الجـــمـهـوريـة الجـــزائـريـة الديمـقـراطيـة الـشـعـبيـة

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التـــــعـليـم العالي والبحث العـلـمـي

Ministry of Higher Education and scientific research

جـامعة أبي بكـر بـلـقـايد – تـلمسـان –

University of Aboubakr Belkaïd – Tlemcen –
Faculty of TECHNOLOGY

Master's Thesis

Presented for obtaining **MASTER's DEGREE**

In: Industrial Engineering

Speciality: Production Engineer

BY:
BELBALI Abdellah & HADJ ABDALLAH Hamza

Subject:

Solving the capacitated vehicle routing problem using the physarum polycephalum metaheuristic

Defended on: 23/06 /2022 , before the jury composed of:

Ms. MENADJELIA Nardjes	MCB	University of Tlemcen	President
M. BENSMAIN Yassir	MCB	University of Tlemcen	Examiner
Ms. DIB Zahera	MCB	University of Tlemcen	Examiner
M. MEKAMCHA Khalid	MCB	University of Tlemcen	Supervisor

Academic year: 2021 / 2022

Thanking

We wish first of all to thank Allah the Almighty and Merciful, who gave us the strength and patience to carry out this modest work. A special thanks to our coach, Mr. Khalid MEKAMCHA for his presence, his help, and especially for his valuable advices that helped us to achieve our project.

Our sincere thanks also go to the members of the jury for their interest in our research by agreeing to examine our work and enrich it with their proposals. Our thanks to our dear parents, brothers, sisters, and respective friends who have encouraged us and supported us throughout our journey.

We would like to thank all the teachers of the Faculty of Technology for all the knowledge that we acquired through them during our university course.

Finally, we would like to thank all those who supported and encouraged us in the preparation of this modest work. We thank the students of the class of 2021/2022.

Dedication

Every time we complete an important step in our lives, we take a position to look back and remember all those people who shared all the good moments of our existence, but especially the bad moments. Those people who helped us not tell them, supported us without precaution, loved us without reckoning, those people whom our happiness becomes directly to, turn into tears. We devote this humble act as evidence of recognition and respect.

To our dear parents.

To our teachers and everyone, we have a right to.

To our brothers and sisters.

To all our friends.

For All Our Families.

Abstract

The Capacitated vehicle routing problem is one of the hardest problems in the domain of combinatorial optimization. It consists of forming vehicle routes for a fleet of customers with a vehicle capacity constraint to minimize the total traveled distance. There are many methods to solve this problem. Our goal was to use a new approach to solve such kinds of problems. Recently a new intelligent creature called **physarum polycephalum** has taken the interest of many of the Scientifics and researchers. The most interesting thing about this creature is the foraging behavior, It creates an optimal network to connect the food sources. Many models have been proposed that imitate this behavior, but the most popular model is the flow conductivity which later developed into an algorithm called the **physarum solver** algorithm. In this work, we demonstrate that the **physarum solver** can be used to solve the capacitated vehicle routing problem **CVRP** and also the traveling salesman problem **TSP**. we have compared the results obtained by the **physarum solver** with those tabu search and simulated annealing. The results showed that the **physarum solver** performed good results in some cases.

Keywords: metaheuristics, physarum polycephalum, slime mold, vehicle routing problem, traveling salesman problem.

Résumé

Le problème de tourneé de véhicules avec capacité est l'un des problèmes les plus difficiles dans le domaine de l'optimisation combinatoire. Il consiste à former des itinéraires de véhicules pour une flotte de clients avec une contrainte de capacité de véhicules afin de minimiser la distance totale parcourue. Il existe de nombreuses méthodes pour résoudre ce problème. Notre objectif était d'utiliser une nouvelle approche pour résoudre ce genre de problèmes. Récemment, une nouvelle créature intelligente appelée **physarum polycephalum** a suscité l'intérêt de nombreux scientifiques et chercheurs. La chose la plus intéressante à propos de cette créature est le comportement de recherche de nourriture. Il crée un réseau optimal pour connecter les sources de nourriture. De nombreux modèles ont été proposés pour imiter ce comportement, mais le modèle le plus populaire est la conductivité de flux qui s'est ensuite développée en un algorithme appelé **physarum solver**. Dans ce travail, nous démontrons que **physarum solver** peut être utilisé pour résoudre le problème de tournées de véhicules avec capacité CVRP ainsi que le problème de voyageur de commerce TSP. Nous avons comparé les résultats obtenus par **physarum solver** avec ceux de la recherche tabou et du recuit simulé. Les résultats ont montré que **physarum solver** donnait de bons résultats dans certains cas.

Mots-clés : metaheuristiques, physarum polycephalum, Problème de tournée de véhicule, probleme de voyageur de commerce.

ملخص

تعد مشكلة توجيه المركبات ذات السعة من أصعب المشاكل في مجال التحسين الاندماجي. وهو يتمثل في تشكيل مسارات لمركبات ذات سعة محدودة لمجموعة من العملاء من أجل تقليل إجمالي المسافة المقطوعة. هناك طرق عديدة لحل هذه المشكلة. كان هدفنا استخدام نهج جديد لحل مثل هذه الأنواع من المشاكل. في الآونة الأخيرة ، نال مخلوق ذكي جديد يسمى **فيزاروم بوليسيفالوم** على اهتمام العديد من العلماء والباحثين. الشيء الأكثر إثارة للاهتمام في هذا المخلوق هو سلوك البحث عن الطعام ، فهو يشكل شبكة مثالية لربط مصادر الغذاء. تم اقتراح العديد من النماذج التي تحاكي هذا السلوك ، ولكن النموذج الأكثر شيوعًا هو نموذج موصلية التدفق الذي اشتقت منه لاحقًا خوارزمية تسمى **فيزاروم صولفر**. في هذا العمل ، أوضحنا أنه يمكن استخدام ا **فيزاروم صولفر** لحل مشكلة توجيه المركبات ذات السعة **CVRP** وأيضًا مشكلة البائع المتجول **TSP**. لقد قمنا بمقارنة النتائج التي تم الحصول عليها بواسطة فيزاروم صولفر مع تلك النتائج المحظورة ومحاكاة التلدين. أظهرت النتائج أن حلالا الفيزاروم كان له نتائج جيدة في بعض الحالات.

الكلمات المفتاحية : مشكلة البائع المتجول، مشكلة توجيه المركبات ذات السعة،فيزاروم بوليسيفالم،metaheuristics .

Table of Contents

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

VRP	Vehicle routing problem
CVRP	Capacitated vehicle routing problem
DVRP	Dynamic vehicle routing problem
SVRP	Stochastic vehicle routing problem
VRPTW	Vehicle routing problem with time window
TSP	Traveling salesman problem
MH	Metaheuristic
TS	Tabu search
SA	Simulated annealing
GA	Genetic algorithm
ACO	Ant colony optimization
MST	Minimum spanning tree
SA	Sensor angel
RA	Rotation angle
OOP	Object-oriented programming
API	Application programming interface
OSRM	Open source routing machine
SMA	Slim mold algorithm
TPSMA	Tow-way parallel slim mold algorithm
TSPLIB	Traveling salesman problem library

General introduction

Life is based on the principle of facilitating things and choosing the best and easiest ways to eliminate hardship and complexity. Humans always seek to spend their needs while saving time, effort, and profit. They always seek to optimize and over time new and complex problems arise that need research or studies to access only a way to solve them and every time over time other methods and studies may emerge that give better solutions to previous problems.

Optimization problems are one of the most studied areas since age because of its interconnection with the lives of generations and the purpose of their study is to reduce or minimize damage and maximize the benefits and their solutions by types. Optimization methods are divided into two types. The first is mono-objective optimization, which is a maximization or minimization of the objective function. The only goal is to find the optimal and unique solution that can be easily found because the problem is one-way and a clear objective. The second type is multi-objective optimization, the aim being to find the value that mediates these objectives because these objectives are generally contradictory and the solutions therefore vary according to the importance of each objective.

Distribution problems are the search for solutions to cover the needs of all customers, whether by delivering customers, distributing products or other things. These types are generally found in traders, factory owners, and even carriers. The aim is to shorten the time, distance with increased profit, and satisfy all customers.

The vehicle routing problem (VRP) is one of the most prominent distribution problems within the field of improvement and is the distribution of products to customers through vehicles. Each vehicle includes a group of customers so that each customer takes his or her demand through one vehicle such vehicles may be governed by some limitations such as capacity, time, and perhaps some other limitations. The aim of studying this problem is to find the shortest possible route for vehicles while covering all customers and obtaining the shortest total distance traveled by all vehicles.

Methods for solving optimization problems vary depending on the type of problem. Methods of solving also fall into two sections: accurate methods and approximate methods. Precise methods are used to solve minor or small-sized problems, these methods such as the Branch and Bound method. Approximate methods are used to solve complex or large-sized problems, which in turn are divided into heuristics and metaheuristic sections.

Since this **VRP** has no exact solution, especially when the number of customers is large, it is solved by approximate methods. We will use a method of metaheuristic in that.

A metaheuristic is one of the methods of solving difficult problems of improvement, their ideas usually come from the normalcy or from the actions of living organisms, translating these ideas into algorithms used to solve complex problems such as ant algorithms, and bee algorithms, genetic algorithms, etc. We will use a new algorithm in this area, the slime mold algorithm (physarum polycephalum).

This master's thesis is divided into three chapters:

In the first chapter, we will provide the generality of **VRP** and metaheuristics and we will explain its meaning, some types, examples, and lots of details and information.

In the second chapter, we will present the slime mold organ (physarum polycephalum) and will provide his biological background and his life cycle, and his foraging behavior, and finally, present the slime mold algorithm and its details.

In the last chapter, we will try to solve our problem through the slime mold algorithm and finally compare the results with the results of other algorithms to see the effectiveness of this algorithm in this area.

Finally, this master's thesis concludes with a general conclusion.

Chapter I

Generality of VRP and metaheuristics

I.1 Introduction

In the manufacturing section the transportation process plays the role of the lifeline of the industries, it manages the movement of materials (goods, raw materials…) between two or multiple locations. Like all the processes, this one confronts many problems (transport mode distance, cost…..). The vehicle routing problem is one of the most classical problems in this section.

The vehicle routing problem is a class of combinatorial optimization and operational research. It consists of finding optimal routes for each vehicle in order to minimize the total traveled distance; determine the single-vehicle route according to customer positioning as well as some limitations. Finding the optimal solution for the **VRP** is NP-hard. The complexity of this problem is proportional to the number of customers, so the proper methods that can give a feasible solution in a reasonable time are metaheuristics.

In this chapter, we will present a brief introduction to the vehicle routing problem and its variants, also we will talk about the metaheuristics, classification, and the most commonly used methods in the combinatorial optimization problems domain.

I.2 The Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is one of the most studied combinatorial optimization problems. Following a very close collaboration between the specialties of mathematical programming and combinatorial optimization on the one hand, and transport managers on the other hand. Its aim is to find a set of roads (tours) for a fleet of vehicles, in order to satisfy the demands of a set of customers by minimizing the cost of transportation with tours starting and ending at a deposit. [2]

The (VRP) is an extension of the problem of the commercial traveler. It was first introduced by Dantzig in 1954 under the name of (Truck Dispatching Problem) and has since been the subject of intensive studies to model and solve it. In its most basic version (CVRP) or **VRP** with capacity constraints, a fleet of vehicles of finite capacity, based at a depot, must provide tours between several clients (or cities) having each requested a certain quantity of goods. All customers visited by a vehicle refer to the vehicle's tour. Each customer must be served once and only once and each tour starts and ends at the depot as shown in the graph in **Figure I.1**. The objective of the **CVRP** is to minimize the total cost, the sum of the distances or journey times of the tours while respecting the capacity constraints of the vehicles: the number of goods delivered on a tour must not exceed the capacity of the vehicle that insures it. [7]

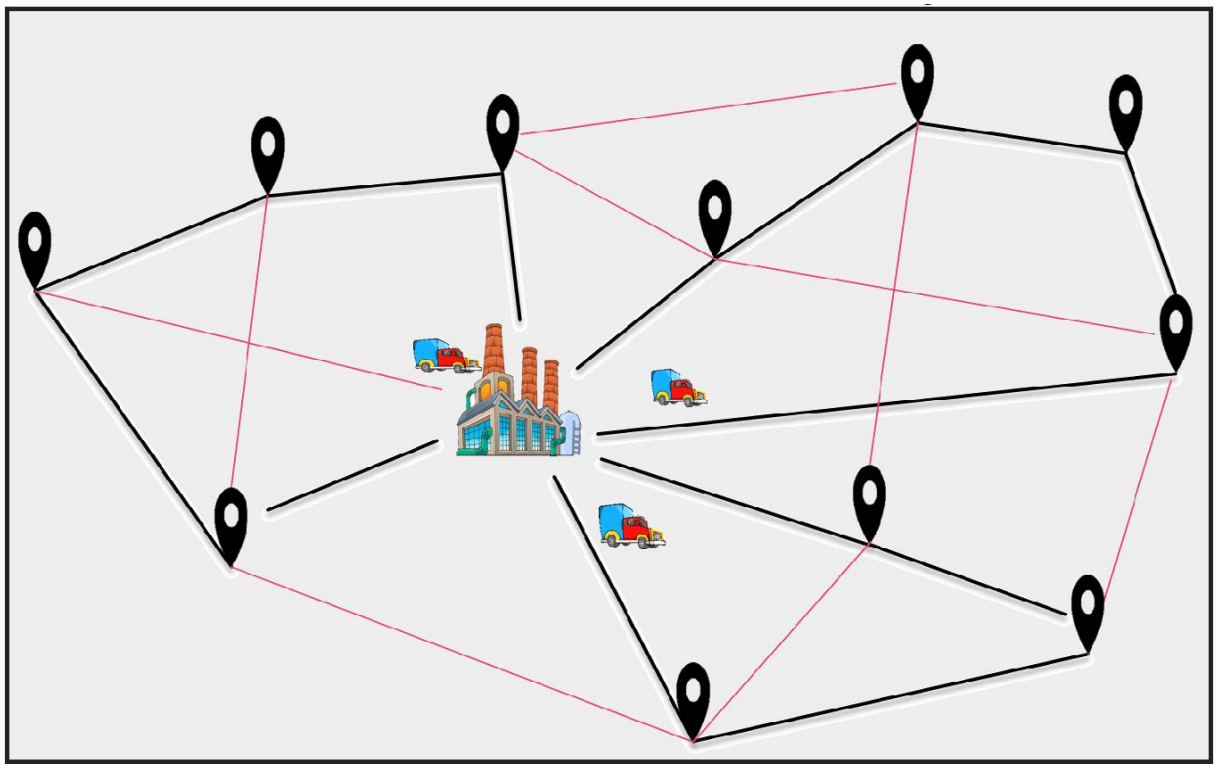


Figure I.1: Graphical presentation of the VRP vehicle routing problem.

