 while ejecting another request j from r2. j is inserted into a 3rd route, thereby.

Rejecting a 3rd request and then on. The chain of ejections ends with the insertion of the last request into a route without ejecting a brand new request. Two crossover methods are proposed, both can produce infeasible offspring, where some requests don't seem to be visited or some requests are visited twice.

The algorithm also incorporates mutation operators supported local search. The heuristic was tested on the 50 request instances from Li and Lim [2001], but the answer quality obtained was worse than that obtained by Li and Lim's tabu search heuristic. Another genetic algorithm was proposed for the PDPTW where the genetic encoding stores the partitioning of requests on vehicles, but not the particular routing of the requests. The heuristic was tested on instances with around 50 requests from Li and Lim and Nanry and Barnes. The computational results seem to be better than those obtained by the opposite genetic algorithm. Bent and Hentenryck applied a two-stage heuristic to the PDPTW and obtained good results on the instances proposed by Li and Lim. the primary stage minimizes the amount of vehicles wont to serve the requests, this can be done employing a simulated annealing algorithm whose neighborhood consists of moving letter of invitation from one position within the solution to a different. A modified objective function is employed while minimizing the amount of vehicles. The modified objective function encourages solutions that contain routes with some requests and routes with many requests. The second stage minimizes the traveled distance using large neighborhood search (BRUTE FORCE). The Brute Force alternates between removing requests from the present solution and reinserting the requests again. Removal of requests is administered by a heuristic that removes related requests as proposed by Shaw for the VRPTW. Re-insertion of the requests is performed employing a truncated branch and-bound search that only allows a specific amount of branching.

Chapter 3

Evaluation of Brute Force Algorithm

Brute Force can be categorized broadly into three different categories: construction Brute Force, improvement Brute Force and metaheuristics. These three categories are explained here. The propose two main classes classical Brute Force and metaheuristics. The class of classical heuristics is divided into three groups: constructive Brute Force two-phase Brute Force and improvement methods. The term two-phase Brute Force covers heuristics that divide the construction into two phases: a clustering phase and a routing phase. In the classification of Brute Force used in this thesis, two-phase brute forces are seen as construction heuristics.

3.1 Construction of Brute Force Algorithm

Constructive Brute Force gradually builds a feasible solution while keeping a watch on solution cost, but they are doing not contain an improvement phase intrinsically. Many construction brute force for vehicle routing problems are proposed during the last 40 years. within the recent years it appears that their popularity has faded somewhat within the scientific literature as meta brute force became more dominant, however papers about construction brute force still appear. Many construction brute force for vehicle routing problems are proposed during the last 40 years. within the recent years it appears that their popularity has faded somewhat within the scientific literature as meta brute force became more dominant, however papers about construction brute force still appear. Fast brute force may be used as subroutines in additional time consuming meta brute force, this approach is employed during this thesis. Many construction brute force for vehicle routing problems constitute one in all the three classes: insertion brute force, savings brute force and clustering brute force. Insertion brute force builds an answer by inserting one customer at a time. Insertion brute force can build one route at a time (sequential insertion brute force) or build many or all routes in parallel (parallel insertion brute force). the selection of which customer to insert and where to insert the customer is what differentiates the insertion brute force.

very simple insertion heuristic could favor to insert the customer that increases the cost the smallest amount. Savings brute forces initially build an answer where each customer is served on its own route. Routes are then merged one by one in line with some criteria. There are several algorithms are varied by the criterion used for merging routes and by how routes are merged. For the CVRP the foremost simple merge operation deletes a footing between the depot and a customer from each of the 2 routes that's being merged and joins the route by adding a footing between the 2 customers that are adjacent to just one edge. More advanced merging procedures consider all the purchasers served by the 2 routes and solve a TSP (in case of the CVRP) on these customers. Another clustering approach is that the sweep algorithm for the CVRP which was presented by Gillet and Miller [1974]. during this algorithm customers are clustered in sectors of the revolve around the depot as shown on Figure 3.1. In practice the algorithm works

by sorting customers in keeping with their co-ordinate angle with the depot as (0,0). The algorithm starts from the primary customer within the list and adds this customer to a cluster. The algorithm continues to process the purchasers per the ordering and adds the customer to this cluster as long because the cluster is served by one vehicle. When it's now not possible to feature a customer to the present cluster a brand new cluster is started and becomes the present cluster. When all customers are assigned to a cluster a TSP tour is found for every cluster to provide a CVRP solution.

3.2 Local search Brute Force

Local search heuristics are heuristics that take an answer as input, modify this solution by performing a sequence of operations on the answer and produce a brand new, hopefully improved solution. in any respect times the heuristic incorporates a current solution and it modifies this solution by evaluating the effect of fixing the answer in systematic way. If one amongst the changes results in an improved solution, then the present solution is replaced by the new improved solution and also the process is repeated. in additional advanced local search heuristics the algorithm sometimes perform changes that result in an answer that's worse than the present. this can be done collectively can hope to search out an excellent better solution after some more changes. We define a part of an answer $s \in S$ as $N(s) \subseteq S$. That is, N may be a function that maps from an answer to a collection of solutions. an answer s is claimed to be locally optimal or an area optimum with regard to a region N if $c(s) \leq c(s\ 0)\ \forall s\ 0 \in N(s)$. With these definitions it's possible to define a steepest descent algorithm (see Algorithm 1). This algorithm gets an initial result as input. It repeats line 3–7 as long because it found an improved solution within the last iteration.

The neighborhood of s is searched in line 3 and s 0 is that the best solution within the neighborhood. In line 4 it's determined if the new solution is best than the previous. If it is, then we update this solution in line 5 and reiterate. If this solution wasn't improved then the algorithm terminates with the most effective solution observed during the search. The algorithm is named a steepest descent algorithm because it always chooses the simplest solution within the neighborhood. Another strategy is to settle on the primary improving solution observed within the neighborhood. This kinds of algorithm could be a ancestry algorithm. Funke et al. [2005] use the terms best search and first seek for a steepest descent algorithm and a descent algorithm, respectively.

