Spring 5-22-2019

Benchmarking Optimization Algorithms for Capacitated Vehicle Routing Problems

Pratik Surana
San Jose State University

Benchmarking Optimization Algorithms for Capacitated Vehicle Routing Problems

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Pratik Surana

May 2019

The Designated Project Committee Approves the Project Titled

Benchmarking Optimization Algorithms for Capacitated Vehicle Routing Problems

by

Pratik Surana

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2019

Dr. Sami Khuri	Department of Computer Science
Dr. Natalia Khuri	Department of Bioengineering, Stanford University
Dr. Philip Heller	Department of Computer Science

ABSTRACT

Benchmarking Optimization Algorithms for Capacitated Vehicle Routing Problems

by Pratik Surana

The Vehicle Routing Problem (VRP) originated in the 1950s when algorithms and mathematical approaches were applied to find solutions for routing vehicles. Since then, there has been extensive research in the field of VRPs to solve real-life problems. The process of generating an optimal routing schedule for a VRP is complex due to two reasons. First, VRP is considered to be an NP-Hard problem. Second, there are several constraints involved, such as the number of available vehicles, the vehicle capacities, time-windows for pickup or delivery etc.

The main goal for this project was to compare different optimization algorithms for solving Capacitated Vehicle Routing Problems (CVRP). The three specific aims for this project were to (1) survey existing research and identify suitable optimization algorithms for CVRP and (2) implement a work-flow in the Python programming language, to evaluate their performance, (3) perform different computational experiments on existing CVRP benchmarks.

Experiments were conducted by leveraging Google's OR-Tools library on the well-known benchmarks. Different strategies were evaluated to see if there exists a solution or a better solution than the best-known solutions for these benchmarks. The results show that almost 60% of the problems in the benchmarks have a better solution than the current best-known solution. The second finding of this project is that there is not one strategy which can provide the best solution for all types of CVRPs.

ACKNOWLEDGMENTS

I would first like to thank my advisor Dr. Sami Khuri for being there to guide me always. I really appreciate the efforts he took to encourage me for deeper research into the topics and its associated challenges.

I would also like to take this opportunity to specially thank Dr. Natalia Khuri. I was extremely fortunate to have found a teacher, a guru, a mentor, an inspiration, a role-model and a pillar of support in Dr. Natalia Khuri who has been with me since the very beginning of my Master's of Science journey.

I would like to thank Dr. Philip Heller for overseeing the progress throughout the course of this project and providing timely feedback and guidance.

Finally, I must express my very profound gratitude to my parents, my sister, Pranjal and to my partner, Harshada and all my dear friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this project report. This accomplishment would not have been possible without them.

Thank you.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Background and Significance

There are several types of problems based on transportation routing and scheduling that involve assigning vehicles to delivery or pickup jobs which aims at minimizing the assignment cost and overall routing cost. These problems are important to manufacturing industries, service and transportation companies, such as courier mail, on-demand transportation and taxi services. Below, I describe some of the applications of these problems.

In a **Dynamic Travelling Repairman** problem, the total traveled distance is minimized and/or the urgency of the call is prioritized to determine the route for the repairman. Scheduling a route to repair broken bank teller machines in urban and remote areas is an example of such problems.

In a **Dynamic Dial-a-Ride** problem, one or more commodity kinds or clients must be picked up at one place and delivered to another location. Transport facilities for the elderly and disabled are instances of these issues where clients call for service one day before the desired journey takes place.

In a **Courier Mail service** problem, mail and/or packages are picked up from one location and delivered to some other location in a certain time limit. The mail / packages to be supplied are often not local, but they are shipped from other towns or countries. The shipments are therefore delivered to a centre first and then circulated to other locations from this centre. Before cars leave the facility, all receiving places are known to the driver and the dispatcher. But, the dispatcher and the drivers are uncertain about the pick-ups to be handled during the deliveries.

In **Taxi Cab** applications, the number of dynamic customers is quite large. Hence the planner does not know all customers before the taxi cab leaves the taxi hub.

To get the most number of customers the driver chooses to get back to the centrally located taxi stand rather than waiting at the last customer location as there are more chances of the taxi being requested from the central location. The central location and the dispatch center is shared by all the contractors. Then, based on the number of taxi cabs owned by each contractor, customers are then assigned to the taxi.

All mentioned applications belong to special type of optimization problems,called the Vehicle Routing Problems.

1.2 Vehicle Routing Problems

Vehicle Routing Problem (VRP) is a class of planning problems, which include Static Vehicle Routing Problem (SRP) and Dynamic Vehicle Routing Problem (DVRP) [1]. Inputs to VRP are either deterministic or known with certainty or known with uncertainty, or probabilistic, i.e. follow some probability distribution.

In the classical VRP a set of routes are found, the costs of which are aimed to be minimal. The starting location and ending locations of the route are the same, called depot, to fulfill the demands at each node. The capacity of the vehicles is limited and each node is visited only once by one vehicle. There also exists come constraints on the maximum travelling time in some types of VRPs [2, 3]. The Travelling Salesman Problem (TSP) is an example of VRP. Recall that for the TSP, we are given a set of cities and the cost of travel between each pair. The goal of TSP is to find the cheapest route to visit and return to the starting point. Each city needs to be visited exactly once. A route consists in the order in which the cities are visited [4].

The VRP can be defined more formally as follows:

Input:

- A set of locations C.
- K vehicles available in a depot.

- The cost of a travel from location i to location j.

Assumptions:

- The fleet of vehicles is homogeneous, i.e. all vehicles are of the same size and capacity.
- Depot is denoted by two nodes, node 0 and node n+1.

Constraints:

- The starting node of each route should be a depot and after visiting a subset of nodes the route should end at the depot.
- Each node can be visited exactly once in any given route
- Each vehicle has a maximum capacity Q, which limits the number of nodes it can visit before returning to the depot. Each node has a demand q_i, such that $q_i > 0$ for each $i \epsilon C$ and $q_0 = q_n+1 = 0$.

Output:

- A route scheduled for the entire time period with the locations to be visited, the order in which they need to be visited and the vehicles assigned to visit each location.

Optimization:

- Minimization of the total cost of the route.

Next, I will review a few differences between static VRP and dynamic VRP. The assumption in SVRP is that all data appropriate to route scheduling is known to the planner before the routing process starts. In addition, routing-related data will not alter after the paths have been generated. The included attributes of the customers may be predetermined, such as time of service, their geographic location and the duration of the trip. For example, in Courier Mail Service companies, packages are shipped to various locations from a central hub and all the recipients are known by the dispatcher before the vehicles leave the hub.

In DVRP, the planner does not have all the information when the routing process is done. Also, there can be additional changes in the information even after the initial routes have been generated. For example, in Taxi Cab services, very few customers are pre-planned and known before the departure of the vehicle. Most of the customers are added dynamically after the vehicle has been dispatched from the depot. And thus, the DVRP is much more of a complex problem than SVRP. It can be said that SVRP is a subset of DVRP and is denoted as $P(SVRP) \subset P(DVRP)$ where the problem class of SVRP is denoted by $P(SVRP)$ and the problem class of DVRP is denoted by $P(DVRP)$ [1].

In comparison with SVRP, DVRP presents several additional challenges, such as abiding by time window constraints, managing addition of new customers at real time, managing additional assumptions and constraints at real time, handling different geographical locations, calculating shortest paths each time a new customer is added, updating real-time locations of the vehicles and the customers, working with missing data items, optimizing the costs involved, and so on (Table 1). Applications of DVRP typically involve near term events and not long-term events. The real time information can be available locally (e.g. Only with the dispatcher) or globally (e.g. With the driver along with the dispatcher) and information processing can be centralized or decentralized [1].

Both, SVRP and DVRP are NP-hard problems, i.e. not solvable in polynomial time [5]. Therefore, several heuristic solutions to SVRP and DVRP have been proposed.

1.3 Problem Definition

The main objective of this project is to research the process of transportation scheduling and to evaluate several scheduling algorithms. The process of creation a

Table 1: Comparison of Static and Dynamic Vehicle Routing Problems [1]

SVRP	DVRP
All information known is complete and unchangeable prior to planning the route	Little or no information is known prior to planning the route
No updates of information are needed	Information needs to be up to date at any given point of time
The start and end positions in a route are known	No specific start and end locations are defined making it an unbounded and continuous process
Focus of the planner is well defined and all events carry the same weight	Planner focuses more on near-future and short-term events
There are no updates at real time, so it does not require an update mechanism to be in place	Information update mechanism should be in place to be able to change the solution at real time
The dispatcher must evaluate and assign decisions just once at the beginning	The dispatcher must re-evaluate and re-assign the decisions according to the changes being updated at real time
It may be possible for the dispatcher to spend more time to compute and find the optimal solution for the problem	It may not be possible for dispatcher to provide optimal or even a feasible solution to the problem considering the changing nature of the information
Wait times are irrelevant	Indefinite wait times cannot be afforded in a dynamic setting as the objective would not be met at that point of time
The objective function can be well defined at the time of dispatch	The objective function set at the time of dispatch can be meaningless if there are changes made at real time which would keep changing the objective function each time
The time gap between planning and implementation allows for adjusting the vehicle fleet	There might be a situation where a customer cannot be serviced with existing vehicles and may receive a lower quality service

near perfect or an efficient schedule is complex due to two reasons. First, scheduling is an NP-Hard problem [5, 6]. Second, there are several constraints involved, such as for example, the amount of time customers should be asked to wait for their ride to arrive, the number of days to be scheduled, the number of days window prior to running the scheduling or the maximum number of rides a driver can serve in a day.

