الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليـم العالـي والبحـث العلمـي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Nº Ref :.....

University Centre Abd-Elhafid Boussouf - Mila

Institute of Mathematics and Computer science **Department Computer science**

A Dissertation Submitted in Partial Fulfillment for the Requirement of the Master Degree in computer science

Specialty: Information and Communication Sciences and Technologies (STIC)

Algorithms for the Two-Stage Capacitated Facility Location Problem

Presented by:
Abboud Tariq

Board of Examiners:

Supervisor :	Guemri Oualid	Grade MCB
Chairman :	Lalouci Ali	Grade MAA
Examiner :	Guettiche Mourad	Grade MCB

Academic year : 2022/2023

Acknowledgements

First of all, I endlessly thank Allah for giving me the strength and patience to fulfill this work.

I would like to express my gratitude and deepest appreciation to my teacher and supervisor Dr. GUEMRI OUALID for his serious guidance, constant support, and continuing encouragement all throughout the period of writing my dissertation. I thank him for his precious advice and valuable annotations.

Special thanks go to the board of examiners: Mr. LALOUCI ALI and Dr. GUETTICHE MOURAD for accepting to critically read and sincerely evaluate our work.

Finally, I extend my unlimited gratitude and enormous thanks to my wonderful, beloved family who encouraged and supported me to complete this work.

Table of contents

List of Figures

List of Algorithms

List of Tables

Abbreviations

UFLP: Uncapacitated Facility Location Problem.

CFLP: Capacitated Facility Location Problem.

MUFLP: Multi-product Uncapacitated Facility Location Problem.

TSCFLP: Two-Stage Capacitated Facility Location Problem.

B&B: Branch and Bound.

GRASP: Greedy Randomized Adaptive Search Procedures.

GA: Genetic Algorithm.

LS: Local Search.

SA: Simulated Annealing.

TS: Tabu Search.

VNS: Variable Neighborhood Search.

PSO: Particle swarm optimization.

ACO: Ant Colony Optimization.

CS: Clustering Search.

ALNS: Adaptive Large Neighborhood Search.

LB: Local Branching.

Abstract

This thesis presents a Simulated Annealing based algorithm to solve Two stage capacitated facility location problem. In this problem, a single type of product must be transported from factories to customers, passing through intermediate warehouses. Initially, our algorithm was designed to ensures the selection of the most appropriate facilities (factories\ warehouses). Then, we shifted our focus towards enhancing client allocations to the best-suited facilities (customer→warehouse\warehouse→factory). Experiments show that our algorithm obtains promising results comparing to the literature.

Key words: Location problems, Combinatorial optimization, Meta-heuristics, Simulated annealing, Two stage capacitated facility location problem.

ملخص

خلال هذه الأطروحة اقترحنا خوارزمية تستند على تقنية محاكاة التلدين لحل مشكلة تحديد المواقع ذات السعة المحدودة والموزعة على طبقتين. في هذه المشكلة، يجب نقل نوع واحد من المنتجات من المصانع إلى العملاء، مروراً بالمستودعات الوسيطة. في البداية، تم تصميم خوارزميتنا لضمان اختيار المنشآت المثلى (المصانع/المستودعات). بعد ذلك، حولنا تركيزنا نحو تعزيز توزيعات العملاء على المنشآت الأنسب (العميل ← المستودع/المستودع ← المصنع). تظهر التجارب أن خوارزميتنا تحصلت على نتائج واعدة مقارنة بمختلف الخوارزميات المقترحة سالفا.

الكلمات المفتاحية: مشاكل تحديد المواقع، التحسين التوافقي، الطرق التقريبية، محاكاة التلدين، مشكلة تحديد المواقع ذات السعة المحدودة والموزعة على طبقتين.

Introduction

The efficient and strategic placement of facilities plays a crucial role in the success of various industries and organizations. The Two-Stage Capacitated Facility Location Problem is a well-known optimization challenge and it is an extend to the traditional Facility Location Problem through incorporating capacity constraints into the decision-making process which allows to a more realistic representation of facility operations.

In the first stage of TSCFLP, products produced by capacitated factories are transferred to capacitated warehouses and in the second stage, the products are delivered to customers. The problem to be addressed includes finding an optimal location for facilities to meet the customers in order to minimize both the fixed opening cost of the factories and warehouses and the transportation costs associated with both stages.

The objective of this thesis is to deal with TSCFLP where a single type of product must be transported from two type of facilities to customers with the aim of proposing an efficient algorithm to solve heuristically this problem. To the best of our knowledge and for the first time in the literature, we propose to solve this problem using a simulated annealing algorithm.

The thesis is organized into four chapters as follows:

In first chapter we present location problems and its application. Whereas in the second chapter we will talk about some of the combinatorial optimization methods and algorithms widely used to solve optimization problems. Concerning the third chapter, we will highlight our proposed algorithm to solve TSCFLP. Then, in the fourth chapter we will present the results obtained by our algorithm where testing it on benchmark instances from the literature. The obtained results are compared to most of the literature. Finally, we finished with a conclusion.

Chapter I : Location problems: definitions and applications

I.1 Introduction

Location problems are well-known optimization problems in the literature of the operations research. There are several variants of location problems that have been extensively studied, starting from the capacitated and uncapacitated facility location problems to the most constrained location problems.

In this chapter we will present a literature review on the location problems which are related to our problem studied in this thesis. [1] [2] [3]

I.2 Location problems

In general, in the location problems the goal is to select a sub-set of facilities or locations to be installed from a set of candidates and to allocate the other not selected ones to the selected locations. In this section, we present the location problems related to our work. The following variables are used in the definitions of the location problems:

I : number of factories .

J : number of warehouses .

K : number of customers .

c_{kj} : cost of shipping one unit from facility j to customer k .

z_{kj} : equals 1 if demand customer k is assigned to facility j.

f_j : opening cost of facility .

y_j : 1 if node j is chosen as a facility (opened) .

s_j : the maximum capacity of facility j .

d_k : demand of customer k .

I.2. 1 Uncapacitated, single-stage location problem

This problem is also known in the literature with the name of Uncapacitated facility location problem (UFLP). The UFLP is a basic location problem where the goal is to select a sub-set of facilities to be installed from a set of candidates. Since we don't have any capacity constraint, each not selected facility will be allocated to the near selected one. The following mathematical model presents the UFLP [1]:

$$v(\text{ UFLP }) = \min \quad \sum_{k\in K}\sum_{j\in J} c_{kj}z_{kj} + \sum_{j\in J} f_j y_j, \tag{1a}$$

$$\text{s.t.} \quad \sum_{j\in J} z_{kj} = 1 \ \forall k \in K, \tag{1b}$$

$$z_{kj} - y_j \leqslant 0 \ \forall k \in K,\ j \in J, \tag{1c}$$

$$0 \leqslant z_{kj} \leqslant 1,\ 0 \leqslant y_j \leqslant 1 \ \forall k \in K, j \in J, \tag{1d}$$

$$y_j \in \mathbb{B} \ \forall j \in J, \tag{1e}$$

In this model, (1a) presents the objective function where we sum the setup cost and the allocation cost. The constraints (1b) ensure that each not selected facility is allocated to only one selected facility. The constraints (1c) ensures that each not selected facility is allocated to a selected facility. The constraints (1d and 1e) present the decision variables.

I.2. 2 Capacitated, single-stage problem

This problem is also known as capacitated facility location problem (CFLP). In CFLP, we have a set of facilities candidates and a set of customers. Each customer has a demand and each facility has a setup cost and a capacity.

The goal is to choose a sub-set of facilities from a set of candidates and to allocate each customer to a selected facility where the sum of the demands of the customers allocated to a selected facility must be less than or equal to the capacity of the facility. Here we present the mathematical model which describes the CFLP [1] :

$$
\begin{aligned}
v(\text{ CFLP }) = min \quad & \sum_{k\in K}\sum_{j\in J} c_{kj}z_{kj} + \sum_{j\in J} f_j y_j, \\
\text{s.t.} \quad & \sum_{j\in J} z_{kj} = 1 \ \forall k \in K, \\
& \sum_{k\in K} d_k z_{kj} - s_j y_j \leqslant 0 \ \forall j \in J, \\
& z_{kj} - y_j \leqslant 0 \ \forall k \in K, \ \forall j \in J, \\
& \sum_{j\in J} s_j y_j \geqslant d(K), \\
& \sum_{j\in J_q} z_{kj} \leqslant 1 \ \forall k \in K, \ \forall q \in Q, \\
& 0 \leqslant z_{kj} \leqslant 1, \ 0 \leqslant y_j \leqslant 1 \ \forall k \in K, \ \forall j \in J, \\
& y_j \in \{0,1\} \ \forall j \in J,
\end{aligned}
$$

I.2. 3 Multi-product location problems

In the literature when we talk about a location problem, we consider that the customers demand on only one product (In the above presented location problems we consider that). However, we can have some situations where the customers can demand more than one product and here, we talk about a class of location problems called multi-product location problems. So, each single-product location problem can be transformed to multi-product location problem and consequently we can have a multi-product UFLP, a multi-product CFLP, a multi-product multi-stage location problem, etc... The mathematical model which presents the multi-product MUFLP can be founded in [1] .

I.2. 4 Multi-stage location problems

The Multi-Stage Location problems is a class of location problems describe the situations where we have facilities on several hierarchically related levels. These cases can be found in distribution/collect systems of companies. In this class of problems, in general the goal is to choose a sub-set of facilities to be installed at each stage and then to allocate the selected facilities one stage to the selected facilities of the next stage in order to minimize the total cost including the setup costs and the allocation cost. Please note that in this class of problems, in the first stage customers are allocated to the first stage of facilities and the facilities of the last (the higher) stage are not allocated. In the following image we present an example of one problem of this class:

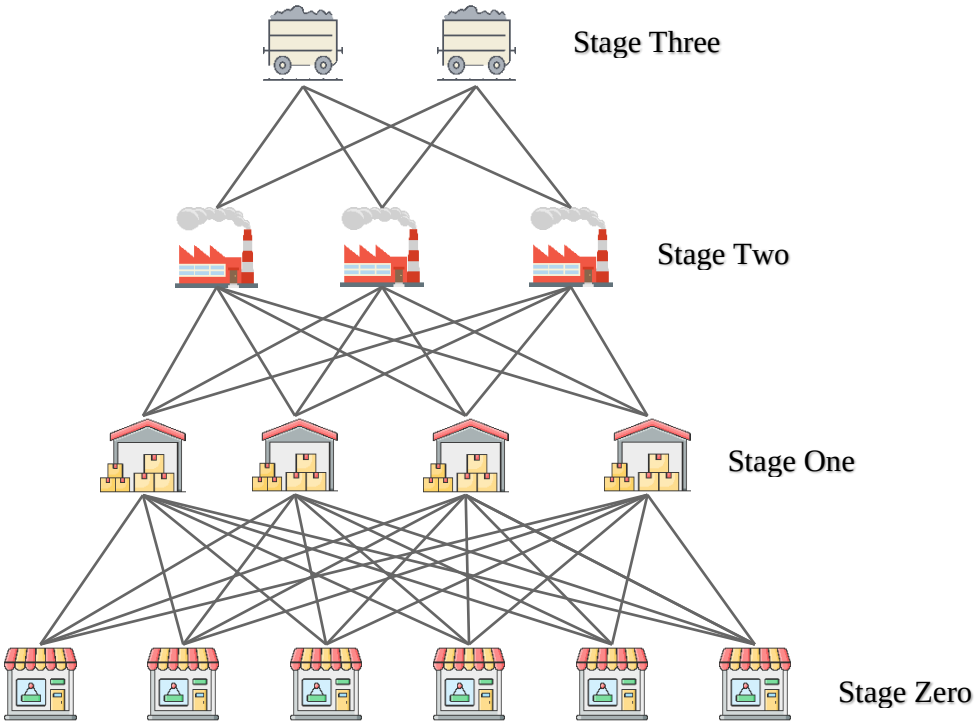


Figure 1: *Muti-Stage Facility Location Problem*

I.2. 5 Dynamic location problems

In the industry, installation of facilities is considered as strategic decisions which token on a long-term basis. On the other hand, the basis used when taking these decisions can be changed over time such as: demand of customers etc... The dynamic location problems are a class of location problems which can deal with these cases (cases of “change-over-time") where the goal is to find high-quality solutions while considering the change-over-time. It is worth mentioning that the dynamic location problems are harder and more complex than the static location problems. In the literature, several dynamic location problems have been studied such as: [4] [5] [6].

I.2. 6 Probabilistic location problems

Probabilistic location problems are considered as a class of location problems which deal with the situations where some variables/data of the problem are subject to uncertainty. In general, the data or variables which are subject to uncertainty are modeled as a random variable. The probabilistic location problems are harder and more complex than the deterministic location problems. In the literature, we can find several studies which dealt with probabilistic location problems, such as: [7] [8].