Algorithm 1

Smallest Distance

1 input: Initial solution s ∈ SI 3 improved = true

2 while (improved)
3 3 s 0 = arg minx∈N(s) {c(x)}
4 4 if c(s 0) < c(s)
5 5 s = s 0
6 6 else
7 7 improved = false
8 8 return s

In order candidate for P after the current one c.

valid (P, c): check whether candidate c is a solution for P.

output (P, c): use the solution c of P as appropriate to the application.

The next procedure must also tell when there are not any more candidates for the instance P, after the present C. Convenient to do this is to return a "null candidate", some conventional data value Λ that's distinct from any real candidate. Likewise the primary procedure should return Λ if there aren't any candidates in the slightest degree for the instance P. The brute-force method is then expressed by the algorithm.

c ← first(P)

while c ≠ Λ do

if valid(P,c) then

output(P, c)

c ← next(P, c)

end while

For example, when searching for the divisors of an integer n, the instance data P is that the number n. the decision first(n) should return the integer 1 if n ≥ 1, or Λ otherwise; the decision next(n,c) should return c + 1 if c < n, and Λ otherwise; and valid(n,c) should return true if and on condition that c may be a divisor of n. (In fact, if we elect Λ to be n + 1, the tests n ≥ 1 and c < n are unnecessary.)The brute-force search algorithm above will call output for each candidate that's an answer to the given instance P. The algorithm is definitely modified to prevent after finding the primary solution, or a specified number of solutions; or after testing a specified number of candidates, or after spending a given amount of CPU time.

it easier to debate the dimensions of a vicinity, we define the dimensions of a locality as Nˆ (s) or |M|. Using N(s) to live the scale of a region is problematic because the size of this set would depend upon s, but a working definition may be max N(s).

3.3 Neighborhoods

In this study we discuss about some neighborhoods policy for vehicle routing problems. it's aloof from a whole description of all the neighborhoods conceived. Giving such an outline is out of the scope of this section. What we wish to convey with this section is a concept of the various types of neighborhoods that are proposed and attempted in practice. For a more complete survey of neighborhoods we refer the reader to Bräysy and Gendreau which discusses VRPTW neighborhoods and Funke et al. for a more general presentation.

VRP neighborhoods are often split into two major categories: Single-Route Improvements and Multi route Improvements, following the terminology from Laporte and Semet, or SingleRoute neighborhoods and Multi route neighborhoods as we opt to call them. Single Route neighborhoods perform changes to at least one route at a time, that is, they permute the purchasers within a route. Thus TSP neighborhoods are often used as Single-Route neighborhoods for the CVRP; TSPTW neighborhoods will be used for the VRPTW and 1-PDPTW neighborhoods may be used for the PDPTW. Multi route neighborhoods exchange and move customers between two or more routes. this suggests that they'll make greater structural changes to an answer. within the following sections we'll only consider multi route neighborhoods.

3.4 Trends in heuristic research for the VRP

This section outlines a number of the trends within the research in brute force methods for static vehicle routing problems and it contains some comments on the direction I foresee and/or hope the research will move within the coming years. The section is kind of subjective in some paragraphs and other researchers within the VRP community may have different opinions or see different opportunities than I do.

The section first lists some possible research directions then comments on the impact we believe these directions will have within the future.

• More complex and rich vehicle routing problems.

• Faster brute forces (disregarding increasing computer speeds) that also produces top quality Solutions.

• Ability to handle larger instances.

• More precise brute forces - better solution quality without concern overly about the time needed for the computation.

• Simpler brute forces.

• Brute forces using mathematical programming - combining ideas from exact optimization with Brute forces.

• Parallel implementations.

• More realistic test.

Chapter 4

Methodology

Methodology is basically the systematic and theoretical analysis or study in a particular activity or area of study. The term 'methodology' is widely used across in industries and scientific discipline, the techniques are used to research in a particular area of study or to accomplish particular project. In simpler way the term 'methodology' can be defined as the system of doing, teaching and studying something. Including multiple methods can also be considered as methodology, , each methods as applied to various facets of the whole scope of the methodology. Research are basically divided into two parts such as qualitative research and quantitative research. An organization can tackle the risk in an integrated manner, comprehensively and systematically with the help of Project Management Methodology. This methodology is also beneficial at various levels such as strategic, tactical and operational levels.

4.1 Process Model

In general definition, Process Model are the collection of processes of same nature which are classified together into a model. Basically process is a condition which operates and changes the state of an object. By using process model activities of a software/app can be shown graphically. For getting a good and beneficial product the process has to be good too. So, we need to process the model because processing in more important than the product.

4.2 Recent trends in Process Model

1. Waterfall Model.
2. Incremental Model
3. Spiral Model
4. Iterative Development Model
5. Agile Model
6. Prototyping:
    a) Throw away prototyping process model.
    b) Evolutionary prototyping process model.

4.3 A particular Process model for "Solving Vehicle Routing Problem Using Brute Force "

While developing the "Vehicle Routing Problem Using Brute Force", we have followed Water Fall model.