1.4 Report Organization

The report is organized as follows: Chapter 2 covers the literature review, briefly describing different algorithms which can be used to solve Vehicle Routing Problems. I go into more specific details for two algorithms which can be used to solve VRP, namely, Greedy algorithms and Genetic algorithms. Each algorithm has a section dedicated to it, which give an overview of the algorithms along with their pseudo-codes. Chapter 3 describes the benchmarks and a computational framework, Google's OR-Tools, which were used to evaluate the performance of several algorithms. It also illustrates the interpretation of the problem and the solution using a toy example. Chapter 4 describes the experiments performed on the benchmarks and how the results were generated and analyzed. Chapter 5 then covers the results of the experiments that were performed using Google's OR-Tools. Finally, Chapter 6 concludes with analyzing the results and major findings along with their shortcomings and future work.

CHAPTER 2

Literature Review

Only a handful of algorithms can solve VRP. These algorithms can be grouped into five different categories [7].

Branch and bound algorithms

A divide-and-conquer strategy is used in the branch and bound algorithm which divides the solution space into local sub-problems and then tries to optimize each sub-problem individually to get the lower and upper bounds to the local sub-problems with the overall ultimate goal of finding a global optimal solution [8].

Heuristic algorithms

The heuristic algorithms explore a limited search space to produce good quality solutions in efficient run times [9]. Genetic Algorithms, Simulated Annealing, Constraint Programming, Ant-Colony and Tabu Search are some of the heuristics that have been applied to solve VRPs [10, 11, 12, 13, 14].

Constructive Methods

In constructive heuristics methods, there exists a minimization function using which, the customers are selected. The routes are then generated using this function along with the capacity and the time constraints [15, 16, 17, 18].

Phase Algorithms

Phase algorithms are divided into two main classes: Cluster-First, Route-Second algorithms [19] and Route-First, Cluster-Second algorithms [20]. In the Cluster-First, Route-Second approach, the customers are initially clustered together and then for every cluster a route is generated. In Route-First, Cluster-Second approach, all the customers are initially considered for generating a tour and then the tour is segmented into smaller feasible routes.

2.1 Greedy Algorithms

A greedy algorithm follows an approach of problem-solving heuristic of approximating a local solution and in an iterative way trying to find an optimal solution to the global problem. In most of the cases a greedy algorithm would not product a globally optimal solution, but it tries to produce a locally optimal solution using reasonable time. For example, in the case of a travelling salesman problem (which is NP-hard [4]) is of the following heuristic: "At each stage visit an unvisited city to the current city with the shortest distance". The algorithm terminates in a reasonable number of steps which may or may not find the optimal solution as it takes unreasonably many steps to find the optimal solution for an NP-hard problem [21].

Greedy Algorithm can optimally solve certain problems and for others, the Greedy Algorithm is just used as a heuristic. [4, 7].

Algorithm 1 Pseudo-code for a general Greedy Algorithm

Result: Feasible Solution

```
while Stop condition is not reached / All Items are visited; do
    Item will be added in a solution set by using some selection function
    if the set would no longer be feasible then
        Reject items under consideration (and never consider the item again)
    else
        Add the current item;
    end
end
```

2.2 Genetic Algorithms

In nature, organisms tend to evolve to adapt to the environment. The theory of natural evolution is the inspiration behind the heuristic optimization method of Genetic Algorithm (GA) [22]. The algorithm works similar to the process of natural selection in which the fittest individuals are selected for reproduction to produce the offspring of the next generation. Researchers have used GAs in the context of

optimization problems. The idea is to effectively search large complex solutions even when the problems consist of high-dimensionality, multimodality, discontinuity and noise [23].

GAs tend to produce better approximations over several iterations on a population of individuals. A new population is generated through the selection process for individuals based on their fitness value. Inspired by natural selection, crossover and mutation are used in GAs. The mutation process may also be applicable to the offspring. This approach generates a population of evolved individuals which are better adapted to the environment than their parents.

There are five steps involved in a general GA, namely initialization, fitness assignment, selection, crossover and mutation Figure 1 .

GA terminology and steps are described below in greater detail.

Encoding

Before a genetic algorithm can be put to work on any problem, a potential solution for that problem should be encoded in a form amiable to processing by a computer. In GA terminology, a solution is encoded in a Chromosome consisting of a set of Genes or a string of values. Usually, binary values are used for encoding the genes in a chromosome.

Step 1: Initialization

In this step, a set of individuals is generated called a Population. A solution to the problem which needs to be solved is given by each individual. At the beginning, the initial population is formed by randomly generating many individual solutions. The given string length should be of an optimal value that is not too small which would hinder effective exploration of the search space, and not too large which would result in reduction of the efficiency of the overall algorithm to find a solution in reasonable amount of computation [24].

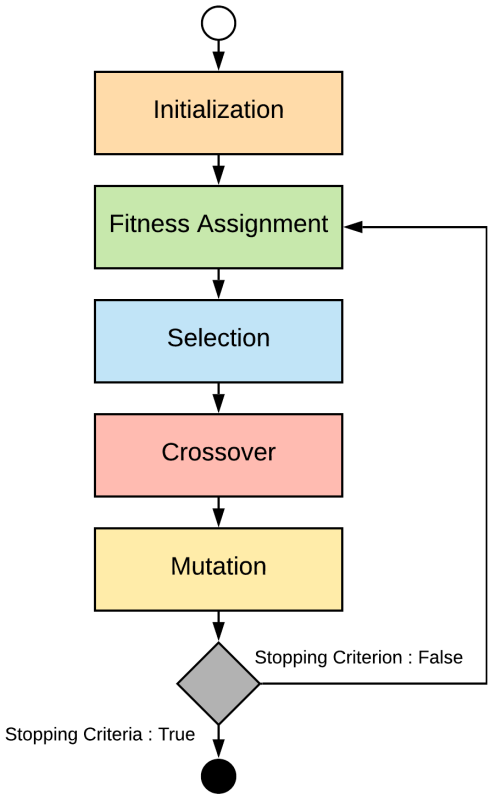

Figure 1: Steps in a Generic Genetic Algorithm

Step 2: Fitness assignment

A fitness value is assigned to every individual in the population which decides its ability of competing with other individuals in the population. The fitness score of every individual helps to determine the probability of being selected or not for the next steps in the algorithm or reproduction.

Step 3: Selection

In this step the fittest individuals are selected to pass their genes to the next generation. Based on their fitness scores, two pairs of individuals or parents are selected. The roulette wheel selection, also known as the stochastic sampling with replacement is one of the most used selection methods [23]. All individuals are placed

on a roulette in which the areas are proportional to their fitness scores.

Algorithm 2 Pseudo-code for roulette wheel selection

Result: Selected pair of individuals

```
while number of desired individuals are selected; do
    r <- random number in the range (0,1)
    sumOfProb <- 0
    while Iterate over all individuals i in the population: do
        if r is in the range of (sumOfProb, sumOfProb + P(X = i)) then
            individual i has been selected
        end
    end
    sumOfProb + P(X = i)
end
```

Step 4: Crossover

In this step the crossover operator will recombine the chosen parents to generate two offspring. This operator picks two individuals at random with a user-defined probability and re-combines their chromosomes using a predefined recombination scheme. .

For example, bit strings of two parent chromosomes are exchanged at random positions between 1 and L - 1 in one-point crossover operation, where L is the chromosome's length. The chromosomes are then split at the selected position and two offspring are created by exchanging their end parts as shown in Figure 2.

Parent 1 : **1 1 0** | **0 0 1**
Parent 2 : 0 1 0 | 1 1 1
Offspring 1 : **1 1 0** | 1 1 1
Offspring 2 : 0 1 0 | **0 0 1**

Figure 2: One-point Crossover at position 3

There are other crossover operators, which may be used for non-binary encoding of chromosomes, such as Partially Mapped crossover, Cycle crossover, Edge recombination, and so on [25]. For instance, in Partially Mapped Crossover (PMX) swapping

operations between two cut points are determined which preserve the absolute positions of elements in the parents. The portions of both parents are mapped with each other and then there is an exchange of the remaining information. The interchange mapping is determined by the section between these two positions as shown in Figure 3.

Parent 1 - 0 | **1 0** | 0
Parent 2 - 1 | **0 1** | 1

Selecting cut points at positions before 2 and after 3 gives the mapping 1 <-> 0 and 0 <-> 1.

Offspring 1 - x | **0 1** | x
Offspring 2 - x | **1 0** | x

The corresponding section of the i^{th} parent is copied and filled in the offspring i (i = 1, 2). In case of the same section being already present in the offspring, it is then replaced as per the mappings.

Offspring 1 - 0 | **0 1** | 0
Offspring 2 - 1 | **1 0** | 1

Figure 3: Partially Mapped Crossover at position 2 and 3

Step 5: Mutation

In this step, the mutation operator processes the bits of the two offspring which were generated in the crossover step. Random perturbations are introduced in the search process by the mutation operator to maintain diversity within the population. A small probability (such as 0.001 for example) is used to apply this operator to each position. For example, the new bit value changes from 0 to 1 or 1 to 0 when this operator is applied at a determined position in a binary chromosome as shown in Figure 4.

Parent : 1 1 0 **0** 0 1
Offspring : 1 1 0 **1** 0 1

Figure 4: Simple Mutation at position 4 of the parent

Step 5: Termination

The entire population is replaced by a new population whenever a new generation is created in a simple genetic algorithm. The fitness function is then used to evaluate the produced offspring again. When the population converges, the algorithm terminates which means that it no longer produces significantly different offspring in the newer generations. Evaluation of convergence can be done in different ways. For example, the algorithm will terminate if any of the following conditions is satisfied.

- No improvement is observed for over a predefined number of iterations
- A predefined number of generations has been reached.
- Fitness function has reached a predefined value.

Illustrated below is how GA can be applied to find a solution to the TSP.

2.2.1 Genetic Algorithms for the Traveling Salesman Problem

Genetic Algorithms have been used to solve TSP [26]. The general framework is as follows.

