PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

University Mohamed Seddik Ben Yahia of Jijel, Algeria.

Faculty of Exact Sciences and Computer Science

Department of Computer Science

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment of the Requirements for the Degree of Master

Option: Artificial Intelligence

Solving the consistent Vehicle Routing Problem with Profit using Simulated Annealing

Submitted by:

Ms. Rima Cheraitia

Supervised by:

Dr. Hamida Bouaziz

September 2021

Dedication

To my dear, beloved mother, the most precious person to my heart, and my dear father

No devotion can express my respect, eternal love and appreciation for the sacrifices you made for my education and well-being.

May this humble work be the fruit of your innumerable sacrifices, though I cannot satisfy a little of your labors with me.

May God Almighty grant you health, happiness and long life.

To my supportive supervisor, who encouraged me until the end.

To my beloved husband ***Hamza*** for his support and encouragement.

To the soul of my dear brother ***Youssef*** - may God have mercy on him -

To my sisters: ***Ibtissame*** and ***Asma***, for their love, encouragement and continuous support.

To my brothers: ***Ammar***, ***Hamza*** and especially to ***Abd al-Rahman***.

To all my friends, and especially to: ***Loubna***, ***Nesrine***, ***Soumia***, ***Inese***.

Finally, to all those who have taught me throughout my school life.

Thanks

First of all I thank Allah the Almighty, the Merciful, who gave me the courage, the will, the strength, the health and the persistence to accomplish this modest work. Thank you for lighting my way to success.

And I would like to express all my gratitude to: My supervisor: ***Dr. Hamida Bouaziz***, for his patience, and above all for his confidence, his remarks and his advice, his availability and his benevolence.

All the teachers in the computer science department, who attended my computer science debut, for their valuable advice. my sincere thanks to the members of the jury for the honor they do me by accepting to judge this work and for all their pertinent comments.

I express my gratitude to everyone who has contributed from near or far to the completion of this work, my teachers, my parents, and my brothers, sisters, my friends.

Finally, we hope that this work will have the desired value.

Abstract

Consistent VRP with profits is a variant of vehicle routing problems, it was introduced in 2019. In this problem, each instance is defined by a depot, a group of customers (frequent customers and non-frequent customers) and a group of vehicles with a specific capacity. In this variant, each customer has a demand, a service time, and a fixed profit. The goal is to find the best set of paths for the group of vehicles, while respecting the constraints of consistency, which are Person consistency, Time consistency, and consistency in the delivered quantity. In this work, we used the simulated annealing meta-heuristic for solving the consistent VRP with profit. In the definition of the neighborhood, we have implemented several operators noted 2-Opt move, 1-1 exchange move and 1-0 exchange move. In the computational experiments, we have compared the results obtained by the simulated annealing algorithm with those of an Adaptive Tabu Search and also with those of branch and-bound exact method. The comparison showed that SA performs good results for small-sized instances. For the large-sized instances we found that SA is not enough efficient to solve this NP-complete problem, comparing with ATS.

Keywords: vehicle routing problem, consistent vehicle routing problem with profit, meta-heuristic, simulated annealing.

ملخص

يعتبر VRP المتناسق مع الأرباح أحد أشكال مشاكل توجيه المركبات، وقد تم تقديمه في عام 2019. في هذه المشكلة ، يتم تعريف كل نموذج من خلال مستودع، مجموعة من العملاء (عملاء متكررون وعملاء غير متكررين) ومجموعة من المركبات ذات قدرة استيعاب محددة. في هذا المشكل، لكل عميل طلب ثابت، وقت خدمة وأرباح ثابتة. الهدف هو العثور على أفضل مجموعة من المسارات لمجموعة المركبات، مع مراعاة قيود التناسق، والتي تتمثل في تناسق الأشخاص، تناسق الوقت والتناسق في الكمية المسلمة. في هذا العمل ، استخدمنا خوارزمية التلدين المحاكي لحل VRP المتناسق مع الربح. لتعريف علاقة الجيرة، قمنا ببرمجة العديد من العديد من العمليات، و هي حركة Opt-2, حركة التبادل 1-1 وحركة التبادل 1-0. في التجارب الحسابية ، قمنا بمقارنة النتائج التي تم الحصول عليها بواسطة خوارزمية محاكاة الصلب مع تلك الخاصة ببحث Tabu التكيفي (ATS), وأيضًا مع تلك الخاصة بالطريقة الدقيقة الفرع والربط (B&B). أظهرت المقارنة أن SA تؤدي نتائج جيدة للحالات صغيرة الحجم. بالنسبة للحالات كبيرة الحجم، وجدنا أن SA ليست فعالة بما يكفي لحل مشكلة NP-Complete، مقارنة بـ ATS.

الكلمات الرئيسية: مشكلة توجيه السيارة ، مشكلة توجيه السيارة المتسقة مع الربح ، خوارزمية محاكاة الصلب.

Table of Contents

List of Figures

List of Tables

List of Acronyms

VRP:	Vehicle Routing Problem.
conVRP:	consistent Vehicle Routing Problem.
TSP:	Traveling Salesman Problem.
ATS:	Adaptive Tabu Search.
VRPP:	Vehicle Routing Problem with Profit.
VRPTW:	Vehicle Routing Problem with Time Windows.
CVRP:	Capacitated Vehicle Routing Problem.
OVRP:	Open Vehicle Routing Problem.
MDVRP:	Multi-depot Vehicle Routing Problem.
ACO:	Ant Colony Optimization.
GA:	Genetic Algorithm.
TS:	Tabu Search.
SA	Simulated Annealing.
EVRP	Electric-Vehicle Routing Problem.

General Introduction

The vehicle routing problem (VRP) is an extension of the travelling salesman problem (TSP), it is a problem of combinatorial optimization and operations research. In these problems falling within the domain of logistics, one or more vehicles must cover a transport network to deliver goods to customers or cover the routes of this network (Ismai, Legras, & Coppin, 2011).

The literature in recent years has flourished with customer-oriented routing problems. Mathematically, VRP is classified as an NP-hard problem, which means that the required solution time increases excessively with size. Solving VRP consists in focusing on finding the optimal paths for a fleet of vehicles. The optimal paths are the shortest and least expensive paths. Various types of consistency have been identified and discussed in the literature, such as time consistency, person consistency, quantitative consistency, etc. The primary goal of consistency constraints is customer satisfaction. The primary work in this field is that of (Groër, Golden, & Wasil, 2009).

Person consistency is important so that employees are able to learn about customer areas, in addition to that they become aware of customers' needs and requirements and create bonds with customers. Time consistency is also important, as it makes it easier to plan and organize the visited customers. Consistency in delivered quantity is also desirable, as low differences in delivery quantity facilitate warehouse management processes for clients (Stavropoulou, Repoussis, & Tarantilis, 2018).

Thus, any route or set of routes, starting and ending at a given depot, can be measured in terms of cost, profit, and time. To that end, we addresses a vehicle routing problem that aims to maximizing the total acquired profit, and to minimizing the total traveling cost when a mixed set of customers is served with consistent service constraints. This variant of VRP is called consistent Vehicle Routing Problem with Profit was firstly introduced by (Stavropoulou, Repoussis, & Tarantilis, 2018) in 2018. Is the problem addressed in this master thesis.

Objective

The vehicle routing problem is an NP-hard problem. Therefore, exact methods cannot be used to solve large-sized problems in a reasonable time. On the other hand, meta-heuristics can generate high-quality solutions in reasonable computing time. In the variant of VRP addressed in this study called is also an NP-hard problem.

Thus to solve this problem, decided to implement a meta-heuristic, which is the Simulated Annealing Algorithm. Our choice of this Mete-heuristic is justify by its ability to escape local optima and its efficiency to solve many kinds of NP-hard problems. Our main objective is to verify if SA may beat the results obtained by ATS.

Document Organization

This master thesis is organized into three chapters:

- In the first chapter, we will introduce the consistent Vehicle Routing Problem with Profit.

- In the second chapter, we will present solution methodologies for solving the Vehicle Routing Problem.

- In the third and final chapter of this thesis, we will present the method we have developed to solve this problem and the results obtained from this study. To evaluate the efficiency of this approach, we compare the results obtained by our algorithm with those found in the literature.

- We conclude with a general conclusion that summarizes the performed study and provides some perspectives.

Chapter 1: consistent Vehicle Routing Problem with Profit

I. Introduction

In the standard version of the Vehicle Routing Problem (VRP), we have a group of customers with known needs and a group of monolithic vehicles of limited capacity located on the depot. The aim is to define a set of paths to satisfy all orders at the lowest cost. The VRP was introduced in 1959, and its initial goal is to reduce the company's dependencies without considering customer satisfaction. This last point represents an important issue in a competitive market. That is why in the recent variants of VRP, more effort is made to incorporate the customer satisfaction in the model. The goal is to obtain more balanced and efficient configuration. By balanced efficient configuration, we mean solutions that implement as trade-off between the satisfaction of customer and the profit of company.

Consistent Vehicle Routing Problem with profits is one of these variants that focus more on customer satisfaction, where the goal is to maximize the company's profit while ensuring customer satisfaction. These objectives are subject to a set of consistency constraints. The purpose behind the incorporation of consistency constraints such as time consistency, person consistency and consistency in the delivered quantity is to achieve the customer comfort and satisfaction.

This chapter is organized as follows: Firstly, in section II, we start by presenting some transportation problems0, and we define the VRP and mention some of its variants. After, in section III, we define the conVRP and enumerate some of consistency constraints. In sectionIV, we present the consistent VRP with profits and its mathematical formulation2. Finally, in sectionV, we conclude the chapter.

II. Transportation Problems

Transportation problems, also known as routing problems, model real issues and problems related to the transportation of goods or people. In this section, we will first present the traveling salesman problem in order to introduce later the vehicle routing problem.

1. Traveling Salesman Problem

The Traveling Salesman Problem (TSP) was first formulated in 1930. It is one of the most intensively studied problems in optimization. In this problem, a traveling salesman wants to find the shortest and less weight way to visit all destinations. Each destination must be visited only once. Formally speaking, the problem is formulated using a graph. Solving the problem consists in finding a Hamiltonian cycle in the graph that contains all destinations (see the example in Figure 1).

Thus, we have a graph G = (V, E), and a weight function w : E → N, where :

- V: is the set of destinations, and
- E: is the set of edges.
- Each edge *(i, j)*∈ *E* has a certain weight w.

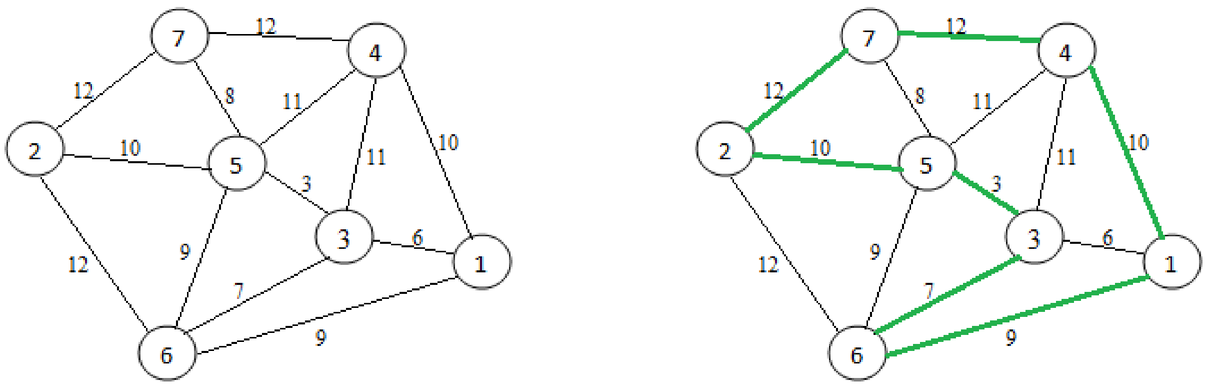


Figure 1: Example of Traveling Salesmen Problem.

More information about TSP can be found at (Floo, 1956).

2. Vehicle Routing Problem

2.1 Definition

The vehicle routing problem (VRP) was introduced by Dantzig and Ramser in 1959 (Dantzig & Ramser, 1956), it is an extension of the traveling salesman problem (TSP). The goal is to find the optimal set of paths for a group of vehicles in order to deliver orders for oil to a customers set. In 1964, the problem was generalized to a linear optimization problem known as VRP by (Clark & Wright, 1964). Solving daily vehicle routing issues allows companies to reduce operating costs and use their fleets effectivelyFigure 2: Example of vehicle routing problem. Figure 2 represents an Example of vehicle routing problem.

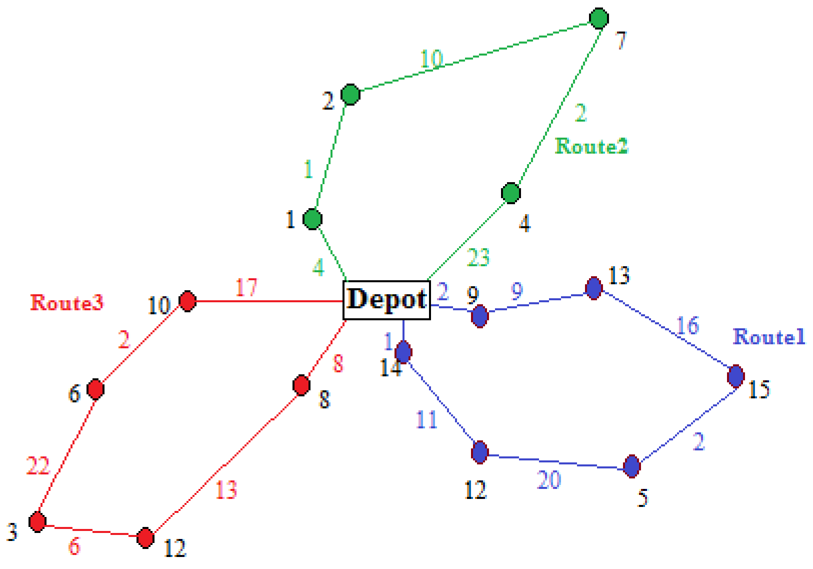


Figure 2: Example of vehicle routing problem.

2.2 Mathematical Formulation

VRP can be defined as a complete undirected graph $G = (V, A)$ (Labbé & Laporte, 1991), where:

- $V = \{1, \dots, n\}$ is the set of customers having a non-negative demand q_i, 0 represent the depot.
- A represents the edges of graph G.

The VRP formulation that we present here corresponds to the mathematical formulation used in linear integer programming. It translates the natural modeling of the problem by the definition of a binary decision variable x_{ijk}. This variable takes value 1if the vehicle k crosses the arc (i,j) (Ismai, Legras, & Coppin, 2011). In this problem, we have a fleet of vehicles, the number of vehicles available can be limited in some cases.

In order to write the mathematical model we use the following variables:n number of clients (or vertices).

- m number of vehicles.
- c_{ij} is the cost of the edge between vertices i and j (distance or travel time).
- $x_{ijk} = \begin{cases} 1\ if\ (i,j)\ is\ traveled\ by\ the\ vehicle\ k \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases}$

Thus, as an optimization problem, the VRP is written (Ismai, Legras, & Coppin, 2011):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \sum_{k=1}^{m} x_{ijk} \tag{1}$$

Subject to the following constraints:

$$\sum_{i=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \qquad \forall\ 1 \le j \le n \tag{2}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \qquad \forall\ 1 \le i \le n \tag{3}$$

$$\sum_{i=1}^{n} \sum_{l=1}^{n} x_{ilk} = \sum_{l=1}^{n} \sum_{j=1}^{n} x_{ljk} \tag{4}$$

$$\sum_{j=0}^{n} x_{0jk} = 1 \qquad \forall\ 1 \le k \le m \tag{5}$$

$$\sum_{i=0}^{n} x_{i0k} = 1 \qquad \forall\ 1 \le k \le m \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ 0 \le i,j \le n\ ;\ 1 \le k \le m \tag{7}$$

- The objective function (1) minimizes the total traveling cost.

- Constraints (2) and (3) require that each customer must be served once and only once.
- Constraints (4) ensure flow conservation.
- Constraints (5) and (6) verify that each tour begins and ends at the depot.
- Finally, the constraints (7) are binarity constraints on the decision variables x_{ijk}.

2.3 *Some Variants of VRP*

In practice, the basic vehicle routing problem is extended with constraints, such as, the permissible capacity of the vehicle, route length, time of arrival departure and repair, time of assortment, and delivery of products. So depending on the type of necessities, there are many variants of VRP (see Figure 3).