4.3.1 Waterfall Model

Waterfall Model could be a consecutive or sequential way where each activity arranged in linear order and every process is represented in separate phase. In Waterfall Model, it's must to plan and schedule all the activities before beginning to work on them. Waterfall Model is additionally called Linear Sequential Life Cycle. As Waterfall Model follow the sequential order so we are able to forward to next step of development or testing if the previous step is successfully completed. If all the wants are clear then Waterfall Model is extremely successful. In Waterfall Model, the testing starts when the event is completed. Waterfall Model is more popular than other models for Software Development Life Cycle. As water falls towards down, similarly in waterfall model the testing and development is dole out in downward mechanism. In Waterfall Model if one step is completed then only the event process of next step are often started. It cannot revert back to previous stage to perform any changes. So for working with waterfall model we must have very clear requirements. Particularly Waterfall Model that's shown in bellow:

Phases in Waterfall Model-

- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Maintenance

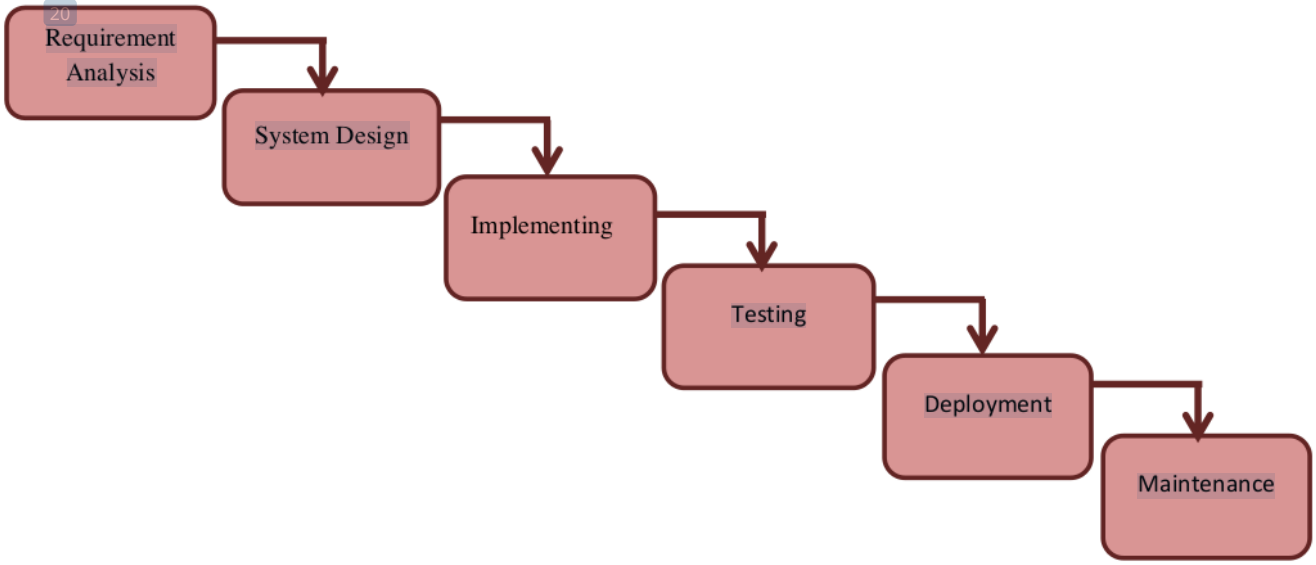


Figure 4.1 : Waterfall Model Process

Requirements: This is the primary phase of development where all the wants gathered and documented. It involves what have to be designed, what's its function and purpose etc. For vehicle routing problem we've got first chosen a subject after observing from our surrounding. Then we've analyzed the prevailing applications associated with our topic. Next after observing all the features and limitations we finalized the features for vehicle routing problem..

Analysis: In this phase, we gather all the requirements from previous step and check if the requirements are valid or not. For vehicle routing problem after gathering all the requirements and finalizing the features we have analyzed if the requirements are valid or not.

Design: In this phase, the system design is prepared and the hardware, system configuration and architecture of system is specified. For vehicle routing problem we have designed the user interface of the application using XML and cloud based server Firebase.

Implementing: In this phase, the software codes are written. All the development works are performed and development components or units handed over to testing team. For vehicle routing problem the implementation is done here. For implementing android application, we have used JAVA and cloud based server Firebase.

Testing: Once the task has finished, testing the system begins and in this way, we test each step and make sure the developed components are working as expected. All the result of testing are performed in this step. In case of vehicle routing problem we have done black box testing and white box testing. We have done various validity check from each panel.

Deployment: Once testing is completed and ensure there's no bug or defect or any quite issue, then project is deployed to production. Once product is deployed to production the top users start using the merchandise. just in case of auto routing problem we've checked the performance of the app installing it in various android versions and tried to repair the obtained bugs.

Maintenance: We always keep eye on the merchandise and supply all the required bug or issue fixes if occurs in production or reported by end users. Also, time to time we keep updated the merchandise with new updates or patches if developed or available. For vehicle routing problem we'll attempt to observe the app continuously and keep the app updated..

Why Waterfall Model:

We Prefer **Waterfall Model,** because of its following advantages: -

- Very good approach for little projects.
- Easy to use and follow.
- Cost effective.
- Each phase completely developed.
- Development processed in sequential manner so very less chance to remodel.
- Easy to manage the project.
- Easy documentation.

Limitation of Waterfall Model:

- Not very useful for the massive project.
- Less effective if requirements aren't very clear at the start.
- Very difficult to maneuver back to vary on the previous phase.
- Testing starts once development completes, so more and more chances of bugs to be found.
- High risk.

When Waterfall Model should be followed:

- If project is tiny and requirements are very clear.
- For low budget projects.
- When changes within the project are stable.
- Can set deadlines for every stage of development and a product can proceed through the event process model phases one by one.
- Easily understandable and explainable phases and thus it's easy to use.
- Easy to manage because of rigidity of model.
- Phases are processed and completed once at a time thus they are doing not overlap.
- Process and output of every phase are clearly mentioned in documentation.

Chapter 5

Neighborhood Search Brute Force for Pickup and Delivery

The pickup and delivery problem with time windows is that the problem of serving variety of transportation requests employing a limited amount of vehicles. Each request involves moving variety of products from a pickup location to a delivery location. Our task is to construct routes that visit all locations specified corresponding pickups and deliveries are placed on the identical route and specified a pickup is performed before the corresponding delivery. The specified routes must maintain time window and capacity constraints. This paper presents a Brute Force for the matter supported an extension of the big Neighborhood Search Brute Force previously suggested for solving the vehicle routing problem with time windows. The proposed Brute Force consists of variety of competing sub-Brute Forces which are used with a frequency equivalent to their historic performance. This kind of general framework is known as Adaptive Large Neighborhood Search.