I.2. 7 Hub location problems

Hub location problems is a class of location problems where the goal is to install a set of facilities called hubs to meet the transportation demands of the customers. In a hub location problem, each customer demand is formulated as a transportation demand from an origin to a destination and the quantity demanded by the customer is transported via the selected or the installed hubs. As examples of the problems of this class we have: the uncapacitated hub location problem, the capacitated hub location problem, etc. [9] [10] [11] [12]

I.2. 8 Routing location problems

In all classes of the location problems, we presented above, we considered the direct link (route, arc, etc...) between a terminal or a client to a selected facility or depot in the allocation part. For example, if we have 3 clients allocated to a facility, then we consider that the client 1 is directly linked with facility, the second client is also directly linked to facility and the same case for the third client. However, in the location routing problems, all clients allocated to a facility are not directly linked to the facility, but they are linked with a route that starts from the facility and ends at this facility. So, in a location routing problem we have two sub-problems: (1) The location of the facilities and the allocation of clients to these facilities and (2) create a set of routes to visit the clients allocated to each facility. More details on location-routing problems and their applications are found in [13] [14] [15] .

I.2. 9 Multi-objective location problems

The multi-objective location problems are a class of location problems where the objective is to optimize more than one criterion. In the formulation of these problems, we find that the objective function contains more than one criterion such as: the construction cost (including installation and affectation cost), the profit (to be maximized), the waiting time (as a service quality), etc... In the literature, we can find several multi-objective location problems which have been studied [16] [17] [18].

I.3 Two-Stage Capacitated Facility Location problems

The Two-Stage Capacitated Facility Location Problems (TSCFLPs) are considered as multi-level location problems where a capacity constraint is imposed. In TSCFLPs, we have a set of facilities candidates in level 1 (in general we call them warehouses) and another set of facilities candidates in level 2 (in general we call them factories) [19] .The goal is to select a sub-set of factories to be installed from the set of candidates, and another sub-set of warehouses to be installed from the set of candidates to meet the demands of the clients while minimizing the total cost, including the installation cost and the allocation cost. The allocation is made as: the clients are allocated to the selected warehouses and the selected warehouses are allocated to the selected factories. In addition, the sum of the demands of clients treated by a selected warehouse must be inferior or equals to its capacity, and the same case for each selected factory, the sum of the demands of the warehouses must be inferior or equals to its capacity. In the literature, there are many TSCFLPs that have been solved such: single-source TSCFLP [20], multiple-source TSCFLP [21], multi-product TSCFLP [22]. etc... **It is worth mentioning that in our thesis we deal with the multiple-source TSCFLP.** In the following image we present as example of the TSCFLP:

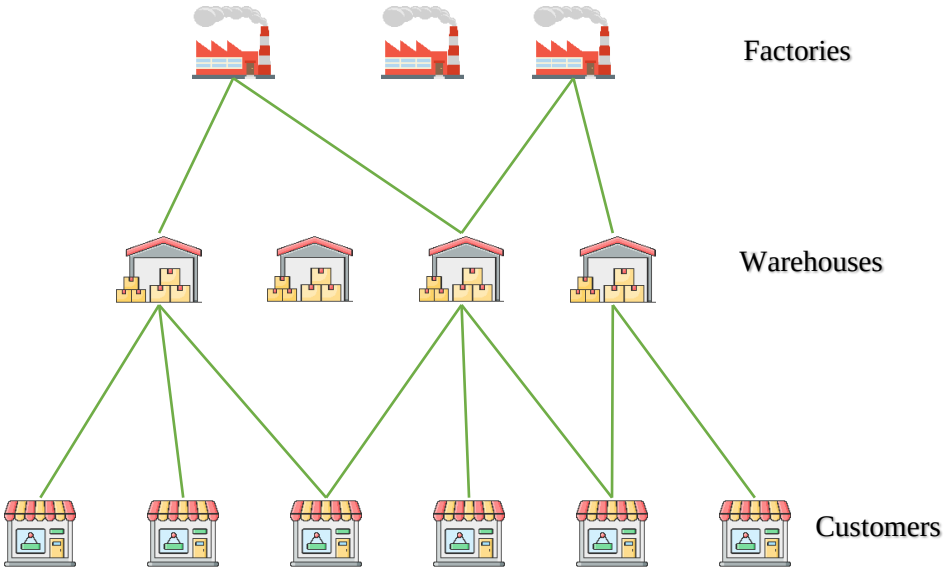


Figure 2 : *Two Stage Capacitated Facility Location Problem*

I.4 Applications

The location problems have a plenty of applications, notably in modelling industrial and real-life problems. Here we present examples of these applications [23] [24]

Cluster analysis: In general, in the cluster analysis, the goal is to group a set of items (or any other entity) into clusters (or groups) where the items belong to one cluster should be homogeneous. In fact, we can solve cluster analysis problems as location problems where the goal is to find the best items that will be the kernel of each group and by the allocation of the rest of the items to these best items, we get a set of groups or clusters. As an example, from the literature, in [25] the authors modelled a clustering task as a p-median problem.

Location of bank accounts: Another important application of the location problems can be found in [26]. In this study, the authors assumed that: when a company pays its suppliers, we can optimize float when choosing the location of the bank accounts used to pay them. This problem has been modelled as an UFLP with some additional constraint(s). Another application of the location problems in the financial sector can be found in [27].

Vendor selection: Selecting the most appropriate vendors is an important task for any company. In fact, the selection process takes in consideration several criteria such as: price, quality, know-how, product-to-buy etc... In [28], the authors discussed that the vendor selection problem can be modeled and solved using location problems such as UFLP and CFLP.

Location and sizing of offshore platforms for oil exploration: In [29] [30] the authors modeled and solved a problem in oil exploitation as a location problem.

Database location in computer networks: In [31]the authors modeled the problem of the installation and the maintenance of databases in a computer network an extended variant of UFLP.

Computer networks and concentrator location: In the literature, location problems are used to solve several complex problems in the design of the telecommunication and computer networks [32] and [33] In addition, many of these complex problems are related to the location of the concentrators [34] and [35]

Index selection for database design: In [36] the authors dealt with an important problem in the physical database design which is the index selection problem. This problem has been modeled and solved as an UFLP.

I.5 Conclusion

In this chapter, at first, we have presented several location problems related to the Two-Stage Capacitated Facility Location Problem. Secondly, we have provided a description of TSCFLPs. Finally, we have presented the real-life applications of TSCFLP.

Chapter II : Optimization methods and algorithms

II.1 Introduction

In this chapter we will present some methods and algorithms wish are used in combinatorial optimization problems solver. The combinatorial optimization methods can be divided in two main sub class: exact methods and Approximation methods.

In general, Exact methods aim to find the globally optimal solution for combinatorial optimization problems. On the other hand, the Approximation methods and algorithms aim to find near-optimal or optimal solutions within a reasonable amount of time, as finding optimal solution for large-scale problems is often computationally infeasible.

II.2 Combinatorial optimization problem

Combinatorial optimization covers all methods that allow determine the optimum of a function with or without constraints. In theory, a combinatorial optimization problem is defined by a set of instances. Each instance of the problem is associated with a discrete set of solutions S, a sub-set $X\ of\ S$ representing the feasible solutions and a cost function f which assigns to each solution $s \in X$ a cost $f(s)$. Solving such a problem consists of finding a solution $s_{best} \in X$ optimizing the value of the cost function f. s_{best} is called an optimal solution or global optimum [37] .

II.3 Exact Methods

Exact methods are methods that search for the optimal solution of a problem by exhaustively examine all possible solutions in the search space. However, the major drawback of these methods is the execution time, because all possible solutions will be examined one by one and the execution time increases exponentially with the size of the problem solved. Therefore, these techniques remain inappropriate for large sizes instances [37]. As an example of these methods, we can cite: the branch and bound.

II.3. 1 Branch and bound

The Branch and bound algorithm (B&B) [38] is appeared for the first time in the 60s and used to solve linear economic programming problems. Later, B&B becomes the most widely used exact method for solving NP-hard optimization problems [39]. Formally, B&B is a

tree-structure based algorithm where its main goal is to examine all possible solutions while eliminating unnecessary or not-beneficial branches. Unnecessary or not-beneficial branches are branches that contain infeasible solutions or bad quality solutions. The branch and bound algorithm consider x_b as an optimal solution if and only if the value of the objective function x_b is less than or equal to the upper bound v_s and is greater than or equal to lower bound v_i , mathematically: $v_i \leq f(x_b) \leq v_s$.

As we highlighted above, the B&B explores all possible solutions while eliminating not-beneficial branches. Therefore, in order to develop a high-quality B&B method you have to focus on the following techniques used in B&B:

- The separation technique: how to divide the search space into subsets of solutions awhile ensuring that the union of the created subsets covers all possible solutions of the problem.
- The evaluation technique: used to determine whether there are possible solutions of good quality in the tree-branch or not by calculating the lower and upper bounds associated to the current branch.
- The exploration technique: which consists of fixing the strategy of exploration of the tree by giving the order of visit to its branches. There are several exploration strategies such as: better first, depth first etc.

Algorithm 1 : *Branch and Bound for minimization*

```
1:  T_root ← Create the root of the search tree according to the separation technique ;
2:  U_bound ← +∞; L ← T_root ;
4:  while (L≠∅)
5:      S_C ← Explorer(L);
6:      If (Evaluation (S_C) ≤ U_bound)
7:          L′ ← All partial solutions S′ that can be obtained from Sc;
8:          For (each S' in L' do)
9:              If (S' is a complete solution)
10:                 update U_bound; update S_best;
12:             Else
13:                 add S′ to L ;
14:             End
15:         End
16:     Else
17:         delate S_C from L ;
18:     End
19: End
20: Return S_best;
```

II.4 Approximation Methods

Optimization problems in the industrial world have usually large size and many constraints, and therefore, exact methods cannot be applied for most of these cases. So, we have to look for a good solution in a reasonable time instead of waiting for an optimal solution after years of computation [37]. In contrast to the exact algorithms, Approximation methods do not guarantee the optimality of the solution, but they allow to find good quality solutions in a reduced execution time, it means, they seek a good compromise between the quality of the solution and the calculation time. In the literature, many Approximation methods have been proposed. In the following we present 3 categories of the approximation methods: heuristics, meta-heuristics and hybrid methods.

II.4. 1 Heuristic

In the literature, there are several definitions of a heuristic. here we present that of [40]: *"A heuristic (heuristic rule, heuristic method) is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large problem spaces. Heuristics do not guarantee optimal solutions; in fact, they do not guarantee any solution at all; all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time. "*

Moreover, in the field of combinatorial optimization, we can say that a heuristic is an Approximation method developed to solve a particular problem and it requires a deep knowledge about the problem being addressed. The goal of a heuristic is to find solutions not necessarily optimal for a given problem in a very short execution time [37].

II.4.1.1 Greedy constructive algorithm

A Greedy constructive algorithm [41] is an algorithm that progressively build a solution from scratch. At each step the locally optimal element according to the evaluation function is selected and added to the solution under construction until obtaining a complete feasible solution. The evaluation function also known as the greedy criterion or greedy choice rule, typically it measures the incremental increase or decrease in the objective function or cost function when incorporating a specific element into the partial solution.

Algorithm 2 : Greedy algorithm for minimization

```
1:  S ← ∅;
2:  C ← {e_1, e_2, ....., e_n };
3:  Evaluate the incremental cost c(e) for all e ∈ C;
4:  While (C ≠ ∅)
5:      e_b ← select e ∈ C with the smallest incremental cost c(e) ;
6:      S ← S ∪ { e_b};
7:      C ← C − { e_b};
8:      Reevaluate the incremental cost c(e) for all e ∈ C ;
9:  End
10: Return S;
```

II.4.1.2 Randomization and Greedy Randomized algorithm

Randomization plays a very important role in algorithm design [41]. It is used to introduce randomness and diversity into the search process, allowing the algorithm to explore different regions of the solution space and avoid being trapped in local optimal. One particularly important use of randomization appears in the context of greedy algorithms.

A Greedy randomized constructive algorithm [41] uses the same principle of a greedy algorithm that we mentioned before but it builds a restricted list of locally optimal element and randomly select an element from the predefined list unsated of selecting the locally optimal element. In general, the greedy randomized algorithms are used in the construction phase of GRASP or to create initial solutions for GA.

Algorithm 3: Greedy randomized algorithm for minimization

```
1:  S ← ∅
2:  C ← {e_1, e_2, ....., e_n };
3:  Evaluate the incremental cost c(e) for all e ∈ C;
4:  While (C ≠ ∅)
5:      Build a list with the candidate elements having the smallest incremental costs;
6:      e_b ← Select random e ∈ the restricted candidate list ;
7:      S ← S ∪ { e_b};
8:      C ← C − { e_b};
9:      Reevaluate the incremental cost c(e) for all e ∈ C ;
10: End
11: Return S;
```

II.4.1.3 Local Search algorithm

A Local search algorithm (LS) is iteratively improving a solution, it replacing the current solution with a better solution in the neighborhood until there is no better solution is founded. The neighborhood of a solution consists of the solutions that can be obtained by making small modifications or changes to the current solution. The efficiency of LS depends on several aspects, such as the initial solution & the neighborhood structure.