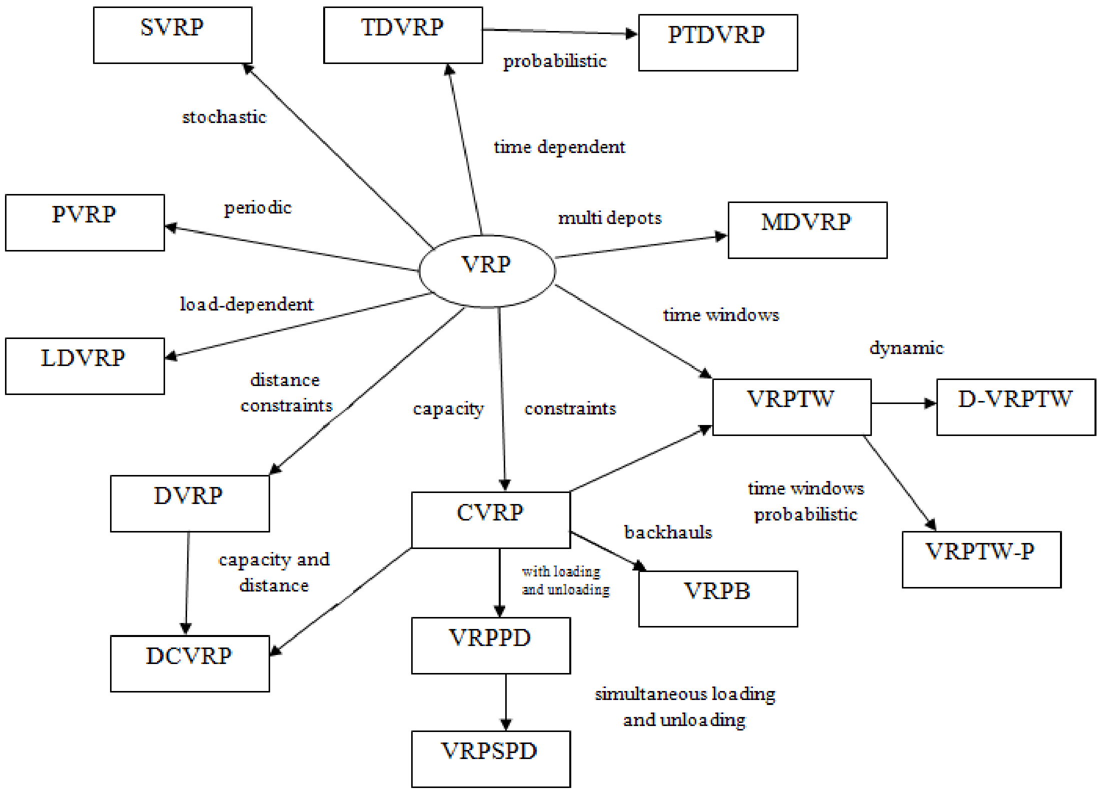


Figure 3: Hierarchy of VRP variants (Režnar, Martinovič, Slaninová, Grakova, & Vondrák, 2017).

We can mention the following variants:

a. Vehicle Routing Problem with Profits (VRPP)

The problem of maximization seeks to increase the profit, as visiting all customers is not mandatory. The goal is to visit, as much as possible, customers in order to increase the total collected profits while respecting the time specified for the vehicles. Vehicles must start and end at the depot.

b. Vehicle Routing Problem with Time Windows (VRPTW)

The delivery locations have time windows within which the deliveries (or visits) must be made. These time windows limit the times at which a customer is available to receive a delivery. VRPTW is NP-hard, and even finding a feasible solution to VRPTW is an NP-hard problem. A review of research work related to VRPTW can be found in (Cordeau, Gendreau, Laporte, & Potvin..., 2002), (Kallehauge, 2008).

c. Capacitated Vehicle Routing Problem (CVRP)

Vehicles have a limited carrying capacity of the goods to be delivered. A review of research work related to CVRP can be found in (Mazzeo & Loiseau, 2004).

d. Open Vehicle Routing Problem (OVRP)

The vehicles either are not required to return to the depot, or they have to return by revisiting the customers assigned to them in the reverse order. Therefore, the vehicle routes are not closed paths but open ones.

e. Multi-Depot Vehicle Routing Problem (MDVRP)

In this variant of the problem, there are multiple depots from which vehicles can start and finish.

In this master thesis, we are interested with a variant of VRP called “ConVRP with profit”. Thus, we reserve the next section for exposing more this variant.

III. Consistent Vehicle Routing Problem (conVRP)

The main objective of Consistent Vehicle Routing Problem (conVRP) is to provide consistent service and ensure customer satisfaction by imposing some consistency constraints, such as time constraints, vehicle capacity constraints, road duration constraints, etc. (Barros, Linfati, & Escobar, 2020).

1. Definition

In the standard version of (VRP), the goal is to visit a group of customers in order to deliver their known requests, each customer is visited only once during the day by one vehicle, which has a specific capacity. A road is created for each vehicle, and reduces the total distance traveled by the fleet. But, in conVRP there is a group of customers who must be served over a planning horizon, i.e. over a set of days, where the customer is visited only once a day by a vehicle. The same vehicle delivers orders for the rest of the planning horizon days. The visit of the same customer must be performed within a specified time range, not exceeding L time units. In addition, the time taken by the vehicle to complete the road should not exceed T units of time. The goal is to find a set of routes for a group of vehicles that reduce the operating time of the vehicles over several days. Also, we point out that the fleet is homogeneous, where all the vehicles having the same capacity and ability to serve any customer (Groër, Golden, & Wasil, 2009; Groër, Golden, & Wasil, 2009).

3. Consistency constraints

In general, customer satisfaction is directly related to providing a consistent service. Service consistency has three different dimensions: Time consistency, Person consistency and Consistency in delivered quantity.

3.1 Time Consistency

Time consistency ensures that clients are visited at the same time or within a specified time range throughout the days of the planning horizon. Time consistency is important, as it makes it easier to plan and organize the visited customers (Talarico & Duque, 2015)

3.2 Person Consistency

Person consistency verify that orders are delivered to customers by the same driver, throughout the days of the planning horizon, making drivers aware of customer locations. In addition, to being aware of customer needs and requirements, creating links with customers and providing good service. Makes the customers feel more comfortable by dealing with the same drivers.

3.3 Consistency in Delivered Quantity

Consistency in the delivered quantity, means that the customer’s order must be delivered in one go and not in parts. But we also should verify that the load should not exceed the vehicle’s carrying capacity.

The importance of each of these three dimensions of consistency changes from market to market.

IV. Consistent Vehicle Routing Problem with profit (conVRP with profits)

1. Definition

Consistent VRP with profits is one of vehicle routing problems, was introduced by (Stavropoulou, Repoussis, & Tarantilis, 2018) in 2018 it is the problem addressed in this master thesis. In this problem, we have: the warehouse or the depot where the goods that will be transported to customers are located. A group of vehicles with a specific capacity that used to serve customers with the ordered goods. They start from the depot and return to it. A group of customers. The customer set is divided into two subsets, the set of frequent customers or regular customers, and the set of non-frequent customers or on the spot customers. Frequent customers are the clients who request the service more than once during the planning horizon, they must to be visited in all days they requested the service. Non-frequent customers only request the service once during the planning horizon, they need to be visited at most once on an ad hoc basis, so that the customers who contribute to the maximization of profit are selected. In this problem variant each customer has a demand, service time, and profit.

The goal is to find the best set of paths for the group of vehicles, while respecting the constraints of consistency. Good paths are those that include all frequent customers that must be served, and include the largest possible number of non-frequent customers. It is worth noting that there are cases where adding a non-frequent customer decreases the objective function. There are two categories of non-frequent customers that decrease the objective value, (i) the ones with an estimated profit equal to zero and (ii) the ones whose estimated profit is greater than zero, but when including them in the initial solution the required traveling cost is greater than their estimated profit (Stavropoulou, Repoussis, & Tarantilis, 2018). Figure 4 represent the impact of change vehicle move on the assignment of frequent customers for vehicles.

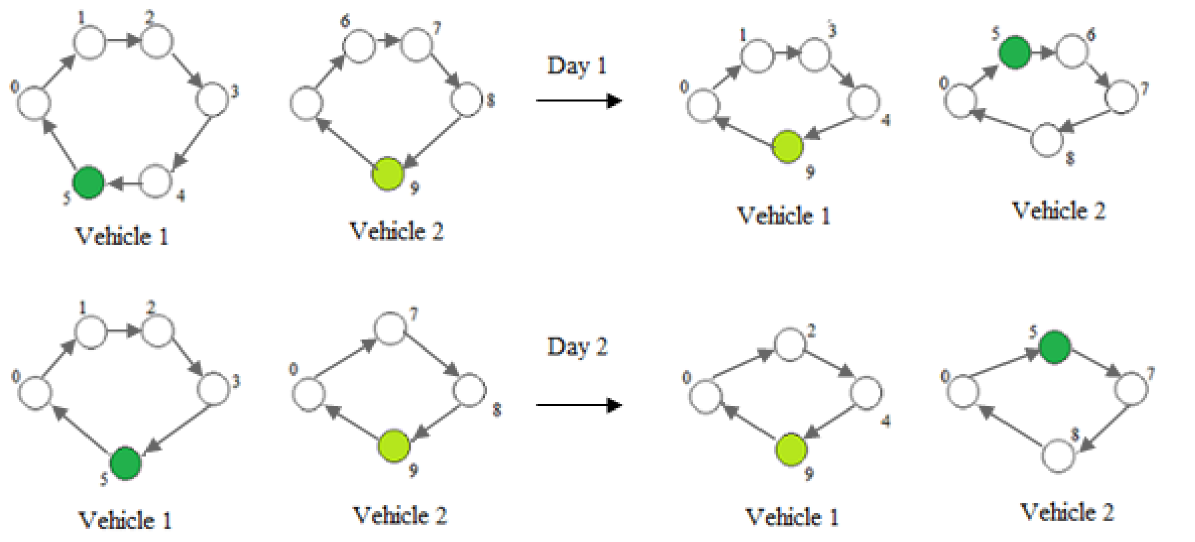


Figure 4: Change vehicle move - affecting the assignment of frequent customers to vehicles (Stavropoulou, Repoussis, & Tarantilis, 2018).

2. Mathematical Formulation

conVRP with profit can be defined on a complete undirected graph $G = (N, E)$. Where:

$N = \{0,1,2, \dots, n\}$ is the node set, and

$E = \{(i, j) | i, j \in N, i \neq j\}$ is the edge set.

The mathematical model for conVRP is introduced by stavropoulou (Stavropoulou, Repoussis, & Tarantilis, 2018). In order to write the mathematical model for conVRP with profit, the following variables are used:

- $N_c = N\backslash\{0\}$ is the set of customers, where 0 is the depot. N_c can be divided into two disjoint subsets N_f, N_{nf} based on the customers' service requirements.
    - N_f is the set of frequent customers.
    - N_{nf} is the set of non-frequent customers. N_{nf} can be divided into two disjoint subsets N_s and N_u.
        - N_s is the set of serviced non-frequent customers.
        - N_u is the set of non-serviced non-frequent customers.
- c_{ij} is a non-negative cost associated with each edge $(i,j) \in E$, while the corresponding travel cost matrix $[c_{ij}]$ is symmetric, i.e. $c_{ij} = c_{ji}$.
- Q is the maximum carrying capacity of a vehicle.
- T is the maximum available time units for a vehicle.
- $P = \{1, \dots, h\}$ is a set of periods, it represents the horizon of the tour.
- r_{ip} is a specific service requirement for each customer $i \in N_c$ on period $p \in P$, so:
- $r_{ip} = \begin{cases} 1 & \text{if customer } i \text{ requires service on period } p \\ 0 & \text{otherwise} \end{cases}$
- Each customer $i \in N_c$ has:
    - a predefined profit g_{ip},
    - a service time s_{ip}, and
    - a non-zero demand q_{ip} ($0 < q_{ip} \leq Q$).
- $E_p = \{(i,j) \in E, r_{ip}r_{jp} = 1\}$ is a reduced set of edges for each period $p \in P$.
- $V_p = \{i \in V_c, r_{ip} = 1\}$ is a reduced set of customers, for each period $p \in P$. It represents customers that require service in day p.

The goal behind solving this problem is to determine the subset of customers to be served and the corresponding order of visits, so as to maximize the difference between the total collected profit minus the total travel cost, and to satisfy the following constraints (Stavropoulou, Repoussis, & Tarantilis, 2018):

- The start/end of each vehicle route must be from/at the depot.
- The carrying load of each vehicle $k \in K$ in each day does not exceed the total carrying capacity Q.
- Each vehicle route lasts at most T time units.

- Each frequent customer $i \in N_f$ must be visited only once by the same vehicle $k \in K$ on each period $p \in P$ when they need to be served (or each profit g_{ip} of customer $i \in N_f$ is collected only once).
- Each non-frequent customer $i \in N_{nf}$ can be visited by one vehicle $k \in K$ at most once during the time period p ∈ P when they need to be served (each profit g_{ip} of customer $i \in N_s$ can be collected at most once).
- The maximum difference between the earliest and latest vehicle arrival times to a *frequent customer* $i \in N_f$ over the planning horizon does not exceed L time units
- Within the planning horizon, the maximum difference between the earliest and latest vehicles arriving *frequent customer* $i \in N_f$ does not exceed L time unit.

Three groups of variables associated with the customer visiting sequence, the customers' assignment to vehicle routes and the vehicle arrival time to serviced customers are used (Stavropoulou, Repoussis, & Tarantilis, 2018). Let:

- binary variables x_{ijp} count the number of times edge $(i,j) \in E$ is traversed on period p,
- binary variables y_{ik} indicate if customer i is serviced by vehicle k, and
- continuous variables a_{ip} depict the arrival time to customer i on period p in the optimal solution.

Considering the above notations, ConVRP with profits can be mathematically described as follows:

$$max \sum_{i\in Nc}\sum_{k\in K}\sum_{p\in P} y_{ik}\, g_{ip} - (\sum_{p\in P}\sum_{(i,j)\in E} x_{ijp}\, c_{ij} + \sum_{p\in P}\sum_{k\in K}\sum_{i\in Nc} y_{ik}\, s_{ip}) \qquad \textbf{(8)}$$

Subject to:

$$\sum_{k\in K} \boldsymbol{y_{ik} = r_{ip} \quad \forall\, p \in P, i \in N_f} \qquad (9)$$

$$\sum_{k\in K} \boldsymbol{y_{ik} < r_{ip} \quad \forall\, p \in P, i \in N_{nf}} \qquad \textbf{(10)}$$

$$\sum_{j\in Nc} \boldsymbol{x_{0jp}} = \sum_{i\in Nc} \boldsymbol{x_{i0p} = K \quad \forall p \in P} \qquad \textbf{(11)}$$

$$\sum_{i\in N} x_{ijp} = \sum_{j\in N} x_{jip} = \sum_{k\in K} y_{ik} \quad \forall i \in N(i \neq j), p \in P \tag{12}$$

$$\sum_{i\in Vp} q_{ip}\, y_{ik} \leq Q \quad \forall k \in K, p \in P \tag{13}$$

$$1 - x_{ijp} - x_{jip} \geq y_{ik} - y_{jk} \quad \forall (i,j) \in V_p \times V_p : i \neq j, k \in K, p \in P \tag{14}$$

$$a_{0p} = 0 \quad \forall p \in P \tag{15}$$

$$a_{ip} + x_{ijp}(s_{ip} + c_{ij}) - (1 - x_{ijp})T \leq a_{jp} \quad \forall (i,j) \in V_c \times V_c : i \neq j, p \in P \tag{16}$$

$$a_{ip} + r_{ip}(s_{ip} + c_{i0}) \leq r_{ip}T \quad \forall i \in V_c, p \in P \tag{17}$$

$$a_{ip} - a_{ip'} \leq L \quad \forall i \in V_p \cap V_{p'}, p \in P, p' \in P : p \neq P' \tag{18}$$

$$y_{0k} = 1 \quad \forall k \in K \tag{19}$$

$$y_{ik}, x_{ijp} \in \{0,1\}, a_{ip} \geq 0 \quad \forall i,j \in N, k \in K \tag{20}$$

$$r_{ip}c_{0i} \leq a_{ip} \leq T - s_{ip} - c_{i0} \quad \forall i \in N_f, p \in P \tag{21}$$

- The objective function (8) maximizes the net collected profit, i.e. the difference between the total collected profit and the total traveling cost.
- Constraints (9) and (10) verify that frequent customers must be visited on each period they require service while the non-frequent customers must be visited at most once on the period they require service.
- Constraints (11) impose that the K vehicles leave and return to the depot in every period.
- Constraints (12) ensure route connectivity.
- Constraints (13) are capacity restrictions for each vehicle.
- Constraints (14) show that each frequent customer must be serviced by the same vehicle.
- Constraints (15), (16) and (17) calculate the arrival times to the depot and to each serviced customer on each period, and impose that it must be less than T.
- Constraints (18) ensure that the maximum difference between the earliest and latest arrival time to each frequent customer does not exceed a predefined threshold L.
- Constraints (19) impose that all vehicle routes start from the depot on every period of the planning horizon.
- Finally, the last sets of constraints impose binary conditions to x and y variables as well as lower and upper bounds for the continuous variables.

V. Conclusion

In this chapter, we have presented in the first part the travelling salesman problem (TSP). Next, we have exposed VRP, its variants, and its mathematical formulation. After we presented consistent VRP and types of consistency. Then, we presented conVRP with profit and it mathematical formulation. In the next chapter, we will show some exact methods and meta-heuristics as methods to solve the problem.

Our problem belongs to the NP-hard class. The problems of this class are algorithmically solvable but computationally intractable. There is no exact method that can find the optimal global solutions to NP-hard problems in polynomial time. Fast approximate heuristics and meta-heuristics are the popular approaches to search for practical solutions. In our study, we will use the simulated annealing, which is often used to solve complex large-scale optimization problems.