I.2.1 VRP Classification

As suggested by Desrochers and Al in 1990 on the classification of **VRP**, they are of the view that the aim of this design is to distinguish between the multiple characteristics of practical problems relating to the conduct of vehicles and their schedules and to achieve some consistency in the literature on the subject.

These features identify four aspects of the problem considered:

The characteristics of the problem, the customers (clients), the fleet, and the optimized objective. [8]

I.2.2 The problem characteristics

Can be the graphical modeling of the problem, which can be oriented, not oriented, or mixed. These can also be parameters of the problem such as the matrix of travel costs that may or may not respect triangular inequality or applications that may be delivered in whole or in part. Finally, travel requests and costs are either deterministic or random.

- **Clients**

In a graphical representation, nodes, arcs, or both at the same time represent clients. There are two types of customers: retail customers called deposits and single customers. Some problems use multiple deposits, which can be intermediate. Customers have sometimes characterized by weights to be delivered. These weights are deterministic or random in nature and are delivered and sometimes picked up by vehicles. Time intervals (flexible or hard) may

be imposed for delivery and weight pick-up operations. In addition to weights, clients are sometimes also affected by priorities or the frequency of visits for a given period. In other cases, some customers cannot be visited due to some fleet constraints.

- **Fleet**

Restrictions on the fleet are used to consider the number of vehicles as fixed, to be used all, or as a variable to be determined. A capacity constraint, which limits loads, assigns similar values for all vehicles (homogeneous) or different values (heterogeneous). Capacity can be represented as a single volume or as a set of compartments. Availability intervals are sometimes imposed on vehicles, which subsequently require synchronization. Similarly, maximum allowances may be imposed on travel costs incurred by vehicles. As with capacity, quotas may be the same or different.

- **The problem objective**

The variants of the **VRP** problem also differ according to the optimized criteria. The possible criteria are the cost of travel, the size of the fleet, the expected level of service, and one or more penalties due to the violation of capacity constraints, quota, or vehicle availability intervals. When multiple objectives are considered, they can be optimized simultaneously. [8]

I.3 VRP Types

During years of research into the **VRP**, other derivatives of this problem have emerged. These manifestations are mainly attributed to researchers who are increasingly addressing the transport and distribution problems facing companies. In the following, we will present the main problems derived from the **VRP** problem:

I.3.1 Capacitated Vehicle Routing Problem (CVRP)

The **CVRP** is a vehicle reorganization, whereby vehicles with limited endurance must pick up or deliver materials to different locations. Items have a quantity, such as weight or size, and vehicles have a maximum carrying capacity. The problem is that materials are transported or delivered at the lowest cost, while vehicles never exceed capacity. [1]

I.3.2 Dynamic Vehicle Routing Problem (DVRP)

In the Dynamic Vehicle Routing Problem (DVRP), new customer commands appear over time, and new routes need to be reconfigured during the execution of the current solution. Montemanni et al. [4] considered a **DVRP** as an extension to the standard vehicle routing problem (VRP) by decomposing a DVRP as a sequence of static **VRPs** and then solving them with an ant colony system (ACS) algorithm. [5]

I.3.3 Stochastic Vehicle Routing Problem (SVRP)

A vehicle routing problem is stochastic if at least one of its elements is stochastic, these elements can be one or many demands of the client, the time, or the transport cost. The problem with the stochastic demands is the most studied in the literature [2]

I.3.4 Vehicle Routing Problem with Time Windows (VRPTW)

VRPTW determines that each client has a time window. This is a period of time during which his service must be performed. The vehicle may arrive early, before the start of the time window, but it must wait until it can be served. In this case, the total wait time can be factored into the model and can be a goal that needs to be minimized. If this happens later, the service cannot be rendered and the corresponding customer will never be satisfied. It can also be subject to penalties if the client does not serve in his or her period of employment. In this case, the problem is said to be "hard constraints" the number of vehicles can be fixed in advance, or be considered as one of the variables of the problem. The objective of the problem is then to minimize the total distance and the number of vehicles to serve the customers without veiling the time windows. [2]

I.3.5 Travelling Salesman Problem (TSP)

The **TSP** is one of the simplest and most studied **VRP**, with the number of vehicles equal to 1. This car revolves around all the cities or customers and eventually goes back to the point where the vehicle went from. The purpose of this problem is to find the shortest distance a vehicle can travel by visiting all customer points.

I.4 VRP Mathematical modelling

Through our source [2] that: G = (V, A) is a graph where V = {1… n} is a set of vertices with 1 top taken as a depot, and A = {(i, j) / i, j € V and i ≠ j} is the set of arcs.

V'=V {1} refers to the set clients except the depot.

Each arc has a non-negative cost C_{ij}. This can be interpreted as the cost of the trip or the travel time between j and i.

We assume here that we have m vehicles of transport capacity D. The **VRP** consists of determining a set of vehicle towers of minimum cost in such a way that:

- Each V' client be visited once by one and only one vehicle;
- All routes start and end at the depot;
- Certain constraints are met.

The constraints are taken into account in this formulation:

- **Capacity restrictions**: for each client, I of V' are associated with a non-negative weight representing the demand, the sum of the weights of around not exceeding the capacity of the vehicle.

- **The total time restrictions**: the total time of a tour must not exceed a terminal T. This time is constituted by the times of the journeys between the clients and the times of stop at each client on the road.

Note: Our objective is to minimize the total distance for all vehicles, so we have to ignore the second rule. "The restrictions of the total time" that: T=0

I.5 CVRP Mathematical formulation

Rego and Raucairol propose a mathematical formulation of **CVRP** as said [2], with the following observations:

- **The Constants**

n = number of clients.

m = number of vehicles.

D = capacity of a vehicle.

R_i=client request I with (R1=0).

C_{ij}=cost or distance from client i to client j.

- **Decision Variables**

X_{ij}^k =1 ----- If the vehicle k travel from client i to client j .

X_{ij}^k =0 ----- if not.

- **The objective function**

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}\sum_{k=1}^{m} X_{ij}^k$ (1.1)

- **Constraints**

a) Each customer is served by one and only one vehicle:

$\sum_{j=1}^{n}\sum_{k=1}^{m} X_{ij}^k = 1$ With: i = 2,……,n. (1.2)

b) The continuity of a tour: a vehicle visiting a summit must exit.

$\sum_{j=1}^{n} X_{ip}^k - \sum_{j=1}^{n} X_{pj}^k = 0$ With: k=1,….,m ; p=1,….,n. (1.3)

c) the capacity constraint of a vehicle:

$\sum_{i=1}^{n} R_i \sum_{j=1}^{n} X_{ij}^k \leq D$ with: k=1,……..,m. (1.4)

d) not exceeding the availability of a vehicle:

$\sum_{i=2}^{n} X_{i1}^k \leq 1$ with: k=1,……..,m. (1.5)

The **VRP** development is an NP-difficult problem that is to say that there is not so far a deterministic algorithm that can solve this problem in polynomial time. For issues with a large number of clients (<100 clients), approaches are required to resolve them. [9]

I.6 The VRP resolution methods:

The **VRP** problems are problems with a wide range and multiple types and sections. Therefore, researchers in this field have used many ways to solve them.

The **VRP** belongs to the large family of NP-difficult combinatorial problems. This means that there is no method that can give the optimal solution of the problem in a reasonable time. In this case, it is necessary to use methods that try to give a solution not necessarily optimal but close to the optimal in an acceptable time. Thus the resolution methods devoted to the **VRP** and its variants are mainly classified into two categories: the exact methods and the methods approached, the latter is also divided into two groups: heuristics and metaheuristics [2]. The following figure presents **VRP** resolution methods.

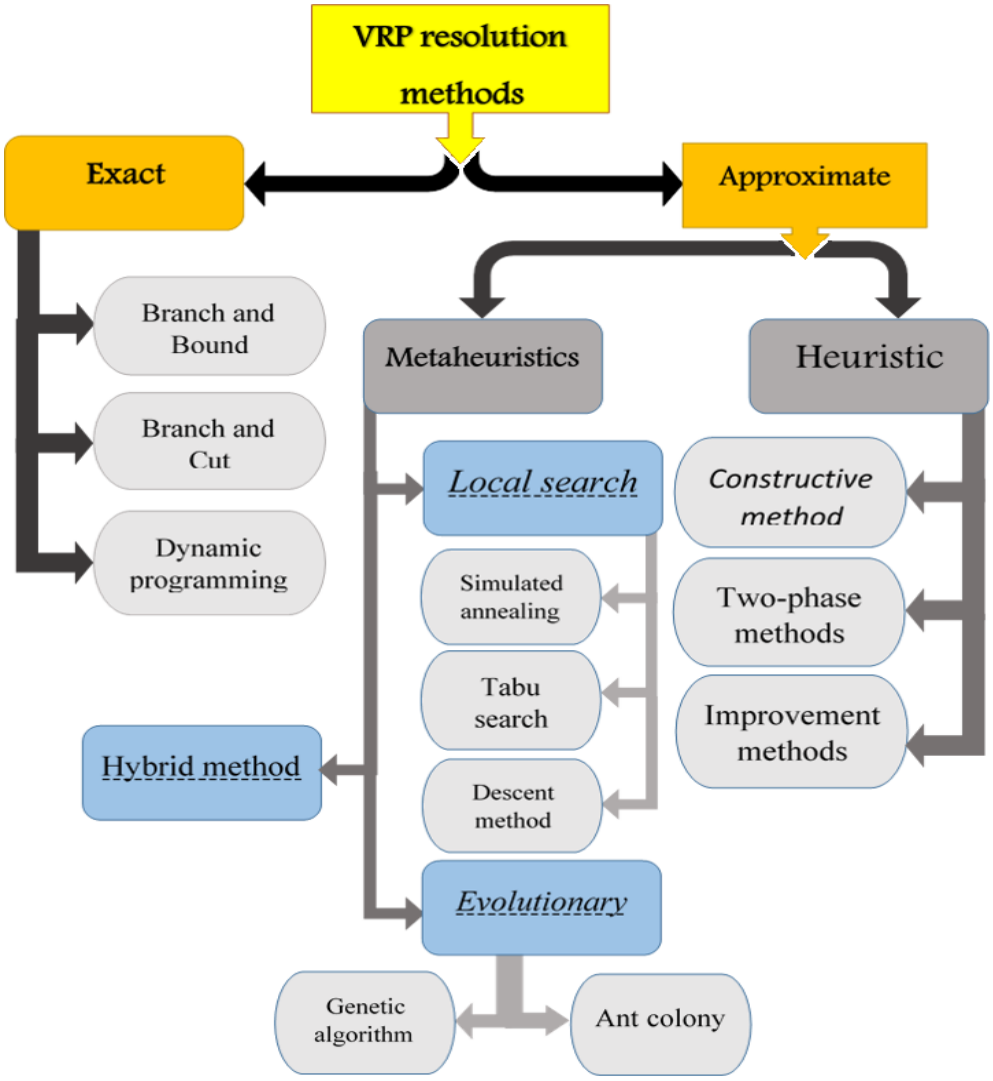


Figure I.2: Some VRP resolution methods graphical presentation.

I.7 Metaheuristics generality

I.7.1 Definition

Most Combinatorial optimization problems are hard to solve because they are difficult to formulate, additionally, choosing the proper method and the best configurations for solving these problems is also difficult due to the existence of serval methods with different parametrizations. [16]

The terminology "metaheuristic" is composed of two terms," meta" means high level, and "heuristics" means methods designed for a specific problem. Metaheuristics are generic algorithms, which can be used to solve complex optimization problems; they aim to find an approximate solution for a given problem. [14]

I.7.2 Application domain

In the last years, metaheuristics have shown that they are useful to solve a large number of problems. Due to the efficiency of complex optimization problem solving and the quality of the solution.

In real life, many problems search to find a high-quality solution in the shortest time possible. Such problems, usually take a long time to solve by using the classical exact methods, in order to solve these problems in a reasonable time, metaheuristics are used.

Nowadays, metaheuristics are used in many industrial, scientific, and logistics application domains such as neural networks, image processing, vehicle routing, network optimization, traffic optimization, and many applications of real-world optimization problems. [11]

I.7.3 Diversification and intensification

These two concepts are important for the success of the optimization problem. Diversification is the capability of the method for exploring diverse solutions of the search space in order to cover large possible solutions and avoid stagnation in local optima. On the other hand, intensification allows the exploitation of the best solutions already found in the search in order to improve their quality. [11]

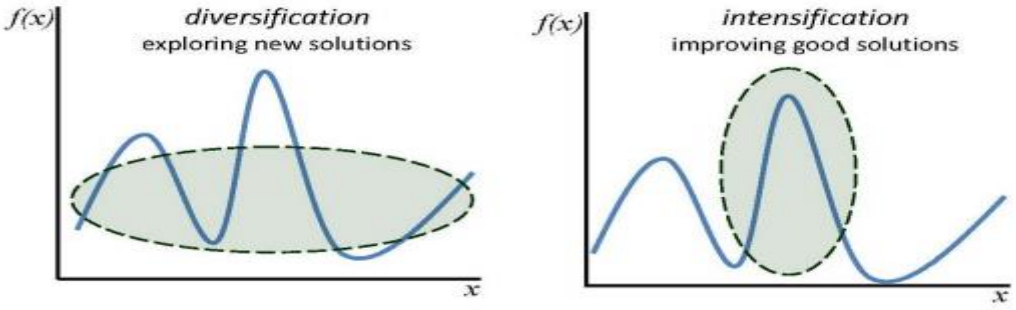


Figure I.3: Diversification and intensification. [11]

I.7.4 Metaheuristics classification

Metaheuristics can be classified into many categories, we will introduce the commonly used categories, and one metaheuristic can belong to many categories.

I.7.4.1 Trajectories methods

These methods manipulate one candidate solution, which is modified at each iteration of the algorithm and replaced with a new one found in its neighborhood, this is defined as the possible movements in the space of solutions .such methods are faster in finding local optimum solutions. Population-based metaheuristics, work with more than one candidate solution at the same time e.g., ant colony optimization and genetic algorithm. [11]

I.7.4.2 Nature-inspired method

Many methods are inspired by a natural phenomenon (biological or physical), where these methods mimic the behavior of a certain intelligent system or a physical process. e.g., genetic algorithm and ant colony algorithm inspired from the evolutionary theory and the foraging behavior of ants, respectively. [10]

I.7.4.3 Memory-usage methods

Some metaheuristics maintain a memory structure, which records information during the search .for instance short-term memory is used to reject the most recently found solutions and avoid recycling, while long-term memory is used for the exploitation and exploration features. For instance, tabu search (TS) is considered a memory-usage metaheuristic. On there are memoryless MHs, use no information from the past during the search such as SA metaheuristic.