Algorithm 4: Local Search

```
1: S ← start solution;
2: While (S is not a local optimal)
3:     S_n ← select S ∈ N(S);
4:     If (f(S_n) is better than f(S))
5:         S ← S_n;
6:     End
7: End
8: Return S;
```

II.4. 2 Meta-Heuristic

In the literature and according to [42] "*A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions."*. Another definition can be found in [43]*:*

"A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a constructive method." [43]

In general, a heuristic is an algorithm developed to solve a specific problem. However, a metaheuristic is a general strategy that can be applied to solve a large number of optimization problems. In the literature, several metaheuristics have been developed which can be subdivided into two main families: single-solution based metaheuristics (based on a single

solution) and population-based metaheuristics (based on a population of solutions). These two families are described below.

II.4.2.1 Single-solution based metaheuristics

Single-solution based metaheuristics are single solution algorithms that are generally based on the exploration of the neighborhood of the current solution. They start from an initial solution, then at each iteration then they try to improve the current solution by exploiting its neighborhood. Many single-solution based metaheuristics methods have been proposed in the literature, here we cite the most known: simulated annealing, taboo search, variable neighborhood search, GRASP, etc.

II.4.2.1.1 Simulated Annealing

Simulated Annealing (SA) was first proposed by Kirkpatrick [44] , inspired from the physical process of annealing in metallurgy. The annealing process is to modify the properties of metal by heat it to a specific temperature and then slowly cool it in a regular way to ensure that the atoms reorganize themselves in a regular way. this process helps to reduce metal defects when it is transformed from a liquid to a solid state.

SA algorithm attempts to simulate the annealing process described above to find a good quality solution for a given optimization problem. Starting with very high temperature and an initial solution. During the annealing process and iteratively, the temperature decreased and a close neighbor of the current solution is randomly selected and accepted if it is better than the current solution, otherwise It will be accepted with a probability proportional to the temperature: the lower the temperature, the lower the probability of the solution being accepted. Over time, the algorithm accepts much better solutions and converges to good quality solutions.

Algorithm 5: Simulated annealing

```
1:  S ← initial solution;
2:  T ← initial temperature;
3:  while (the stop criterion is not met) do
4:      randomly choose S_n ∈ N(S)
5:      r ← a random number between 0 and 1.
6:      calculate Δ;
7:      If (S_n is better than S Or r < e^(−(Δ/T)))
8:          S ← S_n ;
9:          if (S is better than S_b)
10:             S_best ← S;
11:         End
12:     End
13:     update T;
14: End
15: return S_best;
```

II.4.2.1.2 Tabu Search

Tabu search (TS) introduced by Glover in the 1986 [45]. Technically, TS is a form of local search with additional rules and a tabu list to keep track of previously visited solutions and prevent the algorithm from revisiting them in the near future. The tabu list acts as a memory mechanism that helps the algorithm to escape the local optima's and to explore different regions of the search space. There are various types of memory mechanisms employed in TS such as: short, medium and long memories.

Formally, TS algorithm starts with an initial solution S. For each iteration, the neighborhood $\mathrm{N}(\mathrm{s})$ of the current solution is generated and the best solution S_n in it which does not appear in the tabu list L is selected. Afterwards, the tabu list is updated by adding the selected solution to it and remove the oldest solution in it (FIFO method). After that, the selected solution becomes the current solution. The best overall solution S_b is kept as the result and the algorithm ends when the stop criterion is satisfied.

Algorithm 6 : Tabu Search

```
S ← initial solution;
while the stop criterion is not met do
    Generate N(s);
    find the best solution S_n ,{ S_n ∈ N(s) and S_n ∉ L} ;
    Update L;
    S ← S_n ;
    If (S is better than S_b) Then
        S_best ← S;
    End
End
Return S_best;
```

II.4.2.1.3 Greedy Randomized Adaptive Search Procedures

Greedy randomized adaptive search procedure (GRASP) was first introduced in 1989 [46] as multi-start metaheuristic approach that combines both of greedy randomized algorithm and local search algorithm. At each iteration, the greedy randomized algorithm is used to construct a solution. Once the solution is obtained, a repair procedure might be called to fix the solution if it is not feasible or create a new solution that reach feasibility. After that, the local search algorithm is applied on the created feasible solution. The best overall solution is kept as the result of the algorithm.

Algorithm 7 :Greedy Randomized Adaptive Search Procedures for Minimization

```
f_i ← ∞;
While (the stop criterion is not met)
    S ← Greedy Randomized Algorithm ( );
    if (S is not feasible)
        S ← RepairSolution(S) ;
    End
    S ← LocalSearch(S) ;
    if ( f(S) < f_i)
        S_best ← S ;
        f_i ← f(S) ;
    End
End
return S_best;
```

II.4.2.1.4 Variable Neighborhood Search

Variable Neighborhood Search (VNS) is a metaheuristic introduced by Mladenović & Hansen in1997 [47] .The basic idea of VNS is the systematic change of a neighborhood combined with solution perturbation and local search procedures. During algorithm running, the neighborhood of a solution is explored using a set of predefined neighborhood structures. VNS has undergone various modifications and enhancements. A discussion of the basic concepts and successful applications of VNS can be found in survey papers [48].

Algorithm 8: Variable Neighborhood Search

```
S ← initial solution;
N(L), L = 1.2..... L_max ;
While the stop criterion is not met do
    L ← 1;
    While (L < L_max)
        S_x ← Shaking(S, N) ;
        S_y ← LocalSearch(S_x);
        if ( f(S_y) < f(S_b))
            S ← S_y ;
            L ← 1;
            S_best ← S ;
        End
            L ← L + 1;
        End
    End
End
Return S_best ;
```

II.4.2.2 Population-based metaheuristics

Population-based metaheuristics are methods based on a population of solutions and which are in general inspired by nature [49]. These methods use a set of solutions called population. They start with an initial population and, at each iteration, they try to build a new and better population based on the previous one in order to converge to good solution(s). As examples of these methods, we can cite: genetic algorithms, particle swarm optimization, ant colony algorithm, etc.

II.4.2.2.1 Genetic algorithm

Genetic algorithm (GA) was introduced in the 1975's by Holland [50]; it is inspired from the biological evolution of living beings based on the principles of natural selection and genetics. The basic genetic algorithm generally consists of two processes, the first is the selection of the individual to produce the next generation, and the second is the manipulation of the selected individual to produce the next generation through the crossing and the mutation techniques [51].

GA starts with the creation of an initial population of solutions. Then, at each iteration, the algorithm creates a set of solutions called parents by making a copy of the selected solutions from the population. A solution that belongs to the population can be selected zero, one or more than one time. After the selection process, the crossover is applied on the parents to generate a new set of solutions called children. Then, the algorithm applies the mutation on the children. At the end of the current iteration, the algorithm chooses a set of solutions from the population and the children to build the next population of the next iteration.

Algorithm 9 : Genetic algorithm

```
P ← Create an initial population ();
While (the stopping criterion is met)
    P_n ← Selection(P);
    E ← Crossover(P_n);
    E ← Mutation(E);
    P ← Replacement (E, P);
End
Return the best solution found;
```

II.4.2.2.1.1 Selection

Selection is the operator that allows you to choose good solutions from the population to create the set of parents that will produce the children. There are many selection methods in the literature, such as:

- **Uniform selection method:** select randomly one solution and the finesses of the solutions are considered. Therefore, all solutions have the same probability of being selected.
- **Roulette selection method:** consists of randomly selecting a solution where the probability of choosing a solution is proportional to the fitness of that solution.
- **Rank selection method:** each solution is chosen randomly where the probability of selection a solution is proportional to its rank in the population. Each solution

ranked according to his fitness where the worst solution has rank=1 and best solution take the highest rank.

- **tournament selection method:** create a group of solutions from the population randomly. Then the best solution of the group is selected.

II.4.2.2.1.2 Crossover

The crossover is the operator that allows to build one or two children (new solutions) from two parent solutions by recombination the parental genes (genes mean parts of the solution). The crossover is applied to each pair of parents selected with a probability that usually between 65% and 90%. There are several crossover methods such as:

- **Single point crossover method:** it is the most popular crossover method where a random point is chosen and we cut each parent on two parts.

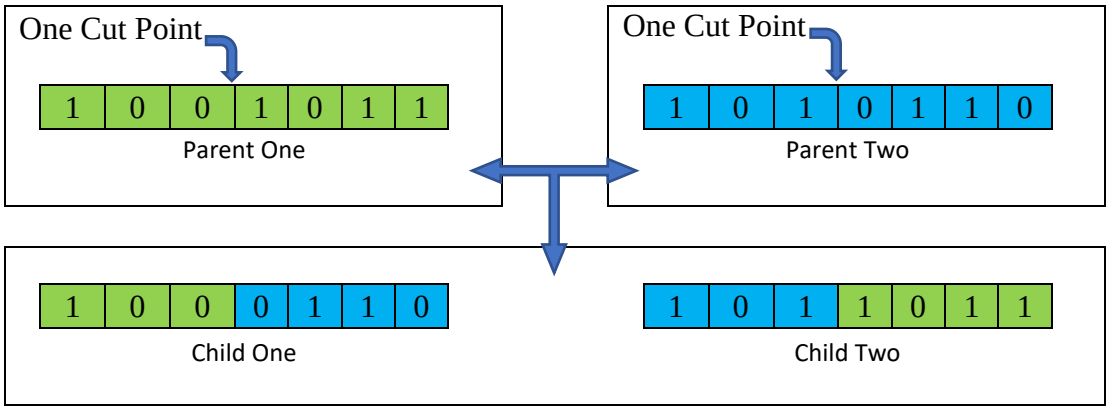


Figure 3 : *Single point crossover in GA*

- **Two-point crossover method:** This method cuts both parents into three parts by two cutting points. The two cutting points are chosen randomly.

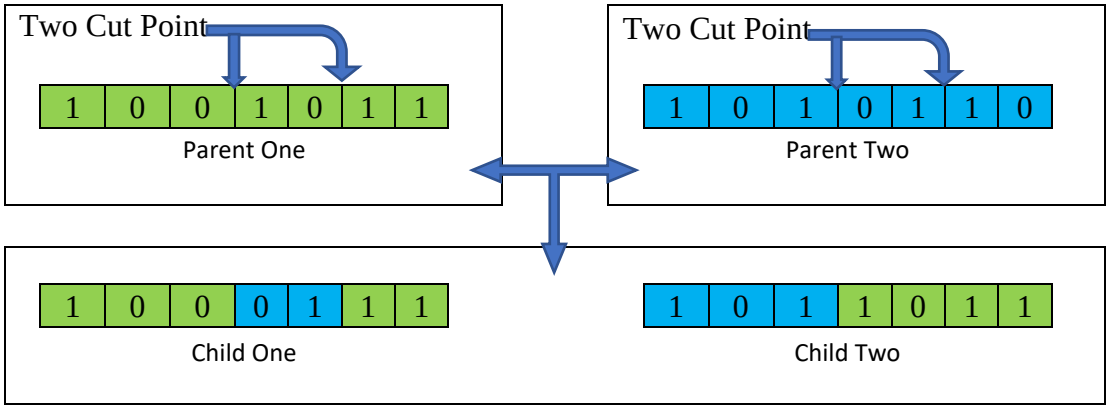


Figure 4: *Two-point crossover in GA*

- **uniform crossover method:** This method consists of going through both parents' gene by gene and each time one of the two genes is selected. The child solution is built by the selected genes.

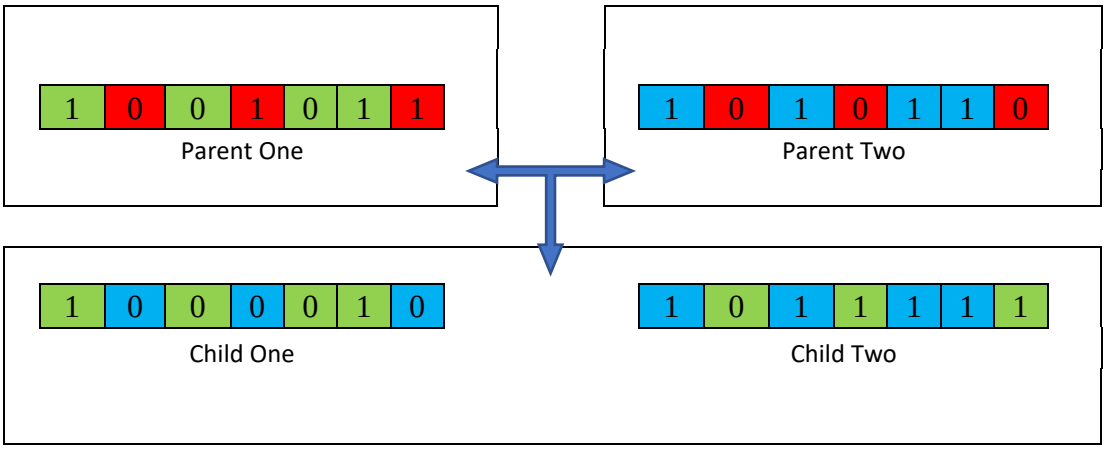


Figure 5 : *uniform crossover in GA*

II.4.2.2.1.3 Mutation

After selection and crossover, we get new population of solutions. Some are directly copied, and others are produced by crossover. Furthermore, Mutation involves making a small random change to the solution. For example, altering one or two genes in the solution. The purpose of the mutation is to ensure a good exploration of research space. The mutation is applied with a probability between 1% and 5%.