Encoding

Given a set of cities, each city is given a unique integer in the range of 0, 1,...,*N*-1. Thus, a possible solution is a permutation of the set {0,1,..., *N*-1}, where the order of the cities traversed is specified from left to right.

For example: Let us consider a set of 4 cities {1, 2, 3, 4}.

The encoding 3124 encodes the following tour:

City 3 -> City 1 -> City 2 -> City 4

Step 1: Initialization

In the example for TSP initial population is generated in the size of 6 as follows:

- Tour Chromosome 1 - **1 2 3 4**
- Tour Chromosome 2 - **2 1 3 4**
- Tour Chromosome 3 - **4 3 2 1**

- Tour Chromosome 4 - **3 4 2 1**
- Tour Chromosome 5 - **3 2 1 4**
- Tour Chromosome 6 - **1 2 4 3**

Step 2: Fitness

The fitness function used for TSP evaluates the cost of completing the entire tour, which can be measured by the total traveled distance, for example. Thus, given a tour C_1,...,C_N, the length of a tour is defined as:

$$l = \text{dist}\,(C_N, C_1) + \sum_{i=1}^{N-1} dist\,(c_i, c_{i+1}) \tag{1}$$

where $dist\ (\mathrm{C}_i, \mathrm{C}_{i+1})$ = distance between the cities i and $i + 1$,

$dist\ (\mathrm{C}_n, \mathrm{C}_1)$ = total distance of the complete tour.

In order to describe the shorter tours with higher fitness, the inverse of the total cost of the tour can be chosen as the fitness function:

$$fitness = \frac{1}{l} \tag{2}$$

Step 3: Selection

With the roulette wheel selection for the TSP problem, the probability for individual i to be selected:

$$P(X = i) = \frac{\text{fitness}(i)}{\sum_{j=0}^{N} \text{fitness}(j)} \tag{3}$$

Step 4: Crossover

In the case of TSP, the classical one-point crossover operator cannot be easily applied. Consider, for example the following two parents.

Parent 1 - 1 2 | **3 4**

Parent 2 - 2 1 | **4 3**

Suppose these two ordinal tours are crossed in the third and fourth positions the

following two offspring are created:

Offspring 1 - 1 2 **4 3**

Offspring 2 - 2 1 **3 4**

The sub-tours corresponding to the genes in the tours to the left of the crossover point do not change. However, the sub-tours corresponding to genes to the right of the crossover points are disrupted. The closer the crossover point is to the front of the tour, the greater the disruption of the sub-tour in the offspring. Thus, an invalid solution may be generated. For example:

Parent 1 - 1 2 | **3 4**

Parent 2 - 4 3 | **2 1**

Offspring 1 - 1 2 **2 1**

Offspring 2 - 4 3 **3 4**

Using Partially Matched Crossover (PMX) on the above invalid solution example-

Parent 1 - 1 | **2 3** | 4

Parent 2 - 4 | **3 2** | 1

Selecting cut points at positions before 2 and after 3 generates the mapping 2 <-> 3 and 3 <-> 2.

Offspring 1 - x | **3 2** | x

Offspring 2 - x | **2 3** | x

Then offspring i ($i = 1, 2$) is filled up by copying the respective section of the ith parent. In case, it is already present in the offspring it is replaced according to the mappings.

Offspring 1 - 1 | **3 2** | 4

Offspring 2 - 4 | **2 3** | 1

Mutation

Since the encoding of chromosomes is not binary, a classical swap mutation cannot

be applied because it will produce invalid solutions. Therefore, mutation operators have been devised to preserve the uniqueness of the tour [26].

- Random swap mutation operator chooses 2 random points in the chromosome and swaps them.

  Original chromosome - **1** 2 **3** 4

  Mutated chromosome - **3** 2 **1** 4

- Adjacent swap mutation operator chooses 1 random point in the chromosome and swaps it with the gene to its right

  Original chromosome - **1** 2 3 4

  Mutated chromosome - 2 3 4 **1**

- Inverted exchange mutation operator selects 2 random points in the chromosome and inverts the order of the genes.

  If sub tour 2 3 is selected at random using 2 points, then the result would be

  Original chromosome - 1 2 **3** 4

  Mutated chromosome - 1 3 **2** 4

Next, a GA-based solution to a general VRP is described below.

2.2.2 Genetic Algorithms for Vehicle Routing Problem

A heterogeneous representation of vehicle fleet, passenger loading and vehicle route information is difficult to encode into a single genetic code and formulate and evaluate it, a two-level encoding can be used [10].

The problem is addressed independently in both levels after it is split in two levels.

The allocation of the passengers to the vehicles is done in the upper level and the shortest route for a given set of passengers in a single vehicle is found in the lower level. The variety in the size, distribution and number of vehicles is considered in the upper

level. In the lower level, the selection of fleet and passenger allocation optimization problem is solved. The lower level GA can be used to solve a single vehicle VRP but upper and lower level GA are required for solving VRP with multiple vehicles [27].

GA for Lower Level

- Encoding of Vehicle Routes

  It is non-trivial to encode the vehicle routes genetically which is done by the GA in the lower level. A permutation of integers from 1 to $2n$ is used to represent the vehicle routes, where n is the number of passengers. The pickup of passenger n has the value $P_n = 2n$ - 1 and the drop-off of passenger n has value $D_n = 2n$. The passengers must be picked up before they are dropped off, which leads to a subset of the permutations of a set of numbers that are valid vehicle routes. There exists a fix for this problem which is done by applying a repair algorithm to each chromosome in the initial population as also to every offspring generated by the crossover. The fix includes exchanging the pickup and drop-off positions of the passengers which cause the chromosome to be invalid [25].

- Crossover Operator

  For chromosomes which are permutations, the application of traditional simple crossover operator may result in the production of offspring which are not permutations. Partially Mapped Crossover (PMX) is the method which is used in this case to ensure valid offspring generation [10]. The optimal method is not yet determined as there have been no comparisons made between PMX and other permutation crossover operators.

- The Fitness Function

  In this genetic algorithm, a linear combination of vector of waiting time, w, and travel time (distance), d, for all passengers on the route is defined as the fitness

function, F. In the lower level GA, therefore, an objective function is defined as :

$$F(w,d) = \min \left[\sum_{i=1}^{n} (C_1 w_i + C_2 d_i) \right] \tag{4}$$

GA for Upper Level

- Allocating Passengers to Vehicles

  Alleles are used to represent vehicles in the upper level in the GA. The encoding used for this allocates passengers to multiple vehicles. A simulation can be done for multiple scenarios including heterogeneous fleet of vehicles by storing historical information for each taxi number [25].

- The Fitness Function

  The sum of the fitness values for each single vehicle in case of multiple vehicles is considered as their respective fitness values in the upper level GA.

In the next chapter 3, materials used (benchmarks) and proposed methods to solve VRPs is described along with the implementation of the methods.

CHAPTER 3

Materials and Methods

3.1 Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is a vehicle routing problem with constraints on the capacities of the vehicles. In a CVRP, each location has a demand for items to be picked up or delivered there. Each time a vehicle visits a location, the total amount of items the vehicle is carrying increases (for a pickup) or decreases (for a delivery) by the demand at that location [28].

A graph can be used to represent a CVRP in which the edge corresponds to distances and the nodes correspond to the demands.

3.1.1 Sample CVRP Problem and Solution

Objective : To find the shortest total distance for all vehicles such that the vehicle capacities serving customers demands are not exceeded and the items are picked up / delivered starting and ending at a common depot node.

Expected Output : A route for each vehicle with the total distance traveled.

Given :

Number of locations (n) = 5

Number of vehicles (k) = 2

Capacities of vehicles (c) = (100, 100)

The input files for this example as shown in Figure 3.1.1, derived from the benchmarks used in the experiments section, would be as follows [29]. The example file below describes the problem in sections format. NAME of the file states the values for n and the k. In the COMMENT section the minimum vehicles being used for this problem can be seen (here, 2). Then the TYPE of the problem is given (here, CVRP). DIMENSION would give the number of locations(n) again (here, 5). EDGE_WEIGHT_TYPE states the distance metric being used for this

input (here, EUC_2D). CAPACITY gives the capacities of each vehicle (here, 100). NODE_COORD_SECTION lists all the node locations in the form of x and y coordinates (here, {0,0}, {2,1}, {1,2}, {3,3}, {0,4}). DEMAND_SECTION gives the demand of each location (here, {0, 20, 35, 30, 45}). DEPOT_SECTION gives the node which wouuld be used as the depot for the problem (here, 1). -1 denotes the end of the previous section and EOF denotes the end of the input file.