I.7.5 Popular metaheuristics algorithms

I.7.5.1 Tabu search

TS was introduced by Glover (1986), it is a trajectory-based MH, based on a local search method, enhanced with a memory to store all the visited solutions in the past to avoid returning to them, in order to explore large solutions of search space. [11]

TS metaheuristic has various applications in solving combinatorial problems. Nowadays, many Scientifics and engineers use TS due to the simplicity of the method compared with the quality of the solutions that can give.

TS starts with one candidate initial solution $s_0 \epsilon\, C(s)$. At each iteration, generate a new candidate solutions N(s) in the neighborhood of the current solution, then the best solution among N(s) is chosen .to explore more solutions, the visited ones are avoided by maintaining a tabu list. Typically there are two types of tabu lists, one that carries history during the exploration process (long term memory), the other one keeps only the most recently visited solutions(short term memory). [15]

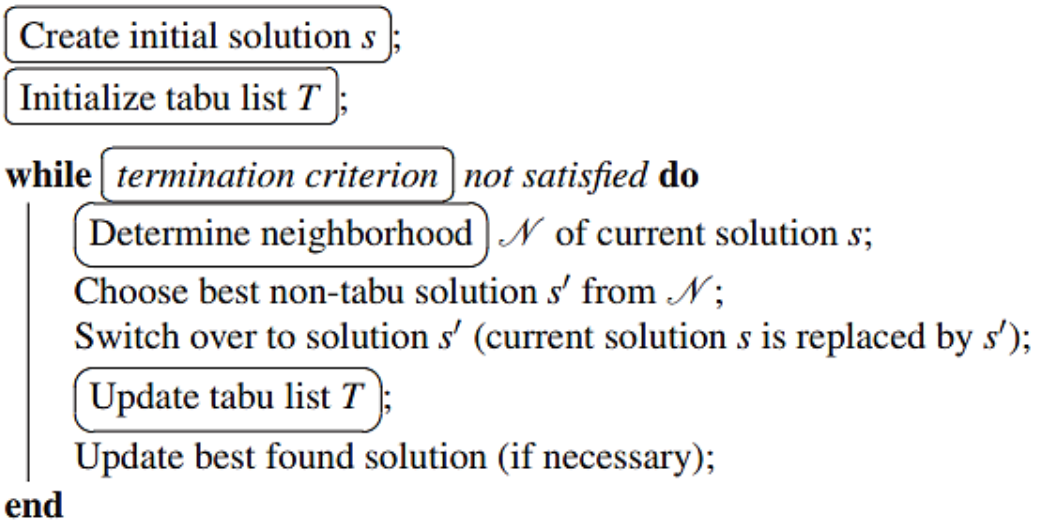


Figure I.4: Tabu search algorithm. [14]

Due to its efficiency, many combinatorial problems have used TS, and made great successes in solving the classical problems such as traveling salesman problem (TSP), vehicle routing problem….ext. [10]

I.7.5.2 Simulated annealing

SA was first introduced by Metropolis et al. (1953) .it based on the annealing process of the materials. Annealing is a process of heating a material until it reaches the temperature of melting and then cooling it slowly, in order to obtain the desired crystalline structure[12].

Simulated annealing starts with an initial solution s $\in \theta$ and an initial temperature T, then a new solution $s' \in N(s)$ is generated.SA uses the Metropolis algorithm, which describes how thermodynamics systems move from the current solution s to the new solution s', the new solution s' is accepted as the current solution by the following acceptance probability [13]:

$$P(\text{accept } s' \textit{ the current solution }) = \begin{cases} e^{\frac{[-\Delta E]}{T}} & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases} \qquad (1)$$

ΔE represents the difference in the objective value between the current solution s and the neighbor solution s':

$$\Delta E = f(s') - f(s) \qquad (2)$$

Template of simulated annealing algorithm.

Input: Cooling schedule.
$s = s_0$; /* Generation of the initial solution */
$T = T_{max}$; /* Starting temperature */
Repeat
  Repeat /* At a fixed temperature */
    Generate a random neighbor s' ;
    $\Delta E = f(s') - f(s)$;
    If $\Delta E \leq 0$ **Then** $s = s'$ /* Accept the neighbor solution */
    Else Accept s' with a probability $e^{\frac{-\Delta E}{T}}$;
  Until Equilibrium condition
  /* e.g. a given number of iterations executed at each temperature T */
  $T = g(T)$; /* Temperature update */
Until Stopping criteria satisfied /* e.g. $T < T_{min}$ */
Output: Best solution found.

Figure I.5: Simulated annealing algorithm. [18]

I.7.5.3 Genetic algorithms:

A genetic algorithm is a powerful global optimization technique based on the principles of genetics and natural selection. In the 1970s, Holland applies the operator of crossover, recombination, mutation, and selection in the studies of artificial system studies. These operators are the main components of GA for solving optimization problems. [19]

The principle of GA is based on several components, which are:

- Population
- Initialization
- Evaluation
- Selection
- Recombination
- Mutation
- Replacement

At first, an initial population is created by the initialization method. Then the fitness of each individual is evaluated. In the selection step, two solutions are chosen and recombined using the crossover operator by merging the proprieties present in the two solutions to form a new solution. After recombination, the new individuals are mutated with a percentage, which means a small modification is applied to the new solutions. In The final step, the new solutions

are evaluated and collected in a new, temporary population (P'). For the next iteration, a new population P is formed by choosing individuals from the set $P' \cup P$ to replace the old ones. [14]

```
P ← initial population;
evaluate(P);
while termination criterion not satisfied do
    P' ← recombine(select(P));
    mutate(P');
    evaluate(P');
    P ← replace(P' ∪ P);
end
```

Figure I.6: The basic genetic algorithm. [14]

I.7.5.4 Ant colony metaheuristic:

Ant colony optimization is a natural-based metaheuristic, inspired by the foraging behavior of ants. It was first introduced in 1991 by Dorigo and Colorni, the most interesting aspect of the cooperative behavior of ants it's the way they communicate with each other during the foraging process.

This intelligent system shows that the ant can find the shortest path between the nest and the food sources. On the way back to its nest from where the food exists, the ants lay down chemical trails (pheromones). The ants follow the trails with the higher intensity, which are the shortest paths, while the pheromone trails of the largest path evaporate. [17]

```
Initialize the pheromone trails ;
Repeat
   For each ant  Do
      Solution construction using the pheromone trail ;
      Update the pheromone trails:
          Evaporation ;
          Reinforcement ;
Until Stopping criteria
Output: Best solution found or a set of solutions.
```

Figure I.7: Ant colony optimization algorithm. [14]

I.8 Conclusion

In this chapter, we presented general information and basic concepts of both the problems of **VRP** and the methods of solving them. We focused on the Meta-heuristic branch of them because it is the most effective one.

Meta-heuristic algorithms are one of the best theoretical ways to solve problems of improvement in transport and distribution, especially in problems such as **VRP** and **TSP**. Their effectiveness, such as their sources of development, is inspired by nature, such as ant algorithm, bee algorithm, and genetic algorithm. All of these algorithms have been detected through the

behaviors of these creatures because they act with their natural instincts to solve their problems, whether in nutrition, reproduction, or other things.

Moreover, in the next chapter, we are going to talk about a new algorithm for another creature that scientists and researchers as of today are still trying to discover its behaviors and the way it thinks about solving problems in general.

Chapter II

Slime mold physarum polycephalum

II.1 Introduction

In the 1990s, Scientifics and researchers in computer science began focusing on biological systems for inspiration to design optimization algorithms. For around ten years, ant colony optimization was the paradigm, especially when it came to short path problems. In static environments, ant colony optimization performed well, but most optimization problems, on the other hand, are dynamic and require the algorithm to change its solution constantly. Later, ACO becomes less based on the natural behavior of ants because of the assumption that the trails laying by the ants are incapable of quickly adapting to change their foraging environment.

Consequently, the focus turned to another creature, **physarum polycepalum**, the true slime mold. This unicellular organism has no brain and no nervous system, but it has the capability to solve shortest path problems like the maze and to develop adaptive networks. [38]

II.2 Physarum polycephalum

II.2.1 Biological background

Slime Molds are among the strangest organisms in the world. Long confused with mushrooms, they are now classified as a type of amoeba. As single-celled organisms, they have neither neurons nor brains. However, for the past decade, scientists have been wondering whether slime molds have the capacity to know their environment and adjust their behavior accordingly. [31]

Slime Mold is a general term that applies to many fungal protests. The taxonomy of this group is still being revised and changes are occurring frequently. They were originally considered mushrooms, but are now classified as protists [25]. The true slime mold **Physarum polycephalum** is a single-celled amoeboid organism that is able to find the shortest path connecting food resources in a maze. [24]

Physarum polycephalum belongs to the upper class of Myxomycetes, also referred to as real viscous molds. Together with cellular molds, they form the Mycetozoa group. **Physarum polycephalum** is visibly pigmented in yellow or orange (for example, see Figure 2.1) and does not perform photosynthesis. The vegetative stage of food consumption of the life cycle of **Physarum** is called plasmodium and consists of a single amoeboid cell as with multiple diploid nuclei where all nuclei divide at exactly the same time. [25]

Figure II.1: Physarum polycephalum (slime mold). [30]

Slime molds feed on microorganisms in decaying plant matter. They may be found on the ground, on lawns, and in the forest commonly on broad-leaved logs. They are also common on the mulch or even in the mold of the leaves that collects in the gutters. They start life-like cells like amoebas. These unicellular amoebae are commonly haploid and multiply if they encounter their favorite food bacteria. These amoebas can mate if they meet the correct type of mating and form zygotes that then develop into plasmodia that contain many nuclei without cell membranes between them, which can become meters in size. A variety is often considered a viscous yellow network in and on decaying logs. Amoeba and plasmodia engulf microorganisms. Plasmodium develops into an interconnected network of protoplasmic strands. [28]

II.2.2 Life cycle

The plasmodial slime molds begin as amoeboid cells, each with a unique haploid nucleus. These can begin to feed on bacteria and multiply. Under very wet conditions, they can turn into biflagellated swarming cells. Mygale haploid amoeba reproduced by binary fission. [27]

Most organisms reproduce by cellular fission when their cells reach a certain size, that is, the nucleus is divided by cutthroat, and the cell is divided with it to produce two smaller daughter cells, each with its own nucleus. That is what haploid slime mold Amoeba does. However, the zygote behaves differently from most other organisms. Instead of cell divisions, myxomycete zygotes grow more; the nucleus is divided by vomiting, but the cell does not divide with it, resulting in a new larger cell with two nuclei. This is called coenocyte (or plasmodium). As coenocytes grow, the nucleus continues to divide. In some species, the size of the plasmodium will be centimeters with thousands of nuclei. [28]

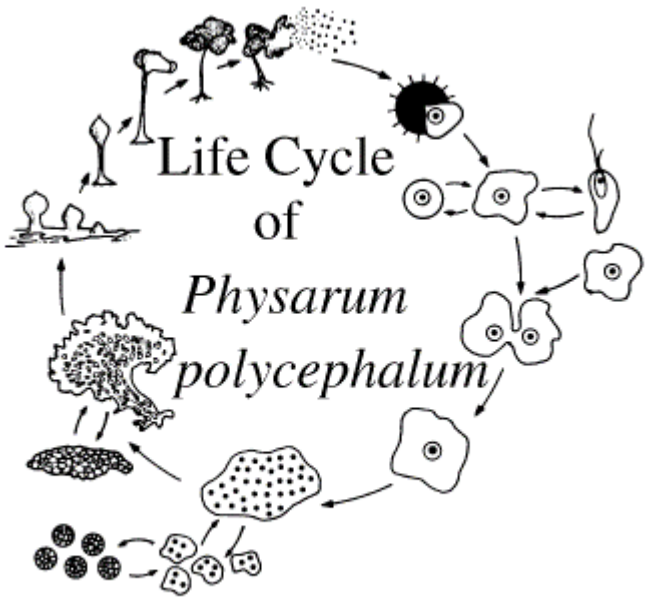


Figure II.2: Life cycle of physarum polycephalum. [32]

Physarum polycephalum appears alternately in the life cycle of generations, with one stage consisting of a stimulating, nutritious, and growing animal-like stage, and the other stage being an unappealing plant-like stage. This life cycle is superficially similar to that of fungi and involves, in the plant-like breeding phase, forming clusters of glands, sporangia, often on the tip of streams, resulting in the release of glands to produce new single cells and start the life cycle again. [29]

In dry conditions, plasmodia can also form resting structures called sclerotia, which start to develop again when humid conditions return. [28]

II.2.3 Foraging behavior

All animals need resources such as food, shelter, and colleagues. Success in achieving these goals is crucial for survival and reproduction. Indeed, the need to obtain resources and avoid predators is a key factor in the physical morphology and sensory and cognitive abilities of organisms. Given the enormous diversity of life forms and ecological sites in nature, it is not surprising to see a corresponding diversity in transport strategies [22]. Since the slime mold is one of these organisms, it also has a feeding system to which it is subjected for the continuation of the life cycle.