II.4.2.2.1.4 replacement

There are several methods that can be used to select the new population of solutions of the next generation. In the following present three methods:

- Completely remove the old population and replace it with the children.
- Merge the two sets the old population and the children and use one of the selection methods (used in the selection phase) to select the solutions.
- Merge the two sets the old population and the children and choose the best solutions (elitist method).

II.4.2.2.2 Particle Swarm Optimization

Particle swarm optimization (PSO) was proposed in 1995 by Kennedy and Eberhart [52]. It is inspired from the swarms of birds that move in groups where every bird can profit from the experience of all other members.

In PSO, a group of particles moves through the search space, representing potential solutions. Each particle adjusts its position based on its own experience and the experiences of neighboring particles. The position and movement of particles are guided by two main factors: the personal best (the best solution found by the particle itself) and the global best (the best solution found by any particle in the swarm).

Algorithm 10 : Particle Swarm Optimization

```
randomly initialized position Xi and velocity Vi of particles ;
While (the stopping criterion is met)
    For (each particle)
        evaluate the fitness function;
        update: Vi and Xi ;
        update: p_best and g_best ;
    End
End
Return the best solution found;
```

II.4.2.2.3 Ant Colony Optimization

Ant Colony Optimization (ACO) was first introduced by Dorigo in the 90s [53]. It was inspired from the behaviors of real ants, which leaving pheromone trails to find their ways back to the nest or to find food. The pheromone trails serve as a form of communication between the ants, allowing them to indirectly exchange information.

In general, ACO is based on the indirect communication of a colony of simple agents, called artificial ants, mediated by artificial pheromone trails. The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem and which the ants adapt during the algorithm's execution to reflect their search experience.

Algorithm 11 : Ant Colony Optimization

```
randomly initialized pheromone values; ;
While (the stopping criterion is met)
    For (each Ant)
        construct a solution;
        update local pheromone values;
    End
End
Return the best solution found so far ;
```

II.4. 3 Hybridization

Hybridization is one of the recent approaches in the field of optimization. In the hope of obtaining better results, many independent optimization algorithms have been combined. Considering the good results that hybridization has obtained, it has become a widely used strategy to solve optimization problems. The huge number of efficient hybrid metaheuristics proves that hybrid metaheuristics represent actually the most efficient algorithms for many classical and real-life difficult problems [54]. In this section we are going to present some hybridization method such as: Coupling metaheuristics with exact methods and Coupling metaheuristics with other metaheuristics

II.4.3.1.1 Coupling metaheuristics with exact methods

Initially, the primary focus of hybridization was on the collaboration between different metaheuristics [55]. This approach was perceived as the most direct and obvious way to combine metaheuristic techniques, leading to the neglect of other potential methods for hybridization. However, when researchers start to explore alternative hybridization approaches, they realized the complementarity between specific exact methods and metaheuristics. In fact, exact methods are known for their capability to solve small instances of the problems and asses their optimality but they are not used to solve large NP-hard problems because they are computationally expensive.

By coupling metaheuristics with exact methods, researchers aimed to leverage the strengths of both approaches. This hybridization allows for the efficient exploration of solution spaces using metaheuristics, while exact methods are employed to refine and improve the solutions obtained. The exact methods can be used to verify the quality of

solutions found by the metaheuristics, potentially reaching optimality for smaller instances within a reasonable time frame.

In [56], the authors presented different state-of-the-art approaches of combining exact algorithms and metaheuristics to solve combinatorial optimization problems and they classed these hybrids in two main categories:

- The first category was called 'collaborative combinations', where the algorithms exchange information but are not part of each other. This category was divided into two sub-categories: Sequential Execution, Parallel and Interleaved Execution.

- The second category was called "Integrative Combinations"; where one technique is an integrated component of another technique. It was also subdivided in two subcategories: Incorporating exact algorithms in metaheuristics and Incorporating metaheuristics in exact algorithms.

II.4.3.1.2 Coupling metaheuristics with other metaheuristics

The combination of different metaheuristics is the most common type of hybridization found in the literature [55]. Coupling metaheuristics is a technique used to combine multiple metaheuristics to improve their overall performance in solving optimization problems. There are several ways to couple metaheuristics with other metaheuristics. Here are a few commonly used approaches [57] [58] [59]:

II.4.3.1.2.1 Parallel hybrids

Parallel hybrids contained multiple metaheuristics that executed simultaneously or in parallel. Each metaheuristic operates independently, exploring the search space and generating solutions concurrently. The solutions generated by the individual metaheuristics are then combined or compared to determine the best solution. parallelization is mainly used for the following reasons: speed-up the search, improve the quality of the obtained solutions and improve the robustness and to solve large scale problems [57].

According to the authors in [58], the parallelization techniques of a "standard" metaheuristic vary depending on whether it is a trajectory-based (single solution) or a

population-based metaheuristic. For trajectory-based metaheuristics, three types of parallelization are often found in the literature:

Parallel moves model: A master–slave approach is conducted here. Where, at the beginning of each iteration, the master duplicates the current solution between distributed nodes. Each solution separately manages their own solution/candidate and the results are then returned to the master. This technique of parallelization does not alter the behavior of the metaheuristic. A relatively recent example of this hybridization can be found in [59].

Parallel multi-start model: This approach of parallelization involves simultaneously launching several trajectory-based methods for computing better and robust solutions. They may be homogeneous or heterogeneous, cooperative or independent, start from the same or different solution(s), and configured with the same or different parameters. An example of this category is in [60]

Move acceleration model: Techniques that fit in this category evaluate the quality of each move in a parallel centralized way. This model becomes attractive when the evaluation function can be parallelized as its computationally expensive. In that case, the function can be regarded as an aggregation of a certain number of partial functions that can be run in parallel. The interested readers are referred to the work of [61].

II.4.3.1.2.2 Sequential hybrids

With regards to the hybridization purpose, non-parallel hybrid algorithms can loosely be divided into two categories [62]

Collaborative Hybrids: Under this category of hybrid algorithms, multiple algorithms work together to solve the same problem directly, with each algorithm being utilized in different search stages. In the simplest case, the contribution weight of each participating algorithm can be considered equal. An example on collaborative Hybrids can be found in [63].

Integrative Hybrids: In this type of hybridization, one primary algorithm is utilized to solve the problem, while another algorithm is applied to optimize the parameters for the primary algorithm. In this aspect, one algorithm is regarded as a subordinate, embedded in a master metaheuristic. For this category, the contributing weight of the secondary algorithm

is Approximatively 10 to 20% [55]. This involves the incorporation of a manipulating operator from a secondary algorithm into a primary algorithm. For example, many algorithms utilized the mutation operator from GA into PSO, resulted in so called Genetic PSO or Mutated PSO.

II.5 Conclusion

In this chapter, we have presented several combinatorial optimization methods and algorithms. We began by presenting the exact methods, within this category, we highlighted branch and bound. Next, we explored heuristic methods such as Greedy algorithms, Greedy Randomized algorithm and Local Search algorithm. Furthermore, we delved into metaheuristic methods, which are general-purpose optimization algorithms applicable to a wide range of combinatorial problems. As metaheuristics, we presented the most popular and the widely used ones in the literature: SA, TS, VNS, GRSP, GA, PSO and ACO. Finally, we discussed the concept of hybridization, which consists of combining different optimization methods.

In the following chapter, we will present the proposed algorithm used to solve TSCFLP.

Chapter III : Simulated annealing for TSCFLP

III.1 Introduction

In this chapter we will present an algorithm to solve Two stage capacitated facility location problem with single commodity and multi-source. The algorithm that we propose is a Simulated Annealing based method which starts from a randomly generated solution and tries to improve it in order to get a high-quality solution within a reasonable running time. In what follows, in section 2, we present the mathematical definition of the TSCFLP as shown in [64]. Then, we give a short review on the most related works which dealt with the TSCFLP. After that, we present our proposed algorithm, and we finish with a conclusion.

III.2 Problem definition

The set of all customers is represented by K , where each customer $k \in \boldsymbol{K}$ has a demand q_k to be met. $\boldsymbol{J}$ represents the warehouses; for each warehouse $j \in \boldsymbol{J}$, we have: a capacity p_j , an opening cost g_j and the shipping product cost d_{jk} to all customers $k \in \boldsymbol{K}$. Similarly, to the warehouses, $\boldsymbol{I}$ represents the factories; each factory $i \in \boldsymbol{I}$ has: a capacity b_i , an opening cost f_i and a shipping product cost c_{ij} of to all warehouses $j \in \boldsymbol{J}$.

To meet demands of all customers, TSCFL can be defined as determining a subset of open warehouses $\overline{W} \subseteq \boldsymbol{J}$ and open factories $\overline{\boldsymbol{F}} \subseteq \boldsymbol{I}$ while the sum of total opening and total shipping costs is minimal.

In order to represent the TSCFL as Mixed Integer Programming (MIP) problem, the decisions to be made at each step have to be defined in term of decision variables. Given that, we define $z_j . j \in \boldsymbol{J}$ and $y_i . i \in \boldsymbol{I}$ as decision variables that indicate whether the warehouse j and factory i will be opened or not. In addition, the decision variables $x_{ij} . i \in \boldsymbol{I}, j \in \boldsymbol{J}$ refer to how much flow is being sent from the factory i to the warehouse j and $s_{jk} . j \in \mathbf{J}, k \in \mathbf{K}$ indicates to how much flow is being sent the warehouse j to the customer k .

The Mixed Integer Programming used for this TSCFL is the same presented in [64]

$$in \quad \sum_{i\in I} f_i\, y_i + \sum_{j\in J} g_j\, z_j + \sum_{i\in I}\sum_{j\in J} c_{ij}\, x_{ij} + \sum_{k\in K}\sum_{j\in J} d_{jk} s_{jk} \qquad (\mathbf{A})$$

$$s.t \sum_{j\in J} s_{jk} \geq q_k \quad \forall\ k \in K \qquad (\mathbf{B})$$

$$\sum_{i\in I} x_{ij} \geq \sum_{k\in K} s_{jk} \quad \forall\, j \in J \qquad (\mathbf{C})$$

$$\sum_{j\in J} x_{ij} \leq b_i y_i \quad \forall\, i \in I \qquad (\mathbf{D})$$

$$\sum_{k\in K} s_{jk} \leq p_j z_j \quad \forall\, j \in J \qquad (\mathbf{E})$$

$$x_{ij} \in \mathbb{R}^+ \quad \forall\, i \in I, j \in J \qquad (\mathbf{F})$$

$$s_{jk} \in \mathbb{R}^+ \quad \forall j \in J, k \in K \qquad (\mathbf{G})$$

$$y_i \in \{0,1\} \quad \forall\ i \in I \qquad (\mathbf{H})$$

$$z_j \in \{0,1\} \quad \forall\ j \in J \qquad (\mathbf{I})$$

The objective function (A) represents the total cost of the shipping system. Constraints (B) ensure that each customer is served. Constraints (C) are conservation constraints, i.e. the total amount of products shipped from a warehouse must be at most the total shipping to it from the factories. Constraints (D) and (E) are capacity constraints assigned to factories and warehouses, respectively. Finally, constraints (F) and (G) are assigned to flow variables, and constraints (H) and (I) impose binary values for the respective variables.

III.3 Most related work

In this section, we present the most related works proposed to solve TSCFLP. In this thesis, we consider the TSCFLP with multiple-source and single-commodity and to the best of our knowledge there are six papers that have been published and dealt with this variant. In the following, we give a short review of some works:

In 2014 Fernandes [64] proposed a set of instances with different characteristics and presented a simple and effective Genetic Algorithm to solve the TSCFLP. Computational results are reported comparing the heuristic results with those obtained by two state-of-the-art Lagrangian heuristics proposed in the literature for the problem

In 2016 Louzada [65] came up with a hybrid method that combined a clustering search (CS) method to define the factories and warehouses to be installed with an exact method to

define the flow of products between factories, warehouses and customers. This work was able to find better solutions compared to the GA of [64]in lower computational times.

In 2019 González [66] presented hybrid method based on the Greedy Randomized Adaptive Search Procedure (GRASP) with a Local Branching procedure. This method was able to obtained relevant results.

Recently in 2021 González [19], the authors developed a hybridization of Clustering Search (CS) and Adaptive Large Neighborhood Search (ALNS) metaheuristics with the Local Branching (LB) technique for the TSCFLP. This hybridization has found high quality solutions in low computational time.

All the mentioned work above used, in the experiments and in the comparison, the same instances proposed in [64]

III.4 Proposed Algorithm

In this section, we present our algorithm for solving the TSCFL problem, which attempts to find solution of good quality in a reasonable time. The algorithm considers that the two levels are independent of each other and at each level there is potential facilities to be opened to satisfy the total demand of customers. In addition, the algorithm treats the level 1 and then deals with the level 2; where in level 1, the warehouses are considered as the facilities (to be opened) and in level 2 the opened warehouses are considered as customers.