5.1 Mathematical Model

This section presents a mathematical model of the matter, it's supported the model proposed by Desaulnier set al. The mathematical model is a proper description of the matter. As we solve the matter heuristically we don't try and write the model on integer-linear form.

With some modifications, the initial roadman problem can support multiple salesmen. rather than making each facility only be visited once, the origin facility are going to be visited multiple times. If we've two salesman then the origin is visited exactly twice so on.

For **K** vehicles or sales people:

Variables:

$$x_{ij} = \begin{cases} 1 : \text{the path goes from city i to j} \\ 0 : \text{otherwise} \end{cases}$$

Order dummy variables:

$$u_i - u_j + C * x_{ij} \leq C - d_j, C = N/K \begin{cases} u_i : \text{order that site i is visited} \\ d_j : \text{the cost to visit city j}, 0 \leq u_i \leq C - d_j, \forall i \in V \setminus \{0\} \end{cases}$$

Goal:

$$min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \begin{cases} c_{ij} : \text{distance from city i to city j} \\ x_{ij} : \text{whether there's a path between i and j} \end{cases}$$

Constraints:

$$\sum_{i \in V} x_{ij} = 1, \forall j \in V \setminus \{0\} \sum_{j \in V} x_{ij} = 1, \forall i \in V \setminus \{0\} \sum_{i \in V} x_{i0} = K \sum_{j \in V} x_{0j} = K$$

The purpose of this function minimizes the sum of the distance traveled, the total of the time passes by every vehicle, and the number of requests not maintain as schedule.

5.2 Solution Method

Local search heuristics are often built on neighborhood moves that make small changes to this solution, like moving a call for participation from one route to a different or exchanging two requests as in Berlin and Budapest. These quite local search heuristics are ready to investigate a large number of solutions in a very short time, but an answer is barely changed little or no in each iteration. it's our belief that such heuristics can have difficulties in moving from one promising area of the answer space to a different, when faced with tightly constrained problems, even when embedded in Brute Force.

```
Berlin -> Budapest -> Bucharest -> Istanbul -> Sofia -> Belgrade -> Vienna -> Berlin
Berlin -> Hamburg -> Dublin -> London -> Paris -> Brussels -> Berlin
Berlin -> Prague -> Rome -> Barcelona -> Madrid -> Milan -> Munich -> Berlin
Berlin -> Warsaw -> Kiev -> Moscow -> Saint Petersburg -> Stockholm -> Copenhagen -> Berlin
```

5.3 Speeding up Brute Force Searches

One way to hurry up a brute-force algorithm is to scale back the search space, that is, the set of candidate solutions, by using heuristics specific to the matter class. as an example, within the eight nodes problem the challenge is to put eight queens on a regular chessboard in order that no queen attacks the other. Since each queen may be placed in any of the 64 squares, in essence there are 64^8 = 281,474,976,710,656 possibilities to contemplate. However, because the queens are all alike, which no two queens will be placed on the identical square, the candidates are all possible ways of selecting of a collection of 8 squares from the set all 64 squares; Further, no arrangement with two queens on the identical row or the identical column may be an answer. Therefore, we are able to further restrict the set of candidates to those arrangements.

As this instance shows, a bit little bit of analysis will often cause dramatic reductions within the number of candidate solutions, and should turn an intractable problem into a trivial one.

In some cases, the analysis may reduce the candidates to the set of all valid solutions; that's, it should yield an algorithm that directly enumerates all the specified solutions (or finds one solution, as appropriate), without trifling with tests and therefore the generation of invalid candidates. for instance, for the matter find all integers between 1 and 1,000,000 that are evenly divisible by 417" a naive brute-force solution would generate all integers within the range, testing each of them for divisibility. However, that problem will be solved far more efficiently by starting with 417 and repeatedly adding 417 until the amount exceeds 1,000,000 – which takes only 2398 (= 1,000,000 ÷ 417) steps, and no tests.

5.4 Minimizing the number of vehicles used

Minimization of the quantity of vehicles accustomed serve all requests is usually considered as first priority within the vehicle routing literature. The heuristic proposed to this point isn't able to deal with such an objective, but by employing a simple two stage algorithm that minimizes the quantity of vehicles within the first stage then minimizes

A secondary objective (typically traveled distance) within the second stage, we will handle such problems. We also assume that the quantity of vehicles available is unlimited, such constructing an initial feasible solution always are often done.

A two-stage method was also utilized by Bent and Van Hentenryck, but while they used two different neighborhoods and metaheuristics for the 2 stages, we use the identical heuristic in both stages. The vehicle minimization stage works as follows: first an initial feasible solution is made employing a sequential insertion method that constructs one route at a time until all requests are planned.

The number of vehicles employed in this solution is that the initial estimate on the amount of vehicles necessary. Next step is to get rid of one route from our feasible solution. The requests on the removed route are placed within the request bank. The resulting problem is solved by our Brute Force. If the heuristic is in a position to seek out an answer that serves all requests, a replacement candidate for the minimum number of vehicles has been found.

When such an answer has been found, the Brute Force is instantly stopped, yet another route is far from the answer and therefore the process is reiterated. If the Brute Force terminates without finding an answer where all requests are served, then the algorithm steps back to the last solution encountered during which all requests were served.

This solution is employed as a starting solution within the second stage of the algorithm, which simply consists of applying the traditional Brute Force. so as to stay the period of the vehicle

minimization stage down, this stage is simply allowed to spend Φ brute force iterations all at once specified if the primary application of the Brute Force for instance spends a iterations to seek out an answer where all requests are planned, then the vehicle minimization stage is barely allowed to perform Φ−a brute force iterations to attenuate the quantity of vehicles further.

in a different way to stay the time period limited is to prevent the Brute Force when it seems unlikely that an answer exists within which all requests are planned. In practice this can be implemented by stopping the Brute Force if 5 or more requests are unplanned and no improvement within the number of unplanned requests has been found within the last τ brute force iterations. within the computational experiments Φ was set to 25000 and τ was set to 2000..