Chapter 2: Solution Methods for Vehicle Routing Problem

I. Introduction

In the literature, there are many strategies for solving VRP problems, such as exact methods and approximation methods. Exact methods provide the optimal solution, i.e. the best solution, but they are generally used to solve small problems, because despite their ability to find the optimal solution, they are not suitable for large-sized problems (NP-hard problems), because it takes a long and unacceptable time to reach the optimal solution. While there is no limit to the scale of problems that can be solved by meta-heuristics, it produces close optimal solutions and can handle a large number of constraints efficiently.

This chapter is organized as follows: Firstly, in section II, we start by presenting some solution methodologies for VRP, where we present, exact methods and meta-heuristics and enumerate some of its advantages and drawbacks, and we mention some recent works on solving VRP using exact methods and meta-heuristics. After, in section III, we present the simulated annealing algorithm2. Finally, in section IV, we conclude the chapter.

II. Solution Methodologies for VRP

A variety of VRP solution strategies are provided in the literature (see Figure 5). These range from exact methods to the approximate solution methods. Although the Exact methods provide the best solution, the approximation method commonly referred to as meta-heuristics usually produces a near-optimal solution.

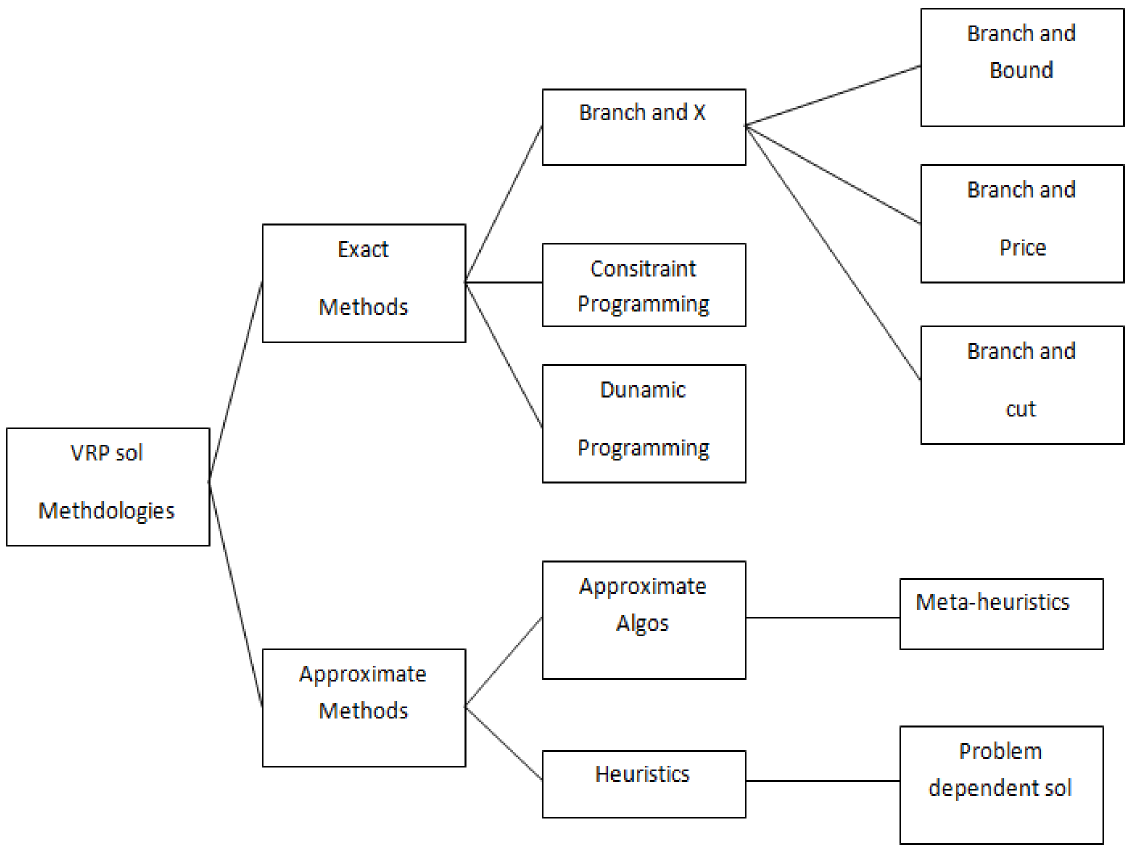


Figure 5: Various approaches to solve VRP.

1. Exact Methods

Exact methods (also called complete) are generally based on a tree search and partial enumeration of the solution space. Exact methods obtain optimal solutions and guarantee their optimality (Manen, November 2019). It is a method for solving a problem, and it is able to find the optimal solution and ensure its optimality, but it is not suitable for NP-hard problems, unless the problem size is small. In combinatorial optimization, the exact methods are mostly used when the number of customers in particular problems is relatively low and thus the optimal solution can be found in a reasonable time. For example, if we want to solve the traveling salesman problem, we end up with n! possible combinations. So, looking for a Hamiltonian path of 3 cities gives $3! = 6$ or 4 cities gives $4! = 24$ possible combinations, while solving the same problem with 10 cities requires $10! = 3,628,800$ possible combinations.

Among the most famous exact algorithms used to solve vehicle routing problems is branch-and-bound.

The branch-and-bound method is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. It is based on a tree-like method in which branches that do not generate better solutions are cut down. All separations are permitted provided that no information is lost (Thanina & Katrin, 2017). The idea of this method is divide to conquer, as well as to use optimal cost boundaries to avoid exploring parts of the set of acceptable solutions. Where the main problem is divided into a group of sub-problems. Sub-problems can be as difficult as the original problem, in this case, we apply the same system, we partition the sub-problem (s) as shown in Figure 6, This stage is called the separation phase.

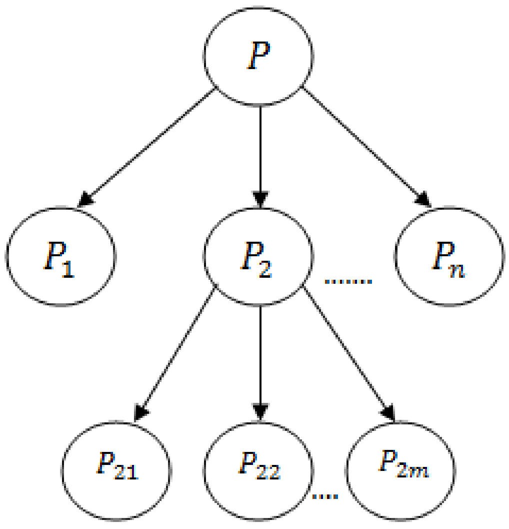


Figure 6: Representation of the problem's division into sub-problems.

The nodes of the search tree are evaluated to determine the best set of possible solutions associated with the node, or to prove that this set does not contain an optimal solution, so it is not necessary to check its solution space but is rather separate. At a given node, the optimum of the sub problem can be determined when the sub problem becomes "sufficiently simple", this stage is called the evaluation phase.

The most general method to determine that a set of feasible solutions does not contain an optimal solution, consists in determining a lower bound (upper bound, if it is a maximization problem) of the cost of the solutions contained in the set. If we manage to find a lower bound (upper bound) which is greater (smaller) than the cost of the best solution found so far, then we have the assurance that the subset does not contain the optimum. The most classic techniques for calculating bound are based on the idea of relaxation of certain constraints.

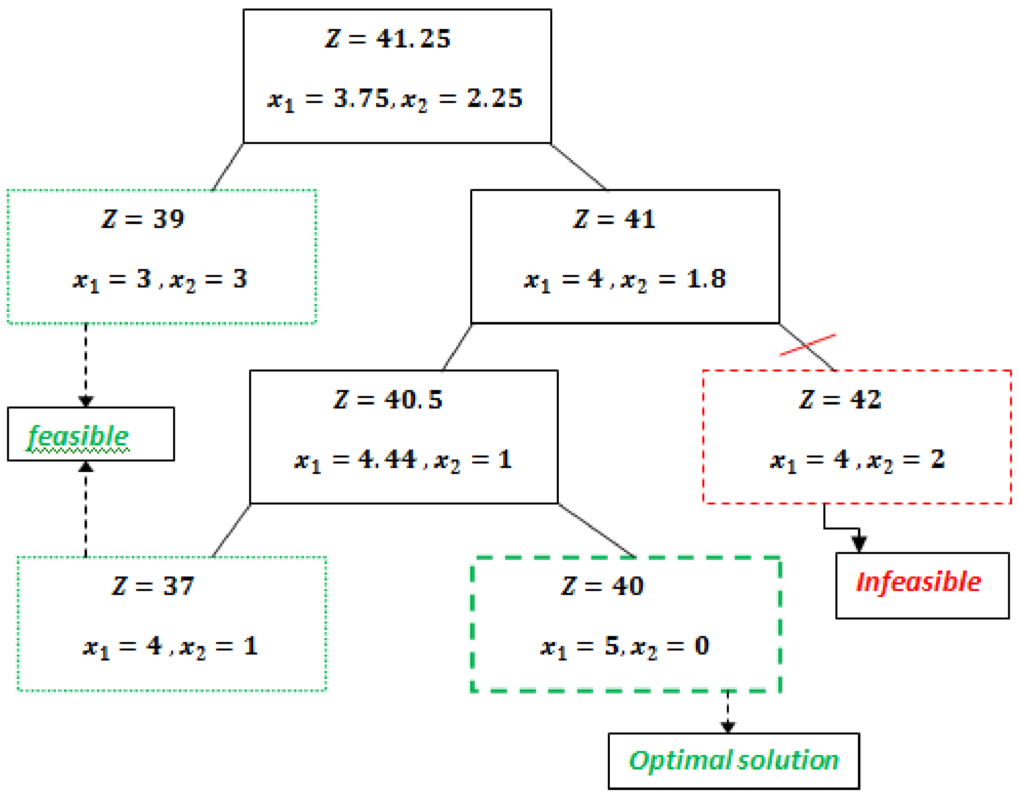


Figure 7: Solution for the problem (p) using the branch and bound method.

Consider the following problem:

$$\max Z = 3x_1 + 5x_2$$

Subject to:

$$(P)\quad \begin{cases} 9x_1 + 5x_2 \leq 45 \\ x_1 + 5x_2 \leq 6 \\ x_1, x_2 \geq 0, \quad and \;\; x_1, x_2 \; integers \end{cases}$$

The solution of this problem using the branch and bound method is represented in Figure 7.

So, the optimal solution of problem (p), is $Z = 40$ where $x_1 = 5, x_2 = 0$.

1.1 Advantages of Exact Methods

Although exact algorithms may not be the most suitable option for solving some problems, they have some advantages that sometimes make them useful and required, we can mention the fallowing advantages:

- Confidence of finding the optimal solution.
- Valuable information on the upper/lower bounds to the optimal solution are obtained even if the algorithm is stopped before completion (Dumitrescu & Stützle, 2003) .
- Methods allow pruning parts of the search space in which optimal solutions cannot be located (Dumitrescu & Stützle, 2003).

1.2 Drawbacks of Exact Methods

Most exact algorithms solve small-sized problems, the larger the problem size is, the larger the execution time will be. Although, these algorithms have some advantages that we mentioned earlier, they have some disadvantages. We can mention the fallowing drawbacks (Dumitrescu & Stützle, 2003):

- They take a lot of time (Prohibitive execution time).
- The memory consumption of exact algorithms may lead to the early abortion of a programme.
- They are often difficult to extend high-performance exact algorithms for a single problem, if some details of the problem formulation change.
- For many combinatorial problems the best performing algorithms are highly problem specific and that they require large development times by experts on integer programming.

1.3 Recent Works on Solving VRP Variants Using Exact Methods

In recent years, several exact algorithms have been proposed for the vehicle routing problem and its variants. In this section, we will mention some recent works on solving VRP variants using exact methods.

In (Blocho, 2020), the most important and exact methods for solving various variants of VRP problems are described. The authors proposed a new precise branch-price-and-cut algorithm has been developed to solve cases where there are few customers and many vehicles per route for the VRPSD problem. In (Lee & Chungmok, 2021), a branch-and-price algorithm is introduced to solve the EVRP problem. Its goal is to minimize the total travel and charging times without approximating the charging time function on the expanded charging station network. In (Florio, Hartl, & Mi, 2020), a branch-price-and-cut exact algorithm relies on an efficient labeling procedure, exact and heuristic dominance rules, and completion bounds to price profitable columns, is developed. The algorithm proved its efficiency to solve many instances with many vehicles and few customers per route. In (Hebler, 2021), a branch-and-cut algorithm based on a three-index formulation for the multi-compartment vehicle routing problem with flexible compartment sizes was modified, and an exact solution approach is introduced that is tailored to the continuous problem variant. Moreover, two other exact solution approaches are proposed, a branch-and-cut algorithm based on a two-index formulation and a branch-price-and-cut algorithm based on a route-indexed formulation.

2 Meta-heuristics

2.1 Definition

In the early years, specialized inference methods were usually developed to solve complex combinatorial optimization problems, such as VRP. The term Meta-heuristics was introduced in 1986 by Fred Glover to refer to heuristics with a higher level of abstraction (Pedemonte & Martín, 2017).

A meta-heuristic is an optimization algorithm that is generally recursive stochastic algorithms, and approaches the global maximum, i.e. the global optimum of the function, by sampling an objective function. Meta-heuristic aims to solve difficult problems (often from the fields of operations research, engineering or artificial intelligence), that require a very large amount of time if they are solved by effective and exact methods (Amraoui, Mhamdi, & Elloumi, 2017). Despite the popularity of meta-heuristics, there is no agreed upon definition of heuristics and meta-heuristics in the literature (Gandomi, Yang, Talatahari, & Alavi, 2013).

2.2 Examples of Meta-heuristics

In literature many meta-heuristic algorithms are exist. We will mention the most popular meta-heuristic algorithms for optimization and for solving VRPs.

a. Ant Colony Optimization

Ant Colony Optimization (ACO) is a recent meta-heuristic approach for solving hard combinatorial optimization problems (Dorigo & Stützle, 2001). The idea of (ACO) is derived from the way the real ants live, and the way they discover the shortest path to reach food, where the ants secrete a substance called pheromone through which they can determine the paths, where the concentration of pheromone rises in the shortest path, the ants follow the paths with the highest concentration of pheromone. As shown in Figure 8Figure 8. ACO is based on the indirect communication of a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails (Dorigo & Stützle, 2001).

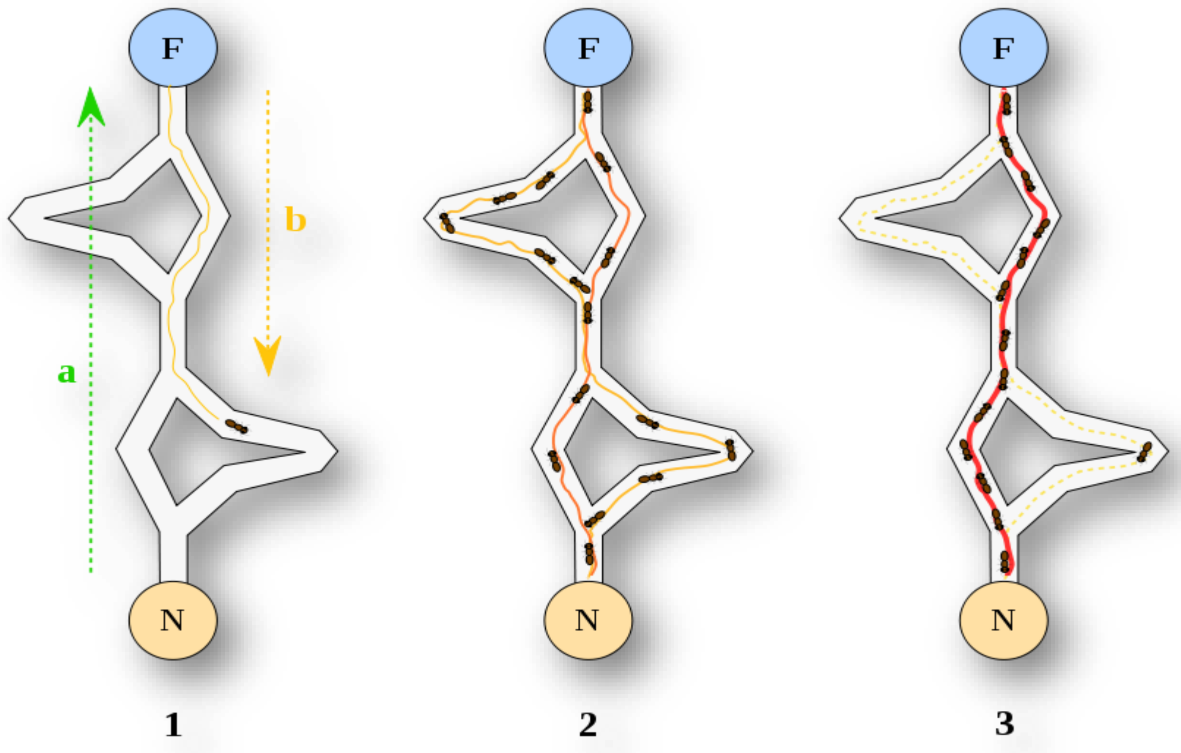


Figure 8: Ant Colony Optimization Algorithm (Neural Networks Tutorial:Model selection).