Plasmodium **Physarum polycephalum** behaves and moves like a giant amoeba. It feeds on bacteria, spores, and other microbial creatures. By seeking its food, the plasmodium spreads to sources of food particles, surrounds them, secretes enzymes, and digests food. Plasmodium typically forms a community of protoplasms in a food source it occupies. When multiple sources of nutrients are dispersed across the plasmodium range, plasmodium forms a network of protoplasmic tubes connecting protoplasm masses to food sources. When we consider food

sources as nodes and protoplasmic tubes as edges, we say that plasmodium develops a planar graph. [21]

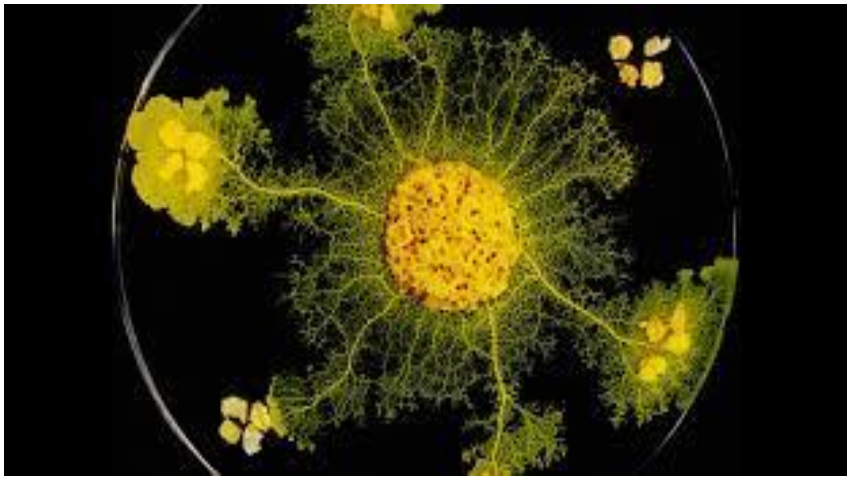


Figure II.3: The physarum polycephalum foraging behavior of . [31]

These amoebas can mate if they meet the correct type of mating and form zygotes that then develops into a plasmid that contains many nuclei without cell membranes between them, which can become meters in size. A variety is often considered a viscous yellow network in and on decaying logs. Amoeba and plasmodia engulf microorganisms. Plasmodium develops into an interconnected network of protoplasmic strands. [28]

This Plasmodium spreads like an amoeba and migrates slowly to about 1 cm. per hour. It continues to grow and feed on various microorganisms in the decomposing plant matter on which molds live. When environmental conditions change or food runs out, mold changes its behavior. It migrates to the outer surface of a decaying log if that's where it lives or migrates to the grass blades, etc. It then turns into fructificators or sporangia. In different species, they take many forms, from amorphous spots to delicate lacy structures. [28]

II.3 Physarum real biological experiments

Physarum polycephalum (P. polycephalum), a large single-cell amoeboid organism, has been experimentally shown to be capable of solving many theoretical problems. To our knowledge, several biological models have been proposed to mimic the behavior of **P. polycephalum** [23]. In this research, we will present some problems that this creature has theoretically been able to solve.

II.3.1 Maze solving

Physarum polycephalum is a single-celled amoeboid organism that is able to find the shortest path connecting food resources in a maze. [24]

According to a 2008 study by Hickey and Noriega, **physarum polycephalum** has demonstrated an ability to solve labyrinth problems, which is a standard test of intelligence in animal psychology. In one Nakagaki and Al experiment in 2000, a large piece of plasmodium was cut into a number of smaller pieces, which were then distributed at intervals in a two-dimensional physical labyrinth structure. Initially, the pieces merged to form a single organism

that filled the entire maze, but when nutritious blocks of oatmeal (technically, it is the bacterial colonies on the oat flakes, rather than the oat flakes themselves, which are harvested by the slime mold) were placed in two places in the maze, the pseudopodia reaching dead ends decreased. This gave a single thick pseudopodium covering the shortest path between the nutrient blocks, indicating the capacity of the viscous slime mold to solve the maze by adapting its physical structure. [22]

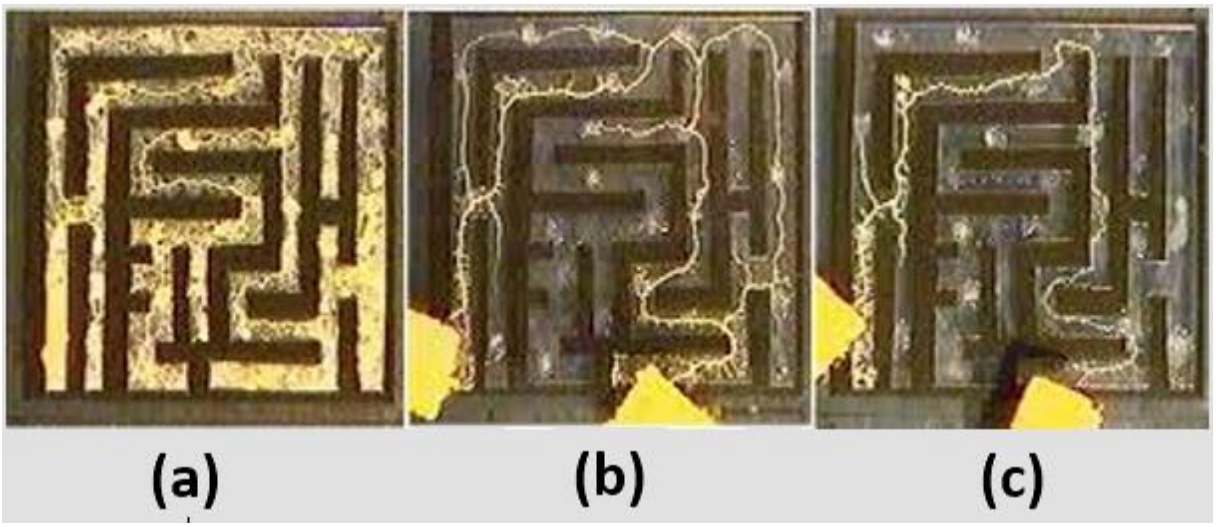


Figure II.4: Physarum polycephalum solution to the maze over time. [20]

At first, this object spreads through the entire labyrinth (Maze) and covers it up to the dead ends of it as shown in (a). Then, four hours later, the pipes spread across the blocked roads of the maze disappear, and only the pipes connecting the food sources (the maze exits) remain so that the shorter the road between the food sources, the thicker the slime mold pipe and vice versa as in (b). Moreover, after another four hours, only the shortest tube remains the link between the food sources and its thickness increases as the other tubes disappear as in figure (c). Therefore, scientists and researchers began to discover the algorithms of this creature in order to solve the problems of finding the shortest path and many other problems.

II.3.2 Networks construction

The slime mold has received a lot of attention after it has been proven to solve a maze. It has been shown that the mold algorithm can overlook dead ends in a maze and choose a shorter path from several alternative means. A Steiner tree building capacity of the slime mold was then noticed and studied. The construction of a Steiner tree is a major network optimization problem that is similar to the construction of the minimum spanning tree (MST) [26], in addition to the problem of Tokyo Rail System.

The yellow slime mold **Physarum polycephalum** grows as a single cell that is large enough to be seen with the naked eye. When it encounters many separate food sources in space, the mold cell surrounds the food and creates tunnels to distribute nutrients. In the experiment, researchers led by Toshiyuki Nakagaki of Hokkaido University in Sapporo, Japan, placed oat flakes (a delicacy of slime mold) in a pattern that mimics how cities are scattered around Tokyo, then released the slime mold. [33]

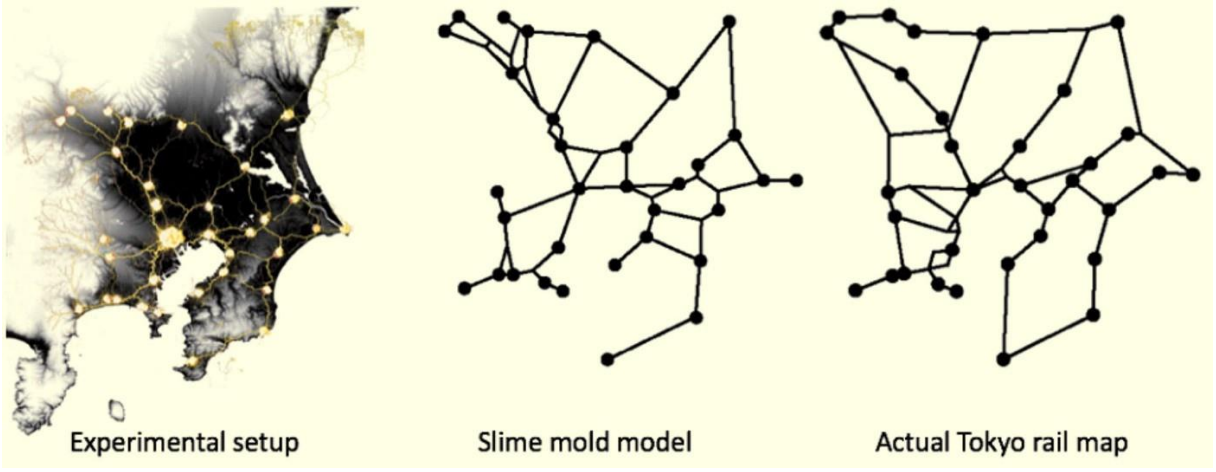


Figure II.5: Comparing the solution of the Physarum polycephalum to a problem of Tokyo Rail System with the current planning of the city. [34]

At first, this creature spread all over the scheme, just like it did in the maze solution, but hours later, it formed trails with its pipes connecting all the food points, and then it got rid of the long tracks, so it would be the perfect solution for the train network.

II.4 Physarum polycephalum modeling

II.4.1 The flow conductivity model

The flow conductivity model simulates the adaptation of the tube's radius during the foraging behavior of the **physarum polyphacrlum** due to flow throw the system. The radius of these tubes is proportional to the amount of flow that carries. Where the shorter tubes carry more flow than longer tubes. The total flow in and out of the tubular system remains constant to conserve the mass of the **physarum polycephalum**.

This model can be represented as a graph with N nodes and E edges where e_{ij} is the tube that connects the nodes N_i and N_j, D_{ij} is the conductivity of the tube e_{ij}, Q_{ij} is the flux through e_{ij} and L_{ij} is the length of eij. The following formula expresses the flux from node N_i to N_J through e_{ij}:

$$Q_{ij} = \frac{D}{L_{ij}}(p_i - p_j) \qquad (1)$$

Where p_i and p_j are the pressure values of the nodes Ni and Nj, respectively. The mass conservation of **physarum polycephalum** is modeled by the following equation:

$$\sum_i \frac{D}{L_{ij}}(p_i - p_j) = \begin{cases} -I_0 & if\ j = in \\ I_0 & if\ j = out \\ 0 & otherwise \end{cases} \qquad (2)$$

Where $-I_0$ is the inflow flux and I_0 is the outflow flux.

Finally, the conductivity D is described by the following equation:

$$\frac{d}{dt}D_{ij} = f(|Q_{ij}|) - rD_{ij} \qquad (3)$$

Where r is a decreasing rate constant of the tube, $f(|Q_{ij}|)$ is the growth function of the tube. The survived tubes depend on the convergence of D_{ij}, when D_{ij} converges to 0, the corresponding tube e_{ij} is removed, while only the tubes with high values of D_{ij} survive. [37]

II.4.2 Agent-based model

This model the most approach to the complex behavior of **physarum polycephalum**. It was first proposed by Jones in 2009. The physarum is modeled by a population of agent particles moving on a two-dimensional flat plane, each particle have three sensors used to measure the concentration of a local variable called 'attractant', which exists on the same plane as the particles. These particles can make move forward or make a rotation to the left side or the right side depending on how many attractants are underneath each sensor. The interesting part of these particles is that they are themselves the source of the attractant, for each successful move of a particle it lay down an amount of attractant. If a particle tries to make a move to an intended place that is already taken, it chooses a new random direction to attempt in the next time without deposing the attractant.

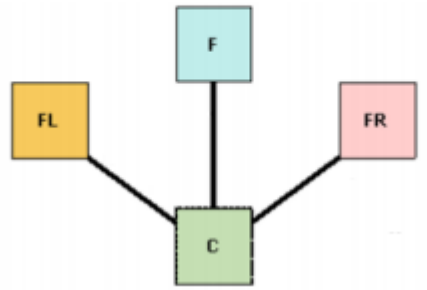


Figure II.6: A particle in the Physarum model, showing the particle position C and the Offset sensor positions FL, F, and FR. [35]

The behavior of these particles depends on the initial parameter settings. The location and the facing for each particle are randomized. The sensor angle (SA) and the rotation angle (RA) of the particles are particularly important, which their values are set universally fixed and do not change. For a set of experiments the rotation angle RA and the sensor angle, SA was set to 45 and 15 degrees, respectively. This means that the particles can rotate in both directions 45 degrees, and the two sensors (FL and FR) were positioned 15 degrees from the front sensor with these parameters, the particles develop a dynamic network spontaneously.

The particles at the beginning take random movements, but over time, they combine and draw a path. A path like this is made up of particles moving in both directions along an approximate line of attractants. The particles deposit attractant as they move along it, reinforcing the path. Then a dynamic network is formed from these paths. [35]

II.4.3 Cellular automation model

In 2008, Gunji et al. proposed a new model of **physarum polycephalum** to design networks with high quality. In this model, the optimization process corresponds with the properties of the real cells. This model consists of two phases, the development phase, and the foraging phase. In the development, phase the components of the cell aggregate from an initial seed, while in the foraging phase the cell behaves just like that in the vegetative state of **physarum polycephalum**. Given a planner lattice, each lattice site has a state, a cell is described as an aggregation of sites in particular states: the inside (state 1) is surrounded by a boundary (state 2) in a lattice space consisting of the outside (state 0). The boundary state is assumed to correspond to an assembly of cytoskeleton fibers in the body of **Physarum polycephalum**. In the foraging phase, a cell eats 0, and this causes migration and modification of the cell, which corresponds to the process of softening a particular part of the membrane of **physaruym polycephalum**. The protoplasmic flow to the softened areas by the transportation of the eaten 0, which is termed the bubble. During this process, the bubble is accompanied by the cytoskeleton, which results in a re-organization of the distribution of the cytoskeleton.

This model was applied to simulate the **physarum** motion and solve the classical Steiner tree problem in planes .however Liu et al. pointed out that this model has a low efficiency since the cell contains only one bubble. Therefore, they improved the original model by increasing the number of bubbles in the cell, which has more efficiency and stability than the original one. [36]

II.5 Physarum solver

The foraging behavior of the **physarum polycephalum** attracts many engineers and Scientifics trying to design an algorithm that mimics the intelligent behavior of the single-cell **Physarum polycephalum**. In the last few years, a new algorithm has been proposed called **physarum solver**; this algorithm implements the flow conductivity model of **Physarum polycephalum**. It proved that could solve easily the problems of finding the shortest path between two nodes in the weighted graphs.

Input: L: the cost matrix, s: the start node, e: the destination node

Output: the shortest s-e path S^*.