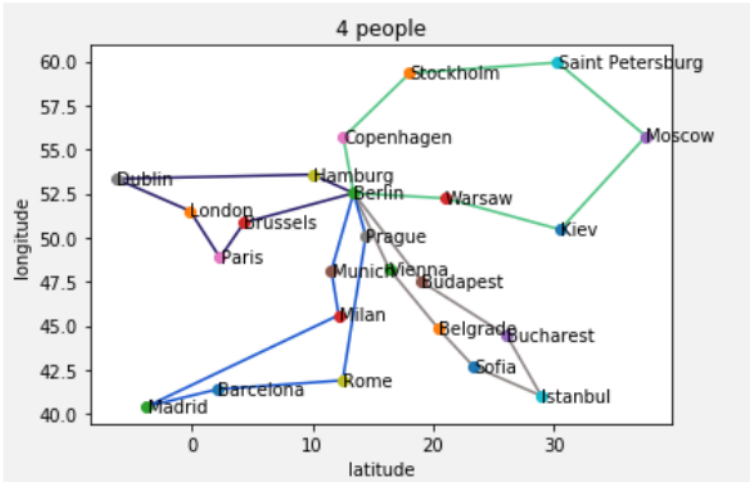


Figure: Path of All Nodes

5.5 Discussion

Several using of removal and insertion heuristics in that time the search may be seen as using local search with several neighborhoods. To the best of our knowledge this idea has not been used in the Brute Force literature before. Using several removal Brute Force insertion heuristics during the search may be seen as using local search with several neighborhoods. To the best of our knowledge this idea has not been used in the Brute Force literature before..

5.6 Datasets

This model is considered in this paper is complicated, it is hard to find any benchmark instances that is considered exactly the same model. In order to test all aspects of the model proposed in this paper, we also introduce some new, randomly generated instances.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	City	Barcelon	Belgrade	Berlin	Brussels	Buchare:	Budapes	Copenha	Dublin	Hamburg	Istanbul	Kiev	London	Madrid	Milan	Moscow	Munich	Paris	Prague	Rome	Saint Pet	Sofia	Stockhol	Vienna	Warsaw
2	Barcelon	0	1528.1	1497.6	1062.9	1968.4	1498.8	1757.5	1469.3	1471.8	2230.4	2391.1	1137.7	504.64	725.12	3006.9	1054.6	831.59	1353.9	856.69	2813	1745.6	2276.5	1347.4	1862.3
3	Belgrade	1528.1	0	999.25	1372.6	447.34	316.41	1327.2	2145.4	1229.9	809.48	976.02	1689	2026.9	885.32	1711	773.33	1445.7	738.1	721.55	1797.8	329.46	1621	489.28	826.66
4	Berlin	1497.6	999.25	0	651.62	1293.4	689.06	354.03	1315.2	254.51	1735	1204	929.97	1867.7	840.72	1608	501.97	876.96	280.34	1181.7	1319.6	1318.7	810.38	523.61	516.06
5	Brussels	1062.9	1372.6	651.62	0	1769.7	1131.5	766.67	773.2	489.76	2178.9	1836.2	318.72	1314.3	696.61	2253.3	601.87	261.29	721.08	1171.3	1903.7	1697.8	1280.9	914.81	1159.9
6	Buchare:	1968.4	447.34	1293.4	1769.7	0	639.77	1571.5	2534.7	1544.2	445.62	744.44	2088.4	2469.7	1331.5	1497.6	1186.4	1870	1076.8	1137.4	1740.4	296.68	1742.3	855.32	946.12
7	Budapes	1498.8	316.41	689.06	1131.5	639.77	0	1011.3	1895	927.92	1064.8	894.29	1450.1	1975.4	788.56	1565.2	563.93	1247.6	443.26	811.11	1556.5	629.63	1316.6	216.98	545.29
8	Copenha	1757.5	1327.2	354.03	766.67	1571.5	1011.3	0	1238.4	287.97	2017.2	1326.3	955.13	2071.8	1157.9	1558.5	838	1025.9	633.05	1529.7	1143.4	1635.5	521.68	868.87	667.8
9	Dublin	1469.3	2145.4	1315.2	773.2	2534.7	1895	1238.4	0	1073.4	2950.1	2513.7	462.6	1450	1413.4	2792.4	1374.9	776.83	1465.6	1882.2	2314.2	2471	1626.6	1680	1823.7
10	Hamburg	1471.8	1229.9	254.51	489.76	1544.2	927.92	287.97	1073.4	0	1983.8	1440.3	720.12	1785.3	900.01	1779.9	610.17	744.63	492.25	1307.5	1414.2	1554.8	809.65	742.79	750.49
11	Istanbul	2230.4	809.48	1735	2178.9	445.62	1064.8	2017.2	2950.1	1983.8	0	1053	2496.4	2734.6	1669.4	1754	1582.2	2254	1507.6	1373.8	2099.3	502.61	2171.7	1273.9	1386.1
12	Kiev	2391.1	976.02	1204	1836.2	744.44	894.29	1326.3	2513.7	1440.3	1053	0	2131.2	2859.3	1672.7	756.61	1391.4	2022.8	1138.6	1673.7	1051.4	1020.8	1265.8	1052.8	690.12
13	London	1137.7	1689	929.97	318.72	2088.4	1450.1	955.13	462.6	720.12	2496.4	2131.2	0	1263.4	957.91	2498.3	916.23	340.55	1034.6	1431.2	2093.7	2012.7	1431.1	1233.5	1445.9
14	Madrid	504.64	2026.9	1867.7	1314.3	2469.7	1975.4	2071.8	1450	1785.3	2734.6	2859.3	1263.4	0	1187.7	3437.7	1484.5	1053.4	1773.7	1360.8	3183.4	2250.1	2591.5	1807.1	2288.4

Figure: Distance between every two places

For mapping, there are place latitude and longitude for every place. Here is the dataset of these places.