The process of finding the shortest path is divided into these steps **(Müller, 2009)**:

1) The first ant finds the food source (F), via any path (a), and then returns to the nest (N) leaving behind a pheromone trail (b).

2) The ants take the four possible paths indifferently, but the reinforcement of the track makes the shorter path more attractive.

3) The ants take the shortest path; the long portions of the other paths lose their trail of pheromones.

There are many experimental works on VRP resolution using ACO. In (Wang, Wang, Chen, Cai, Zhou, & Xing, 2020) a heuristic algorithm based on Improved Ant Colony Optimization (IACO) and Simulate Annealing (SA) called Multi Objective Simulate Annealing - Ant Colony Optimization (MOSA-ACO) is developed to solve the Vehicle Routing Problem with Time Window and Service Choice, Computation experiment results showed that MOSA-ACO algorithm has a good performance on solving (VRPTW). In (Mutar, Burhanuddin, Hameed, Yusof, & Mutashar, 2020) an improvement of Ant Colony System (ACS) is presented to solve the Capacitated Vehicle Routing Problem (CVRP), and better results were obtained compared with the results of other methods. In (Jia, Mei, & Zhang, 2021) a new bi-level ant colony optimization algorithm is proposed to process CEVRP.

b. Genetic Algorithm

A genetic algorithm (GA) is a research heuristic inspired by Charles Darwin's theory of natural evolution, and developed by John Holland and his collaborators in the 1960s and 1970s (Yang, 2021). As in the process of natural selection, in this algorithm, the most appropriate solutions are selected from a population, for reproduction in order to produce the next generation offspring that bear the characteristics of the parents. This process keeps repeating itself and in the end a generation with the fittest individuals will be found.

There are five phases in a genetic algorithm, initial population, fitness function, selection, crossover, and mutation (see Figure 9).

There are many experimental works on solving VRP using this GA. In (Samsuddin, Othman, & Yusuf, 2019) some single and population-based meta-heuristics have been reviewed, that were used from 2013 to 2018 to solve MDVRP, in which were discussed genetic algorithm (GA), simulated annealing (SA), variable neighborhood search (VNS), ant colony algorithm (ACO), and particle swarm optimization (PSO).

In (Yahyaoui, Kaabachi, Krichen, & Dekdouk, 2020) two algorithms (meta-heuristics) are proposed, an adaptive variable neighborhood search (AVNS) and a Partially Matched

Crossover PMX-based Genetic Algorithm to solve the multi-compartment vehicle routing problem in order to ensure better quality of the solution.

A waiting strategy was proposed for the vehicle routing problem with simultaneous pickup and delivery, and a genetic algorithm was used to solve the problem (Park, Son, Koo, & Jeong, 2021).

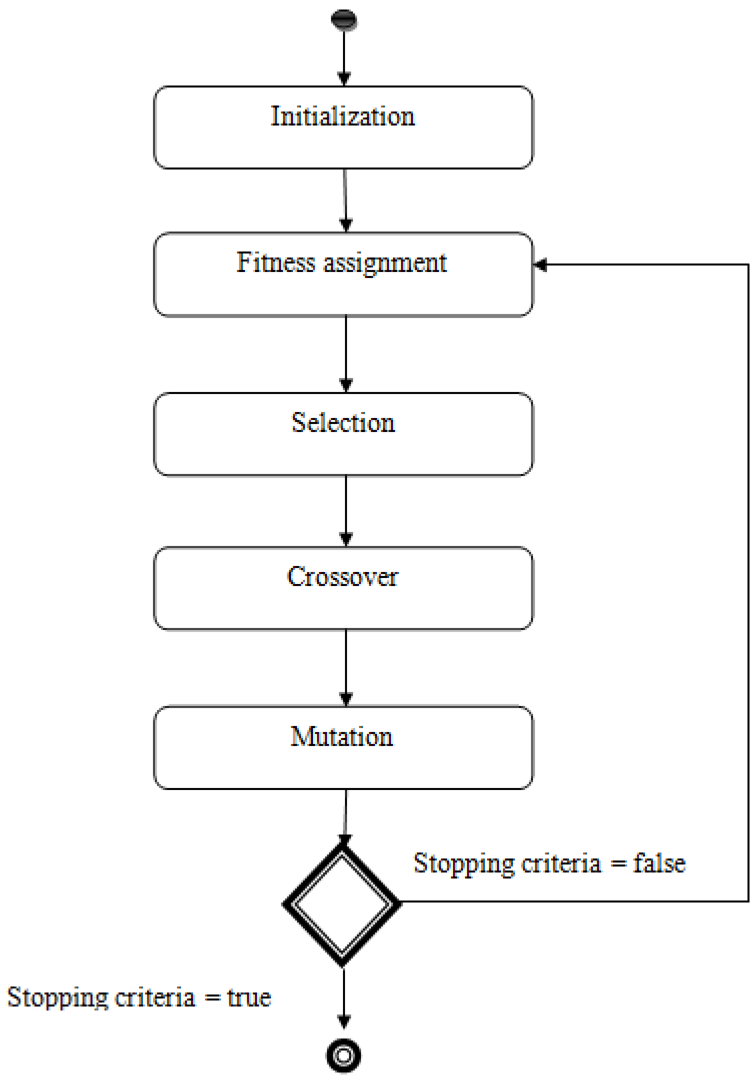


Figure 9: Genetic Algorithm phases (Yang, 2021).

c. Tabu Search

TS is a meta-heuristic search method employing local search methods for mathematical optimization. It was created by Fred W. Glover in 1986 and formalized in 1989. The basic idea of TS is to penalize moves that take the solution into previously visited search spaces (also known as tabu). TS, however, does deterministically accept non-improving solutions in order to prevent getting stuck in local optimums.

The short term memory, being a key component of the TS algorithm, prevents cycling and revisiting the same solutions during the local search, while the long-term memory records the “good” solution characteristics obtained during the execution of the TS algorithm (Stavropoulou, Repoussis, & Tarantilis, 2018). The construction of the tabu list is based on the FIFO principle, i.e. first in is first out.

Termination Criteria is dependent upon the problem at hand but some possible examples are: setting a maximum number of iterations without improving x^*, or setting a time limit after which the search should stop. Figure 10 represent a flowchart of TS algorithm.

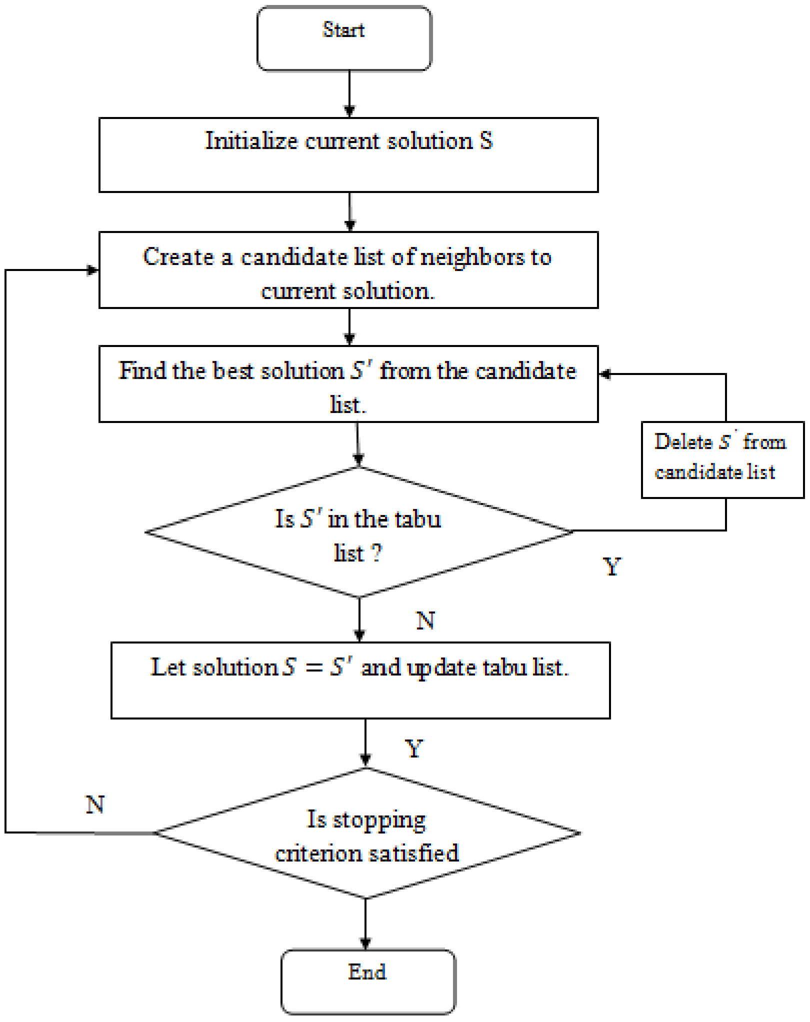

Start
Initialize current solution S
Create a candidate list of neighbors to current solution.
Find the best solution S' from the candidate list.
Delete S' from candidate list
Is S' in the tabu list ?
Y
N
Let solution $S = S'$ and update tabu list.
Y
N
Is stopping criterion satisfied
End

Figure 10: Flowchart of TS algorithm (Wang Z.).

There are many experimental work on solving VRP using this meta-heuristics. We mention some recent works. In **(Gmira, Gendreau, Lodi, & Potvin, 2021)** a TS heuristic were proposed to solve the time-dependent vehicle routing problem with time windows on a road network, the performance of algorithm were assessed by comparing it to an exact method on a set of benchmark instances. In **(Rajabi-Bahaabadi, Shariat-Mohaymany, Babaei, & Vigo, 2021)** a Max–Min ant colony system were hybridized with a TS algorithm to solve the vehicle routing problem in stochastic networks with correlated travel times. In **(Xia & Fu, 2018)** an adaptive TS algorithm were designed to solve the capacitated open vehicle routing problem with split deliveries by order, where the comparison with other methods in the literature shown the effectiveness of the algorithm.

2.3 Advantages of Meta-heuristics

Meta-heuristics are often considered the most appropriate choice for solving problems. They have some advantages that make them effective and satisfactory. We will mention some of them below (Dumitrescu & Stützle, 2003):

- It is able to find a good solution (not necessarily an optimal solution) within a reasonable time.
- In practice they are found to be the best performing algorithms for a large number of problems.
- They can examine an enormous number of possible solutions in short computation time.
- They are often more easily adapted to variants of problems and, thus, are more flexible.
- They are typically easier to understand and implement than exact methods.

2.4 Drawbacks of Meta-heuristics

Meta-heuristics solves problems of a large size, and gives results close to the optimal solution, in a short time, but it has some drawbacks or disadvantages, we will mention some of them below (Dumitrescu & Stützle, 2003):

- They cannot prove optimality.
- They cannot provably reduce the search space.
- They do not have well defined stopping criteria.

- They often have problems with highly constrained problems where feasible areas of the solution space are disconnected.
- In practically, most meta-heuristics often require considerable programming efforts, although usually less than for exact algorithms.

III. Simulated Annealing

Simulated annealing (SA) is the oldest meta-heuristic. It was introduced in 1983 by Kirkpatrick et al (Kirkpatrick & Vecchi, 1983). It is a "meta-heuristic" inspired from the physical process of metals, which consists of heating the metal at a high enough temperature, which leads to the movement of the metal atoms in an irregular way. Thus, as many formations as possible are explored, then the metal is gradually cooled which makes the atoms gradually reduce their movement and settle in Static positions forms a natural crystal structure with the least amount of energy, in which case the ductility improves and the metal becomes easier to work with. Scientists call the slow cooling process the annealing process, and It must be contrasted with the quenching process, which consists of the rapid cooling of a metal (Chopard & Tomassini, 2018). Figure 11 represents a flowchart of simulated annealing algorithm.

The algorithm starts with the initialization of SA parameters and the creating of an Initial solution, and the initialization of the current solution and best solution. then the main loop of the algorithm starts by the creating of a neighbor solution based on the current solution. Then the neighbor solution is accepted if it is better the the current one, else the neighbor solution is accepted by a probability. After that the updateing of the temperateur. The process is done until at least one of the termination criterions are met.

At the best of our knowledge, SA did not use before to solve the conVRP. Thus, we decided to implement the simulated annealing to solve the conVRP with profit in order to investigate and evaluate its efficiency to solve this variant of vehicle routing problem.

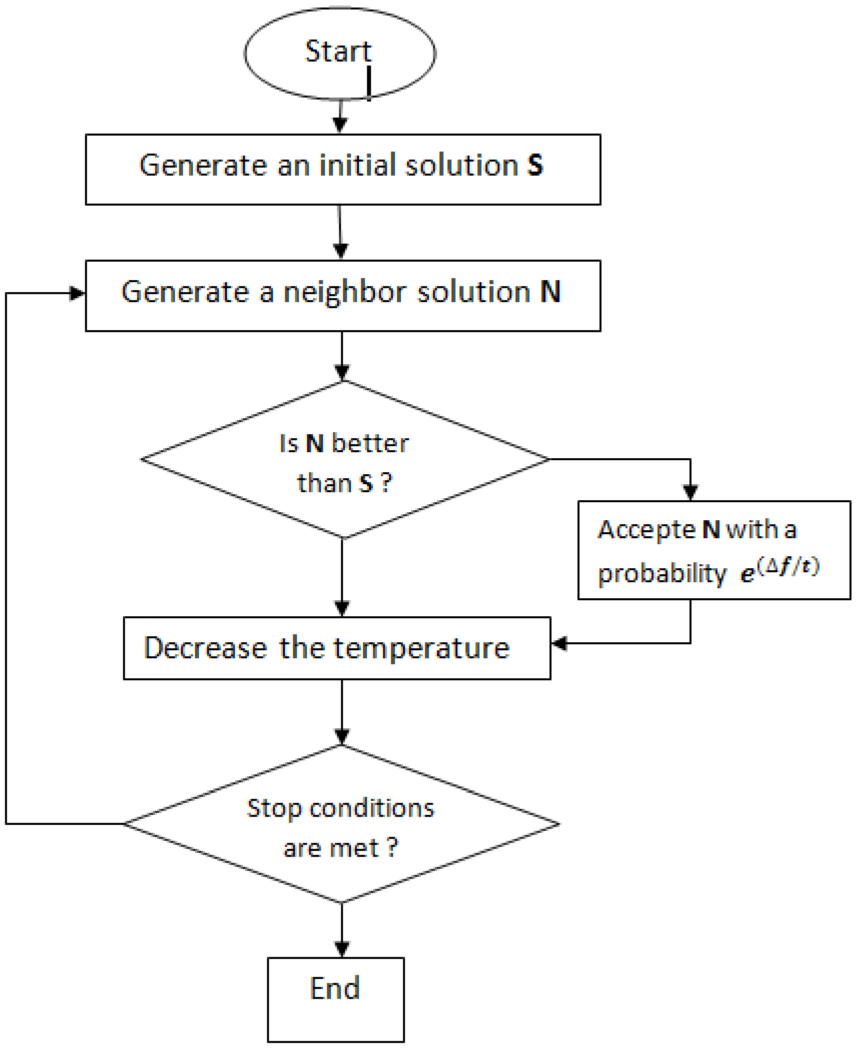

Start
Generate an initial solution **S**
Generate a neighbor solution **N**
Is **N** better than **S** ?
Accepte **N** with a probability $e^{(\Delta f/t)}$
Decrease the temperature
Stop conditions are met ?
End

Figure 11 : Flowchart of simulated annealing algorithm.

IV. Conclusion

In this chapter, we have presented some types of methods used to solve the VRP. We have also mention some works done to solve the different variants of VRP. Lather we will show our developed algorithm which based on simulated annealing, in order to solve the conVRP with profit.

In the next chapter, we will apply the Simulated Annealing Algorithm to the problem already described in chapter 1, which is the consistent Vehicle Routing Problem with profit.

Chapter 3: Our Approach to solve the conVRP with Profit.

I. Introduction

In the previous chapters, we have defined the conVRP with profit, and we have given an overview of the SA meta-heuristic. In this chapter, we will show how we applied the simulated annealing algorithm to solve the conVRP with Profit. Then, we compare the results of our method with other methods, namely CPLEX, and ATS. Thus, this chapter is organized as follow:

In section II, we present the adopted representation and evaluation of the solution. In section III, we detail the solution approach containing a description of the proposed SA. In section IV, we exhibit computational results. In section VV, we show the application's interface and instances. Finally, we conclude in section VII.

II. Solution Representation and Evaluation

1. Solution Representation

In this work, the solution is represented by a matrix, whose size is $P \times K$, where P is the number of days (periods) of the planning horizon (the lines of the matrix represent the days), while k is the number of available vehicles (the columns of the matrix represent the vehicles). Each cell of the matrix represents a route. The route is represented by a list, which starts and ends at the depot (the depot is denoted by 0). In addition, the cell (p, k) contains the list of customers who will be visited in period p by vehicle k. A schema that represents this structure is shown in Figure 12.