First, the algorithm generates randomly an initial solution S_0 which becomes the current solution. Then, at each iteration, the algorithm creates a neighbor solution S_n of the current one S ; If S_n is better than S according to the objective function, then the algorithm will replace S with S_n and update the best solution S_B if it is better the than S_B; otherwise, S_n will be accepted with a probability equal to $e^{-(\Delta/T)}$. At the end of each iteration, the current temperature T is updated. The algorithm stops when the stopping-criterion is met. In the following sub-sections, we highlight the details of each part of the algorithm: (1) the initial solution procedure, (2) the neighborhood creation procedure, (3) the acceptance criterion method and we finish with (4) the annealing method.

Remark: In the most of parts of our algorithm, we treat the level 1 of the problem exactly as the level 2, that means the same techniques and methods used for level 1 are used for level 2. Therefore, and to avoid repetition, the term clients is used to refer to the customers in the level 1 and the opened warehouses in level 2 and we use the term facilities to refer to the warehouses in level 1 and the factories in the level 2.

Algorithm 12 : Simulated Annealing for TSCFLP

```
1:  T ← initial _Temperature;
2:  S_0 ← Initial Solution Procedure ();
3:  while (the stop criterion is not met) do
4:      S_n ← neighbor _Solution (S_0);
5:      r ← a random number between 0 and 1;
6:      Δ ← f(S_n) − f(S_0);
7:      If (f(S_n) < f(S_0) or r < e^(−(Δ/T)))
8:          S_0 ← S_n;
9:          If (f(S_n) < f(S_b) )
10:             S_b ← S_n;
11:         End if
12:     End if
13:     update T;
14: End
15: Return S_b;
```

III.4. 1 Initial Solution Procedure

The Initial Solution Procedure generates the initial solution in a random fashion. it randomly selects warehouses and factories to be opened until the total capacity of the opened warehouses and factories is able to satisfy all demands of customers.

Once we obtain the lists of warehouses and factories, we apply the allocation procedure ***"Allocation Procedure 1"*** described below. This procedure is highlighted in Algorithm 13, the union Sj_0 "Initial Solution of level 1" and Si_0 "Initial Solution of level 2" give us the final Initial Solution.

Algorithm 13 : Initial Solution Procedure

1: $Customers \leftarrow \{k_1, k_2, \ldots\ldots, k_{|K|}\}$;
2: $Warehouses \leftarrow \{j_1, j_2, \ldots\ldots, j_{|J|}\}$
3: $Factories \leftarrow \{i_1, i_2, \ldots\ldots, i_{|I|}\}$;
4: $Sk_0 \leftarrow Customers; Sj_0 \leftarrow \emptyset$; $Si_0 \leftarrow \emptyset$;
5: ***while*** (the stop criterion is not met) do
6: $j_r \leftarrow$ select random $j \in Warehouses$ *and* $j \notin Sj_0$;
7: $Sj_0 \leftarrow Sj_0 \cup \{j_r\}$;
8: ***End***
9: ***while*** (the stop criterion is not met) do
10: $i_r \leftarrow$ select random i $\in$ $Factories$ *and* $i \notin Si_0$;
11: $Si_0 \leftarrow Si_0 \cup \{i_r\}$;
12: ***End***
13: Allocation Procedure $1(Sj_0, Sk_0)$;
14: Allocation Procedure $1(Si_0, Sj_0)$;
15: Return($Si_0 \cup Sj_0$);

III.4. 2 Neighborhood Creation procedure

The neighborhood generation procedure creates a neighbor solution S_n of the current solution S as follows:

Mainly, the algorithm creates S_n based on the swap move. The swap move consists of changing an opened facility (a warehouse or a factory) with a closed one. The facility to be closed is selected randomly however, we open a randomly selected facility but from the best ones. The facility is selected from the top 6 of closed facilities that have the best ratio: capacity/opening-cost. In addition to the swap move, we perform add and drop moves on the solution if the move applied improves the solution quality.

Details of how the S_n solution is created can be found in Algorithm 14. First the algorithm performs the swap move in the first level. In fact, the swap move we propose can create infeasible solution, and therefore, we perform the add move if there is any not satisfied client. After that, we perform the allocation procedure ***"Allocation Procedure 1"*** which allow us to compact all demands of clients into the most appropriate facilities (including the new one). Consequently, after this procedure, we perform the drop move to remove not used facilities (determined by ***Allocation Procedure 1)***. After finishing level 1, the algorithm performs the same steps on the level 2 (see steps from 7 to 12). Then, and after determining the

configuration of the facilities to be opened in the first and the second steps, we delete the old allocation made and we re-allocation clients (customers and opened warehouses) to the facilities using the second allocation procedure ***"Allocation Procedure 2".*** At the end, we construct the final solution S_n by the elements of the level 1 and level 2. The allocation procedure Procedure1 and Procedure2 are presented in detail in the following sub-section.

Algorithm 14 : Neighborhood Creation Procedure (Si_0, Sj_0, Sk)

1: $Sj_n \leftarrow$ swapFacilty(Sj_0) ;
2: ***If*** (customers are unsatisfied) do
3: | addFacility(Sj_n) ;
4: ***End***
5: Allocation_Procedure_1(Sj_n, Sk) ;
6: dropFacility(Sj_n) ;
7: $Si_n \leftarrow$ swapFacilty(Si_0) ;
8: ***If*** (customers are unsatisfied) do
9: | addFacilty(Si_n) ;
10: ***End***
11: Allocation_Procedure_1(Si_n, Sj_n) ;
12: dropFacility(Si_n) ;
13: Allocation_Procedure_2(Sj_n, Sk) ;
14: Allocation_Procedure_2(Si_n, Sj_n) ;
15: Return ($Si_n \cup Sj_n$) ;

III.4. 3 Allocation procedures

The allocation is a very important part of the problem. In this work, we propose two allocation procedures to allocate clients to the opened facilities. The first procedure (***Allocation Procedure 1***) allocates the client to the best facility to minimize ***opening costs***. The second procedure (***Allocation Procedure2***) allocates the client to the nearest facility to minimize ***shipping costs***.

III.4.3.1 Allocation Procedure 1: allocate the best client to the best facility

The Allocation Procedure 1 allocates the best client to the best open facility, where the best client is determined by its demand, and the best open facility is identified based on the highest ratio: capacity /opening-cost.

At each iteration, the best facility $\boldsymbol{\beta}$ with available capacity is selected to meet the demands of unsatisfied clients. Afterwards, the unsatisfied clients are ranked from the best to the worst according to the quantity of demands. Once ranking is made, the facility $\boldsymbol{\beta}$ begins to meet the demands of clients according to their rank, until it is empty. The facility $\boldsymbol{\beta}$ tries to meet the clients demand completely and, if not, partially. If a client is met partially, then we update his demand and we consider him as an unsatisfied client otherwise he is considered as a satisfied client. The procedure ends when all clients are satisfied or there is no available facility.

Algorithm 15 : Allocation_Procedure_1(S_f, S_c)

```
1:  S_f' ← S_f;   S_c' ← S_c;
2:  While (S_f' ≠ ∅ and S_c' ≠ ∅ )
3:      facility_β ← select the best open facility f ∈ S_f' ;
4:      rankingClients(S_c');
5:      While (facility_β.stock ≠ 0 )
6:          client_best ← select best client c ∈ S_c' ;
8:          If (facility_β.stock ≥ c_b.demand)
9:              Update S_f;
10:             Update facility_β.stock;
11:             S_c' ← S_c' − {client_best};
12:         Else
13:             Update S_f;
14:             Update client_best.demand;
15:             S_f' ← S_f' − {facility_β};
17:         End
18:     End
19: End
```

III.4.3.2 Allocation Procedure 2: allocate client to the nearest facility

The main idea of Allocation Procedure 2 is to allocate the client to the nearest open facility with available capacity.

At each iteration, an unsatisfied client **α** is selected randomly to send a request to the nearest open facility ***β*** with available capacity. When the facility β receives the request of **α** he add it to the concurrent clients list with the rest of unsatisfied clients who consider ***β*** as the nearest open facility and arranges the clients in the list from the best to worst where the best client is the client with the big quantity of demand. Once the list of concurrent clients is obtained, the next step for ***β*** is to meet the client's demand based on the priority ranking of each client in the list. ***β*** will continue meeting the demands of clients in the list completely and, if not, partially until either all the clients in the list are satisfied or ***β*** becomes empty. The client that we meet his demand partially, we update his demand and we consider him as an unsatisfied client. The procedure ends when all clients are satisfied.

Algorithm 16: Allocation_Procedure_2(S_f, S_c)

```
1:  S_f' ← S_f;  S_c' ← S_c;
2:  While ( S_c' ≠ ∅ and S_f' ≠ ∅ )
3:      client_α ← select random client c ∈ S_c' ;
4:      facility_β ← nearest open facility(S_f', client_α);
5:      C_list ← concurentClientList( facility_β, S_c');
6:      While (facility_β.stock ≠ 0 and C_list ≠ ∅ )
8:          client_best ← select the best client c ∈ C_list ;
9:          If (facility_β.stock ≥ client_best.demand)
10:             Update (S_f);
11:             Update (facility_β);
12:             C_list ← C_list − {c_b};
13:             S_c' ← S_c' − {c_b};
14:         Else
15:             Update S_f;
17:             Update (client_best.demand);
18:             S_f' ← S_f' − {facility_β};
19:         End
20:     End
21: End
```

III.4. 4 Acceptance criterion method

The acceptance criterion method is the method which determines whether the new neighbor solution is either accepted or discarded using the most popular and common acceptance criterion $e^{-(\Delta/T)}$. At each iteration we compute the fitness variation Δ between the current solution S and the neighbor solution Sn where Δ= f(Sn)-f(S) (f is the function which calculates the cost of the solution). If Δ < 0, then Sn is accepted directly and becomes the current solution (we replace S by Sn); otherwise, the neighbor solution is accepted with a probability p = $e^{-(\Delta/T)}$. where T is the current temperature.

III.4. 5 annealing method

The annealing method is the manner of decreasing the temperature progressively and it is considered as one of the keys of the success of any simulated annealing-based algorithm. First, we start with high-value temperature T0 (T <--- T0) and then we keep decrease the current temperature T when the algorithm is progressed. There are several methods for decreasing the temperature, in our algorithm we use the continuous decrease method where, at each iteration, we decrease the temperature using the formula T=α ×T; and α = 0.99.

III.5 Conclusion

In this chapter we presented our Simulated Annealing based algorithm to solve Two stage capacitated facility location problem with single commodity and multi-source. At first, we started by the mathematical definition of the problem and the most related work, then we presented the algorithm. Moreover, we highlighted all parts of the algorithm by giving details of each one including: the initial solution procedure, neighborhood generation procedure, the acceptance criterion method and the annealing method. In the next chapter, the proposed algorithm will be tested using benchmark data set from the literature and the obtained results will be compared with those of the most related works.

Chapter IV : Experiments

IV.1 Introduction

In this chapter, we will present the experiments performed to test the efficiency of our algorithm. The chapter begins with a detailed description of the benchmark-instances proposed by [64], including the methodology employed for their generation. In addition, an illustrative example of one such instance will be provided. Then, we show the obtained results when we tested the components of the proposed algorithm. Finally, we compare the obtained results with the results which are in the literature.

IV.2 Description of benchmark data set

The instances used were presented in [64]. The authors have generated 50 instances for the TSCFLP using the following parameters:

- number of factories $\boldsymbol{I = 50\ or\ 100}$.
- set number of warehouses $\boldsymbol{J = 2 \times I}$.
- number of customers $\boldsymbol{K = 4 \times I}$
- $B = \frac{\sum_{k \in K} q_k}{I}$
- $P = \frac{\sum_{k \in K} q_k}{J}$

Parameter	Class1	Class2	Class3	Class4	Class5
b_i	[2B 5B]	[5B 10B]	[15B 25B]	[5B 10B]	[5B 10B]
f_i	$[2 \times 10^4 \;\; 3 \times 10^4]$	$[2 \times 10^4 \;\; 3 \times 10^4]$	$[2 \times 10^4 \;\; 3 \times 10^4]$	$[2 \times 10^4 \;\; 3 \times 10^4]$	$[2 \times 10^4 \;\; 3 \times 10^4]$
c_{ij}	[35 45]	[35 45]	[35 45]	$[50 \;\; 1 \times 10^2]$	[35 45]
p_j	[2P 5P]	[5P 10P]	[15P 25P]	[5P10P]	[5P 10P]
g_j	$[8 \times 10^3 \;\; 1.2 \times 10^4]$	$[8 \times 10^3 \;\; 1.2 \times 10^4]$	$[8 \times 10^3 \;\; 1.2 \times 10^4]$	$[8 \times 10^3 \;\; 1.2 \times 10^4]$	$[8 \times 10^3 \;\; 1.2 \times 10^4]$
d_{jk}	[55 65]	[55 65]	$[8 \times 10^2 \;\; 1 \times 10^3]$	$[50 \;\; 1 \times 10^2]$	$[8 \times 10^2 \;\; 1 \times 10^3]$
q_k	[10 20]	[10 20]	[10 20]	[10 20]	[10 20]

Table 1 : *parameters used to generate instances*

Finally, it obtained two set of instances. the first set consists of 25 instances divided into five instance classes whit I=50. In the second set, also with 25 instances, divided into five instance classes whit I=100. All instances can be found at https://github.com/pehgonzalez/OCA, along with the binary file to reproduce the experiments.