Here, EUC_2D or Euclidean distance in 2D D_{ij} between two points i and j is defined as :

$$D_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \tag{5}$$

```
NAME : A-n5-k2
COMMENT : (Min no of trucks: 2)
TYPE : CVRP
DIMENSION : 5
EDGE_WEIGHT_TYPE : EUC_2D
CAPACITY : 100
NODE_COORD_SECTION
1 0 0
2 2 1
3 1 2
4 3 3
5 0 4
DEMAND_SECTION
1 0
2 20
3 35
4 30
5 45
DEPOT_SECTION
1
-1
EOF
```

Figure 3.1.1: A Sample Input File

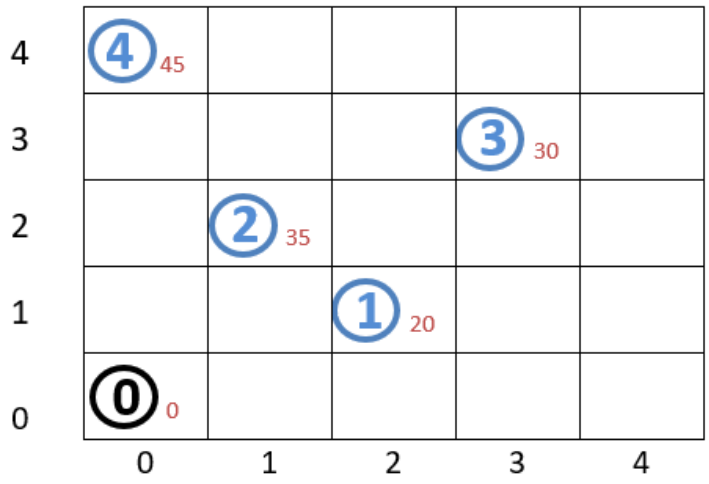

Figure 5: Visual Representation of the sample CVRP problem

Visual representation of this CVRP problem can be depicted as a grid showing the locations to visit in blue and the depot location in black as shown in Figure 5. The demands are shown at the lower right of each location

This example is executed with Google's OR-Tools library using the script given in Appendix **??**, resulting in the solution output given in Figure 3.1.1 [28]

```
Route for vehicle 1:
0 Load(0) ->  2 Load(35) ->  0 Load(35)
Distance of the route: 4
Load of the route: 35

Route for vehicle 2:
0 Load(0) ->  1 Load(20) ->  3 Load(50) ->  4 Load(95) ->  0 Load(95)
Distance of the route: 11
Load of the route: 95

Total Distance of all routes: 15
```

Figure 3.1.1: Solution to the Sample Problem

For each location on a route, the output shows:

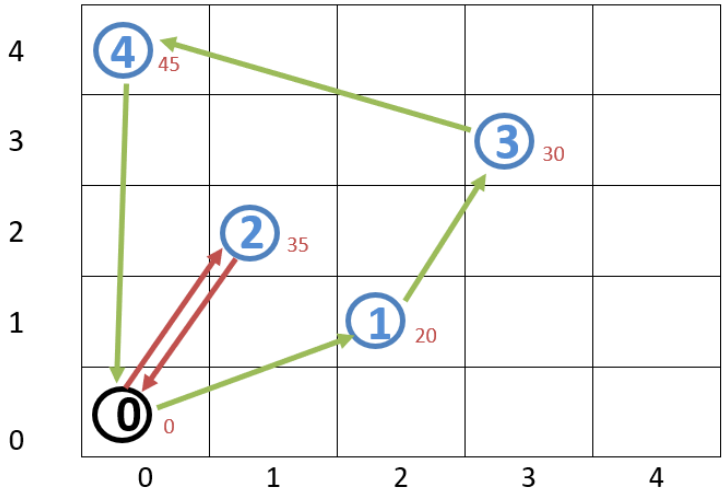

Figure 6: Visual Representation of the solution to sample CVRP problem

- The index of the location.
- The total load carried by the vehicle when it departs the location.

In a visual representation, the solution routes can are as shown in the Figure 6

3.1.2 Program for solving the CVRP (Python Implementation)

The following Python program accepts two command line arguments : [input files directory path] [output files directory path]

To recognize the path correctly an extra \ is added to escape it in the path :

```
python cvrp.py "F:\CS298\A-VRP\\" "F:\CS298\A-VRP-sol\\"
```

The python program given in Appendix **??** is illustrated using Flowchart 7 :

I have uses the three sets of Benchmarks Set A, Set B and Set P by Augerat et al [30]. The input settings information is extracted from these .vrp files from the benchmarks and is fed further to the program. The input settings include number of locations, number of vehicles, depot location, demands at each node, vehicle capacities, generated distance matrix from each location to all other locations and the best-known solution value. For all the three sets of benchmarks, the distance metric used is

Euc_2D.

Each benchmark file is then used with the solver from the OR-Tools with 12 different strategies, each strategy being able to combine with 6 different heuristics. So, I ccould find 72 possible unique solutions using these combinations for each problem in the benchmarks using nested loops. All the 72 possible solutions for each problem in the benchmark sets are written iteratively to a csv file along with the runtime it required to solve. The best OR-Tools solution value out of the 72 solutions is written to a separate csv file for each problem in the benchmarks. This best OR-Tools solution value is then compared with the best-known solution value from the benchmarks to check which strategy-heuristics combination worked for the corresponding input problem and if it is better than the best-known from the benchmarks or not.

3.1.3 Benchmarks for CVRP

The benchmarks used for this experiment are acquired from a repository maintained by the NEO Research group at the Department of LCC from the University of Malaga (Spain) [29]. The benchmarks consist of input and output files. The optimal solutions are determined using the branch-and-cut algorithm which is based on the partial polyhedral description of the corresponding polytope [30]. The input files are given in the similar format as described in the example problem. The output files contain the best-known solutions for the benchmarks as illustrated in Figure 3.1.3 :

[htb]

```
Route #1: 21 31 19 17 13 7 26
Route #2: 12 1 16 30
Route #3: 27 24
Route #4: 29 18 8 9 22 15 10 25 5 20
Route #5: 14 28 11 4 23 3 2 6
cost 784
```

Figure 3.1.3: A Sample Solution File

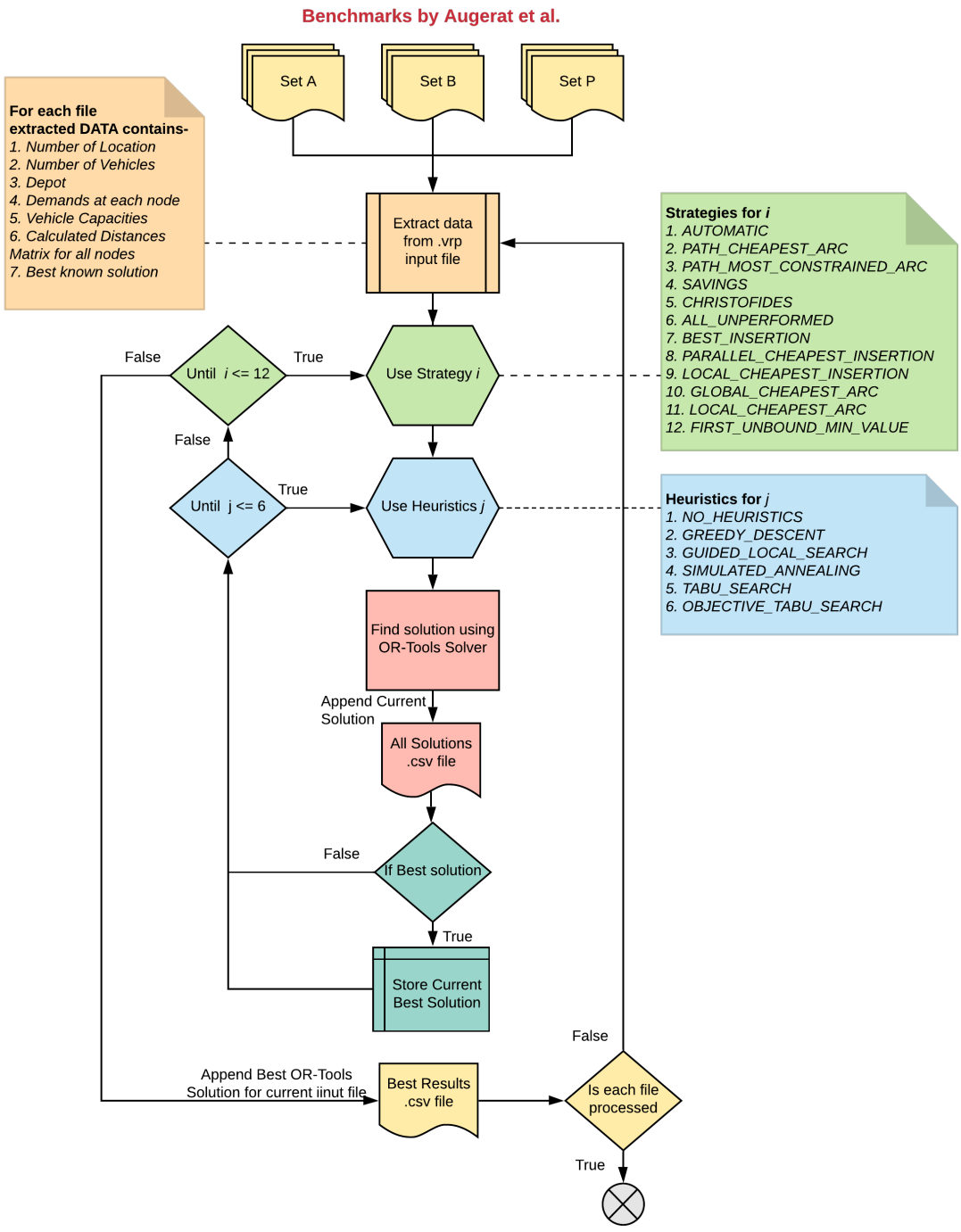

Figure 7: Flowchart of Python Implementation

The next chapter 4 describes the experiments performed on the benchmarks and how the results were generated and analyzed.

CHAPTER 4

Computational Experiments

Experiments were completed on Intel Core i7-7500U processor (2.70 GHz) running Windows 10 with 16.0 GB memory. The following packages and libraries were installed: Python 3.7.2, Google's OR-Tools v7.0 package, json library, re library, math library, os library, sys library, datetime library. Visualization of the results have been done using Tableau.

Google's OR-Tools (v7.0) have been leveraged to test the benchmarks. OR-Tools is an open source tool, developed by Google, to solve optimization problems in vehicle routing, network flows, integer and linear programming, and constraint programming. OR-Tools solver for Vehicle Routing Problems include 12 different optimization strategies, such as PATH_CHEAPEST_ARC, BEST_INSERTION, GLOBAL_CHEAPEST_ARC etc. each of which can be incorporated into 6 different heuristics, such as gradient descent, simulated annealing, tabu search, etc. [28].