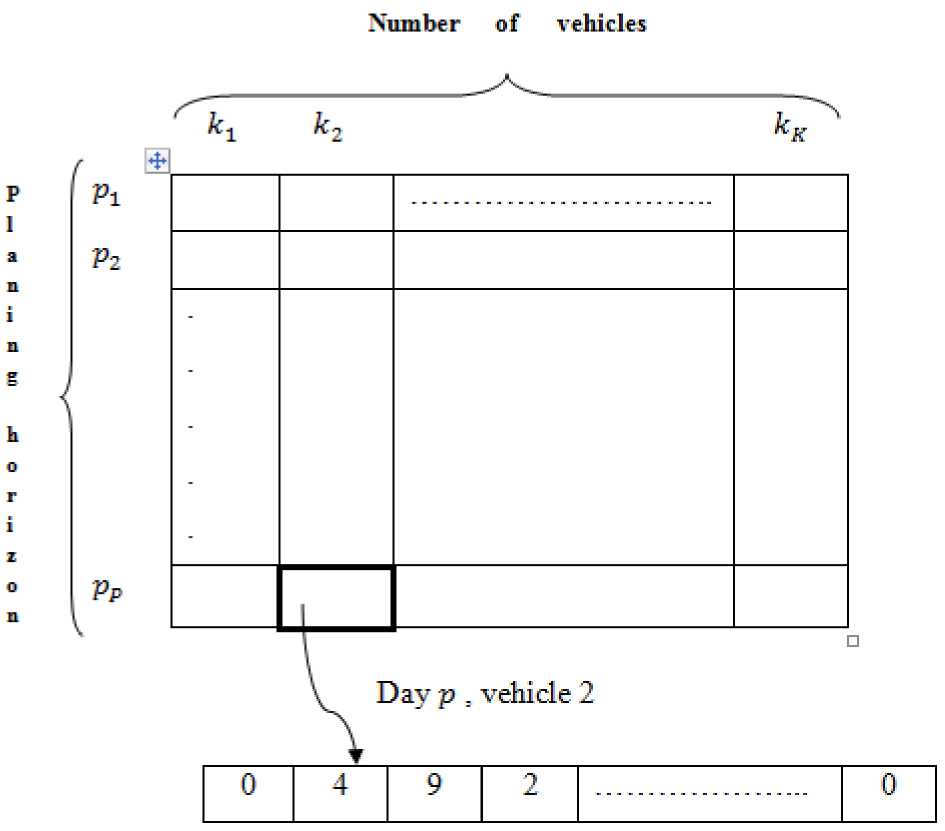


Figure 12: Solution representation.

2. Solution Evaluation

The evaluation of the solution is obtained through three different objectives:

- the net acquired profit,
- the total traveling cost, and
- the maximum difference between the earliest and latest vehicle arrival time to each frequent customer (L_{max}).

The model includes two objectives for minimization:

- the total traveling cost, where we aim to reduce the distance traveled on all roads and take the shortest route as possible, and
- the maximum difference between the earliest and latest vehicle arrival time to each frequent customer (L_{max}).

The model also includes one objective for maximization which is the net acquired profit.

This objective function used to evaluate the solutions, it is equal to the total collected profit minus the total traveling cost. As shown in the mathematical formulation:

$$net\_profit = global\_profit - global\_cost$$

Solutions that do not respect one or more constraints are infeasible solutions. In our resolution process, we can obtain infeasible solution that do not respect constraints of capacity of vehicle and duration of road. Penalizing them using a penalty factor. The solutions that do not respect constraint related to vehicle capacity are penalized by multiplying the number of vehicles that exceed the specified capacity, by the penalty factor, and subtracting the result from the total value net profit of the solution. In the same way, solutions that do not respect constraint of not exceeding a specified period of time for a single path are penalized by multiplying the excess by the penalty factor and subtracting the result from the total value of the solution. As shown in the mathematical formulation:

$$objective = netProfit - 10000 \times nbrInconsistencyVehicle - 10000000 \times timeExeed - L_{max}$$

III. Simulated Annealing Algorithm

We used the simulated annealing method because it is an effective way to find the solution in a reasonable time. Other hand, it doesn't just accept good solutions which leads the objective function in a better direction in the iteration process, but also accepts solutions that will degrade the objective function within certain limits, and this is what effectively saves the algorithm from falling into a local trap.

The algorithm starts with initialize SA parameters: *cooling_ratio, T, L, counter_no_improv, nbr_it_max, stop* [line 1]:

- *cooling_ratio* the factor that causes the temperature to drop continuously. We can imagine any kind of law of decrease, the most common being $T_{i+1} = \propto \times T_i$, with $0 < \propto < 1$.

- T is the temperature, it plays an important role. At high temperature, the system is free to move in solution space ($e^{(\frac{\Delta f}{T})}$ close to 1). At low temperature, the system has less freedom of movement ($e^{(\frac{\Delta f}{T})}$ close to 0).
- L is the number of changes after which the temperature of the system is lowered.
- *counter_no_improv* and *nbr_it_max* are the stop factors of the algorithm, where *counter_no_improv* represents the number of repetitions without improvement the best solution, and *nbr_it_max* represents the number of iterations of the algorithm.

Algorithm 1: Simulated Annealing

```
1: Initialize the SA parameters (cooling_ratio; T; L ; counter_no_improv; nbr_it_max ; stop)
2: Create_Initial_Solution ();
3: Current_Solution ← Initial_Solution; Best_Solution ← Initial_Solution;
4: while (! stop)
5:      Create neighbor solution;
6:      Repair neighbor solution;
7:      if (neighbor_obj > current_obj)
8:           Current_Solution ← Neighbor_Solution;
9:           if (neighbor_obj > best_obj)
10:               Best_Solution ← Neighbor_Solution;
11:               counter_no_improv = initial_value_cno;
12:          else
13:               counter_no_improv--;
14:          endif
15:     else
16:          p ← $e^{(\frac{\Delta f}{T})}$ ;
17:          m ← Random_probability() ;
18:          if (m < p)
19:               Accept neighbor solution as new current solution
20:          endif
21:     endif
22:     if (l==0)
23:          l=L;
24:          T=T*cooling_ratio;
25:     else
26:          l-- ;
27:     endif
28:     cpt++ ;   // cpt is the counter of iteration.
29:     if ((cpt==nbr_it_max) or (counter_no_improv<=0){
30:          stop=true;
31:     endif
```

```
32: endwhile
```

After that, an initial solution is created [line 2]. The process of building the initial solution ensure that the initial solution respects constraints (9), (10), (11), (12), (14), and (19) of the mathematical model. More detail about this process will be given later in chapter. Then we initialize the current solution and best solution by coping the Initial Solution to them [line 3]. Next the main loop of the algorithm starts [line 4]. While we don't yet reach a stopping criterion, the algorithm stays in this loop. At each iteration of the algorithm, we start by creating the neighbor solution of the current one [line 5], and we execute some reparations on the obtained solution [line 6]. This reparations allow solution to respect more constraints. Thus, they allow to reduce the infeasibilities of the solution. After that in [line 7-8] a comparison is made between the current solution and the neighbor one. If the value of the objective function of the neighbor solution is higher than that of the current solution, the neighbor solution is accepted so the current solution is replaced by the neighbor solution. In that case, another comparison is made [lines 9-14] between the best solution and the neighbor solution. If the value of objective function of the neighbor solution is higher than that of the best solution, the neighbor solution is accepted as new best solution, and *counter_no_improv* will be reinitialized by 1. Otherwise the value of *counter_no_improv* will be decreased by 1. In the case when the condition in [line 7] is not satisfied, the neighbor solution is accepted as a new current solution with probability $p.\ e^{(\frac{\Delta f}{T})}$,where $\Delta f = |f\_obj(current_{solution}) - f\_obj(neigbour\_solution)|$ [lines 15-20]. At the end of each iteration, T and l are updated [lines 22-27].The algorithm stops when at least one of the termination criterions are met [lines 29-30], Either the number of iterations without optimizing the best solution reached to 50000, or the number of iterations of the algorithm reached to 100000.

1. Initial Solution

It may be impossible to create a feasible initial solution due to the complexity of the process and due to the imposed consistency constraints. Thus, we can start with a possible solution and compensate by penalizing the infeasibilities.

First, we insert depot at the beginning and the end of each route [lines 1-10]. After that we order the frequent customers according to their request (the number of days in which they requested the service) [line 11]. Next, we find the best vehicle and the best position in this vehicle for each frequent customer, then insert it in each day he ask for a service in same vehicle [lines 12-18]. Finally, we choose a random vehicle and find the best position for each infrequent customer on the day it asks for a service [lines 19-25].

Algorithm 2: Build Initial Solution

```
1: for each day d
2:       for each vehicle v
3:             Initial_Solution.routes[d][v].add(0);
4:       endfor
5: endfor
6: for each day
7:       for each vehicle
8:             Initial_Solution.routes[d][v].add(0);
9:       endfor
10: endfor
11: Sort customers by them requirement;
12: for each frequent customer i
13:       best_vehicle ← Find_Best_Vehicle (i);
14:       for each day d frequent customer i ask for a service
15:             best_position ← Find_Best_Position (d, best_vehicle, i);
16:             Initial_Solution.routes[d][best_vehicle].add (best_position, i);
17:       endfor
18: endfor
19: for each non-frequent customer i
20:       best_vehicle ← Find_Best_Vehicle (i);
21:       for the day d non-frequent customer i ask for a service
22:             best_position ← Find_Best_Position (d, best_vehicle, i);
23:             Initial_Solution.routes[d][best_vehicle].add (best_position, i);
24:       endfor
25: endfor
```

2. Neighborhood Method

Neighborhood of the solution is a set of solutions, some of these solutions may be better than the current solution and others may be worse than it. A neighbor solution is obtained by making changes and transformations on the current solution.

To build a neighbor solution, there are several methods. We chose three methods among them to implement. The three methods we used are:

- 2-Opt move
- 1-1 Exchange move

- 1-0 Exchange move

2.1 The 2-Opt Move

2-opt is an iterative algorithm, at each step, we remove two edges that form an intersection, of the current solution, and we reconnect the two vertices. As shown in Figure 13. This method improves the cost of solutions. Suppose a single route consists of the following set of customers in the given order N1 = {depot , ... i , i+1 , ... , j , j+1; ... , depot} and let {(i , i+1) , (j , j+1)} be a set of two edges in N1 that form a criss-cross. The 2-Opt move deleting the branch (i , i+1) , (j , j+1) and replacing them with their supplement (i, j) , (i+1 , j+1)} to reconstruct the route (see Algorithm 2). In multiple route the 2-Opt move is applied exactly in the same way as in the case of single route (Tarantilis, Kiranoudis, & Vassiliadis, 2002).

Algorithm 3: 2-Opt Move

```
1: for each edge $(x_i, x_{i+1}) \in graph$ do
2:     for each edge $(x_j, x_{j+1}) \in graph$ different from $(x_i, x_{i+1})$ do
3:         if ($(x_i, x_{i+1}), (x_j, x_{j+1})$ intersect) then
4:             Replace edges $(x_i, x_{i+1}), (x_j, x_{j+1})$ by $(x_i, x_j), (x_{i+1}, x_{j+1})$
5:         end if
6:     end for
7: end for
```

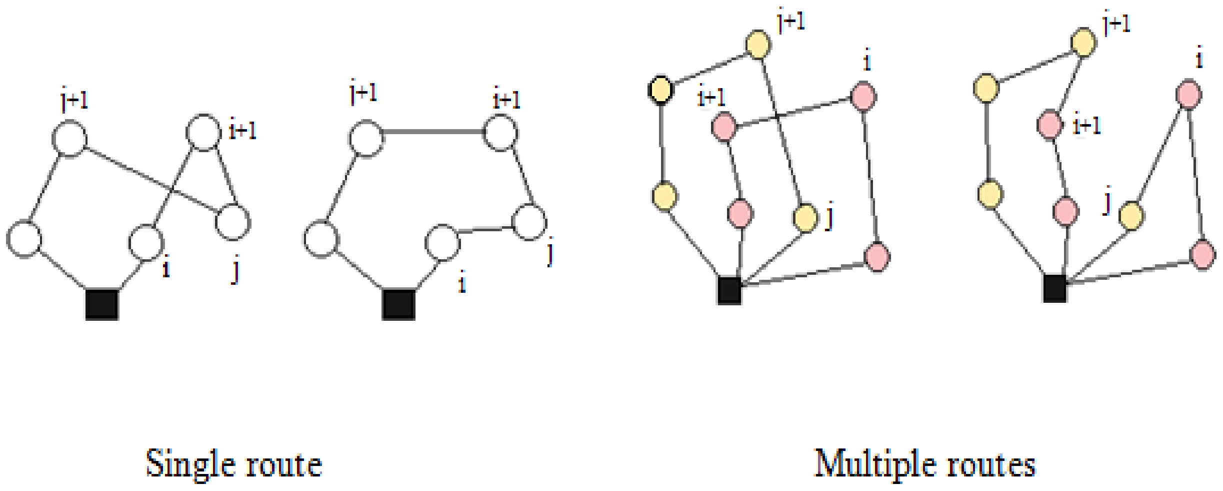


Figure 13: 2-Opt move for single and multiple routes.

2.2 The 1-1 Exchange Move

The 1-1 Exchange move swaps two customers from the same route. Consequently, if it is supposed that the initial tour consists of the set of customers {depot ; ... ; i -1 ; i ; i +1 ; ... ; j- 1; j ; j +1 ; ... ; depot} , the improved one is constructed as {depot ; ... ; i -1 ; j ; i +1 ; ... ; j-1; i ; j +1 ; ... ; depot}(see Algorithm 3).The 1-1 Exchange move is applied exactly in the same way as in the case of single route but the swapping of nodes takes place between different routes (Tarantilis, Kiranoudis, & Vassiliadis, 2002).This move is shown in Figure 14.

Algorithm 4: 1-1 Exchange move

```
1: for each node $x_i \in graph$ do
2:     for each node $x_j \in graph$ different from $x_i$ do
3:         if (($x_{i-1}, x_i$), ($x_i, x_{i+1}$), ($x_{j-1}, x_j$), ($x_j, x_{j+1}$) form two consecutive intersections) then
4:             Replace edges {(($x_{i-1}, x_i$), ($x_i, x_{i+1}$), ($x_{j-1}, x_j$), ($x_j, x_{j+1}$) }by
5:                 {(($x_{i-1}, x_j$), ($x_j, x_{i+1}$), ($x_{j-1}, x_i$), ($x_i, x_{j+1}$))}
6:         end if
7:     end for
8: end for
```

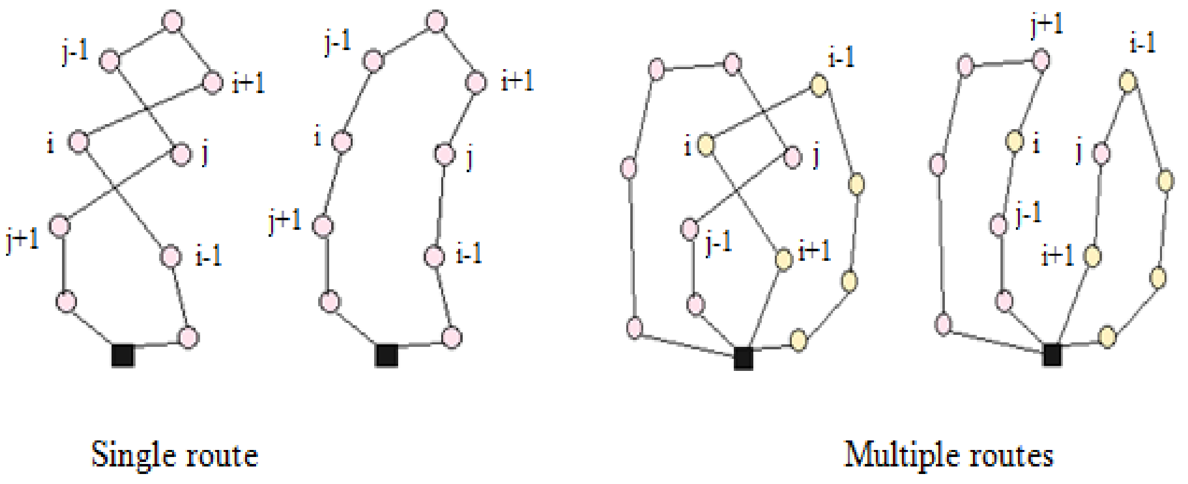


Figure 14: 1-1 Exchange move for single and multiple routes.

2.3 The 1-0 Exchange Move

The 1-0 Exchange transfers a customer from its position in one route to another position in either the same or a different route. If it is supposed that the initial tour consists of the set of customers {depot ; ... ; i ; i +1 ; ... ; j -2 ; j- 1; j ; j +1 ; ... ; depot}, the improved one is constructed as {depot ; ... ; i ; j ; i +1 ; ... ; j -2 ; j- 1 ; j +1 ; ... ; depot} (see Algorithm 4) (Tarantilis, Kiranoudis, & Vassiliadis, 2002). This move is shown in Figure 15.