	A	B	C
1	City	latitude	longitude
2	Barcelona	41.3888	2.159
3	Belgrade	44.804	20.4651
4	Berlin	52.5244	13.4105
5	Brussels	50.8505	4.3488
6	Bucharest	44.4323	26.1063
7	Budapest	47.498	19.0399
8	Copenhag	55.6759	12.5655
9	Dublin	53.3331	-6.2489
10	Hamburg	53.5753	10.0153
11	Istanbul	41.0138	28.9497
12	Kiev	50.4547	30.5238
13	London	51.5085	-0.1257
14	Madrid	40.4165	-3.7026
15	Milan	45.5928	12.1728
16	Moscow	55.7522	37.6156
17	Munich	48.1374	11.5755
18	Paris	48.8534	2.3488

Figure: Latitude and Longitude Dataset

Depending on distance, here is the flight timing for visiting one place to another.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	City	Barcelona	Belgrade	Berlin	Brussels	Bucharest	Budapest	Copenhag	Dublin	Hamburg	Istanbul	Kiev	London	Madrid	Milan	Moscow
2	Barcelona	0	379	373	336	416	383	397	372	369	444	455	343	178	318	533
3	Belgrade	379	0	326	361	274	270	355	433	345	312	323	392	425	290	413
4	Berlin	373	326	0	291	349	305	261	352	145	395	341	318	408	299	398
5	Brussels	336	361	291	0	394	347	301	305	273	436	400	265	359	305	459
6	Bucharest	416	274	349	394	0	295	373	465	369	273	295	425	463	327	388
7	Budapest	383	270	305	347	295	0	333	418	324	339	319	378	428	289	404
8	Copenhag	397	355	261	301	373	333	0	345	252	419	350	321	426	329	392
9	Dublin	372	433	352	305	465	418	345	0	327	508	463	278	371	375	508
10	Hamburg	369	345	145	273	369	324	252	327	0	415	359	296	398	308	410
11	Istanbul	444	312	395	436	273	339	419	508	415	0	330	467	491	364	418
12	Kiev	455	323	341	400	295	319	350	463	359	330	0	429	498	360	320
13	London	343	392	318	265	425	378	321	278	296	467	429	0	356	333	483
14	Madrid	178	425	408	359	463	428	426	371	398	491	498	356	0	362	573
15	Milan	318	290	299	305	327	289	329	375	308	364	360	333	362	0	441
16	Moscow	533	413	398	459	388	404	392	508	410	418	320	483	573	441	0
17	Munich	355	325	190	308	358	312	325	380	198	400	376	338	395	276	449

Figure: Flight Timing

In the vehicle routing problem with backhauls (VRPB) we wish to attenuate the whole traveled distance and that we are allowed to serve linehaul and backhaul customers on the identical routes subject to the subsequent limitations.

(A) At any case, a route contains both linehaul and backhaul customers then delivery boy will serve the backhaul customers after the linehaul customers.

(B) A route isn't allowed to consist entirely of backhaul customers.

(C) The capacity of the vehicle should be maintained, that is, neither the total of the strain of the linehaul customers nor the total of the stress of the backhaul customers get services by a vehicle may be exceed the vehicle capacity.

(D) the quantity of vehicles to use is given ahead. this suggests that whether or not it's possible to seek out better solutions using fewer or more vehicles, we must report the most effective solution we will find that uses the required number of vehicles.

(E) All customers are serviced from one depot.

(F) All vehicles have the identical capacity..

Chapter 6

System Implementation

In order to remodel a delivery zone instance to a acquire instance we map every customer. Such an invitation consists of a acquire at the depot and a delivery at the customer site. the number of products that ought to be carried by the requests is adequate to the demand of the corresponding customer.

6.1 Greedy Algorithm

Here is the comparison of multiple algorithms for vehicle routing problem. First of all, we used greedy algorithm.

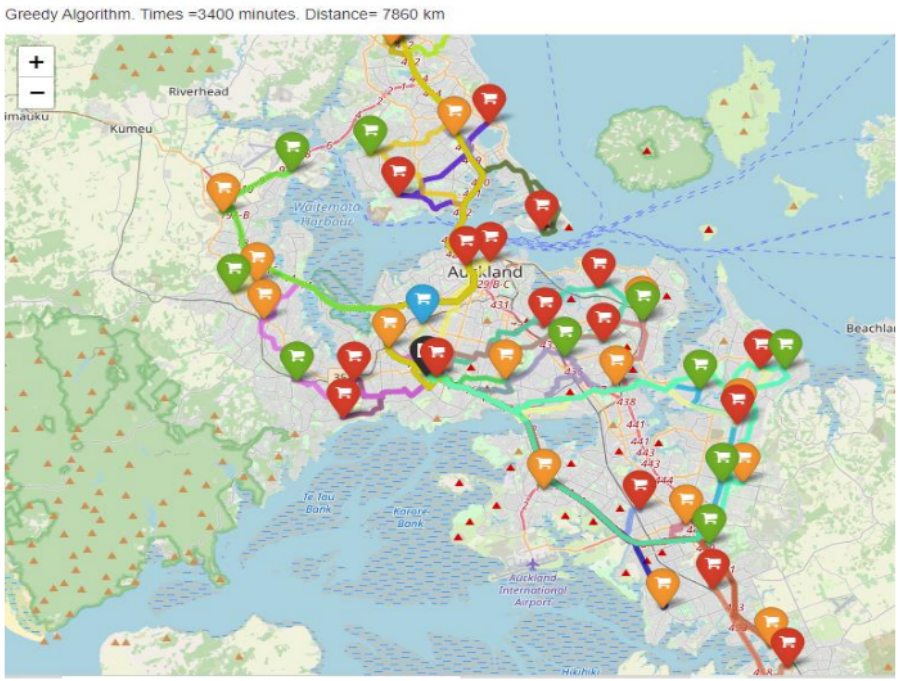


Figure: Using Greedy Algorithm

Here, is needed time is 3400 minutes. In this time, visited distance will be 7860 km.