4.1 Strategies

More specifically, I used following strategies [28]:

- **AUTOMATIC** - The solver automatically decides on which strategy to use depending on the model of the problem.
- **PATH_CHEAPEST_ARC** - It starts from the route "start" node and using a greedy approach it connects each node after that iteratively which produces the cheapest route.
- **PATH_MOST_CONSTRAINED_ARC** - This is similar to the PATH_CHEAPEST_ARC strategy, but in this it will favor the most constrained arc first using a comparison-based selector function.

- **SAVINGS** - This strategy is based on the algorithm given by Clarke and Write [31].
- **CHRISTOFIDES** - This strategy works on the basis of the algorithm given by Nicos Christofides in which it extends the route for the vehicle routing model until no nodes can be inserted further [32].
- **ALL_UNPERFORMED** - In this strategy, all nodes are made inactive and a solution is found only when optional nodes exist.
- **BEST_INSERTION** - In this strategy a solution is built by inserting the cheapest node iteratively, at its cheapest position where the value of the insertion function corresponds to the globally evaluated cost. This strategy only works with optional nodes.
- **PARALLEL_CHEAPEST_INSERTION** - It is similar to the BEST_INSERTION strategy. The only difference is the insertion function which corresponds to the arc cost function.
- **LOCAL_CHEAPEST_INSERTION** - In this strategy, a solution is built by inserting each node iteratively at its cheapest position where the value of the insertion function corresponds to the arc cost function.
- **GLOBAL_CHEAPEST_ARC** - In this strategy two nodes are iteratively connected resulting in the cheapest route segment.
- **LOCAL_CHEAPEST_ARC** - In this strategy the first node with an unbound successor is selected and connected iteratively to nodes which produce cheapest route segment.
- **FIRST_UNBOUND_MIN_VALUE** - In this strategy, the first available node is selected and connected iteratively to the first unbound successor.

4.2 Heuristics

I evaluated the following heuristics, all of which were readily available in OR-Tools [28].

- **AUTOMATIC** - The solver decides which heuristic to use depending on the model
- **GREEDY_DESCENT** - A local minima is reached by accepting and reducing the cost of local search neighbors.
- **GUIDED_LOCAL_SEARCH** - Guided local search is used for escaping local minima which is generally said to be the most used metaheuristic for VRPs.
- **SIMULATED_ANNEALING** - Uses simulated annealing for escaping local minima.
- **TABU_SEARCH** - Uses tabu search for escaping local minima.
- **OBJECTIVE_TABU_SEARCH** - Uses tabu search on objective value of the function for escaping local minima.

4.3 Description of Specific Experiments

There were three specific experiments that were performed on the sample input file and the benchmarks described in the Materials and Methods section:

1. All benchmarks in set A, set B, and set P were tested using OR-Tools solver. Each problem from the benchmarks is evaluated with 12 different strategies from Google's OR-Tools and combining each strategy with 6 different heuristics. This way I can possibly get 72 different solutions for each problem and find the best solution out of the 72 unique solutions. 7.

2. Vehicle capacities were modified such that they were not the same for all vehicles in a sample benchmark file.

3. Node demands were modified such that some exceeded vehicle capacities in the

sample input file.

Next in Chapter 5, the results of the experiments that were performed using Google's OR-Tools are described.

CHAPTER 5

Results

5.1 Capacitated Vehicle Routing Problem

The program was implemented using the OR-Tools package and tested for the benchmark data sets [29, 30]. In the three results tables for the respective benchmarks, the best Or-Tools column contains the best solution out of the 72 solutions generated using distinct combinations of 12 different strategies and 6 different heuristics and the best-known column contains the solution from the benchmarks.

5.1.1 Benchmarks Set A

For each problem instance, a CVRP solution was found; the running time for computing the solution ranged from 2ms to 2100ms. Out of 28 different input files in Set A, 14 solutions generated by the program are better than the best-known solutions from the benchmarks which are marked green in Table 2. Figure 8 visualizes the comparisons of best-known solution from the benchmarks vs best generated solution using OR-Tools. Figure 9 represents the distribution of the strategies which generated the best OR-Tools solution for each input file in Set A. It can be seen that there is not one algorithm which always gives the best solution for each problem.

5.1.2 Benchmarks Set B

For each problem instance in Set B of benchmarks, a CVRP solution was found; the running time for computing the solution ranged from 12ms to 2040ms. The best Or-Tools Solution column contains the best solution out of the 72 solutions generated using distinct combinations of 12 different strategies and 6 different heuristics. Out of 22 different input files in Set B, 15 solutions generated by the program are better than the best-known solutions from the benchmarks which are marked green in Table 3. Figure 10 visualizes the comparisons of best-known solution from the benchmarks vs best generated solution using OR-Tools. Figure 11 represents the distribution of

Table 2: Results from Benchmarks Set A

FILE NAME	N	K	C	BEST KNOWN	BEST ORTOOLS	USING STRATEGY	HEURISTICS	RUNTIME (ms)
A-n32-k5.vrp	32	5	100	784	773	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	999.323
A-n33-k5.vrp	33	5	100	661	650	AUTOMATIC	GUIDED_LOCAL _SEARCH	1004.899
A-n33-k6.vrp	33	6	100	742	726	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	999.729
A-n34-k5.vrp	34	5	100	778	770	SAVINGS	GUIDED_LOCAL _SEARCH	997.85
A-n36-k5.vrp	36	5	100	799	797	AUTOMATIC	GUIDED_LOCAL _SEARCH	999.762
A-n37-k5.vrp	37	5	100	669	656	AUTOMATIC	GUIDED_LOCAL _SEARCH	999.799
A-n37-k6.vrp	37	6	100	949	960	SAVINGS	GUIDED_LOCAL _SEARCH	999.982
A-n38-k5.vrp	38	5	100	730	722	AUTOMATIC	GUIDED_LOCAL _SEARCH	1006.46
A-n39-k5.vrp	39	5	100	822	814	AUTOMATIC	GUIDED_LOCAL _SEARCH	1005.607
A-n39-k6.vrp	39	6	100	831	815	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	1005.477
A-n44-k7.vrp	44	6	100	937	952	PARALLEL_CHEAPEST _INSERTION	TABU_SEARCH	993.901
A-n45-k6.vrp	45	6	100	944	1053	SAVINGS	GUIDED_LOCAL _SEARCH	1003.519
A-n45-k7.vrp	45	7	100	1146	1128	GLOBAL_CHEAPEST _ARC	GUIDED_LOCAL _SEARCH	1002.742
A-n46-k7.vrp	46	7	100	914	903	GLOBAL_CHEAPEST _ARC	SIMULATED_ ANNEALING	1002.205
A-n48-k7.vrp	48	7	100	1073	1072	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	999.798
A-n53-k7.vrp	53	7	100	1010	1037	SAVINGS	GUIDED_LOCAL _SEARCH	1004.589
A-n54-k7.vrp	54	7	100	1167	1173	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	989.424
A-n55-k9.vrp	55	9	100	1073	1048	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	999.326
A-n60-k9.vrp	60	9	100	1408	1394	SAVINGS	SIMULATED_ ANNEALING	999.359
A-n61-k9.vrp	61	9	100	1034	1105	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	999.359
A-n62-k8.vrp	62	8	100	1290	1328	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1000.324
A-n63-k10.vrp	63	10	100	1315	1336	SAVINGS	TABU_SEARCH	1001.353
A-n63-k9.vrp	63	9	100	1634	1678	SAVINGS	GUIDED_LOCAL _SEARCH	999.287
A-n64-k9.vrp	64	9	100	1402	1431	FIRST_UNBOUND_ MIN_VALUE	SIMULATED_ ANNEALING	1000.322
A-n65-k9.vrp	65	9	100	1174	1207	SAVINGS	GUIDED_LOCAL _SEARCH	1000.362
A-n69-k9.vrp	69	9	100	1168	1184	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	999.324
A-n80-k10.vrp	80	10	100	1764	1827	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	999.326

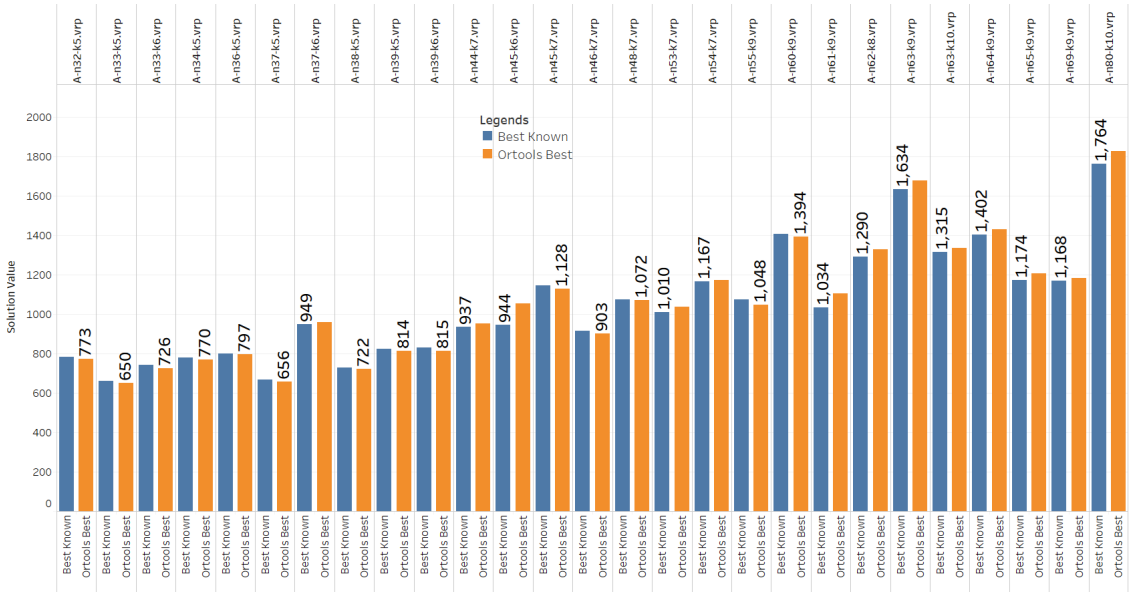

Figure 8: Visualization of Best-Known Solution vs Best OR-Tools Solution for input files in Set A