Algorithm 5: 1-0 Exchange move

```
1: for each node $x_j \in graph$ do
2:     for each edge $(x_i, x_{i+1}) \in graph;\ x_i, x_{i+1}$ different from $x_j$ do
3:         if $(distance\,(x_{j-1}, x_j) + distance\,(x_j, x_{j+1}) >$
4:                 $distance\,(x_i, x_j) + distance\,(x_j, x_{i+1}))$ then
5:             Replace edges $\{((x_i, x_{i+1}), (x_{j-1}, x_j), (x_j, x_{j+1})\}$ by
6:                 $\{((x_i, x_j), (x_j, x_{i+1}), (x_{j-1}, x_{j+1})\}$
7:         end if
8:     end for
9: end for
```

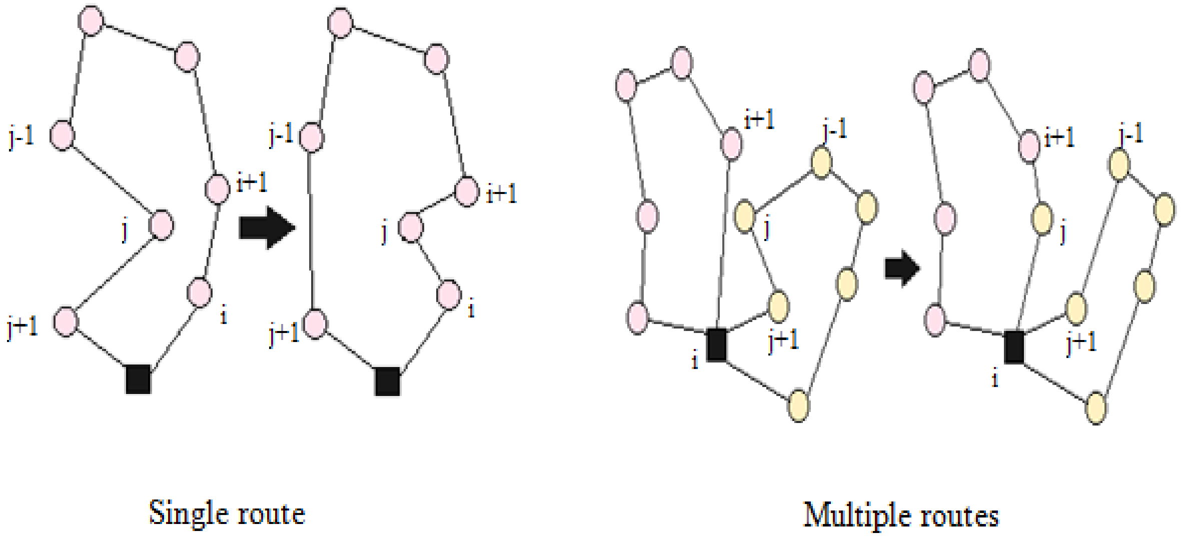


Figure 15: 1-0 Exchange move for single and multiple routes.

IV. Computational Results

The SA was coded in java using the integrated development environment: Net Beans IDE 8.2, under Windows 7 operating system, and run on a PC Intel(R) Core(TM) i3-3110M CPU.

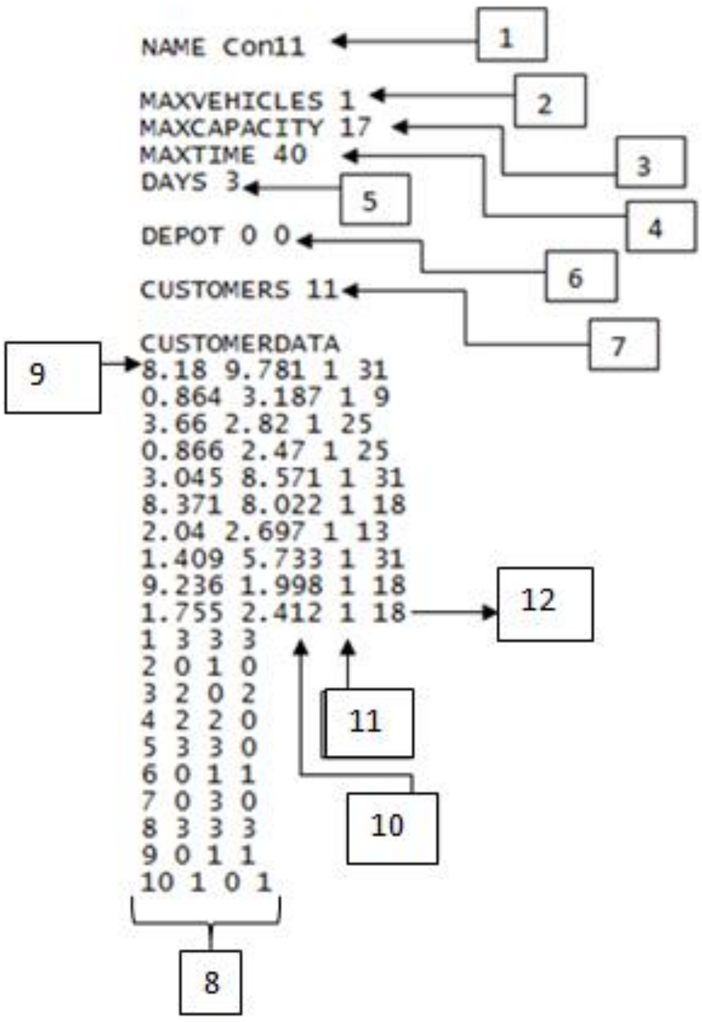
1. Parameters Setting

There is no algorithm that determines exactly the best parameter values of meta-heuristics. So, most developers use the traditional try-and-error method, to set them. In this method, the programmer tries to find the appropriate combination of parameter values by visually analyzing the outputs of the algorithm, i.e. the obtained solution and its objective value. We have adopted the aforementioned method to fix the parameters of our algorithm. Thus, the initial values of *counter_no_improv* and *nbr_it_max;* the stop criteria of the algorithm, are respectively set to be equal to 50000 and 100000. The initial values of *cooling_ratio, T, L* are, respectively, set to be equal to 0.88, 1000 and 10.

2. Instances

Each instance is saved in a text file and organized as shown in Figure 16. The numbered elements on Figure 16, are explained underneath:

1. The name of instance.
2. The number of available vehicles.
3. The vehicle capacity.
4. The route duration.
5. The planning horizon (number of days).
6. The depot coordinates.
7. The number of customers.
8. Customer demand section.
9. The customer coordinate X.
10. The customer coordinate Y.
11. The customer service time.
12. The customer profit.



```
NAME Con11

MAXVEHICLES 1
MAXCAPACITY 17
MAXTIME 40
DAYS 3
DEPOT 0 0
CUSTOMERS 11

CUSTOMERDATA
8.18 9.781 1 31
0.864 3.187 1 9
3.66 2.82 1 25
0.866 2.47 1 25
3.045 8.571 1 31
8.371 8.022 1 18
2.04 2.697 1 13
1.409 5.733 1 31
9.236 1.998 1 18
1.755 2.412 1 18
1 3 3 3
2 0 1 0
3 2 0 2
4 2 2 0
5 3 3 0
6 0 1 1
7 0 3 0
8 3 3 3
9 0 1 1
10 1 0 1
```

Figure 16: Instance representation.

There are two types of instances:

- small instances and
- large instances.

The instance size is related to the number of clients, the larger the number of clients, the larger the instance. Tableau 1 represents the small instances data, the number of permissible vehicles, max capacity of vehicles, max duration of road, number of days and the number of customers.

Tableau 1 : Small instances data.

Instance	Max vehicles	Max capacity	Max time	Number of days	Number of customers
b1	1	17	40	3	11
b2	1	19	38	3	11
b3	2	15	35	3	11
b4	2	15	35	3	11
b5	2	15	35	3	11
b6	2	15	35	3	13
b7	1	15	35	3	13
b8	2	15	35	3	13
b9	2	15	35	3	13
b10	1	15	35	3	13
b11	2	19	40	3	19
b12	2	15	35	3	19
b13	2	15	35	3	19

3. Our Results

Our results of solving the small instances of ConVRP with Profit are presented in Tableau 2.

Three runs of SA were conducted on each test problem. The objective value of the best found solution in these three runs for each test problem is shown in Tableau 2. This table also presents the average objective value obtained for each instance, the average L_{max} , and the average time CPU.

3.1 SA vs. CPLEX

CPLEX is a computer optimization tool marketed by IBM since its acquisition of the French company ILOG in 2009. Its name refers to the C language and the simplex algorithm, more information can be found in (IBM ILOG CPLEX Optimizer).

The obtained results of SA are compared with those of CPLEX in Tableau 3. Tableau 3Tableau 3 shows that SA and CPLEX offer the same results regarding the objective function's values and $L_{\max}$ values for the four test instances b1, b2, b3, b7 and b8. For the test instances b6, b10, b11, b12 and b13, the objective function's values of solutions provided by the SA are almost equal to those provided by CPLEX. But regarding $L_{\max}$, the results of SA are better. For the test instances b4 and b5 the SA provided results slightly lower (one-hundredth or one-thousandth) than CPLEX results, while $L_{\max}$ results of SA are much better than those of

CPLEX. According to the table, for instance b4, SA offers the same value as CPLEX for the net profit, but also a better value for $L_{\max}$. This is due to rounding the net profit values to two digits after comma. For the test instance b9, the results of CPLEX (the net profit's value and $L_{\max}$ value) are better than those of SA. Regarding the computational time, CPLEX takes more time to find the global optimal solution for test instances b6, b9, b11, b12 and b13, but SA takes slightly more time for instances b1, b2, b3, b4, b5, b7, b8 and b10. Figure 17 and Figure 18Figure 18 represent convergence comparison of SA and CPLEX.

Tableau 2: Results SA of small-scale instances.

Instance	Best objective	L_{max}	CT (s)	Average objective	Average L_{max}	Average CT (s)
b1	357.307	4.67	13.9	365.63	3.96	14.71
b2	289.04	4.63	23.78	279.25	3.52	26.42
b3	375.12	2.50	11.41	373.27	4.26	12.07
b4	413.11	2.87	17.24	413.06	2.87	17.39
b5	391.17	0.0	17.07	391.17	0.0	17.07
b6	440.68	3.72	19.71	437.74	3.43	19.50
b7	303.77	4.25	24.43	301.7	3.79	23.92
b8	431.56	3.34	20.45	425.27	4.18	20.48
b9	376.96	5.99	14.56	374.15	5.65	16.58
b10	306.47	5.87	23.45	296.47	3.67	22.45
b11	573.06	2.18	48.03	568.57	3.21	47.51
b12	766.10	3.09	60.37	697.84	2.84	50.68
b13	694.78	362	59.12	684.32	3.94	57.68

Tableau 3: SA vs. CPLEX Small-small instances.

	CPLEX			SA		
Instances	**NP**	L_{max}	**CT (s)**	**NP**	L_{max}	**CT (s)**
b1	357.31	4.67	1.97	**357.31**	**4.67**	14.71
b2	289.04	4.63	2.36	**289.04**	**4.63**	23.78
b3	375.12	2.50	8.36	**375.12**	**2.50**	11.41
b4	413.11	3.98	11.23	**413.11**	**2.87**	17.24
b5	391.69	4.17	12.37	391.17	**0.0**	17.07
b6	440.98	3.93	222.11	440.68	**3.72**	**19.71**
b7	303.77	4.25	0.38	**303.77**	**4.25**	24.43
b8	431.56	3.34	14.65	**431.56**	**3.34**	20.45
b9	377.63	4.21	51.20	376.96	5.99	**14.56**
b10	305.08	4.48	2.43	**306.47**	5.87	23.45
b11	588.61	4.99	5759	573.06	**2.18**	**48.03**
b12	715.63	3.48	340.38	706.10	**3.09**	**60.37**
b13	698.92	4.66	1038.31	694.78	**3.62**	**59.12**

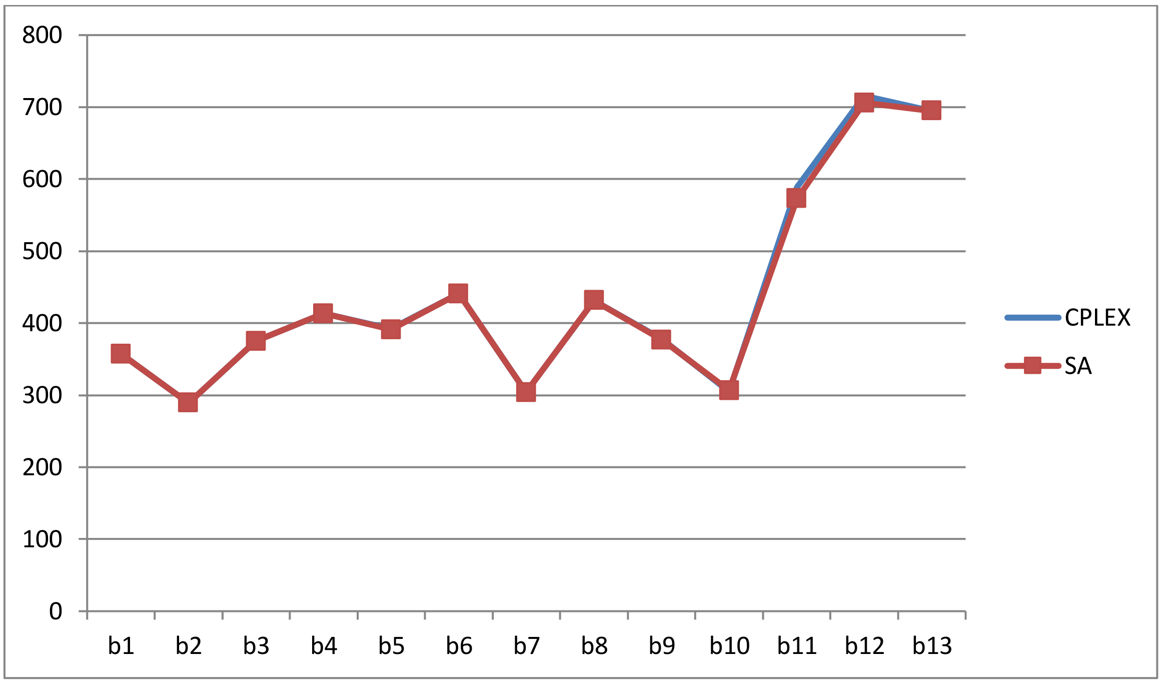


Figure 17 : Comparison of Net profit values between SA and CPLEX.

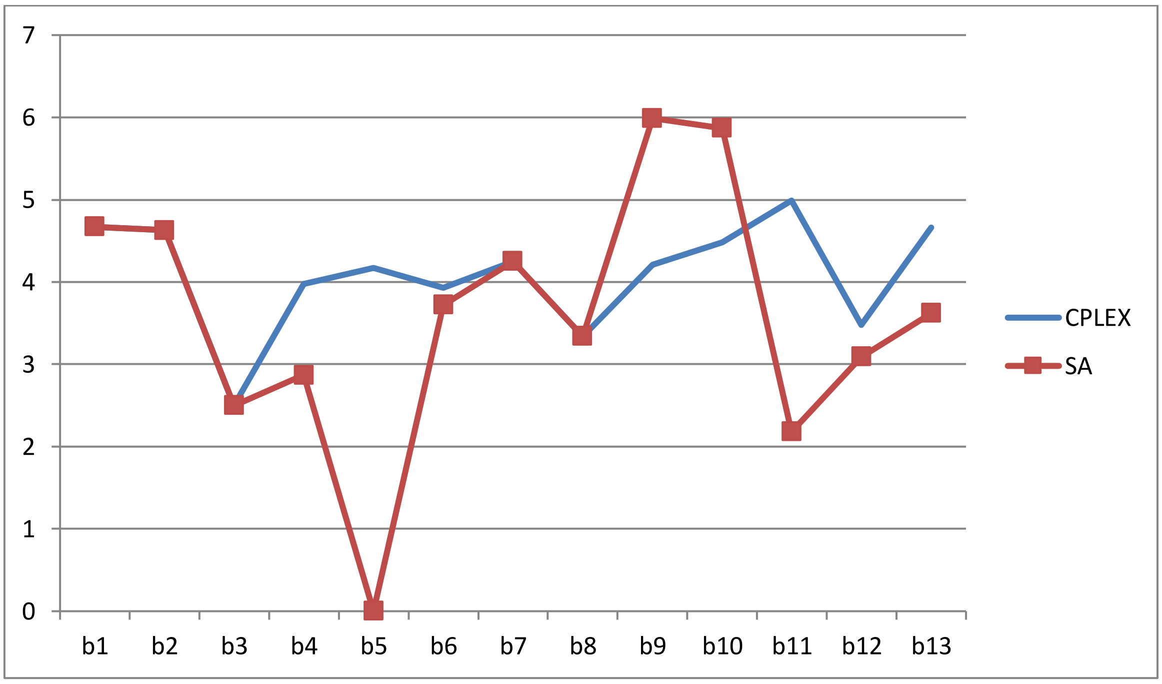


Figure 18 : Comparison of Lmax values between SA and CPLEX.

3.2 SA vs. ATS

The Adaptive Tabu Search is the proposed meta-heuristic algorithm in (Stavropoulou, Repoussis, & Tarantilis, 2018). We compared our results with those obtained by ATS.