1: $G \leftarrow \infty$ // current conductivity gap

2: $\underline{G} \leftarrow 10^{-6}$ // a predefined conductivity gap

3: $itr \leftarrow 0$ // number of iteration

4 $D_{ij} \leftarrow (0, 1]$ $(\forall i, j=1,2\ldots.N)$ // initial conductivity values

5: $Q_{ij} \leftarrow (0, 1]$ $(\forall i, j=1,2\ldots.N)$ // initial flow values

6: $P \leftarrow 0$ $(\forall i=1,2\ldots.N)$ // initial pressure vector

7: **While** *termination criteria not satisfied* ***do***

8: $itr \leftarrow itr + 1$

9: Set $p_e \leftarrow 0$ // the pressure at the ending node **e**

10: Calculate the pressure of every node in the network

11: Using the equation 2;

$$\sum_i \frac{D}{L_{ij}}(p_i - p_j) = \begin{cases} -I_0 & if\ j = in \\ I_0 & if\ j = out \\ 0 & otherwise \end{cases}$$

12: Update the flow values **Q** by using the equation 1;

$$Q_{ij} = \frac{D}{L_{ij}}(p_i - p_j)$$

13: Calculate the new conductivity $\overline{\mathbf{D}}$ by the equation 3;

$$\frac{d}{dt}D_{ij} = f(|Q_{ij}|) - rD_{ij}$$

14: $G \leftarrow sum(|\overline{D} - D|)$

15: $D \leftarrow \overline{D}$

16: **End while**

17: **Return** S^*

Figure II.7: Physarum solver algorithm. [39]

The algorithm takes as an input a cost matrix **L**, the start and the end nod **s**, **e**, respectively.

As a solution, the algorithm returns the shortest path possible S^*.

The way that this algorithm works, simply it continually keeps updating the flow Q and the conductivity D values, each step of the algorithm the edges that can form the shortest path from the source node to the destination node grow up while the edges that cannot be a part of the shortest path disappear. When the termination criteria is satisfied, the last flow values are obtained which contain the shortest path.

II.6 Conclusion

In this chapter, we touched on this new algorithm (the slime mold algorithm) and all about the creature inspired by it (**physarum polycephalum**). In the coming chapters, we will be applying this algorithm to a real problem (solving the problems of a factory or company) and then comparing the results of this algorithm with other algorithms to study its efficiency and effectiveness in this area.

Chapter III

Solving TSP and CVRP using physarum solver

III.1 Introduction

In this chapter, we will apply one of the true slime mold models to solve the capacitated vehicle problem **CVRP** and the traveling salesman problem **TSP**, we chose the flow conductivity model because is the most appropriate for solving this kind of problem, in the end of this chapter we compare our results with other results obtained by different metaheuristics.

III.2 Art state

There are few studies that was interested to solve the vehicle routing problem and the traveling salesman problem with the physarum polycephalum based algorithm. In this section, we will present tow of them, one of the VRP and the other for the TSP.

III.2.1 Solving vehicle routing problem with multi-agent physarum model

In this subsection, we present a study consist of solving the VRP one of the slime mold model, the author use the multi-agent model(mentioned in chapter II section II.) to solve the tow dimensional Euclidian capacitated vehicle routing problems.

The model consist of a large set of particles, which are distributed around the depot. These particles split into different kinds, each kind represents a vehicle. Each city is attracted by many particles and the particles of the same kind are attracted to each other. The route of each vehicle is constructed by a number of cities that are covered by a population of particles.

The model was implemented on 8 virtual different problem sets, each with different problem parameters, each with 20 different problems that were run 20 times each. The parameters of each problem set vary from 2 vehicles on 10 cities with 6 capacity each, to 3 vehicles on 20 cities with 9 capacity each. For the problem sets with fewer cities and larger capacities, the performance was superior in general compared to those with more cities. in the end The author concludes that the model solution quality is not competitive.[35]

III.2.2 A Two-Way Parallel Slime Mold Algorithm by Flow and Distance for the Travelling Salesman Problem

This article was used the SMA (slime mold algorithm) for solving the TSP and it was proposed a new method named Two-way Parallel Slime Mold Algorithm (TPSMA).

The problems that worked on solving in this method was get from the simulation of Traveling Salesman Problem Library (TSPLIB): a library has a lot of Traveling Salesman Problems examples, these problem sets are : **ulysses16**, **city31**, **eil51**, **gr96**, and **bie**r**127**.every last digit in the code means the number of cities . the following table present the exprimental data's resultat of TSP data .

Table III.1: Experimental data's results of ulysses16, city31, eil51, gr96 and bier127. [45]

TSP data	Results of SMA	Results of SMA	Improved percentage
Ulysses16	103.1746	77.8372	24.56%
City31	27073	17300	36.10%
Eil51	798.9	464.3	41.88%
Gr96	1178	591	49.83%
Bier127	274870	129390	52.93

In order to evaluate the efficiency of the proposed algorithm the author made a comparison between **TPSMA** and two other metaheuristics (GA and ACO) .the experiments was applied on **TSP** from **TSPLIB** (euk51, eil76, lin105, bierl127, kroa200, and gi262).the results are shown in the following table:

Table III.2: Comparison of optimization results and algorithm features. [45]

name		**GA**	**ACO**	**TPSMA**
Results of path length	Eil51	519	453	464
	Eil76	727	583	620
	Lin105	30.167	15.303	16.424
	Bier127	196.276	128.147	129.390
	kroA200	87.786	33.471	34.972
	gil262	7769	2779	2881
Algorithm time complexity		N^3	N^4	N^3

In this study the author presented an improved version of the **physarum polycephalum** algorithm named TPSMA, through the experimental results of TSPLIB data , the results that are obtained by TPSMA are obviously better than the standard **physarum polycephalm** algorithm results. The author also made a comparison between the TPSMA ,GA, and ACO results, the ACO and the TPSMA results were very close and better in the most of the examples.

III.3 The work objective

The general objective of this work is the use of meta-heuristics to solve the problem of vehicle routing with capacity constraints, but we will also solve the traveling problem (TSP). The aim is to reduce the distance traveled by all vehicles while taking into account the visitation of all customers N once per customer, we will choose: **the physarum solver Algorithm**. The purpose of using this new algorithm is to monitor its effectiveness in solving these kinds of problems (VRP and TSP) and compare its results with the results of other algorithms to choose the optimal solution and the optimal algorithm to solve these problems.

III.4 Material Environment

This work has been implemented on a PC with the following characteristics:

- The Processor: Intel (R) Core (TM) I3-4005U CPU @ 1.70GHz.
- RAM: 4 GB.
- Under a 64-bit operating system.

III.5 Development environment

- **Python**

Python is an easy-to-learn open source programming language that is expandable, adopting OOP. Python is an interpreted, versatile language that is widely used in many areas, such as building independent software using graphical interfaces and in web applications, and can be used as a text programming language to control the performance of many software such as Blender. In general, Python can be used to make simple programs for beginners and to accomplish megaprojects at the same time. Programming beginners are often advised to learn this language because it is among the fastest-learning programming languages. [40]

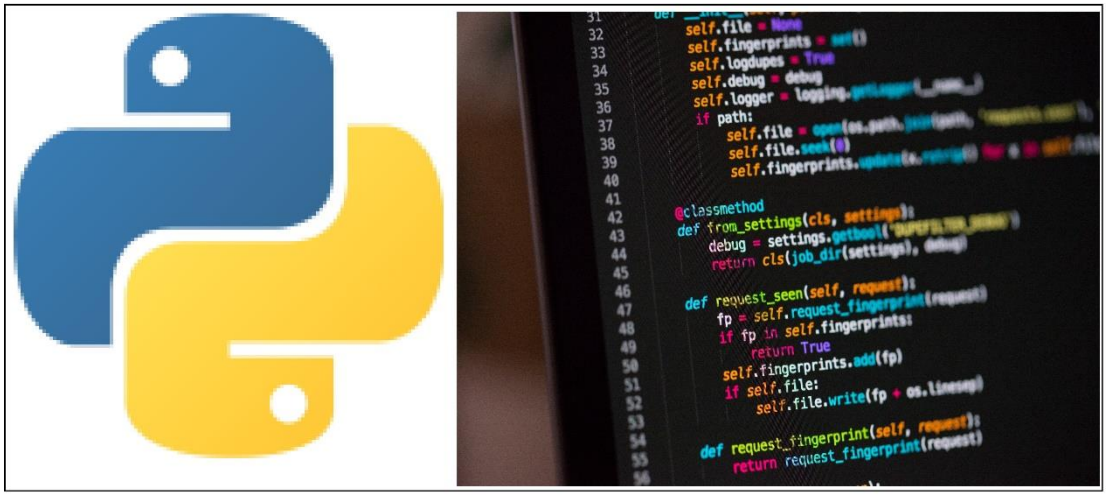


Figure III.1: Programing language Python.

Python has an active community and has many special-purpose software libraries that have been programmed by people from that community. For example, a PyGame library provides a range of functions for game programming. Python can also handle many types of databases such as MySQL and more. [41]

Table III.3: Characteristics of the used tool, python.

Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	20 February 1991; 31 years ago

Stable release	3.10.4 Edit on: 24 March 2022;
Preview release	3.11.0b1: Edit this on 7 May 2022;
Typing discipline	Duck, dynamic, strong typing; gradual (since 3.5, but ignored in CPython)
operating system (OS)	Windows, Linux/UNIX, macOS and more
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyo (prior to 3.5), .pyw, .pyz (since 3.5)[9]
Website	www.python.org
Major implementations	CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython
Dialects	Cython, RPython, Starlark
Influenced by	ABC, Ada, ALGOL 68, APL, C, C++, CLU, Dylan, Haskell, Icon, Lisp, Modula-3, Perl, Standard ML
Influenced	Apache Groovy, Boo, Cobra, CoffeeScript, D, F#, Genie, Go, JavaScript, Julia, Nim, Ring, Ruby, Swift

- **Visual studio code IDE**

Visual Studio Code (vscode) is a Microsoft text editor. The editor is open source and works on Windows, Mac OS, and Linux operating systems. The editor depends on The Electron environment.

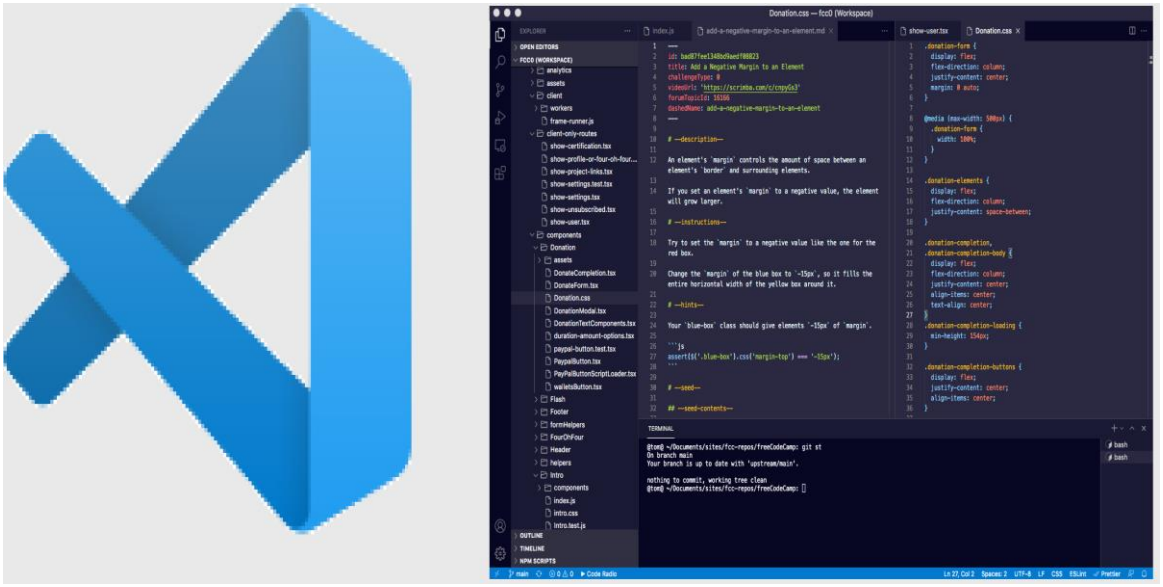


Figure III.2: The text editor Visual Studio Code.

Visual Studio Code essentially supports many different software languages, as it is not limited to a specific area. One of the languages supported in this editor is C&C++ and do not forget Microsoft C Sharp as well as supporting popular web languages, HTML, CSS, and JavaScript and supporting various back-end languages and lots of other languages.

It also essentially supports Python - which is of interest to us - and the editor has additions to support more languages if a language is not primarily supported.

Table III.4: Characteristics of the used tool, Visual Studio Code.

Developer(s)	Microsoft
Initial release	April 29, 2015
Stable release	1.67.1 (10 May 2022)
Preview release	1.68.0
Repository	https://github.com/microsoft/vscode
Written in	TypeScript, JavaScript, HTML, and CSS
Operating system	Windows 7 or later, OS X 10.10 or later, Linux
Platform	IA-32, x86-64, ARM64
Size	Windows: 40.8–68.3 MB Linux: 46.5–66.6 MB

	MacOS: 67.5 MB
Available in	14 languages
List of languages	English (US), Simplified Chinese, Traditional Chinese, French, German, Italian, Portuguese (Brazil), Japanese, Korean, Russian, Spanish, Bulgarian, Hungarian, Turkish
Type	Source code editor
License	**Source code:** MIT License **Binaries built by Microsoft:** Proprietary software
Website	https://code.visualstudio.com/

III.6 Resolution method description

The basic idea of the **physarum** solver algorithm is based on the feedback mechanism of the **physarum polycephalum** during the foraging behavior. It expands in all directions and stretches itself to cover the area that surrounds it. After a period, it forms paths called pipelines between all the discovered food sources; these pipelines optimize the transportation of the chemical sol in the tubular network.