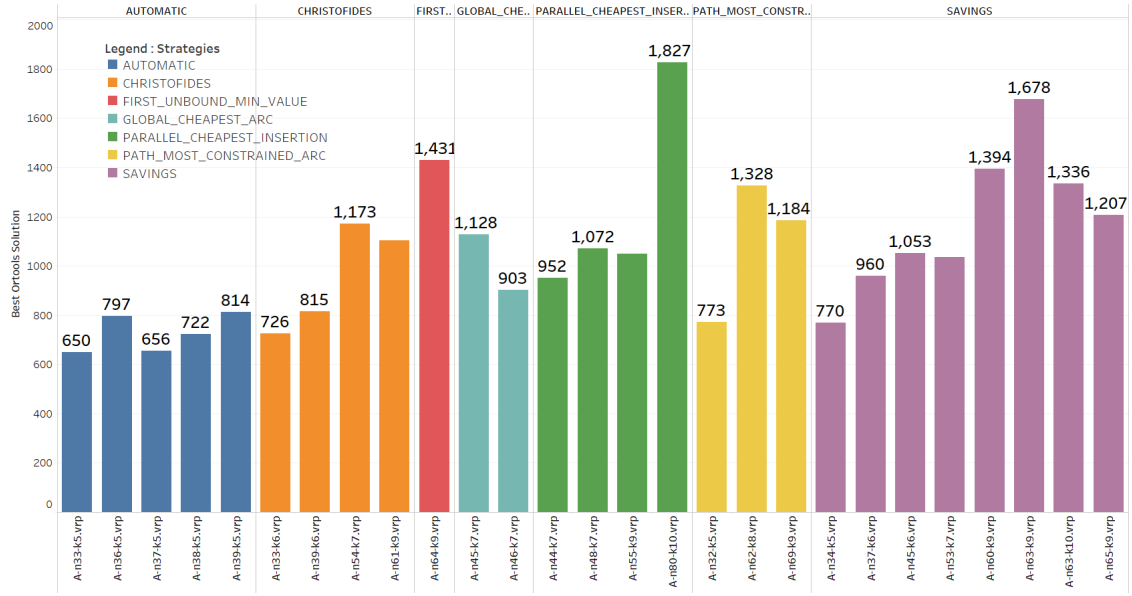

Figure 9: Visualization of OR-Tools Solution for each input file in Set A grouped by strategies

the strategies which generated the best OR-Tools solution for each input file in Set B. Similarly, as the results of Set A, it can be seen here that there is not one algorithm

which always gives the best solution for each problem.

Table 3: Results from Benchmarks Set B

FILE NAME	N	K	C	BEST KNOWN	BEST ORTOOLS	USING STRATEGY	HEURISTICS	RUNTIME (ms)
B-n34-k5.vrp	34	5	100	788	780	AUTOMATIC	GUIDED_LOCAL _SEARCH	2004.609
B-n35-k5.vrp	35	5	100	955	945	SAVINGS	GUIDED_LOCAL _SEARCH	2003.074
B-n38-k6.vrp	38	6	100	805	796	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	2003.639
B-n39-k5.vrp	39	5	100	549	535	CHRISTOFIDES	TABU_SEARCH	2005.638
B-n41-k6.vrp	41	6	100	829	817	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	2003.659
B-n43-k6.vrp	43	6	100	742	734	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	1994.6
B-n44-k7.vrp	44	7	100	909	919	AUTOMATIC	TABU_SEARCH	1999.73
B-n45-k5.vrp	45	5	100	751	751	SAVINGS	TABU_SEARCH	2013.24
B-n45-k6.vrp	45	6	100	678	765	CHRISTOFIDES	SIMULATED_ ANNEALING	2000.617
B-n50-k7.vrp	50	7	100	741	725	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1993.909
B-n50-k8.vrp	50	8	100	1313	1307	PATH_MOST_ CONSTRAINED_ARC	TABU_SEARCH	1993.449
B-n51-k7.vrp	51	7	100	1032	1031	FIRST_UNBOUND _MIN_VALUE	SIMULATED_ ANNEALING	1999.649
B-n52-k7.vrp	52	7	100	747	734	SAVINGS	GUIDED_LOCAL _SEARCH	2000.649
B-n56-k7.vrp	56	7	100	707	696	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	1999.648
B-n57-k9.vrp	57	9	100	1598	1612	FIRST_UNBOUND _MIN_VALUE	TABU_SEARCH	2000.68
B-n63-k10.vrp	63	10	100	1496	1537	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	1999.613
B-n64-k9.vrp	64	9	100	861	882	SAVINGS	GUIDED_LOCAL _SEARCH	1999.65
B-n66-k9.vrp	66	9	100	1374	1320	PATH_MOST_ CONSTRAINED_ARC	TABU_SEARCH	2001.676
B-n67-k10.vrp	67	10	100	1032	1059	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1998.69
B-n68-k9.vrp	68	9	100	1304	1282	LOCAL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	1998.653
B-n78-k10.vrp	78	10	100	1266	1249	PARALLEL_CHEAPEST _INSERTION	TABU_SEARCH	1999.65

5.1.3 Benchmarks Set P

For each problem instance in Set P of benchmarks, a CVRP solution was found; the running time for computing the solution ranged from 19ms to 2078ms. The best Or-Tools Solution column contains the best solution out of the 72 solutions generated using distinct combinations of 12 different strategies and 6 different heuristics. Out of 21 different input files in Set P, 11 solutions generated by the program are better

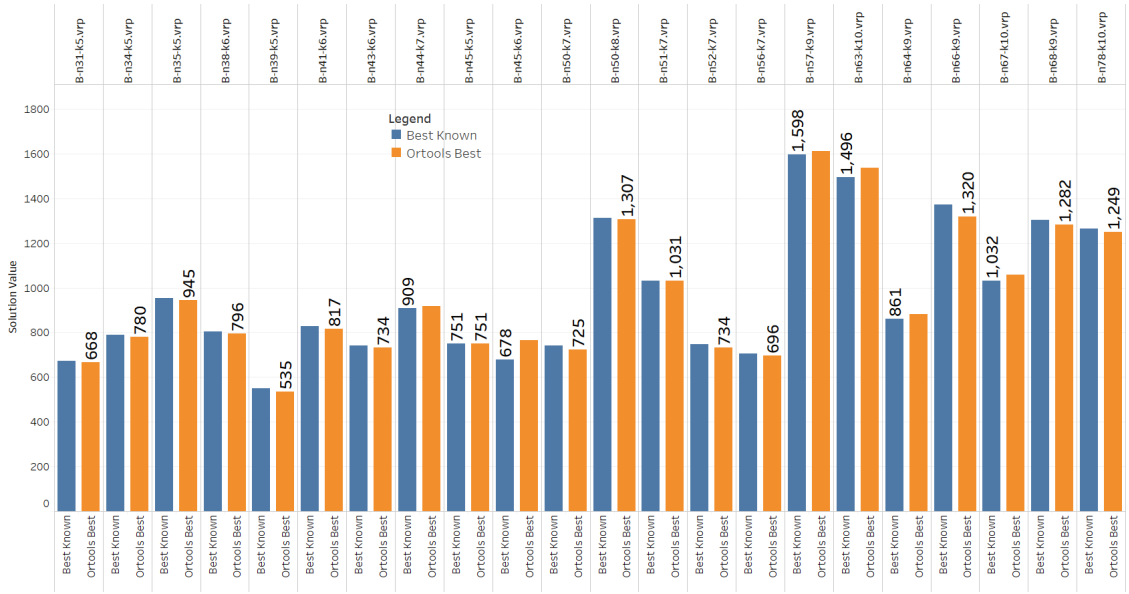

Figure 10: Visualization of Best-Known Solution vs Best OR-Tools Solution for input files in Set B

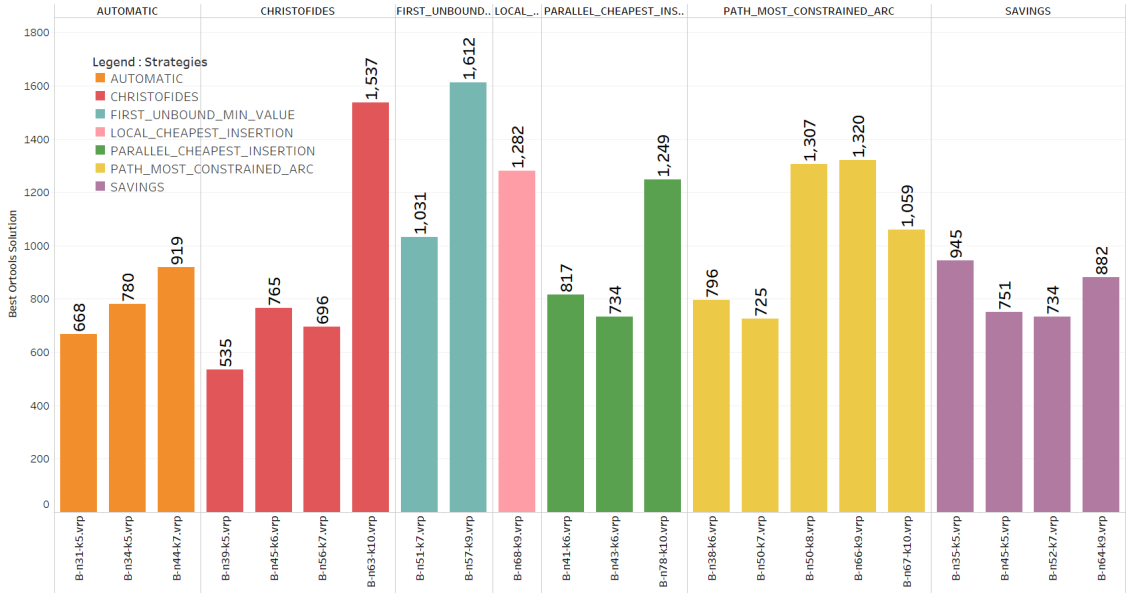

Figure 11: Visualization of OR-Tools Solution for each input file in Set B grouped by strategies

than the best-known solutions from the benchmarks which are marked green in Table 4. Figure 12 visualizes the comparisons of best-known solution from the benchmarks

vs best generated solution using OR-Tools. Figure 13 represents the distribution of the strategies which generated the best OR-Tools solution for each input file in Set P. Similarly, as the results of Set A and Set B, it can be seen here that there is not one algorithm which always gives the best solution for each problem.