The obtained results of SA are compared with those of ATS in Tableau 4. Tableau 4Tableau 4 shows that SA and ATS offer the same results regarding the net profit's values and $L_{\max}$ values for the four test instances b1, b3, b7 and b8. For the test instances b4 and b5, the net profit's values and $L_{\max}$ values of solutions provided by the SA are better than those of ATS. For the test instance b11, the results of ATS (the net profit's value and $L_{\max}$ value) are better than those of SA. For instances b2, b9 and b10 the net profit's values of solutions provided by the SA are better than those by ATS, but L_{max} values of solutions provided by ATS are better than those of SA. On the contrary for instances b6, b12 and b13, the net profit's values of solutions provided by ATS are better than those of SA, but L_{max} values of solutions provided by SA are better than those of ATS. Figure 19 and Figure 20 represent convergence comparison of SA and ATS.

By comparing the results of SA with ATS, we see that the performance of SA is good. SA gave 15.38% of the instances better results than ATS. While only 7.69% of the instances results were found better than SA.

Tableau 4: SA vs. ATS Small-scale instances.

	ATS		**SA**	
Instances	**NP**	L_{max}	**NP**	L_{max}
b1	357.31	4.67	**357.31**	**4.67**
b2	279.04	3.93	**289.04**	4.63
b3	375.12	2.50	**375.12**	**2.50**
b4	413.11	3.98	**413.11**	**2.87**
b5	389.50	4.40	**391.17**	**0.0**
b6	440.87	3.93	440.68	**3.72**
b7	303.77	4.25	**303.77**	**4.25**
b8	431.56	3.34	**431.56**	**3.34**
b9	369.40	4.61	**376.96**	5.99
b10	298.87	4.15	**306.47**	5.87
b11	581.21	2.03	573.06	2.18
b12	715.63	3.48	706.10	**3.09**
b13	698.92	4.66	694.78	**3.62**

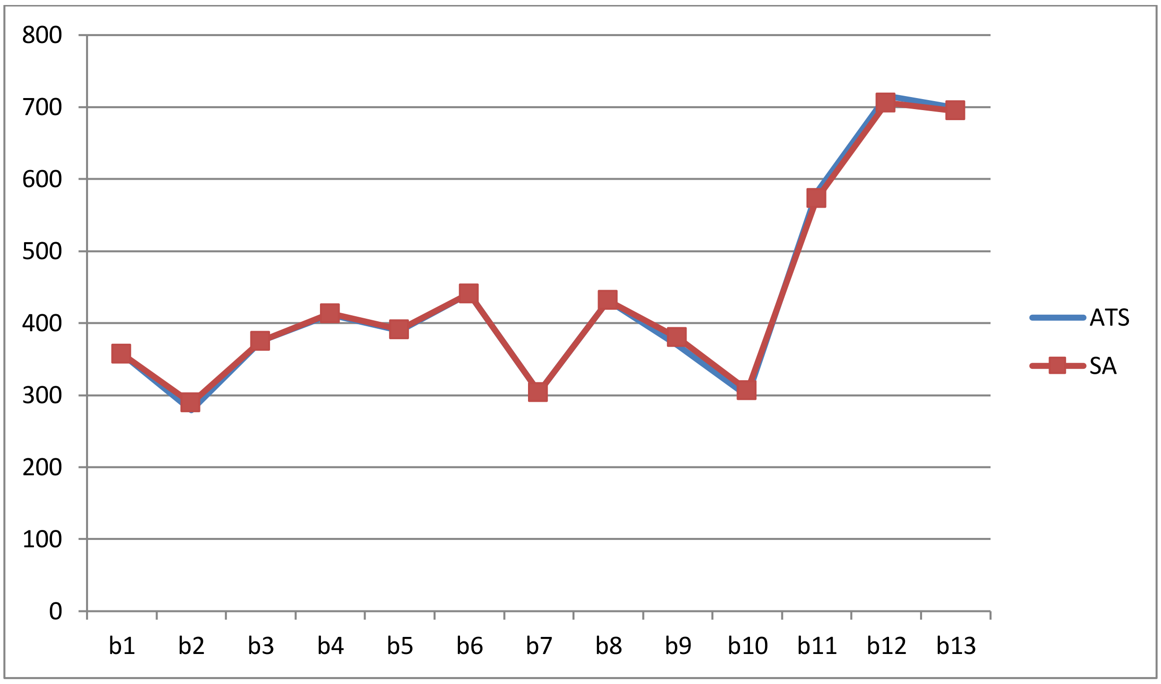


Figure 19 : Comparison of Net profit values between SA and ATS.

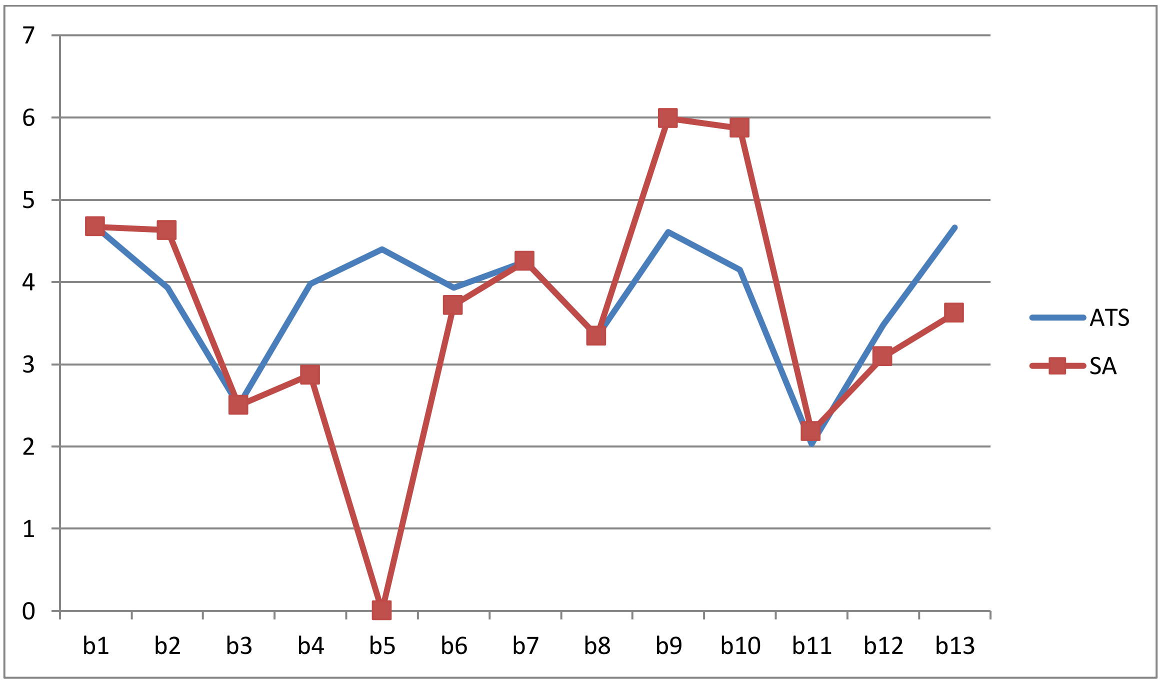


Figure 20 : Comparison of Lmax values between SA and ATS.

V. Application Interface

Our application interface revolves around the main window shown in Figure 21. This window includes the following components:

1: The menu "File", contain two menu items as shown in:

- The menu item "Open". By pushing the item "Open", the window in Figure 22 is shown, through which the user chooses the file he wants to open. If the menu item "Open" is pushed before the end of the SA process, a warning window appears (see Figure 23Figure 23).
- The menu item "Exit", by pushing "Exit" item, the main window is closed, and the program exits.

2: The menu "Run MH", contains two menu items as shown in:

- The menu item "Run", by clicking "Run", the simulated annealing process begins.
- The menu item "Stop", by clicking "Stop", the simulated annealing process stops.

3: The menu "Info", contains one menu item as shown in:

- The menu item "get Information", by pressing "get Information", a help window appears (see Figure 25), where information is displayed. On the left side, we find information on the test instance. However, on the right, we find information on the best_solution yet found.

4: The "textArea", where SA results appear. This text area records the evolution of the best solution at each iteration, as shown in Figure 24.

5: The "daysPanel", where the solution is drawn (i.e. the depot, customers and roads, for each day) as shown in Figure 24.

6: The button "clear ", by clicking "clear", the content of the "textArea" will be cleared.

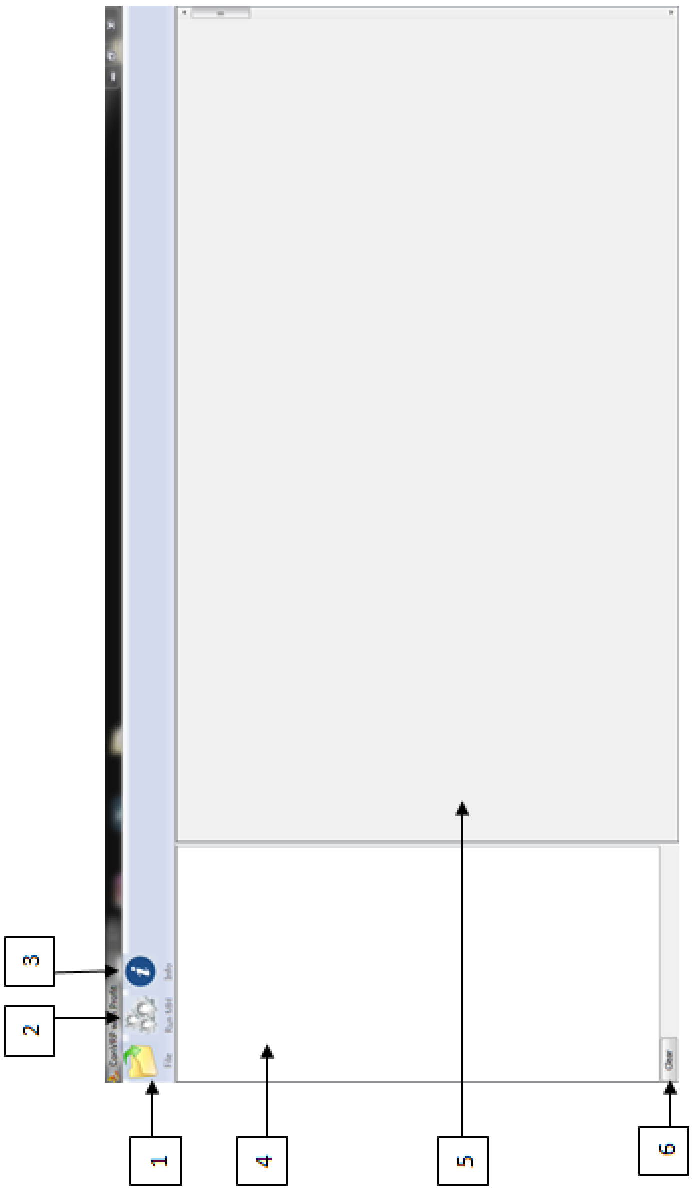


Figure 21: The main window.

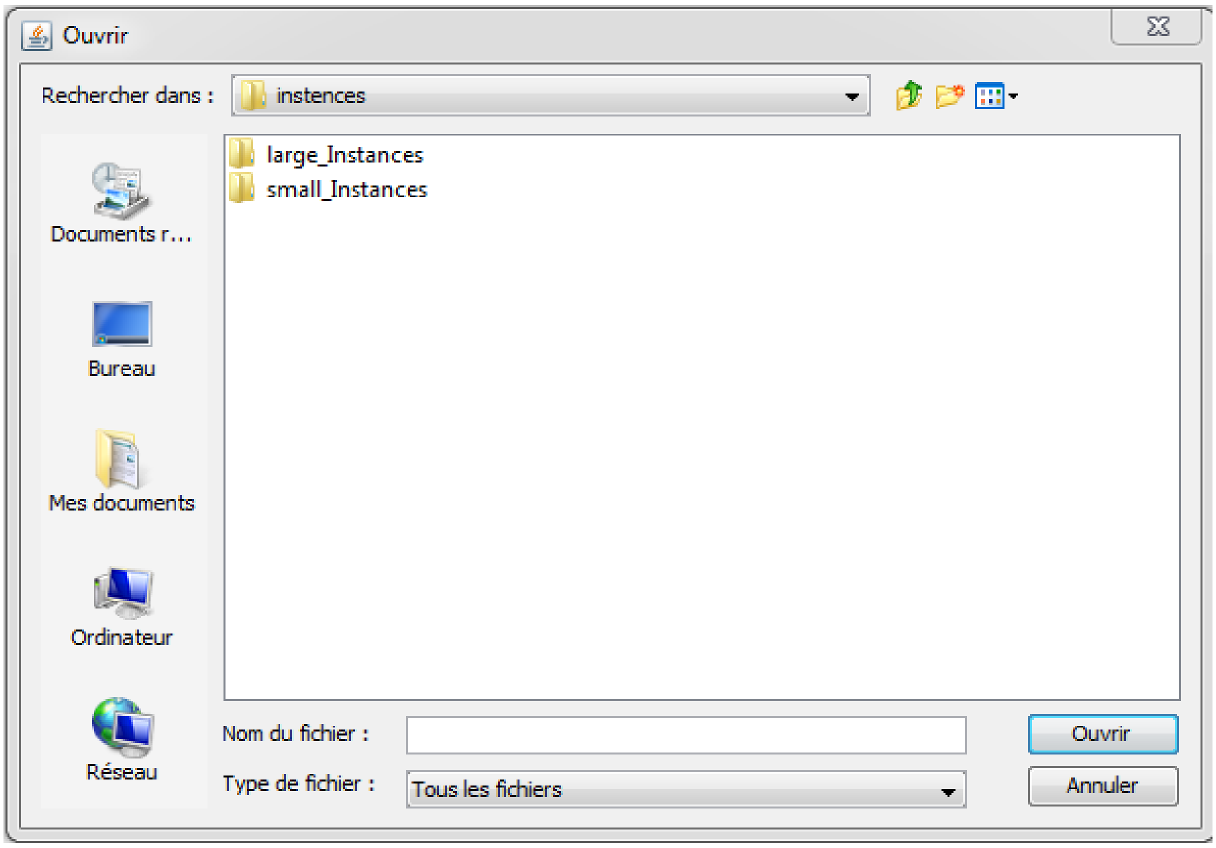
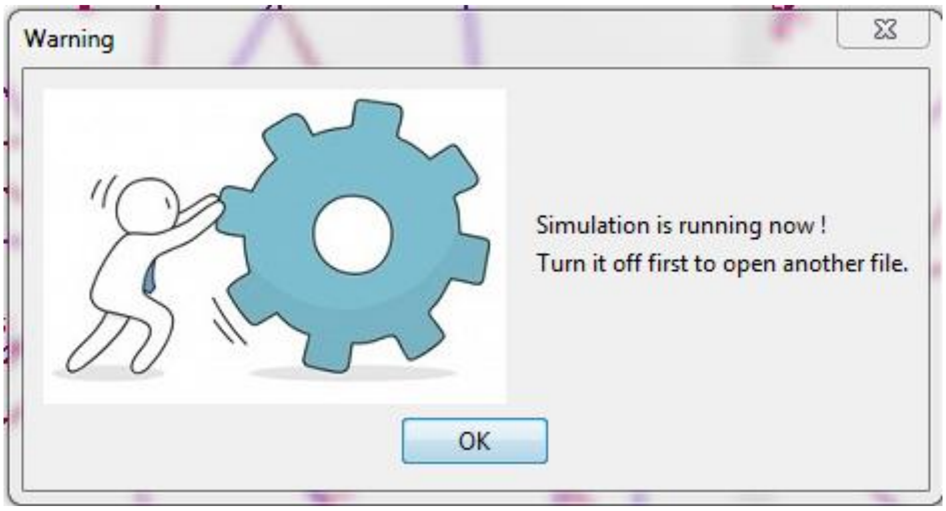
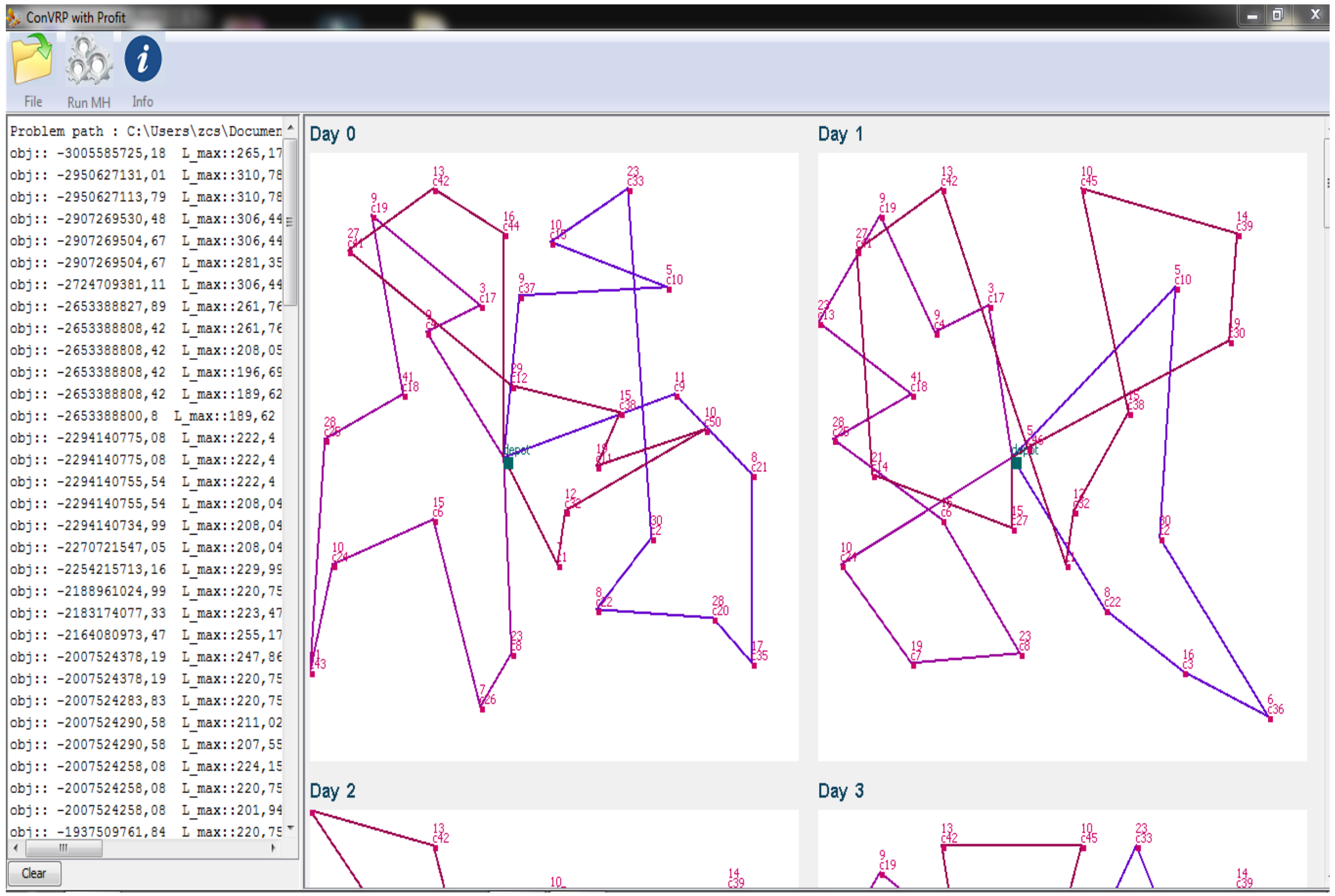


Figure 22: Open window.