6.2 Salesman Algorithm

Another salesman algorithm used in this site. This is using for travelling all nodes.

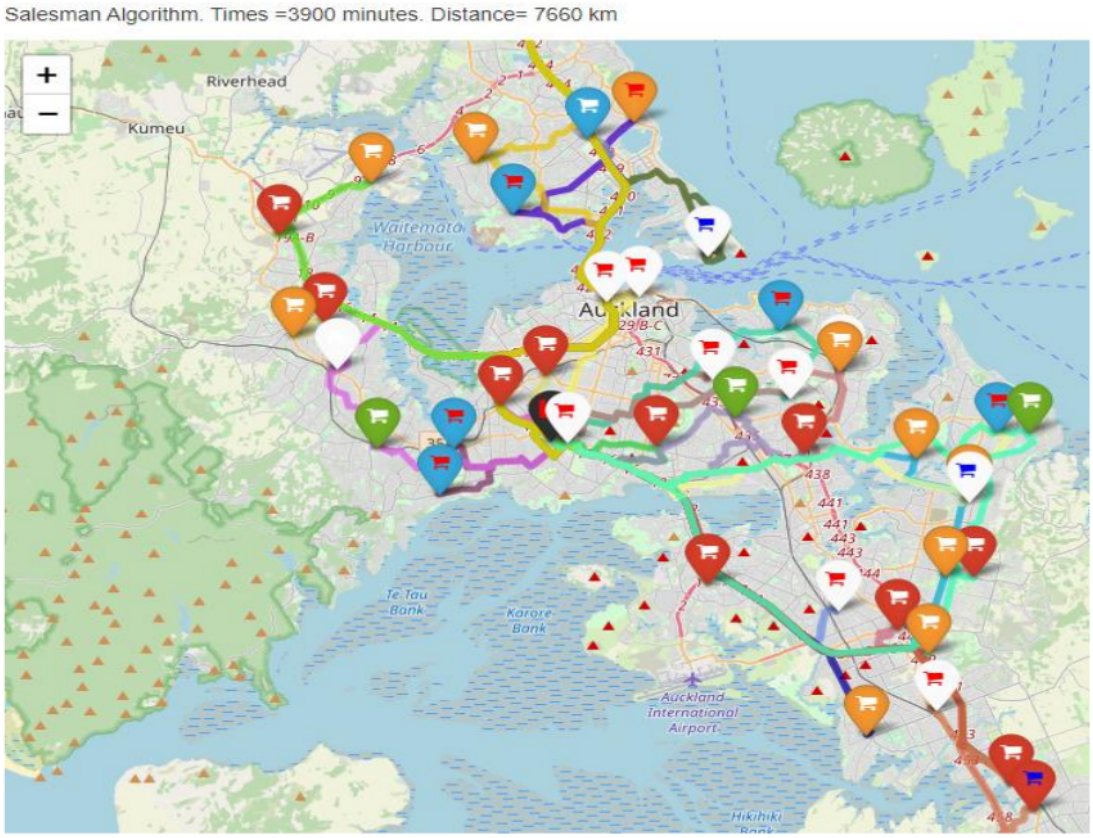


Figure: Using Salesman Algorithm

Here, it takes 3900 minutes to travel 7660 km. This is one of the salesman algorithm which is using for travelling all nodes.

The TSP problem (also called the travelling salesperson problem or TSP) asks the subsequent question: Given a listing of cities and therefore the distances between each pair of cities, what's the shortest possible route that visits each city exactly once and returns to the origin city. The traveling salesman problem was mathematically formulated within the 19th century by a people mathematician W.R. Hamilton and by country mathematician Thomas Kirkman.

6.3 Brute Force Algorithm

Here is the Brute Force Algorithm. This algorithm is used for travel most nodes in short time.

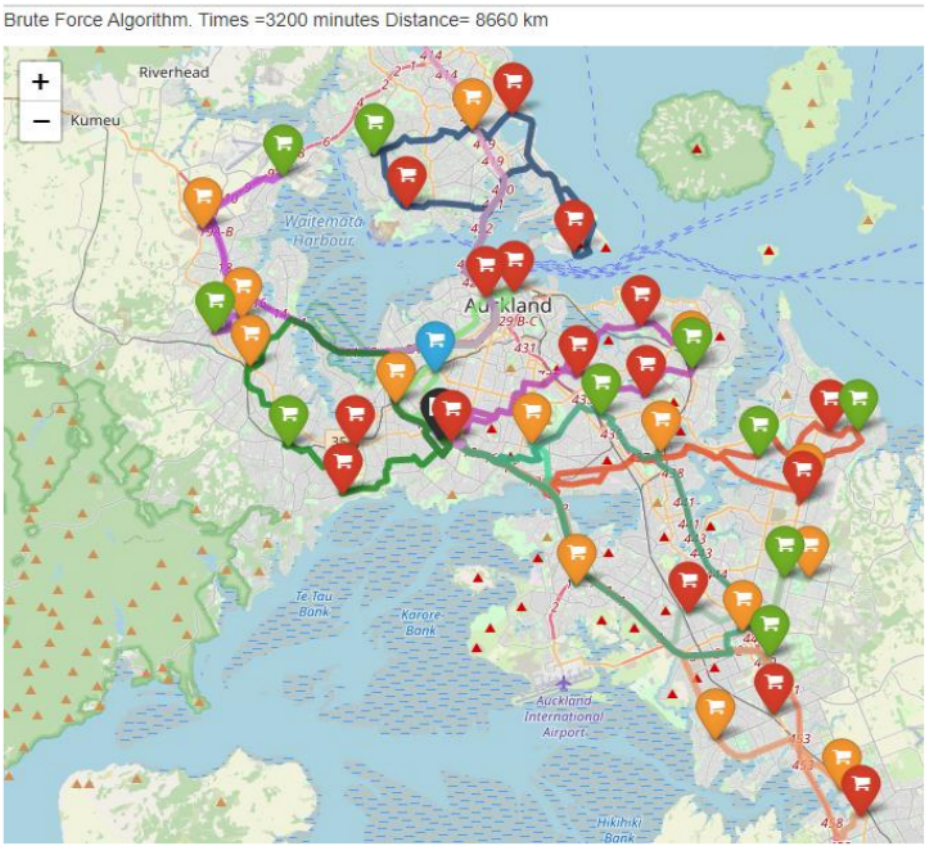


Figure: Using Brute Force

Brute Force takes 3200 minutes to visit 8660 km. This is too much smarter and longest distance visited than others algorithm. It gives us the most fast visited distance.