Table 4: Results from Benchmarks Set P

FILE NAME	N	K	C	BEST KNOWN	BEST ORTOOLS	USING STRATEGY	HEURISTICS	RUNTIME (ms)
P-n101-k4.vrp	101	4	400	681	671	CHRISTOFIDES	SIMULATED_ ANNEALING	2000.647
P-n16-k8.vrp	16	8	35	435	444	AUTOMATIC	GUIDED_LOCAL _SEARCH	2000.707
P-n19-k2.vrp	19	2	160	212	206	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1999.652
P-n20-k2.vrp	20	2	160	220	211	AUTOMATIC	GUIDED_LOCAL _SEARCH	1999.652
P-n21-k2.vrp	21	2	160	211	208	AUTOMATIC	NO_HEURISTICS	45.875
P-n22-k2.vrp	22	2	160	216	212	AUTOMATIC	NO_HEURISTICS	19.222
P-n22-k8.vrp	22	8	3000	603	590	AUTOMATIC	GUIDED_LOCAL _SEARCH	2000.646
P-n40-k5.vrp	40	5	140	458	448	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1999.65
P-n45-k5.vrp	45	5	150	510	499	PATH_MOST_ CONSTRAINED_ARC	GUIDED_LOCAL _SEARCH	1999.652
P-n50-k10.vrp	50	10	100	696	703	LOCAL_CHEAPEST _ARC	GUIDED_LOCAL _SEARCH	2000.656
P-n50-k7.vrp	50	7	150	554	549	AUTOMATIC	GUIDED_LOCAL _SEARCH	2000.648
P-n51-k10.vrp	51	10	80	745	742	SAVINGS	GUIDED_LOCAL _SEARCH	2000.674
P-n55-k10.vrp	55	10	115	669	693	AUTOMATIC	GUIDED_LOCAL _SEARCH	1999.62
P-n55-k7.vrp	55	7	170	524	553	LOCAL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	1999.688
P-n55-k8.vrp	55	8	160	576	563	LOCAL_CHEAPEST _ARC	GUIDED_LOCAL _SEARCH	1999.701
P-n60-k10.vrp	60	10	120	706	756	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	2000.65
P-n60-k15.vrp	60	15	80	905	966	SAVINGS	TABU_SEARCH	1999.666
P-n65-k10.vrp	65	10	130	792	793	LOCAL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	1999.714
P-n70-k10.vrp	70	10	135	834	864	PARALLEL_CHEAPEST _INSERTION	GUIDED_LOCAL _SEARCH	1999.651
P-n76-k4.vrp	76	4	350	589	606	AUTOMATIC	GUIDED_LOCAL _SEARCH	2000.656
P-n76-k5.vrp	76	5	280	631	645	CHRISTOFIDES	GUIDED_LOCAL _SEARCH	1999.839

5.1.4 Other Experiments

I have next examined how changes in program specification affect the solution obtained with OR-Tools package by performing two experiments.

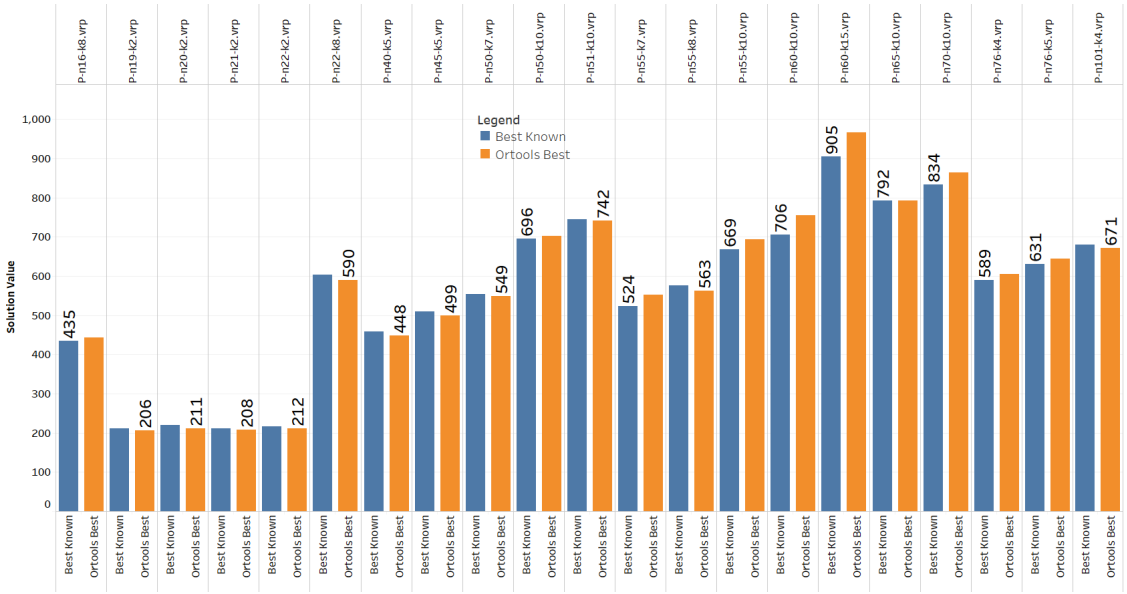

Figure 12: Visualization of Best-Known Solution vs Best OR-Tools Solution for input files in Set P

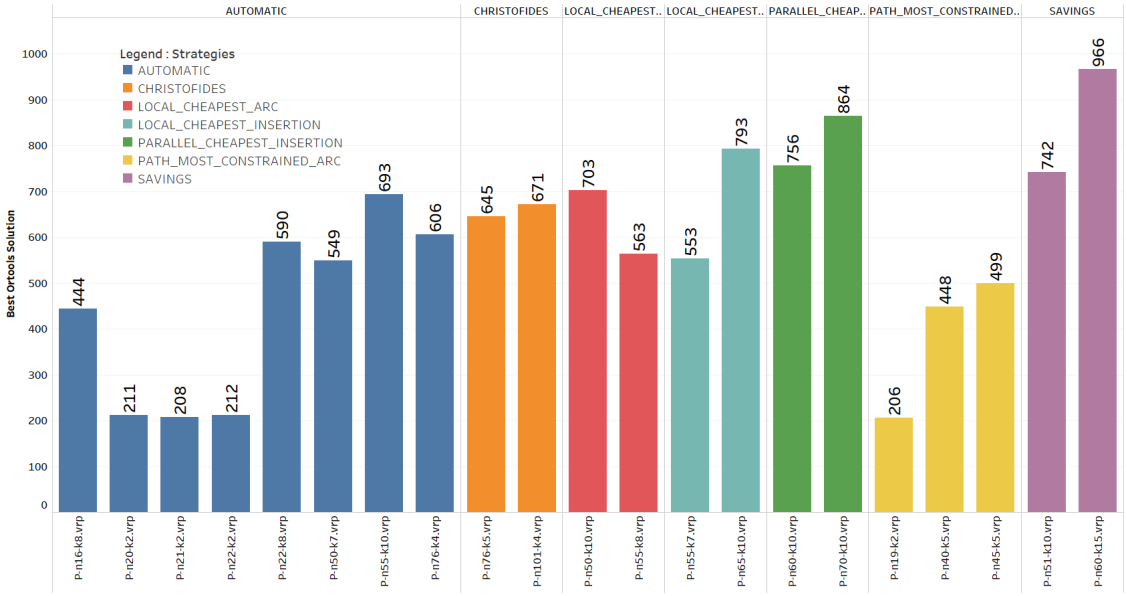

Figure 13: Visualization of OR-Tools Solution for each input file in Set P grouped by strategies

1. Node demands were modified such that they exceeded vehicle capacities:

   In this experiment the program did not return a result. Google's OR-Tools

solver does not find the solution even one node's demand exceeds capacity.

2. Modified vehicle capacities to different capacities for each vehicle for the sample input file :

   Keeping everything else the same i.e. number of locations =4, minimum vehicles being used for this problem = 2, distance metric = Euc_2d, node coordinates = {0,0}, {2,1}, {1,2}, {3,3}, {0,4}, demands at each nodes = {0, 20, 35, 30, 45}, depot node = 1 and just just varying the capacities of 2 vehicles as 50 and 80, the following solution was found :

```
Route for vehicle 1:
 0 Load(0) ->  1 Load(20) ->  3 Load(50) ->  0 Load(50)
Distance of the route: 8
Load of the route: 50

Route for vehicle 2:
 0 Load(0) ->  2 Load(35) ->  4 Load(80) ->  0 Load(80)
Distance of the route: 8
Load of the route: 80

Total Distance of all routes: 16
```

   In a visual representation, the routes can be shown aa in Figure 14 which provides a different valid solution for the problem as the constraints on the capacities of each vehicles were changed.

I have concluded with the analysis of the results and major findings along with their shortcomings and future work in the next chapter.