Warning

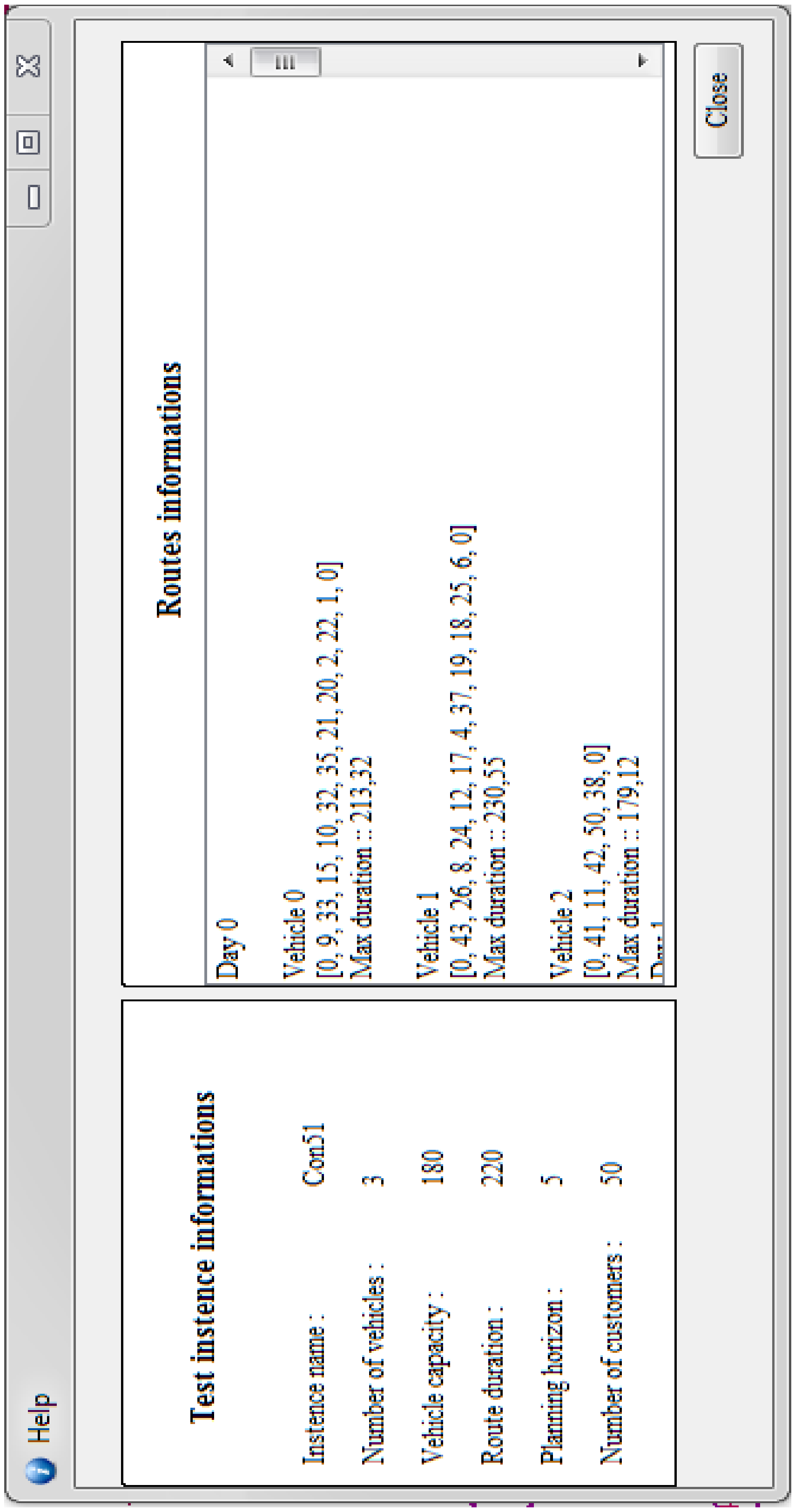
Simulation is running now !
Turn it off first to open another file.

Figure 23: Warning window.



```
Problem path : C:\Users\zcs\Documen
obj:: -3005585725,18  L_max::265,17
obj:: -2950627131,01  L_max::310,78
obj:: -2950627113,79  L_max::310,78
obj:: -2907269530,48  L_max::306,44
obj:: -2907269504,67  L_max::306,44
obj:: -2907269504,67  L_max::281,35
obj:: -2724709381,11  L_max::306,44
obj:: -2653388827,89  L_max::261,76
obj:: -2653388808,42  L_max::261,76
obj:: -2653388808,42  L_max::208,05
obj:: -2653388808,42  L_max::196,69
obj:: -2653388808,42  L_max::189,62
obj:: -2653388800,8  L_max::189,62
obj:: -2294140775,08  L_max::222,4
obj:: -2294140775,08  L_max::222,4
obj:: -2294140755,54  L_max::222,4
obj:: -2294140755,54  L_max::208,04
obj:: -2294140734,99  L_max::208,04
obj:: -2270721547,05  L_max::208,04
obj:: -2254215713,16  L_max::229,99
obj:: -2188961024,99  L_max::220,75
obj:: -2183174077,33  L_max::223,47
obj:: -2164080973,47  L_max::255,17
obj:: -2007524378,19  L_max::247,86
obj:: -2007524378,19  L_max::220,75
obj:: -2007524283,83  L_max::220,75
obj:: -2007524290,58  L_max::211,02
obj:: -2007524290,58  L_max::207,55
obj:: -2007524258,08  L_max::224,15
obj:: -2007524258,08  L_max::220,75
obj:: -2007524258,08  L_max::201,94
obj:: -1937509761,84  L_max::220,75
```

Figure 24: Implementation exemple.



Test instence informations

Instence name :	Con51
Number of vehicles :	3
Vehicle capacity :	180
Route duration :	220
Planning horizon :	5
Number of customers :	50

Routes informations

```
Day 0

Vehicle 0
[0, 9, 33, 15, 10, 32, 35, 21, 20, 2, 22, 1, 0]
Max duration :: 213,32

Vehicle 1
[0, 43, 26, 8, 24, 12, 17, 4, 37, 19, 18, 25, 6, 0]
Max duration :: 230,55

Vehicle 2
[0, 41, 11, 42, 50, 38, 0]
Max duration :: 179,12
```

Figure 25: Information window.

VII. Conclusion

In this chapter, we have shown how we applied the simulated annealing algorithm to conVRP with profit. Firstly, the adopted representation and evaluation of the solution are presented. Next, description of the proposed SA is detailed. After, the computational results are exhibited. Next, the application's interface and instances are shown.

General Conclusion

In this study, we have tackled the consistent Vehicle Routing Problem with Profit, which is a variant of the Vehicle Routing Problem. In this problem, more focus is made on the satisfaction of the customers through the enforcing some consistency constraints. Three of them are considered in the model used in our study, which are: person consistency, time consistency and consistency in the delivered quantity.

In literature, this model was first solved by Adaptive Tabuo Search and also using the exact method branch-and-bound under CPLEX solver. Te ATS showed its performance to solve the problem, especially on large-sized instances. So, our goal in this thesis is to evaluate the performance of another solution-based meta-heuristic, called Simulated Annealing, to solve the problem.

The Comparison with CPLEX reveals that the SA performs good. For 46.15% of the instances, we obtained equal results regarding the net profit's values and the $L_{\max}$ values. For 7.69%, our method gives larger results regarding the net profit's values. However, for 38.46%, our method offers better results than CPLEX regarding the $L_{\max}$ values. Regarding the computational time, SA give a very good performs for 38.46% of the instances.

The Comparison with ATS reveals that the SA gives good results. For 30.77% of the instances, we obtained equal results. However, for 15.38%, our method offers better results, regarding the net profit's values and the $L_{\max}$ values. However, ATS offers better results than those of SA for 7.69%, regarding the net profit's values and the $L_{\max}$ values. For 23.08%, our method gives larger results regarding the net profit's values. Finally, for 23.08% of the instances, our method offers better results regarding the $L_{\max}$ values than ATS.

As it is well known in optimization, the combination of parameter' values has a great impact on the obtained results. In this study, we have used the trial-and-error method to fix them. Our choice of the parameter values enabled us to obtain these results, but there is a possibility that if we make more experiments, we fall on a combination that gives better results than those exhibited in this manuscript.

For the large-sized instances, after several essays to fix the parameters of SA and to develop more preferment neighboring mechanisms, we found that SA is not enough efficient to solve this NP-complete problem, comparing with ATS.

As perspective of this work, we aim in the future to use a population-based meta-heuristic in order to solve the problem. But, in this case, more effort should be done in order to find more appropriate representation for the solution. This representation must be more suitable for executing and applying the operators of the chosen meta-heuristic.

Bibliographies

(s.d.). Consulté le 2021, sur Neural Networks Tutorial:Model selection: https://www.neuraldesigner.com/learning/tutorials/model-selection

Amraoui, H., Mhamdi, F., & Elloumi, M. (2017). *Survey of Metaheuristics and Statistical Methods for Multifactorial Diseases Analyses* .

Barros, L., Linfati, R., & Escobar, J. (2020). *An exact approach for the consistent vehicle routing* .

Blocho, M. (2020). Chapter 3 - Exact algorithms for solving rich vehicle routing problems. 93-99. (J. Nalepa, Éd.)

Chopard, B., & Tomassini, M. (2018). An Introduction to Metaheuristics for Optimization. *Simulated Annealing* .

Clark, W., & Wright, W. (1964). *The Complete Works ofWilluim Shakespeare* .

Cordeau, J., Gendreau, M., Laporte, G., & Potvin..., J. (2002). *A guide to vehicle routing heuristics* .

Dantzig, G., & Ramser, J. (1956). *The truck dispatching problem* .

Dorigo, M., & Stützle, T. (2001). *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances* . (U. L. IRIDIA, Éd.)

Dumitrescu, I., & Stützle, T. (2003). Combinations of local search and exact algorithms. *Conference Paper in Lecture Notes in Computer Science* .

Floo, M. M. (1956). Operations Research 4(1) : The Traveling-Salesman Problem. 61-75.

Florio, A. M., Hartl, R. F., & Mi, S. (2020). New Exact Algorithm for the Vehicle Routing Problem with Stochastic Demands. *54 (4)* .

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). Metaheuristic Algorithms in Modeling and Optimization.

Gmira, M., Gendreau, M., Lodi, A., & Potvin, J.-Y. (2021). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research , 288* (1), 129-140.

Groër, C., Golden, B., & Wasil, E. (2009). The Consistent Vehicle Routing Problem. *11 (4)* .

Hebler, K. (2021). Exact algorithms for the multi-compartment vehicle routing problem with flexible compartment sizes. *European Journal of Operational Research , 294* (1), 188-205.

IBM ILOG CPLEX Optimizer. (s.d.). Consulté le 2021, sur https://www.ibm.com/analytics/cplex-optimizer

Ismai, S. B., Legras, F., & Coppin, G. (2011). Synthèse du problème de routage de véhicules.

Jia, Y.-H., Mei, Y., & Zhang, M. (2021). A Bilevel Ant Colony Optimization Algorithm for Capacitated Electric Vehicle Routing Problem. (IEEE, Éd.) *IEEE Transactions on Cybernetics* , 1 - 14.

Kallehauge, B. (2008). *Formulations and exact algorithms for the vehicle routing problem with time windows* .

Kirkpatrick, S., & Vecchi, M. (1983). Global Wiring by Simulated Annealing. (IEEE, Éd.) *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems , 2* (4), 215-222.

Labbé, M., & Laporte, G. (1991). *Capacitated vehicle routing on trees* .

Lee, & Chungmok. (2021). An exact algorithm for the electric-vehicle routing problem with nonlinear charging time. *Journal of the Operational Research Society* .

Manen, R. v. (November 2019). Automating the capacity planning process for a distribution centre : Comparing heuristic and integer programming approaches. *Master thesis* . University of Twente.

Mazzeo, S., & Loiseau, I. (2004). An Ant Colony Algorithm for the Capacitated Vehicle Routing.

Müller, D. (2009). *Algoithm De Colonies De Fourmis*. Consulté le 2021, sur https://www.apprendre-en-ligne.net/info/algo/fourmis.html

Mutar, M. L., Burhanuddin, M., Hameed, A. S., Yusof, N., & Mutashar, H. J. (2020). An efficient improvement of ant colony system algorithm for handling capacity vehicle routing problem. *11 (4)* , 549-564.

Park, H., Son, D., Koo, B., & Jeong, B. (2021). Waiting strategy for the vehicle routing problem with simultaneous pickup and delivery using genetic algorithm. *165 (1)* .

Pedemonte, & Martín. (2017). Tesis de Doctorado en Informática. *Systolic Genetic Search, A Parallel Metaheuristic for GPUs* .

Rajabi-Bahaabadi, M., Shariat-Mohaymany, A., Babaei, M., & Vigo, D. (2021). Reliable vehicle routing problem in stochastic networks with correlated travel times. *Operational Research* , 299–330.

Režnar, T., Martinovič, J., Slaninová, K., Grakova, E., & Vondrák, V. (2017). Probabilistic time-dependent vehicle routing problem. *Central European Journal of Operations Research volume* .

Samsuddin, S., Othman, M. S., & Yusuf, L. M. (2019). A review of single and population-based metaheuristic algorithms solving multi depot vehicle routing problem.

Stavropoulou, F., Repoussis, P., & Tarantilis, C. (2018). The Vehicle Routing Problem with Profits and Consistency Constraints. European Journal of Operational Research.

Talarico, L., & Duque, P. (2015). *An optimization algorithm for the workforce management in a retail chain* .

Tarantilis, C., Kiranoudis, C., & Vassiliadis, V. (2002). *A List Based Threshold Accepting Algorithm for the Capacitated Vehicle Routing Problem* . International Journal of Computer Mathematics.

Thanina, Z., & Katrin, H. (2017). Optimisation d'un problème de programmation linéaire en nombres entiers par la méthode Branch and Bound.

Wang, Y., Wang, L., Chen, b., Cai, Z., Zhou, Y., & Xing, L. (2020). An Improved Ant Colony Optimization algorithm to the Periodic Vehicle Routing Problem with Time Window and Service Choice. *Swarm and Evolutionary Computation , 55*.

Wang, Z. (s.d.). *Intra-Platoon Vehicle Sequence Optimization for Eco-Cooperative Adaptive Cruise Control - Scientific Figure on ResearchGate.* Consulté le 2021, sur ReaserchGate: https://www.researchgate.net/figure/Flowchart-of-tabu-search-algorithm_fig1_320508257

Xia, Y., & Fu, Z. (2018). An Adaptive Tabu Search Algorithm for the Open Vehicle Routing Problem with Split Deliveries by Order. *Wireless Personal Communications , 103*, 595–609.

Yahyaoui, H., Kaabachi, I., Krichen, S., & Dekdouk, A. (2020). Two metaheuristic approaches for solving the multi-compartment vehicle routing problem.

Yang, X.-S. (2021). *ScienceDirect*. Consulté le 08 15, 2021, sur Genetic Algorithm: https://www.sciencedirect.com/topics/engineering/genetic-algorithm