IV.2. 1 Capture of instance

in the next figures we present a capture of one instance (PSC1-C1-50) of 50 factories 100 warehouses and 200 customers in class 1.

```
202   298 27022
203   274 29545
204   217 22893
205   138 25144
206   224 24140
207   199 28766
208   277 27298
209   264 27157
210   254 27066
211   280 20190
212   203 25250
```

Figure 9: *example of factories instance*

```
302   107 9674
303   115 10108
304   65 8498
305   80 8161
306   62 11501
307   105 8286
308   93 11338
309   92 9217
310   77 11323
311   72 8497
312   117 8886
```

Figure 8 : *example of warehouses instance*

```
1   50 100 200
2   11
3   18
4   15
5   15
6   12
7   10
8   17
9   17
10  20
```

Figure 6 : *example of customer's instance*

```
252   43 42 41 35 41 37 38 42 36 43
253   41 42 42 40 38 43 39 42 38 35
254   39 37 37 42 36 43 36 35 42 36
255   35 36 43 35 40 45 37 37 35 42
256   42 40 41 41 39 36 40 43 35 35
257   40 44 38 44 41 39 43 40 45 39
258   38 42 36 38 42 44 38 36 41 35
259   41 36 37 44 37 36 44 41 41 44
260   43 44 38 37 36 40 35 45 43 36
261   36 43 37 39 41 42 35 45 43 39
262   36 39 39 44 35 40 43 38 45 45
```

Figure 7 : *example of shipment cost from factories to warehouses*

```
402   60 65 61 56 61 60 58 64 62
403   56 64 62 65 65 64 65 58 56
404   60 58 60 55 65 55 64 62 61
405   65 61 65 61 60 62 64 56 56
406   62 56 59 60 65 63 58 63 63
407   58 62 61 61 64 56 63 60 65
408   61 56 64 55 61 59 57 63 59
409   58 59 61 55 64 56 61 65 62
410   59 65 55 64 55 59 56 63 62
411   56 64 61 63 63 64 56 56 62
412   64 58 60 56 64 61 58 59 59
```

Figure 10 : *example of shipment cost from warehouses to customers*

IV.3 Experiments results

In this sub-section, we will present the obtained results of the proposed algorithm. In fact, we will try to highlight the obtained results by each component of the algorithm as following:

First, we show the obtained results by only the initial solution procedure, then we present the obtained results by the SA but only using the ***"allocation procedure 1"*** and finally we present the obtained results by the complete version of SA that means including the two allocation procedures. This will allow us to highlight clearly the contribution of each part of the algorithm in the final obtained results.

IV.3. 1 Parameters and implementation details

The SA-algorithm was implemented in Java, utilizing the Java SE-17 compiler. All experiments were conducted on a PC equipped with an Intel Core i5-4210U processor, operating at 1.70 GHz (with a maximum turbo frequency of 2.40 GHz), and 12GB of RAM. In our algorithm, we use the following parameters: an initial temperature T=350 000, number of iterations i =3 500 and decreasing the temperature at each iteration using the formula $T=\alpha \times T$ where $\alpha = 0.99$.

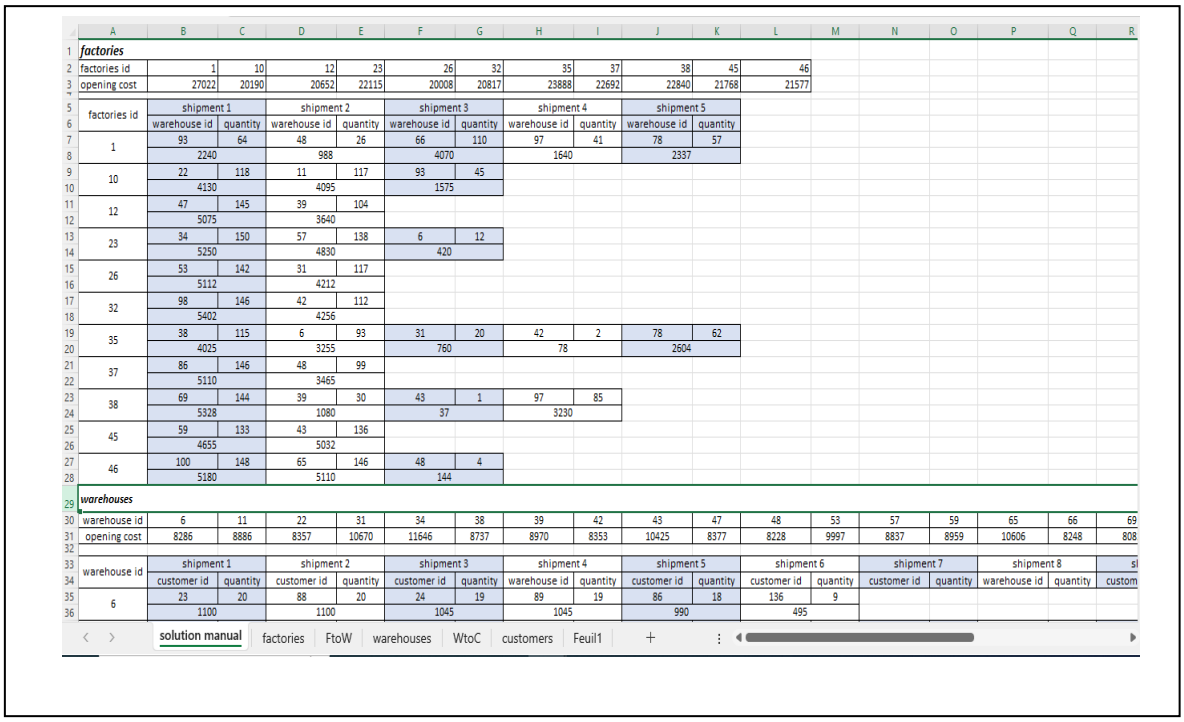
IV.3. 2 Generated solution structure

The generated solution contains the open facilities ids and the shipments from each open facility to its customers. A single shipment is composed of the customer id and the quantity sent. In the next table we present the structure of generated solution.

level two	***Nbr of open factory***	1	2	3	4	5	6	.....	n-1	n
	factory id	1	10	12	20	23	24		45	46
	open factory id	***shipment 1***		***shipment 2***		***shipment 3***		***.....***	***shipment n***	
		warehouse id	***quantity***	***warehouse id***	***quantity***	***warehouse id***	***quantity***		***warehouse id***	***quantity***
	1	66	88	97	22	86	21	.....	69	144
	10	22	118	11	117	/	/	.....	43	45
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	.....	⋮	⋮
	46	100	119	48	132	53	30	.....	42	17
level one	***Nbr of open warehouse***	1	2	3	4	5	6	.....	n-1	n
	warehouse id	11	19	22	31	34	38		97	100
	open warehouse id	***shipment 1***		***shipment 2***		***shipment 3***		***.....***	***shipment n***	
		customer id	***Quantity***	***customer id***	***quantity***	***customer id***	***quantity***		***customer id***	***quantity***
	11	22	16	109	15	123	14	.....	81	5
	19	23	20	41	20	113	19	.....	45	6
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	.....	⋮	⋮
	100	78	18	143	11	13	10	.....	63	4

Table 2 : Generated solution structure

In the next figure, we present a complete solution obtained by the SA algorithm with its corresponding costs and constraints-checking values.



factories

factories id	1	10	12	23	26	32	35	37	38	45	46
opening cost	27022	20190	20652	22115	20008	20817	23888	22692	22840	21768	21577

factories id	shipment 1		shipment 2		shipment 3		shipment 4		shipment 5	
	warehouse id	quantity	warehouse id	quantity	warehouse id	quantity	warehouse id	quantity	warehouse id	quantity
1	93	64	48	26	66	110	97	41	78	57
	2240		988		4070		1640		2337	
10	22	118	11	117	93	45				
	4130		4095		1575					
12	47	145	39	104						
	5075		3640							
23	34	150	57	138	6	12				
	5250		4830		420					
26	53	142	31	117						
	5112		4212							
32	98	146	42	112						
	5402		4256							
35	38	115	6	93	31	20	42	2	78	62
	4025		3255		760		78		2604	
37	86	146	48	99						
	5110		3465							
38	69	144	39	30	43	1	97	85		
	5328		1080		37		3230			
45	59	133	43	136						
	4655		5032							
46	100	148	65	146	48	4				
	5180		5110		144					

warehouses

warehouse id	6	11	22	31	34	38	39	42	43	47	48	53	57	59	65	66	69
opening cost	8286	8886	8357	10670	11646	8737	8970	8353	10425	8377	8228	9997	8837	8959	10606	8248	808

warehouse id	shipment 1		shipment 2		shipment 3		shipment 4		shipment 5		shipment 6		shipment 7		shipment 8		sl
	customer id	quantity	customer id	quantity	customer id	quantity	warehouse id	quantity	customer id	quantity	customer id	quantity	customer id	quantity	warehouse id	quantity	custom
6	23	20	88	20	24	19	89	19	86	18	136	9					
	1100		1100		1045		1045		990		495						

solution manual | factories | FtoW | warehouses | WtoC | customers | Feuil1

Figure 11 : *example of check by hand of a generated solution*

IV.3. 3 The obtained results

In this sub-section, we will present the obtained results for all instances after 10 executions for each instance where table 3 represent the obtained results for the first set of instances and where table 4 represent the obtained results for the second set of instances.

In table 3 and 4 "Cost" represents the best cost obtained, "Best" is calculated as a ratio with the Lower Bound as presented by [67] and measured as following $\frac{\text{Cost} - \text{LowerB}}{\text{LowerB}} \times 100$, "Time" represents the time taken to obtain the results of best solution in seconds, "AVG" refers to the average cost for 10 iterations.

Instances		*lower B*	*SA Initial solution*				*SA without allocation procedure 2*				*Complete Proposed SA*			
Class	*ID*		*Cost*	*Best*	*Time*	*AVG*	*Cost*	*Best*	*Time*	*AVG*	*Cost*	*Best*	*Time*	*AVG*
1	1	721 209,6	919 570	27,50	0,002	952 917,1	746 905	3,56	6,99	747 408,7	723 312	0,29	38,48	724 418,4
	2	730 451,6	951 321	30,24	0,004	966 876,9	757 348	3,68	8,76	759 350,8	733 648	0,44	36,52	734 959,3
	3	731 885,3	926 004	26,52	0,003	972 337,7	755 598	3,24	6,55	758663,6	735 465	0,49	42,09	736 855,9
	4	721 515,0	917 844	27,21	0,003	953 461,7	752 129	4,24	6,29	752 811,7	726 521	0,69	33,78	726 990,1
	5	713 633,8	933 097	30,75	0,004	982 856,9	749 049	4,96	6,23	749 437,1	725 012	1,59	36,53	725 917,6
2	1	479 860,2	588 410	22,62	0,002	614 150,7	515 939	7,52	3,73	517 031,5	495 571	3,27	8,51	496 464,0
	2	483 072,2	587 344	21,59	0,002	610 949,8	520 558	7,76	3,60	521 349,2	499 307	3,36	8,37	499 895,2
	3	486 018,5	607 602	25,02	0,002	623 739,1	515 089	5,98	3,90	517312,6	497 139	2,29	10,68	497 782,5
	4	482 374,6	590 025	22,32	0,001	617 208,1	516 286	7,03	3,64	516 438,5	495 135	2,65	9,01	495 976,8
	5	474 803,3	573 902	20,87	0,002	607 540,2	513 206	8,09	3,77	514 442,8	491 684	3,56	10,12	492 372,2
3	1	2 608 800,0	2 930 202	12,32	0,001	2 956 962,6	2 733 709	4,79	2,17	2 743 738,7	2 705 893	3,72	5,27	2 707 390,2
	2	2 616 252,0	2 933 507	12,13	0,001	2 956 547,7	2 742 314	4,82	2,19	2 754 347,4	2 717 227	3,86	5,22	2 719 006,0
	3	2 598 277,0	2 917 165	12,27	0,001	2 946 060,8	2 719 196	4,65	2,23	2 731 076,7	2 703 275	4,04	5,23	2 704 079,8
	4	2 612 534,0	2 944 401	12,70	0,001	2 964 200,9	2 734 527	4,67	2,23	2746738,2	2 706 294	3,59	5,19	2 708 217,7
	5	2 568 856,0	2 895 540	12,72	0,001	2 911 730,4	2 687 143	4,60	2,22	2 702 786,9	2 662 582	3,65	5,28	2 663 337,9
4	1	525 294,1	737 177	40,34	0,001	767 275,2	626 816	19,33	3,70	637 590,0	551 658	5,02	18,52	552 964,5
	2	526 911,7	736 149	39,71	0,002	765 425,7	629 212	19,42	3,56	639 562,7	549 275	4,24	16,72	549 830,2
	3	532 592,3	742 334	39,38	0,002	775 179,7	631 716	18,61	3,85	635 978,9	552 070	3,66	20,89	553 585,8
	4	529 372,0	749 952	41,67	0,001	768 028,3	631 316	19,26	3,58	635 253,9	549 097	3,73	18,29	550 117,2
	5	521 470,1	726 845	39,38	0,005	763 432,7	629 126	20,64	3,78	635 138,7	541 153	3,77	19,30	543 439,5
5	1	2 743 547,0	3 127 904	14,01	0,002	3 151 768,5	2 851 001	3,92	3,63	2 869 755,5	2 786 366	1,56	24,04	2 787 398,2
	2	2 752 021,0	3 113 600	13,14	0,002	3 142 294,4	2 871 450	4,34	3,65	2 886 021,9	2 792 014	1,45	20,40	2 794 163,7
	3	2 737 769,0	3 104 794	13,41	0,001	3 143 276,4	2 871 790	4,90	3,77	2 886 664,6	2 778 149	1,47	24,05	2 780 151,0
	4	2 748 216,0	3 117 863	13,45	0,001	3 143 600,2	2 854 537	3,87	4,00	2 876 998,4	2 785 792	1,37	20,24	2 786 414,7
	5	2 702 350,0	3 036 409	12,36	0,002	3 093 930,2	2 809 580	3,97	4,14	2 839 225,6	2 746 127	1,62	20,58	2 748 213,6
Average				23,34	0,002			7,91	4,09			2,62	18,53	