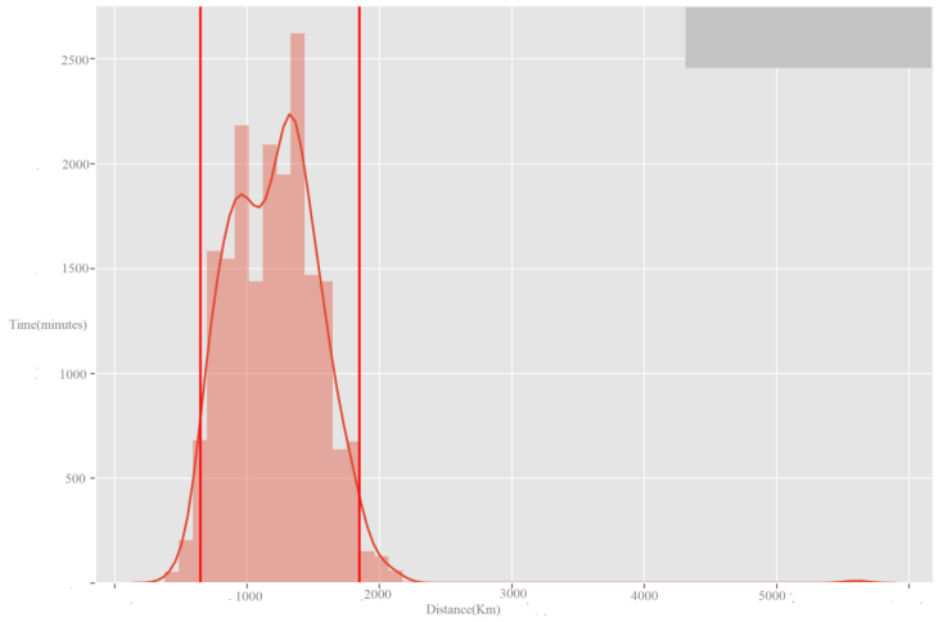
Brute force search is the most common search algorithm as it does not require any domain knowledge; all that is required is a state description, legal operators, the initial state and the description of a goal state. It does not improve the performance and completely relies on the computing power to try out possible combinations.

6.4 Comparison of Algorithms

Analysis of these algorithms, we make a table of their output. Here we are representing by table.

Algorithm	Distance	Time
Greedy Algorithm	7860Km	3400 Minutes
Salesman Algorithm	7660 Km	3900 Minutes
Brute Force Algorithm	8660 Km	3200 Minutes

From this table, we can understand the efficiency of brute force algorithm. This is better than other algorithm. Because, it takes less time than others and by using it we can visit max distances



Here is the efficiency graph, Brute and Force Algorithm is making more visits by cost minimization. Density refers the consumer's number in different places. This algorithm will help Make courier delivery to more people.

Chapter 7

Conclusion

This thesis has discussed many subjects within vehicle routing problems but the core problem, in the methods developed in the papers, has always been a pickup and delivery problem with time windows by Brute Force Algorithm. It is the hope that this thesis significantly has increased the knowledge about the Brute Force Algorithm. The thesis has also improved the boundaries for which Brute Force Algorithm instances that are solvable with exact methods and improved the quality of heuristics for the PDPTW.

• In this paper showed how the delivery can be used to model many other vehicle routing problems. This subject has allowed vehicle routing problem to design a single heuristic that can solve many variants of vehicle routing problem, obtaining just as good solutions as specialized heuristics without the need for any retuning or customization of the heuristics.

• Besides, this paper compared different models for the Pickup to delivery in order to solve the problem to optimality.

• There are several new valid inequalities and corresponding separation algorithms for the problem PDPTW was proposed. We have not performed any polyhedral analysis, but computational experiments show that the valid inequalities are able to improve the lower bounds of the LP relaxations significantly.

Solving Vehicle Routing Problem by using Brute Force Algorithm

ORIGINALITY REPORT

73%	70%	27%	29%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	manualzz.com Internet Source	40%
2	eprints.lancs.ac.uk Internet Source	8%
3	en.wikipedia.org Internet Source	6%
4	Submitted to International Islamic University Chittagong Student Paper	5%
5	backend.orbit.dtu.dk Internet Source	3%
6	www.tandfonline.com Internet Source	2%
7	www.researchgate.net Internet Source	2%
8	testingfreak.com Internet Source	1%
	www.lums.lancs.ac.uk	

9	Internet Source	1 %
10	citeseerx.ist.psu.edu Internet Source	1 %
11	www.greenlogistics.org Internet Source	1 %
12	Submitted to University of Portsmouth Student Paper	1 %
13	www.educba.com Internet Source	1 %
14	Robert A. Nowlan. "Chapter 3 What Else Did Euler Do?", Springer Science and Business Media LLC, 2017 Publication	1 %
15	mafiadoc.com Internet Source	<1 %
16	Submitted to Wawasan Open University Student Paper	<1 %
17	www.gcreddy.com Internet Source	<1 %
18	handwiki.org Internet Source	<1 %
19	www.coursehero.com Internet Source	<1 %

20	Submitted to Adamson University Student Paper	<1 %
21	Stefan Ropke, David Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows", Transportation Science, 2006 Publication	<1 %
22	hdl.handle.net Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Solving Vehicle Routing Problem by using Brute Force Algorithm

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31