Our approach consists of using the **physarum solver** to calculate the flow values between each node and the other nodes, then depending on these flows we solve the **TSP** and the **CVRP** problems in which the cities or the customers represent the nodes and the cost matrix L_{ij} represents the distance matrix. The following figure illustrates the process of calculating the flow values:

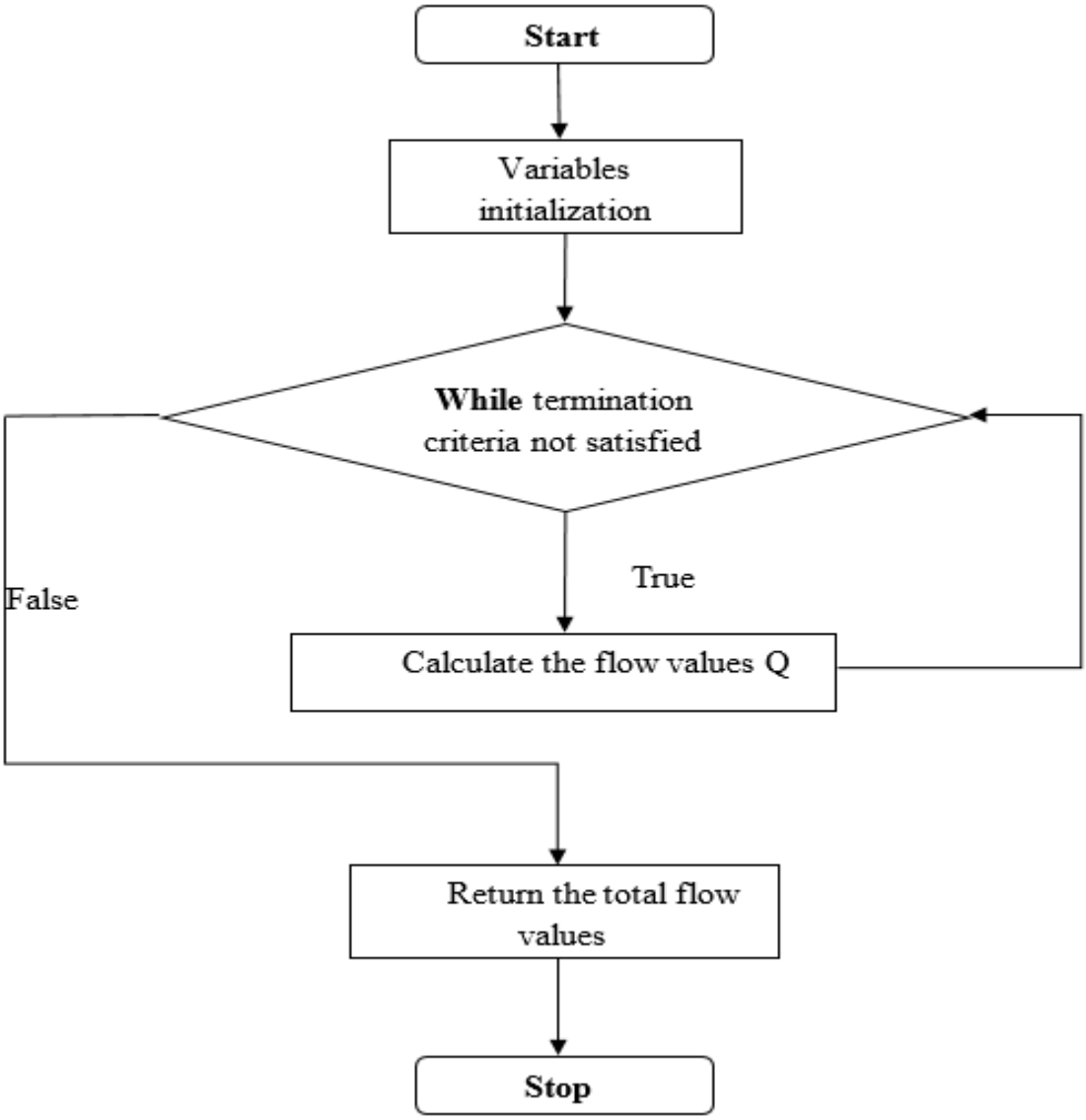


Figure III.3: The physarum solver diagramme.

III.7 TSP resolution with physarum solver

The traveling salesman problem resolution consists of finding an optimal route for one vehicle that visits given cities and then returns to the starting point in order to minimize the travel distance.

The main conditions of **TSP** are never to visit a city twice and all cities should be visited, in the end, the vehicle must return to the starting point.

III.7.1 Resolution steps

Step 1: initialize the **physarum solver** parameters, D_{ij}, Q_{ij}, I_0, the start, and the destination nodes (The cost matrix L_{ij} in this case represents the distance matrix).

Step 2: according to equation (1), the flow Q_{ij} is calculated.

Step 3: keep updating the flow values and the conductivity by the equations (2) and (3), then the flow values are obtained by iterating until the termination criteria is satisfied.

Step 4: according to the final flow values, we define a starting point, and the point which has the largest flow value from the starting point is selected as the next point P_{next}. After completing the selection of one point, the selected point P_{next} will be the current point i for the process of selecting the next point. The selected i point is stored in $L_{best.}$

$$Q_{inext} = \max \{|Q_{i1}| , |Q_{i2}| \ldots\ldots\ldots\ldots |Q_{im}|\}$$

$$L_{best} = \{i_{1,}i_{2,}\ldots\ldots\ldots\ldots i_n\}$$

Q_{inext} represents the pipeline with the largest flow value from the current point i to the other points.

This process continues until all points are selected and the full path L_{best} is obtained.

III.8 CVRP resolution with physarum solver

III.8.1 CVRP parameters

N: number of customers or clients.

T: the demand of each customer.

K: number of vehicles.

C: the capacity of the vehicle.

III.8.2 CVRP constraints

-One customer can be served only by one vehicle.

-The sum of the demands of the customers, which define one route, should not exceed the capacity of the vehicle.

III.8.3 Resolution steps

The resolution of the capacitated vehicle routing problem by the physarum solver is almost the same as the TSP resolution except that the capacity of the vehicle should be taken into consideration.

Step 1: initialize the **physarum solver** parameters, D_{ij}, Q_{ij}, I_0, the start, and the destination nods (The cost matrix L_{ij} in this case represents the distance matrix).

Step 2: according to equation (1), the flow Q_{ij} is calculated.

Step 3: keep updating the flow values and the conductivity by the equations (2) and (3), then the flow values are obtained by iterating until the termination criteria is satisfied.

Step 4: according to the final flow values we define the starting point, a point that has the largest flow value from the starting point is selected as the next point P_{next}.at each

selection, we check if the sum of demands of the selected points is within the range of the truck capacity. Once it exceeds the truck capacity we stop the process selection then a path L_{best} is obtained.

Step 5: keep iterating in step 4 until a path is formed for each truck.

III.9 Methods to measure the distance matrix

The distance matrix is an NxN matrix that represents the distances between each geographic location (N is the number of points). There are two ways to measure the distance between two geographic locations or more. The first one is the tow dimensional Euclidean distance; this one is measured by the Pythagorean Theorem or uses software called QGis.

The second way is to measure the real distance (the routing distance), this could be done manually, but this is a bad option. The best option is to use a web server that provides distance-measuring services, there are two popular web servers commonly used, **the open-source routing machine server** (OSRM), and the **GraphHoper** server, and both of them have their own application-programming interface (API).

III.9.1 GraphHopper server

The GraphHopper is a fast and effective memory steering engine released under Apache License 2.0. It can be used as a Java library or an independent webserver to calculate distance, time, rotation instructions, and many road features for a route between two or more points. [44]

III.9.2 Open source routing machine (OSRM) server

OSRM server is a high-performance routing and map-matching server. The Open Source Routing Machine (OSRM) is an open-source router designed for use with data from the OpenStreetMap project. [42]

III.10 Problem definition

A factory manufactures sachets of cow's milk, which it distributes daily to 20 customers.

To transport the goods to the various customers, the factory has 03 homogeneous trucks, i.e. of the same loading capacity, which is 35 boxes for each truck, and each box can contain up to 10 sachets of milk.

The demands of the customers and the geographic coordinates are shown in **Table III.5**

Table III.5: demands of customers.

Site	Latitude	Longitude	Demand		Site	latitude	Longitude	Demand
Factory	34.877760°	-1.326256°			C11	34.879038°	-1.325492°	7 boxes
C1	34.878290°	-1.327454°	2 boxes		C12	34.875496°	-1.324697°	2 boxes
C2	34.879365°	-1.327990°	3 boxes		C13	34.877411°	-1.323446°	5 boxes
C3	34.878402°	-1.329517°	4 boxes		C14	34.874255°	-1.324057°	1 boxes
C4	34.876132°	-1.329645°	1 boxes		C15	34.874912°	-1.322001°	2 boxes
C5	34.879735°	-1.329644°	2 boxes		C16	34.876732°	-1.323242°	3 boxes
C6	34.880326°	-1.326561°	5 boxes		C17	34.878881°	-1.323038°	6 boxes
C7	34.880487°	-1.330182°	3 boxes		C18	34.881160°	-1.322845°	10 boxes
C8	34.882135°	-1.330063°	9 boxes		C19	34.879148°	-1.321031°	2 boxes
C9	34.881190°	-1.326941°	3 boxes		C20	34.875789°	-1.320439°	5 boxes
C10	34.880753°	-1.324457°	6 boxes		C21	34.873823°	-1.327595°	3 boxes

III.11 Experimentation

The **physarum solver** source code applied here is written in python language and retrieved from [43].

The following figure represents the main source code for solving the traveling problem and the capacitated vehicle routing problem.

```python
from routing_APIs.cordinates import cordinates
from routing_APIs.distance_matrix_APIs import ORSM , GraphHoper
from VRP_TSP_LIB.solve import solve
from drawing.plot import plot
# define files
cordinates_file = "Mini Projet VRP GI913 2021-2022.KMZ"
GH_distance_matrix_file = "GraphHoper_distance_matrix.csv"
ORSM_distance_matrix_file = "ORSM_distance_matrix.csv"

def main():
    demand = [0, 2, 3, 4, 1, 2, 5, 3, 9, 3, 6, 7, 2, 5, 1, 2, 3, 6, 10, 2, 5, 3]
    # cordinates object
    cordinates_object = cordinates(cordinates_file)
    # get the dictionary that contains rll cordinates
    my_cordinates = cordinates_object.get_cordinates()
    GH_distance_object = GraphHoper(cordinates = my_cordinates , key="e594dc06-0a72-4288-921f-2dfb9c5ec495")
    GH_distance_object.GraphHoper_get_distanc_matrix(
        file = GH_distance_matrix_file)
    # problem object
    problem_1 = solve(GH_distance_matrix_file)
    print(problem_1.TSP(0))
    print(problem_1.VRP(3 , 35, demand))
if __name__ == '__main__' :
    main()
```

Figure III.4: The main source code interface.

Test 1

In this test, we will apply the **physarum solver** to solve the problem that was defined in **section III.9** as a **TSP**, so we consider that the factory has only truck with an enough capacity to serve all the customers.

- **Physarum solver parameters:**

  Inflow flux I_0: -1.

  $\underline{G} = 10^{-6}$

  $\mathbf{D_{ij} = 1}$ ($\forall i, j=1,2\ldots.22$)

  $\mathbf{Q_{ij}} = 0$ ($\forall i, j=1,2\ldots.22$)

  $\mathbf{P} = 0$ ($\forall i=1,2\ldots.22$)

  Number of iteration: 5.

- **TSP parameters:**

  Number of clients: 21.

  Number vehicles: 1.

The figure below represents the graphical representation of the solution

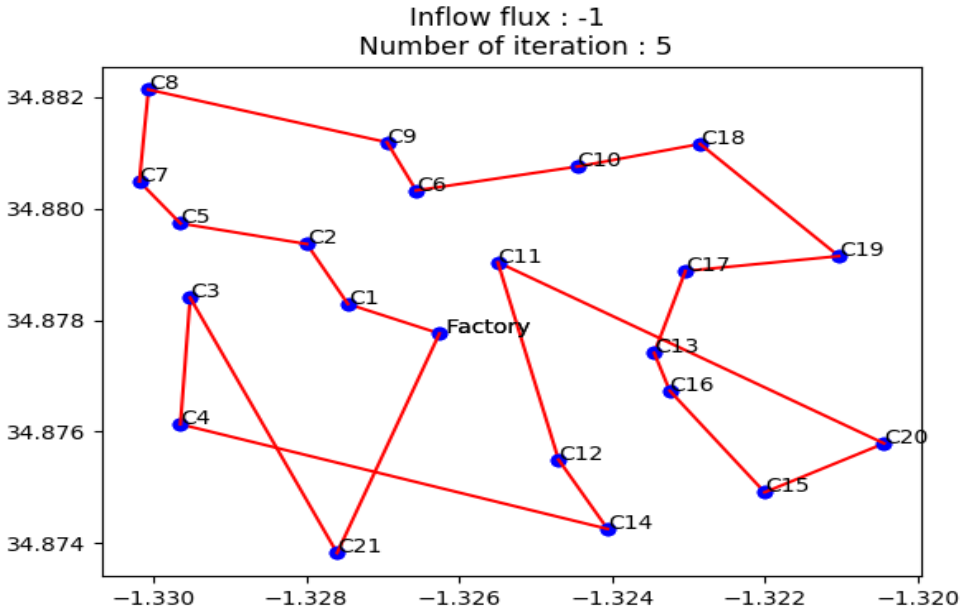


Figure III.5: The graphical window of execution.

Test 2

This test is a complementary for **test 1**,The objective is to see the influence of the number of iterations on the final solution. We set the inflow flux I_0 value and test it with different numbers of iterations.

- **Physarum solver parameters:**

  Inflow flux I_0= -1.

  G = 10^{-6}

  D_{ij} = 1 (∀i, j=1,2….22)

  Q_{ij} = 0 (∀i, j=1,2….22)

  P = 0 (∀i=1,2….22)

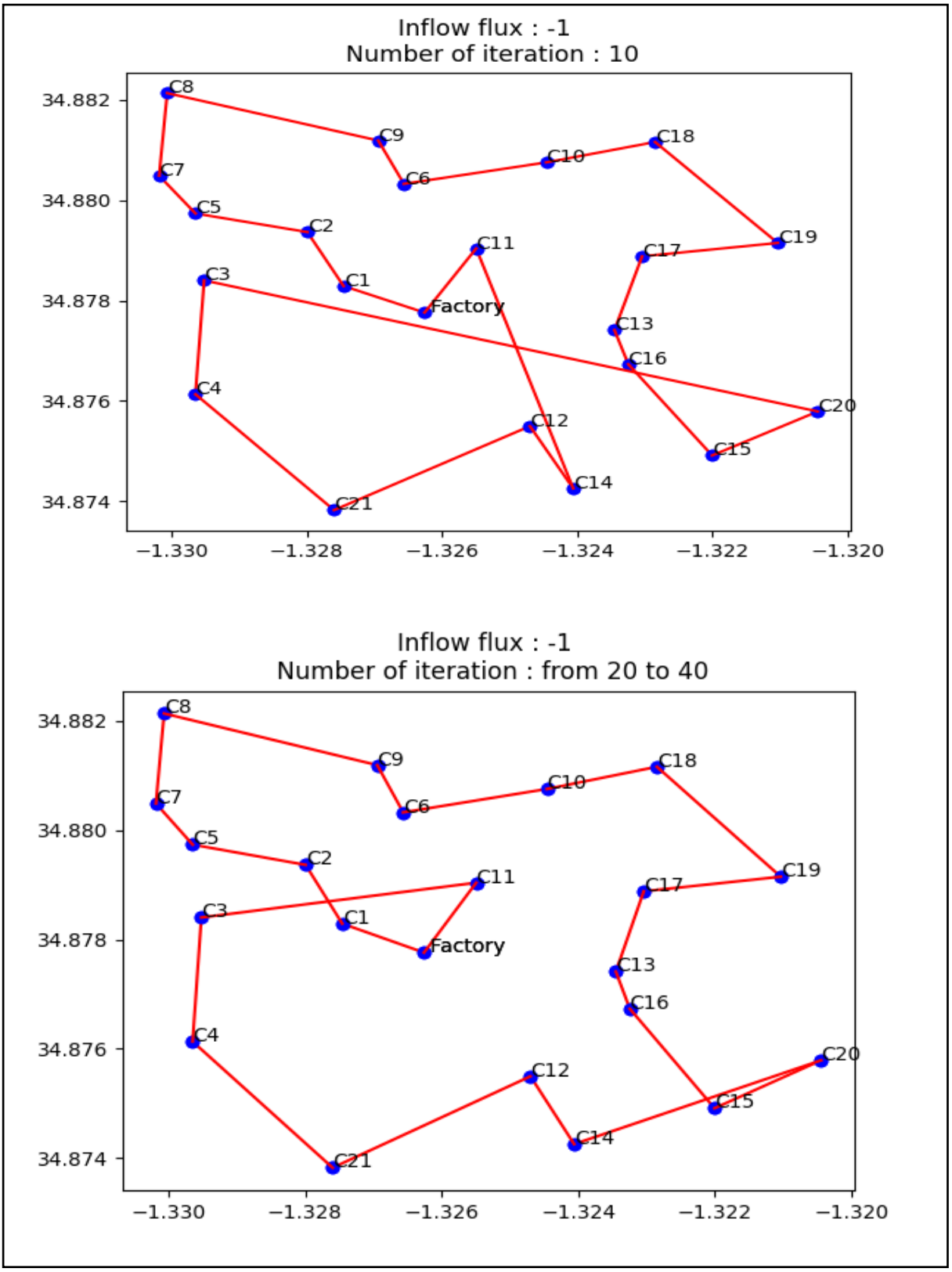
  Number of iterations: from **10** to **100**.