Table 3: *Obtained results for the first set of instances (50 factories, 100 warehouses and 200 customers)*

Instances		*lower B*	*SA Initial solution*				*SA without allocation procedure 2*				*Complete Proposed SA*			
Class	*ID*		*Cost*	*Best*	*Time*	*AVG*	*Cost*	*Best*	*Time*	*AVG*	*Cost*	*Best*	*Time*	*AVG*
1	1	1 475 952,0	1 828 300	23,87	0,019	1904589,5	1 533 052	3,87	41,66	1537779,2	1 480 429	0,30	405,85	1480977,5
	2	1 462 736,0	1 823 845	24,69	0,015	1893230,8	1 528 191	4,47	42,28	1531741,7	1 475 778	0,89	394,56	1476677,2
	3	1 492 163,0	1 875 837	25,71	0,016	1922845,5	1 547 170	3,69	43,11	1555074,6	1 497 995	0,39	456,50	1502311,5
	4	1 459 076,0	1 895 797	29,93	0,010	1918665,5	1 515 897	3,89	41,58	1518511,4	1 465 682	0,45	423,06	1466768,9
	5	1 490 742,0	1 877 397	25,94	0,013	1922156,6	1 545 163	3,65	41,66	1546440,3	1 494 759	0,27	412,23	1495309,8
2	1	970 908,5	1 197 620	23,35	0,008	1229016,9	1 018 378	4,89	22,30	1021266,9	976 560	0,58	74,57	977147,2
	2	965 908,5	1 179 390	22,10	0,008	1218840,7	1 015 569	5,14	22,00	1016285,9	972 748	0,71	74,36	973603,3
	3	975 499,7	1 200 219	23,04	0,008	1227977,0	1 024 070	4,98	22,85	1026119,4	979 345	0,39	75,33	979982
	4	973 019,1	1 197 199	23,04	0,006	1224062,5	1 025 588	5,40	14,12	1026777,2	982 423	0,97	76,81	983180,6
	5	941 567,0	1 160 492	23,25	0,008	1188339,7	1 000 789	6,29	16,67	1002318	955 500	1,48	75,82	956156,2
3	1	5 213 566,0	5 854 408	12,29	0,004	5895507,2	5 357 476	2,76	7,12	5392222,2	5 321 644	2,07	47,06	5322853,1
	2	5 191 321,0	5 845 047	12,59	0,004	5882381,3	5 350 515	3,07	7,13	5369124,5	5 304 652	2,18	46,61	5305048,9
	3	5 145 991,0	5 777 355	12,27	0,004	5827795,3	5 289 782	2,79	7,48	5329680,2	5 243 340	1,89	46,42	5243909,4
	4	5 225 601,0	5 893 207	12,78	0,004	5927335,1	5 385 645	3,06	7,08	5410191,5	5 337 794	2,15	48,77	5338638,1
	5	5 163 182,0	5 820 673	12,73	0,003	5851301,6	5 318 879	3,02	7,11	5347076,3	5 274 329	2,15	47,10	5275829,7
4	1	1 052 172,0	1 460 560	38,81	0,009	1512597,6	1 240 328	17,88	14,34	1260009,9	1 071 651	1,85	202,83	1075428,7
	2	1 043 553,0	1 453 370	39,27	0,009	1514275	1 249 883	19,77	13,99	1259734,9	1 063 134	1,88	190,30	1064129,5
	3	1 050 683,0	1 488 965	41,71	0,008	1529420,7	1 236 387	17,67	14,30	1264481,4	1 078 538	2,65	202,00	1080505
	4	1 044 571,0	1 470 996	40,82	0,009	1510561,3	1 228 296	17,59	14,16	1263466	1 065 739	2,03	204,41	1067538,9
	5	1 053 869,0	1 504 979	42,81	0,008	1528956,3	1 238 348	17,50	14,37	1269588,6	1 074 316	1,94	199,87	1075366,8
5	1	5 486 098,0	6 174 225	12,54	0,007	6272719,7	5 653 295	3,05	14,49	5710603,9	5 520 471	0,63	288,03	5523024,6
	2	5 461 680,0	6 203 103	13,57	0,005	6248659,9	5 679 190	3,98	14,17	5697361,3	5 494 412	0,60	287,40	5495848,1
	3	5 425 391,0	6 171 247	13,75	0,007	6203726,7	5 624 333	3,67	14,32	5646171,8	5 469 275	0,81	319,25	5471081,4
	4	5 494 811,0	6 262 371	13,97	0,006	6294609,5	5 677 817	3,33	13,93	5714610,8	5 531 963	0,68	290,16	5533457,5
	5	5 442 621,0	6 204 209	13,99	0,007	6253702,6	5 618 477	3,23	14,17	5657522,4	5 477 523	0,64	459,14	5478812,2
Average				23,15	0,008			6,75	19,46			1,22	213,94	

Table 4: *Obtained results for the second set of instances (100 factories, 200 warehouses and 400 customers)*

Table 3 and Table 4 present the obtained results for all instance. Initially, the solutions obtained by the initial solution procedure are not of good quality but as we can see the procedure is very fast and it has an average running time of **0.002** second for the first set of instances and **0.008** second for the second set of instances. By this running time, we assume that this procedure can be transformed to a greedy or greedy randomized algorithm and used for real-time system (where we need results in very short running time). Then we can see from Table 3 & 4, that the SA without ***"allocation procedure 2"*** considerably enhances the results obtained by the initial solution procedure by an average of **15.43%** in the first set of instances and **16.40%** in the second set of instances while it still has good running times. However, the complete SA consumes more running times than SA without ***"allocation procedure 2",*** we can see that the complete SA improves the results of SA without ***"allocation procedure 2"*** by an average of **5.29%** for first set of instances and by an average of **5.53%** for second set of instances. The results of the complete SA highlight clearly the contribution of the use of the ***"allocation procedure 2"*** in the proposed SA.

IV.4 Comparison with literature

In this sub-section, we compare our obtained results of the complete SA with those of the literature. In table 5 and table 6 column "BST" represent of the best solution obtained and column "AVG" represent the average of best solution obtained in 10 executions where they are calculated as a ratio with the Lower Bound as presented by [67] and measured as following $\frac{\text{sol}- \text{LowerB}}{\text{LowerB}} \times 100$, where sol indicates the BST or AVG from each method. The column "Time" is calculated as the average time in seconds for 10 executions.

Looking at Tables 5 and 6, we can observe that the obtained solutions are very competitive comparing to the solutions of the literature. In term of solutions quality, we can see that the average ratio of the best solutions obtained for the first set of instances is **2.62%** that means we get near to the literature methods by **0.64%** to GA, **0.66** to CS+CPLEX, **0.62%** to GRASPH and **0.69%** to CS-ALNS-LB and the average ratio of the best solutions obtained for the second set of instances is **1.22%** that means we get near to the literature methods by **0.26%** to GA, **0.56** to CS+CPLEX, **0.47%** to GRASPH and **0.58%** to CS-ALNS-LB.

Instances		lower B	GA [64]		CS+CPLEX [65]			GRASPH [66]			CS-ALNS-LB [19]			SA		
Class	ID		AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time
1	1	721 209,6	0,13	264,78	0,13	0,22	48,90	0,13	0,13	4,50	0,13	0,13	15,41	0,29	0,44	40,64
	2	730 451,6	0,40	257,17	0,23	0,31	76,86	0,24	0,24	21,34	0,23	0,23	21,02	0,44	0,62	36,26
	3	731 885,3	0,24	263,35	0,21	0,29	51,94	0,22	0,22	30,46	0,21	0,21	55,22	0,49	0,68	41,24
	4	721 515,0	**0,81**	242,93	**1,19**	1,41	96,62	0,50	0,50	29,08	0,50	0,50	18,98	**0,69**	0,76	34,05
	5	713 633,8	0,82	251,79	0,81	0,88	56,14	0,81	0,81	160,01	0,81	0,82	64,69	1,59	1,72	36,49
2	1	479 860,2	2,69	144,39	2,68	3,27	27,69	2,69	2,69	383,27	2,68	2,68	15,43	3,27	3,46	8,49
	2	483 072,2	2,30	144,16	2,30	2,62	81,36	2,30	2,34	368,58	2,30	2,30	64,89	3,36	3,48	8,37
	3	486 018,5	2,14	150,60	1,86	2,03	30,82	1,88	1,94	590,94	1,86	1,86	48,02	2,29	2,42	10,85
	4	482 374,6	2,04	142,25	2,01	2,01	83,02	2,02	2,02	365,57	2,01	2,02	47,37	2,65	2,82	9,133
	5	474 803,3	3,14	126,08	3,12	3,39	36,98	3,12	3,12	590,87	3,12	3,12	33,33	3,56	3,70	9,62
3	1	2 608 800,0	3,07	125,90	3,07	3,07	19,89	3,22	3,30	596,03	3,07	3,10	104,13	3,72	3,77	5,32
	2	2 616 252,0	3,12	130,22	3,10	3,10	73,09	3,37	3,39	594,73	3,13	3,20	94,18	3,86	3,93	5,23
	3	2 598 277,0	3,11	123,56	3,09	3,10	52,98	3,23	3,32	591,33	3,09	3,14	77,21	4,04	4,07	5,237
	4	2 612 534,0	3,07	107,73	3,05	3,05	45,21	3,18	3,29	593,68	3,05	3,10	80,60	3,59	3,66	5,22
	5	2 568 856,0	3,01	110,36	3,01	3,01	47,45	3,14	3,22	593,27	3,01	3,03	76,79	3,65	3,68	5,26
4	1	525 294,1	3,14	138,25	3,14	3,60	89,47	3,29	3,29	591,54	3,14	3,14	39,42	5,02	5,27	18,50
	2	526 911,7	2,33	139,83	2,43	2,71	102,38	2,65	2,80	592,19	2,43	2,45	57,23	4,24	4,35	16,82
	3	532 592,3	2,66	144,88	2,41	2,44	118,38	2,45	2,92	591,35	2,30	2,44	67,49	3,66	3,94	21,12
	4	529 372,0	2,53	127,30	2,35	2,66	133,69	2,36	2,50	591,31	2,35	2,36	55,50	3,73	3,92	18,41
	5	521 470,1	3,13	120,27	3,15	3,53	115,67	3,15	3,23	388,72	3,12	3,12	46,46	3,77	4,21	18,99
5	1	2 743 547,0	1,20	164,42	1,19	1,19	157,20	1,24	1,31	591,33	1,16	1,18	64,60	1,56	1,60	21,87
	2	2 752 021,0	1,07	156,71	1,08	1,11	89,13	1,15	1,17	591,31	1,07	1,07	68,15	1,45	1,53	20,49
	3	2 737 769,0	1,10	191,60	1,09	1,10	126,33	1,29	1,30	591,78	1,09	1,13	70,37	1,47	1,55	25,37
	4	2 748 216,0	1,07	136,87	1,05	1,12	149,59	1,06	1,07	591,67	1,05	1,07	67,09	1,37	1,39	20,55
	5	2 702 350,0	1,25	145,07	1,24	1,24	54,96	1,29	1,34	592,50	1,23	1,25	67,90	1,62	1,70	21,74
Average			1,98	162,02	1,96	2,10	78,63	2,00	2,06	449,09	1,93	1,95	56,86	2,62	2,75	18,03

Table 5: *Comparison obtained results for the first set of instances with literature*