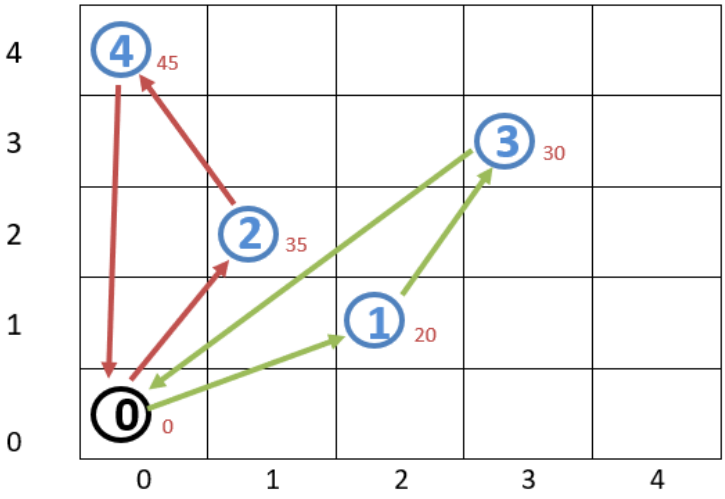

Figure 14: Visual Representation of the solution to experiment 2 problem

CHAPTER 6

Conclusion

In this project, I reviewed existing research to identify suitable optimization algorithms for CVRP including greedy and genetic algorithms. I evaluated 12 different strategies and combined them with 6 different heuristics to potentially be able to produce 72 different solutions for a problem using Google's OR-Tools. I used benchmarks by Augerat et al. and performed different computational experiments like varying capacities of vehicles, exceeding demands of the nodes more than the capacities of the vehicles and using different distance metrics.

It can be observed that we have atleast one result for each of the problems in the benchmarks, produced by the python implementation using OR-Tools. Out of all the 72 different solutions, the best results produced by OR-Tools for each problem are generated using different strategies in different problems. When the strategies which provide the best results using OR-Tools are clustered it can be seen that there is not one strategy-heuristics combination which can be said to work best for all the benchmarks. The results show that almost 60% of the problems in the benchmarks have a better solution produced by OR-Tools than the current best-known solution.

Future extensions of this project would include testing with a large data set as well as with the live source of data from a real-life application. Moreover, OR-Tools could be extended to find feasible solutions for problems with varied capacities, for example. Finally, more extensive testing could be done to find correlation between the types of benchmarks and the performance of strategy-heuristics combinations.

LIST OF REFERENCES

[1] L. Allan, ''The dynamic vehicle routing problem,'' *Kgs. Lyngby, Denmark: Technical University of Denmark (DTU)*, vol. IMM-PHD, No. 2000-73, /6 2000.

[2] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, *Chapter 2 Time constrained routing and scheduling*, ser. Handbooks in Operations Research and Management Science. Elsevier B.V, 1995, vol. 8, pp. 35--139. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927050705801069

[3] P. Belfiore and H. T. Y. Yoshizaki, ''Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil,'' *European Journal of Operational Research*, vol. 199, no. 3, pp. 750--758, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221708006978

[4] ''The traveling salesman problem: a computational study,'' *Choice Reviews Online*, vol. 45, no. 2, p. 0928, Oct 1, 2007.

[5] J. K. Lenstra and A. H. G. Rinnooy Kan, ''Complexity of vehicle routing and scheduling problems,'' *Networks*, vol. 11, no. 2, pp. 221--227, Jan 1, 1981. [Online]. Available: https://www.narcis.nl/publication/RecordID/oai:cwi.nl:20665

[6] M. R. Garey, *Computers and intractability : a guide to the theory of NP-completeness*, United States, 1979. [Online]. Available: http://catalog.hathitrust.org/Record/000187861

[7] P. Munari, T. Dollevoet, and R. Spliet, ''A generalized formulation for vehicle routing problems,'' Jun 6, 2016. [Online]. Available: https://arxiv.org/abs/1606.01935

[8] C. Archetti, N. Bianchessi, and M. G. Speranza, ''Branch-and-cut algorithms for the split delivery vehicle routing problem,'' *European Journal of Operational Research*, vol. 238, no. 3, pp. 685--698, Nov 1, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037722171400352X

[9] A. V. Breedam, ''A parametric analysis of heuristics for the vehicle routing problem with side-constraints,'' *European Journal of Operational Research*, vol. 137, no. 2, pp. 348--370, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221701001503

[10] T. Potter and T. Bossomaier, ''Solving vehicle routing problems with genetic algorithms,'' vol. 2. IEEE, 1995, p. 793 vol.2. [Online]. Available: https://ieeexplore.ieee.org/document/487486

[11] Z. J. Czech and P. Czarnas, ''Parallel simulated annealing for the vehicle routing problem with time windows.'' IEEE, 2002, pp. 376--383. [Online]. Available: https://ieeexplore.ieee.org/document/994313

[12] D. Pisinger and S. Ropke, ''A general heuristic for vehicle routing problems,'' *Computers and Operations Research*, vol. 34, no. 8, pp. 2403--2435, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054805003023

[13] C.-H. Chen, C.-J. Ting, and P.-C. Chang, *Applying a Hybrid Ant Colony System to the Vehicle Routing Problem*, ser. Computational Science and Its Applications – ICCSA 2005. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3483, pp. 417--426.

[14] A. Amberg, W. Domschke, and S. Voß, ''Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees,'' *European Journal of Operational Research*, vol. 124, no. 2, pp. 360--376, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221799001708

[15] K. Altinkemer and B. Gavish, ''Parallel savings based heuristics for the delivery problem,'' *Operations Research*, vol. 39, no. 3, pp. 456--469, May 1, 1991. [Online]. Available: http://or.journal.informs.org/cgi/content/abstract/39/3/456

[16] P. M. Thompson and H. N. Psaraftis, ''Cyclic transfer algorithms for multivehicle routing and scheduling problems,'' *Operations Research*, vol. 41, no. 5, pp. 935--946, Sep 1, 1993. [Online]. Available: https://www.jstor.org/stable/171656

[17] J.-Y. Potvin, ''Evolutionary algorithms for vehicle routing,'' *INFORMS Journal on Computing*, vol. 21, no. 4, p. 518, Oct 1, 2009. [Online]. Available: https://search.proquest.com/docview/200524814

[18] L. Zeng, H. L. Ong, and K. M. Ng, ''A generalized crossing local search method for solving vehicle routing problems,'' *The Journal of the Operational Research Society*, vol. 58, no. 4, pp. 528--532, Apr 1, 2007. [Online]. Available: https://www.jstor.org/stable/4622725

[19] M. Ammi and S. Chikhi, ''A generalized island model based on parallel and cooperating metaheuristics for effective large capacitated vehicle routing problem solving,'' *Journal of Computing and Information Technology*, vol. 23, no. 2, p. 141, 2015. [Online]. Available: https://hrcak.srce.hr/139792

[20] R. Mole, D. Johnson, and K. Wells, ''Combinatorial analysis for route first-cluster second vehicle routing,'' *Omega*, vol. 11, no. 5, pp. 507--512, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0305048383900439

[21] ‘‘Greedy algorithms.’’ [Online]. Available: https://en.wikipedia.org/wiki/Greedy_algorithm

[22] D. Goldberg and K. Sastry, *Genetic Algorithms*, 2nd ed. Berlin: Springer US, 2007.

[23] K. D. Jong, ‘‘Adaptive system design: A genetic approach,’’ *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 9, pp. 566--574, Sep 1980. [Online]. Available: https://ieeexplore.ieee.org/document/4308561

[24] S. Sverchkov, ‘‘Shestakov i.p.,sverchkov s.r.“algebraic approach to optimal initial populations and initial populations of the optimal size of a genetic algorithm”, abstracts of the international conference maltcev meeting, sobolev institute of mathematics, novosibirsk, (2012), p. 120.’’ Unpublished, 2012. [Online]. Available: https://search.datacite.org/works/10.13140/2.1.3105.7920

[25] A. Hussain, Y. S. Muhammad, M. N. Sajid, I. Hussain, A. M. Shoukry, and S. Gani, ‘‘Genetic algorithm for traveling salesman problem with modified cycle crossover operator,’’ *Computational intelligence and neuroscience*, vol. 2017, pp. 7 430 125--7, 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29209364

[26] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, ‘‘Genetic algorithms for the travelling salesman problem: A review of representations and operators,’’ *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129--170, Apr 1999. [Online]. Available: https://search.proquest.com/docview/198028773

[27] J. A. Gromicho Dos Santos, J. J. van Hoorn, A. L. Kok, and J. M. J. Schutten, ‘‘Restricted dynamic programming: a flexible framework for solving realistic vrps,’’ *Computers and Operations Research*, vol. 39, no. 5, pp. 902--909, 2012. [Online]. Available: https://www.narcis.nl/publication/RecordID/oai:research.vu.nl:publications%2F86339457-6482-4a5c-b929-e54a91434c9d

[28] G. D. (n.d.), ‘‘Capacitated vehicle routing problem, or-tools.’’ [Online]. Available: https://developers.google.com/optimization/routing/cvrp

[29] ‘‘Vehicle routing problem.’’ [Online]. Available: http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/

[30] P. Augerat, E. Benavent, C. Prins, C. Prodhon, R. W. Calvo, and J.-M. Belenguer, ‘‘A branch-and-cut method for the capacitated location-routing problem,’’ *Computers and Operations Research*, vol. 38, no. 6, pp. 931--941, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054810002145

[31] G. Clarke and J. W. Wright, ‘‘Scheduling of vehicles from a central depot to a number of delivery points,’’ *The roots of logistics*, pp. 229--244, 2012. [Online]. Available: http://www.econis.eu/PPNSET?PPN=719327458

[32] N. Christofides, ‘‘Worst-case analysis of a new heuristic for the travelling salesman problem,’’ Tech. Rep., Feb 1976. [Online]. Available: http://www.dtic.mil/docs/citations/ADA025602