- **TSP parameters:**

  Number of clients: 21.

  Number of vehicles: 1.

The results are plotted in the figure **III.6**


Inflow flux : -1
Number of iteration : 10
C8
C9
C18
C10
C7
C6
C5
C2
C11
C19
C17
C3
C1
Factory
C13
C16
C4
C20
C12
C15
C14
C21
34.882
34.880
34.878
34.876
34.874
−1.330
−1.328
−1.326
−1.324
−1.322
−1.320
Inflow flux : -1
Number of iteration : from 20 to 40
C8
C9
C18
C10
C7
C6
C5
C2
C11
C19
C17
C3
C1
Factory
C13
C16
C4
C20
C12
C15
C14
C21
34.882
34.880
34.878
34.876
34.874
−1.330
−1.328
−1.326
−1.324
−1.322
−1.320

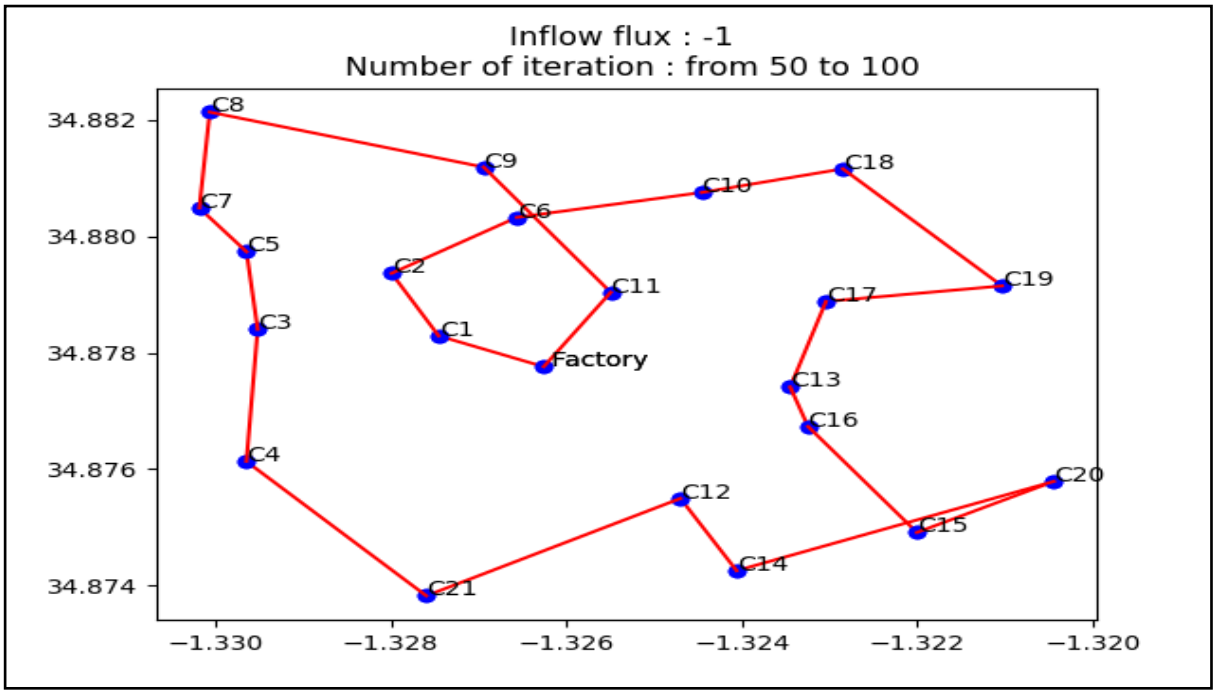


Figure III.6: The graphical window of execution.

Table III.6: Experimentation for measuring the iterations number influence.

Number of iteration	Results(meter)
10	7602.2
From 20 to 40	5891.8
From 50 to 100	5710.0

Test 3

In this test, we solve the **CVRP** problem that was defined in **section III.9** with the following parameters**:**

- **Physarum solver parameters:**

  Inflow flux I_0= -1.

  $\underline{G}$ = 10^{-6}

  $\mathbf{D_{ij}}$ **= 1** (∀i, j=1,2….22)

  $\mathbf{Q_{ij}}$ **=** 0 (∀i, j=1,2….22)

  P = 0 (∀i=1,2….22)

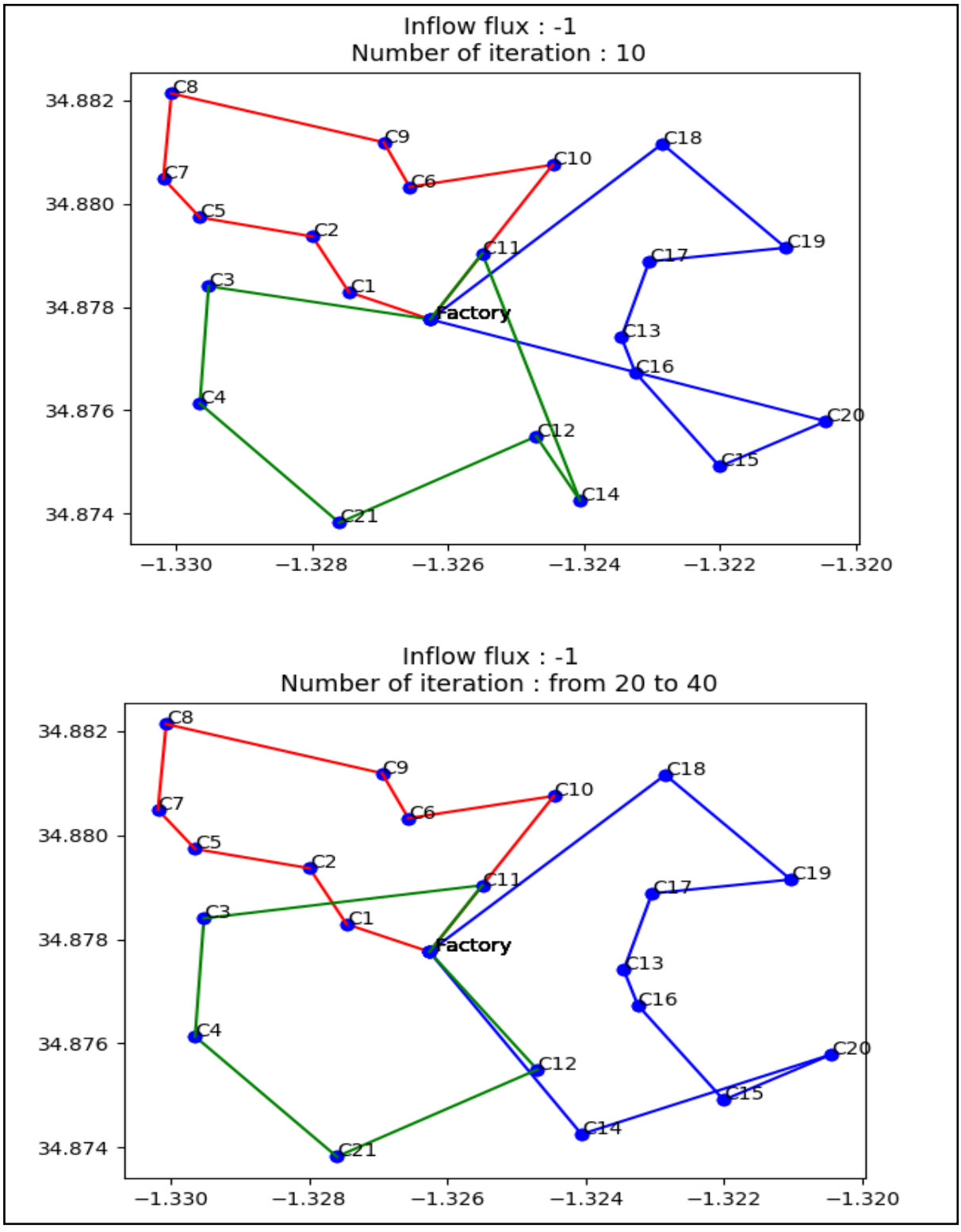
  Number of iterations: from **10** to **100**.

- **CVRP parameters:**

  Number of clients: 21.

  Number of vehicles: 3.

  Vehicle capacity:35.



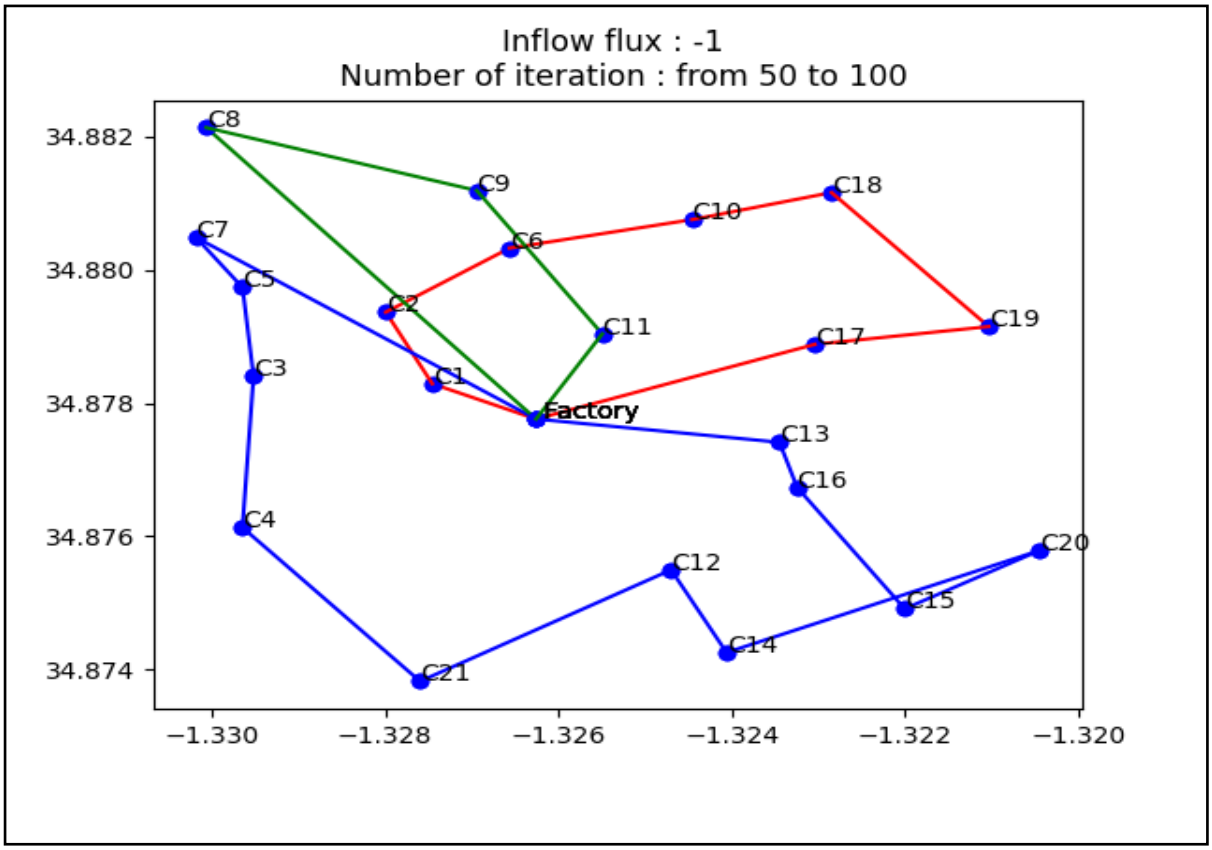


Figure III.7: The graphical window of execution.

Table III.7: Experimentation for measuring the iterations number influence.

Number of iteration	Result(meter)
10	8757.7
From 20 to 40	7738.8
From 50 to 100	7337.9

III.12 Sensibility analysis

In the previous experimentations, we focused only on the influence of the number of iteration on the results of TSP and CVRP. In order to see the influence of the both parameters (number of iteration and inflow flux I_0), we made experiments on the same **TSP** and **CVRP** problems with different parameters. The initial value of other parameters (D_{ij} ,Q_{ij},P,$\underline{G}$) have no influence on the final result. The results are shown in the **table III.8** and **table III.9**

Table III.8: 6TSP analysis results.

Inflow flux	Number of iteration	total distance(meter)	execution time(seconds)
-1	10	7602.2	7.834141254
	30	5891.8	21.583009
	50	5710	35.06829333
	80	5710	53.12596464
	100	5710	64.08805275
-2	10	5891.8	8.018057823
	30	5891.8	21.5914166
	50	6683.7	36.00674438
	80	6683.7	59.79356146
	100	6683.7	66.8718791
-3	10	5891.8	8.26726985
	30	5891.8	22.62200165
	50	5891.8	36.35655499
	80	8165.6	55.58615017
	100	8165.6	68.16773939
-4	10	5891.8	8.498732328
	30	5891.8	23.32553911
	50	7737.8	37.54572248
	80	8599.2	56.48398185
	100	8599.2	69.10215807
-5	10	5891.8	8.377806425
	30	5891.8	22.52003741
	50	7737.8	36.59331298
	80	7132.7	56.57092834
	100	7132.7	71.84345984
-6	10	5891.8	10.6803782
	30	5891.8	24.17401266
	50	7737.8	38.62305474
	80	7594.8	58.26187825
	100	7594.8	72.24321198
-7	10	5891.8	9.047389746
	30	5891.8	24.2439692
	50	5891.8	40.66278911
	80	6165.7	57.36443639
	100	6165.7	69.99160862

Table III.9: CVRP analysis results.

Inflow flux	Number of iteration	total distance(meter)	execution time(second)
-1	10	8757.7	7.524096727
	30	7738.8	19.60730815
	50	7337.9	31.95557213
	80	7337.9	47.82367849
	100	7337.9	57.74822474
-2	10	7738.8	7.185727358
	30	7738.8	19.76292944
	50	8497.4	31.95381832
	80	8497.4	52.97416353
	100	8497.4	63.25219202
-3	10	7738.8	7.982460976
	30	7738.8	22.14014792
	50	7738.8	32.64844489
	80	9959.9	50.27132368
	100	9959.9	60.32736874
-4	10	7738.8	7.479362726
	30	7738.8	20.45309591
	50	9182.3	33.02009559
	80	10043.7	49.59699249
	100	10043.7	60.64795446
-5	10	7738.8	7.315947771
	30	7738.8	20.1190877
	50	9182.3	32.62794662
	80	8288.2	49.36891079
	100	8288.2	59.32575011
-6	10	7738.8	7.36637044
	30	7738.8	19.94904661
	50	9182.3	32.26922774

	80	9023.8	53.24086332
	100	9023.8	61.58474374
-7	10	7738.8	8.041015148
	30	7738.8	20.79886794
	50	7738.8	33.17920828
	80	8012.7	50.1903913
	100	8012.7	60.81929922

From this analysis, we conclude that the best result is obtained when the inflow flux I_0 is set to -1, for the other values of I_0 the results did not effected too much.

III.13 Results discussion

Tables III.6 and **III.7** represent the results of the experimentation that was applied to **TSP** and **CVRP** instances, respectively. We conclude from these experiments that the optimal solution is proportional to the number of iterations.

This imitates the behavior of the true **slime mold**, when the **physarum** finds food sources, it forms multiple tubes that connect them, at the beginning, all the tubes have almost the same radius, but after a period, only the tubes that form the shortest that connect them grow up.

III.14 Comparison with other metaheuristics

Tables III.10, **III.11** show the best solutions of the **TSP** and **CVRP** obtained by **physarum solver** in addition with two other well-known metaheuristics results (simulated annealing and tabu search).

Table III.10: TSP results.

Metaheuristic	**parameters**	**Execution time(seconds)**	**The best result(meter)**	**Time complexity**
Simulated annealing	**-Number iteration :** 29950000 **-Maximum temperature** :100 **-Minimum temperature** :5	10	5878.5	N*log(N)

Tabu search	**-Number of iterations :**1000000	24	5839.1	N
Physarum solver	**-Inflow flux I_0:** -1 **-Number of iterations :**50 **-*G*** = 10^{-6} **-D_{ij} = 1** (∀i, j=1,2….22) **-Q_{ij}** = 0 (∀i, j=1,2….22) **-P** = 0 (∀i=1,2….22)	30	5710	N^3

Table III.11: CVRP results.

Metaheuristic	parameters	Execution time(seconds)	The best result	Time complexity
Simulated annealing	**-Number iteration :** 29950000 **-Maximum temperature** :100 **-Minimum temperature** :5	125	7415.1	N*log(N)
Tabu search	**-Number of iterations :**200000	19	7009.1	N
Physarum solver	**-Inflow flux I_0:** -1 **-Number of iterations :**50 **-*G*** = 10^{-6} **-D_{ij} = 1** (∀i, j=1,2….22) **-Q_{ij}** = 0 (∀i, j=1,2….22)	30	7337.9	N^3

	-P = 0 (∀i=1,2….22)			

Table III.10 represents the **TSP** results, it is clearly demonstrates the **physarum solver** results are better than tabu search and simulated annealing results, While **table III.11** represents the **CVRP** results, the **physarum solver** result is not the best but it better than simulated annealing results.

III.15 Conclusion

In this chapter, we demonstrated that the **physarum solver** algorithm which is inspired by the true slime mold is able to solve the traveling problem and the capacitated vehicle routing problem. We test it with different parameters, and in some cases, it gives results better than other well-known metaheuristics.

General Conclusion

The supply chain field consists of many problems related to the transportation and distribution of goods. The vehicle routing problem and the traveling salesman problem are the most studied during the last few years.

In this work, we focused on solving a particular type of the **VRP** problem, which is the capacitated vehicle routing problem, which consists of performing each customer to one route, which is performed by one limited capacity vehicle. There are many methods to solve such kinds of problems.in this work, we were interested in a new method that is inspired by an intelligent creature called **physarum polycephalum**.

Our goal was to study the capability of solving the **TSP** and **CVRP** by imitating the foraging behavior of the **physarum polycephalum**. There are many models that imitate one aspect or more of this behavior, among these models; we chose the flow conductivity model. The model was then applied to real datasets of **TSP** and **CVRP**. The experimentation demonstrated what we were looking up to, and even gave good results compared with other metaheuristics.

REFERENCES

- **[1]** Google OR-Tools. (2020, 26 Jun). : VRP Capacity Constraints. Retrieved from: https://developers.google.com/optimization/routing/cvrp?hl=en
- **[2]** Salma., B. & Zahia, T. (2016). Résolution de problème de tournée de véhicule à base d'une méta-heuristique. Department of mathematic and informatics, Mila University, Algeria.
- **[3]** Martin, D., Jan K.L., & Martin, W.P.S. (2011).A classification scheme for vehicle routing and scheduling problems. EuropeanJournal of Operational Research, vol. 46, no. 3, pages 322–332, 1990.
- **[4]** Montemanni, R., Gambardella, L.M., AE, Donati, A.V., & Rizzoli, A-E. (2005) a new algorithm for a dynamic vehicle routing problem based on Ant colony system. J Comb Optim 10:327–343.
- **[5]** Franklin, T.H., Beatrice, O.B. (2007). Dynamic vehicle routing using genetic algorithms. Department of Computer Science, Brock University, St. Catharines, ON, Canada, L2S 3A1.
- **[6]** E.D. Taillard. (1999).A heuristic column generation method for the-heterogeneous fleet VRP.RAIRO-Operations Research, 33, 1-14.
- **[7]** Ayache, A. (2019). Résolution du problème de tournées de véhicules avec fenêtre de temps. Departement of informatic, M'sila University, Algeria.
- **[8]** Hiba, B. (2018). « Problèmes de tournées de véhicules robustes multi-objectifs. (Doctoral thesis) University of Picardie Jules Verne.
- **[9]** Lilia, M., & Yamina, G. (2021). Le problème de la tournée de véhicule avec contrainte de capacité par l'algorithme génétique. (Master thesis). Mohamed El Bachir El Ibrahimi University Burj Bouarej Algeria.
- **[10]** Hettab, S., & Latrache, A. (2021). Une Approche pour la Résolution et l'Optimisation d'un Problème de Tournées de Véhicule (Capacitated Location Routing Problem (CLRP)) (master thesis). Abd Elhafid Boussouf university center, Mila.
- **[11]** Nesmachnow, S. (2014). An overview of metaheuristics: Accurate and efficient methods for Optimisation. International Journal of Metaheuristics, 3. https://doi.org/10.1504/IJMHEUR.2014.068914.
- **[12]** Liang, F. (2020, 21 April). Optimization Techniques — Simulated Annealing.

  Retrieved from: https://towardsdatascience.com/optimization-techniques-simulated-annealing-d6a4785a1de7
- **[13]** Henderson, D., Jacobson, S.H., & Johnson, A.W. (2006). The Theory and Practice of Simulated Annealing. https://doi.org/10.1007/0-306-48056-5_10.
- **[14]** Zäpfel, G., Braune, R., & Bögl, M. (2010). Metaheuristic Search Concepts. Berlin Heidelberg: Berlin Heidelberg. https://doi.org/10.1007/978-3-642-11343-7.
- **[15]** Gabarro, K. (2000, 1 Mar). Tabu Search Algorithm. Retrieved from: https://www.cs.upc.edu/~mallba/public/library/firstProposal-BA/node11.html
- **[16]** Peres, F., & Castelli, M. (2021). Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design, and Development. Applied Sciences, 11(14), https:// doi.org/10.3390/app11146449.

- **[17]** Anaci, M., Koylu, F., & Al-Sumaidaee, Z.(2021). Identification of Dynamic Models by Using Metaheuristic Algorithms. ADI Journal on Recent Innovation (AJRI), 3(2685-9106), 36-58. https://doi.org/10.34306/ajri.
- **[18]** Talbi, AL.G. (2009). Metaheuristics From Design To Implementation. Hoboken, New Jersey: John Wiley & Sons. https://doi.org/10.1002/9780470496916.
- **[19]** Gandomi, A.H., Yang, X., Talatahari, S., & Alavi, A.H. (2013). Metaheuristic Applications in Structures and Infrastructures. London: Elsevier Inc. https://www.readallbooks.org/book/metaheuristic-applications-in-structures-and-infrastructures.
- **[20]** Yahui Sun. (2019). Physarum-inspired Network Optimization: A Review. arXiv:1712.02910v2 [cs.ET].
- **[21]** Andrew, A. (2009). If BZ medium did, spanning trees these would be the same trees as Physarum built. University of the West of England, Bristol BS16 1QY, United Kingdom.
- **[22]** Anthony, B., & Seàn, M.G. (2020). Slime mould foraging: an inspiration for algorithmic design. Smurfit School of Business, University College Dublin.
- **[23]** Cai, G., Chao, Y., Zili, Z., Yong, H., Sankaran, M., & Yong, D. (2014). An amoeboid algorithm for solving linear transportation problem.
- **[24]** Hilal A. (2019). Dynamic Physarum Solver: a bio-inspired shortest path method of dynamically changing graphs. Turkish Journal of Electrical Engineering & Computer Sciences.
- **[25]** Torsten, S. (2013). Physarum Learner: A Novel Structure Learning Algorithm for Bayesian Networks inspired by Physarum Polycephalum. Dissertation for obtaining a doctorate of natural sciences (Dr RER NAT.) Faculty of Biology and Preclinical Medicine. The University of Regensburg.
- **[26]** Verner, P., Robert, J. M., Eero, S., Ken, H., Tahitoa, A., & Matti, L. (2020). Slime Mold Inspired Distribution Network Initial Solution.
- **[27]** ck12.(2017). Characteristics and Classification of Fungus-like Protists – Advanced Retrieved from: https://www.ck12.org/book/ck-12-biology-advanced-concepts/section/12.16/
- **[28]** bionity site: slime mold. Retrieved from: https://www.bionity.com/en/encyclopedia/Slime_mold.html
- **[29]** New world encyclopedia: slime mold. Retrieved from: https://www.newworldencyclopedia.org/entry/slime_mold#:~:text=Plasmodial%20slime%20molds%2C%20characteristic%20of,encounter%20their%20favorite%20food%2C%20bacteria
- **[30]** Retrieved from: https://commons.wikimedia.org/wiki/File:Slime_Mold_-Olympic_National_Park_North_Fork_Sol_Duc.jpg.
- **[31]** Quanta magazine: Slime Molds Remember — but Do They Learn? https://www.quantamagazine.org/slime-molds-remember-but-do-they-learn-20180709/

- **[32]** shkrobius. (2005). WordPress: Wonders. IV. Physarum polycephalum (The origin of human intelligence?). Retrieved from: https://shkrobius.wordpress.com/2005/-12/21/wonders-iv-physarum-polycephalum-the-origin-of-human-intelligence/
- **[33]** Sandes, L.(2010, 22 Jan). Slime Mold Grows Network Just Like Tokyo Rail System. Retrieved from: https://www.wired.com/2010/01/slime-mold-grows-network-just-like-tokyo-rail-system/
- **[34]** Petrov, P.(2021, 19 Aug). Meet Professor Physarum polycephalum Retrieved from: https://medium.com/@petarpetrov004/meet-professor-slime-c635227d6ba1.
- **[35]** Versluis, D.M. (2018). Solving the vehicle routing problem with a multi-agent Physarum model (Master Thesis).Utrecht University.
- **[36]** Sun, Y. (2017). Physarum-inspired Network Optimization: A Review. Retrived from: https://www.researchgate.net/publication/321719515_Physaruminspired_Netw-ork_Optimization_A_Review.
- **[37]** Koprinkov, R., Nguyen, T., Alter, D.M., Rehm, R., & van Buuren, T.(2020). Analysing and Modelling the Oscillatory Pattern of Physarum Polycephalum with a Constraint Radius.
- **[38]** Gao, C., Liu, C., Schenz, D., Li, X., Zhang, Z., Jusup, M., Wang, Z., Beekman, M., & Nakagaki, T. (2018). Does being multi-headed make you better at solving problems? A survey of Physarum-based models and computations. Physics of Life Reviews,29,1-26. https://doi.org/10.1016/j.plrev.2018.05.002.
- **[39]** Gao, C. Yue., Z., Wei, D., & Zhang, X. (2020). An AcceleratedPhysarumSolver for Network Optimization.IEEE Transactions on Cybernetics.50 (2):2168-2267. https://doi.org/10.1109/TCYB.2018.2872808.
- **[40]** Dave, K. (2013). A Python Book: Beginning Python, Advanced Python, and Python Exercises.
- **[41]** Retrieved from: https://www.python.org/about.
- **[42]** Luxen, D. (2022, 18 Jan). Open Source Routing Machine. retrieved from: https://wiki.openstreetmap.org/wiki/Open_Source_Routing_Machine.
- **[43]** Gao, C. (2019, 30 Nov).PhysarumOptimization.retrived from: https://github.com/caigaoub/PhysarumOptimization.
- **[44]** Graphhoper. (2004).graphhoper. Retrieved from: https://github.com/graphhopper-/graphhopper.
- **[45]** Meijiao, L., Yanhui, L., Qi, H., Ang, L., Mingchao, Z., Nan, Q., & Liheng, C. (2020). A Two-Way Parallel Slime Mold Algorithm by Flow and Distance for the Travelling Salesman Problem.