Instances		lower B	GA [64]		CS+CPLEX [65]			GRASPH [66]			CS-ALNS-LB [19]			SA		
Class	ID		AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time
1	1	1 475 952,0	**0,55**	1 268,12	0,10	0,30	384,79	0,10	0,10	381,21	0,09	0,11	339,69	**0,30**	0,34	400,20
	2	1 462 736,0	**1,01**	1 250,09	0,34	0,70	716,06	0,12	0,12	130,86	0,12	0,20	231,41	**0,89**	0,95	397,38
	3	1 492 163,0	0,34	1 367,04	**0,54**	1,00	654,10	0,15	0,15	281,03	0,16	0,20	266,18	**0,39**	0,68	456,65
	4	1 459 076,0	**0,49**	1 285,78	0,24	0,49	740,44	0,22	0,28	484,13	0,22	0,24	240,00	**0,45**	0,53	428,89
	5	1 490 742,0	**0,67**	1 303,93	0,11	0,33	850,42	0,12	0,12	82,79	0,11	0,12	192,28	**0,27**	0,31	413,74
2	1	970 908,5	**0,89**	675,48	0,26	0,52	989,39	0,27	0,36	584,20	0,26	0,30	310,07	**0,58**	0,64	75,51
	2	965 908,5	**0,74**	662,96	0,28	0,46	668,55	0,28	0,34	559,24	0,28	0,33	257,51	**0,71**	0,80	75,16
	3	975 499,7	**1,42**	650,19	0,14	0,25	992,58	0,14	0,14	487,39	0,14	0,17	236,91	**0,39**	0,46	73,61
	4	973 019,1	0,56	657,63	0,28	0,40	688,28	0,35	0,41	592,81	0,29	0,32	326,07	0,97	1,04	76,90
	5	941 567,0	1,12	646,23	0,60	0,65	858,06	0,86	1,06	592,22	0,60	0,69	283,63	1,48	1,55	76,96
3	1	5 213 566,0	1,63	617,24	1,62	1,63	1 113,37	1,79	1,92	598,74	1,62	1,66	600,74	2,07	2,10	47,11
	2	5 191 321,0	1,67	601,51	1,65	1,65	1 312,48	1,84	1,94	596,33	1,67	1,71	503,07	2,18	2,19	46,59
	3	5 145 991,0	1,58	597,43	1,57	1,58	958,88	1,77	1,84	594,41	1,58	1,61	491,88	1,89	1,90	46,37
	4	5 225 601,0	1,74	622,04	1,72	1,73	1 033,46	2,01	2,09	600,38	1,72	1,76	498,75	2,15	2,16	47,17
	5	5 163 182,0	1,72	629,84	1,67	1,69	1 073,08	2,02	2,03	594,38	1,73	1,75	518,88	2,15	2,18	47,45
4	1	1 052 172,0	0,82	577,91	0,61	0,87	1 040,75	0,73	0,74	594,13	0,65	0,78	313,52	1,85	2,21	200,26
	2	1 043 553,0	0,93	560,18	0,83	0,88	852,22	0,77	0,85	593,65	0,67	0,71	314,25	1,88	1,97	188,17
	3	1 050 683,0	1,88	584,40	0,62	0,81	677,12	1,20	1,77	592,49	0,78	1,12	299,23	2,65	2,84	195,95
	4	1 044 571,0	0,96	592,63	0,74	0,94	1 099,22	0,98	1,01	594,38	0,80	0,90	303,95	2,03	2,20	205,05
	5	1 053 869,0	0,64	607,94	0,56	0,89	543,26	0,56	0,65	594,12	0,52	0,63	295,09	1,94	2,04	197,22
5	1	5 486 098,0	0,48	706,40	0,38	0,40	801,30	0,43	0,54	593,59	0,38	0,40	326,89	0,63	0,67	286,15
	2	5 461 680,0	0,47	683,08	0,39	0,44	849,41	0,40	0,42	593,52	0,38	0,41	285,26	0,60	0,63	286,64
	3	5 425 391,0	0,62	672,63	0,49	0,53	770,91	0,59	0,59	594,61	0,39	0,48	259,29	0,81	0,84	317,31
	4	5 494 811,0	0,52	689,38	0,43	0,47	657,66	0,50	0,51	593,16	0,41	0,43	322,00	0,68	0,70	291,60
	5	5 442 621,0	0,47	670,38	0,39	0,45	1 323,85	0,46	0,48	592,62	0,38	0,40	262,75	0,60	0,63	457,41
Average			0,96	767,22	0,66	0,80	865,99	0,75	0,82	523,86	0,64	0,70	331,17	1,22	1,30	213,42

***Table 6**: Comparison obtained results for the second set of instances with literature*

Furthermore, the SA outperforms the GA in the instances PSC4-C1-50, PSC1-C1-100, PSC2-C1-100, PSC4-C1-100, PSC5-C1-100, PSC1-C2-100, PSC2-C2-100 PSC3-C2-100. Also, it outperforms the CS+CPLEX in the instances PSC4-C1-50 and PSC3-C1-100.

On the other hand, we can observe that the SA has the best running time over all instances with an average of **115.73** seconds comparing to **464.64** seconds of GA, 472.31 seconds of CS+CPLEX, **486.48** seconds of GRASPH and **194.02** seconds of CS-ALNS-LB. So, we can say that the SA proposed has a very good comprise between solutions quality and running times comparing to the literature. In addition, we can assume that the SA can be improved by adding other techniques/methods which will improve its solutions quality without losing its competitivity in term of running times.

IV.5 Conclusion

In this chapter we presented detail description of the benchmark-instances from the literature. Then, we presented the results obtained by our algorithm where testing it on all instances. Finally, we compared the obtained results with the results in the literature.

Conclusion

The supply chain is the process of moving goods or services from the point of the origin to the point of destination. The supply chain optimization is important for businesses because it allows to reduce costs, improve customer satisfaction, minimize times and lead to respond effectively to market demand.

In this thesis, we have proposed and validated a method to solve multi-source, single-product TSCFLP, where the main objective was to find the best sub-set of facilities that meet the demands of all customers with the lowest cost.

In the first chapter, we presented the most related location problems to our case such as Capacitated, single-stage problem, multi-stage location problems. A general and brief descriptions of the several variants of location problems (including two-stage capacitated facility location problems) have been given with the most popular application in the real life.

In the second chapter we focused on the optimization methods and algorithms, where exact, heuristic, metaheuristic and hybrid methods were presented. We Also provided some of the most popular and efficient algorithms used in the field of optimization.

In the third chapter, we presented the TSCFLP problem where a mathematical model of the problem from the literature is given. Then, we presented our proposed simulated annealing algorithm to solve the problem that it has two main processes: the first is the selection of facilities and the second is the allocation of customers. For allocation we have proposed two procedures with the aim of improving the quality of solutions.

In the fourth chapter, we presented the obtained results and we compared them with the most recent results found in the literature. The proposed SA obtained very competitive results comparing to the results of the literature and it has the best running time over all. Also, it outperforms the GA on 8 instances and the CS+CPLEX on 2 instances.

Finally, we are looking forward to:

-Improve our algorithm using other facilities selection techniques .

- Propose another metaheuristic to solve the TSCFLP.

-Propose a hybrid algorithm which combine the proposed SA with another algorithm.

-Propose a similar algorithm to solve the Multi Stage Facility Location Problems.

-

Bibliography

[1] A. Klose et A. Drexl, «Facility location models for distribution system design,» *European Journal of Operational Research ,* vol. 162, p. 4–29, 2005.

[2] C. Ortiz-Astorquiza, I. Contrerasn et G. Laporte, «Multi-level Facility Location Problems Volume,» *European Journal of Operational Research,* vol. 267, n° %13, pp. 791-805, 16 June 2018.

[3] S. Basu, M. Sharma et G. P. Sarathi, «, Metaheuristic applications on discrete facility location problems: a survey,» *Operational Research Society of India,* 2014.

[4] D. A. Schilling, «Dynamic location modeling for public-sector facilities: A multicriteria approach,» *Decision Sciences,* vol. 11, p. 714–724, 1980.

[5] D. Erlenkotter, «A comparative study of approaches to dynamic facility location problems,» *European Journal of Operational Research,* vol. 6, p. 133–143, 1981.

[6] A. Shulman, «An algorithm for solving dynamic capacitated plant location problems with discrete expansion sizes,» *Operations Research,* vol. 39, n° %13, p. 423–436, 1991.

[7] G. LAPORTE, F. V. LOUVEAUX et L. V. HAMME, «Exact solution to a location problem with stochastic demands,» *Transportation Science ,* vol. 28, p. 95–103, 1994.

[8] O. Listes et R. Dekker, Stochastic approaches for product recovery network design: A case study, Econometric Institute Report EI : Erasmus University Rotterdam, 2001.

[9] J. G. Klincewicz, «A dual algorithm for the uncapacitated hub location problem,» *Location Science,* vol. 4, p. 173–184, 1996.

[10] H. W. Hamache, M. Labbé, S. Nickel et T. Sonneborn, Polyhedral properties of the uncapacitated multiple allocation hub location problem., Kaiserslautern: Fraunhofer Institut für Techno- und Wirtschaftsmathematik, 2000.

[11] T. Aykin, «Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem,» *European Journal of Operational Research,* vol. 79, p. 501–523, 1994.

[12] J. Ebery, M. Krishnamoorthy, A. Ernst and N. Boland, "The capacitated multiple allocation hub location problem: Formulations and algorithms," *European Journal of Operational Research,* vol. 120, p. 614–631, 2000.

[13] C. Prodhon et C. Prins, «A survey of recent research on location-routing problems,» *European Journal of Operational Research,* vol. 238, pp. 1-17, 2014.

[14] A. Klose, Standortplanung in distributiven Systemen. Physica, Heidelberg.], Physica-Verlag Heidelberg, 2001.

[15] T. Aykin, «The hub location and routing problem,» *European Journal of Operational Research,* vol. 83, p. 200–219, 1995.

[16] J. Current, H. Min and D. Schilling, "Multiobjective analysis of facility location decisions," *European Journal of Operational Research,* vol. 49, p. 295–307, 1990.

[17] C. S. Revelle et G. Laporte, «The plant location problem: New models and research prospects,» *Operations Research,* vol. 44 , n° %16, p. 864–874, 1996.

[18] E. Fernández et J. Puerto, «Multiobjective solution of the uncapacitated plant location problem,» *European Journal of Operational Research,* vol. 145, p. 509–529, 2003.

[19] P. H. González, I. M. Gabriel Souto, G. R. Mauri and G. M. Ribeiro, "A hybrid matheuristic for the Two-Stage Capacitated Facility Location problem," *Expert Systems With Applications,* vol. 185, 2021.

[20] S. Tragantalerngsak, M. Rönnqvist and J. Holt, "Exact method for the two-echelon, single-source, capacitated facility location problem," *European Journal of Operational Research,* vol. 123, no. 3, pp. 473-489, 2000.

[21] H. Pirkul et V. Jayaraman, «A Multi-Commodity, Multi-Plant, Capacitated Facility Location Problem: Formulation and Efficient Heuristic Solution,» *Computers & Operations Research,* vol. 25, pp. 869-878, 1998.

[22] H. PIRKUL and V. JAYARAMAN, "Production, Transportation, and Distribution Planning in a Multi-Commodity Tri-Echelon System," *Focused Issue on the Transportation/Manufacturing Interface ,* vol. 30, no. 4, pp. 291-302, 1996.

[23] F. Vasko, D. Newhart, K. Stott and F. Wolf, "A large-scale application of the partial coverage uncapacitated facility," *Journal of the Operational Research Society,* vol. 54, p. 11–20, 2003.

[24] B. Boffey, D. Yates et R. D. Galvão, « An algorithm to locate perinatal facilities in the municipality of Rio de Janeiro. Journal of,» *Journal of the Operational Research Society* , vol. 54, pp. 21-31, 2003.

[25] J. M. Mulvey et H. P. Crowder, «Cluster analysis: An application of Lagrangean relaxation. Management Science,» vol. 25, n° %14, p. 329–340, 1979.

[26] G. Cornuejols, M. L. Fisher and G. L. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Management Science,* vol. 23, p. 789–810, 1977.

[27] R. M. Nauss et R. E. Markland, «Theory and application of an optimizing procedure for lock box location analysis,» *Management Science,* vol. 27, p. 855–865, 1981.

[28] J. Current et C. Weber, «Application of facility location modeling constructs to vendor selection problems,» *European Journal of Operational Research,* vol. 76, n° %13, p. 387–392, 1994.

[29] P. Hansen, E. d. L. P. Filho et C. C. Ribeiro, «Location and sizing of offshore platforms for oil exploration.,» *European Journal of Operational Research,* vol. 58, p. 202–214, 1992.

[30] P. Hansen, E. d. L. P. Filho et C. C. Ribeiro, «Modelling location and sizing of offshore platforms,» *European Journal of Operational Research*, vol. 72, p. 602–606, 1994.

[31] M. L. FISHER et D. S. HOCHBAUM, «Database location in computer networks,» *Journal of the ACM* , vol. 7, p. 718–735, 1980.

[32] T. Boffey, «Location problems arising in computer networks.,» *The Journal of the Operational Research Society*, vol. 40 , n° %14, p. 347–354, 1989.

[33] B. Gavish, «Topological design of telecommunication networks—local access design methods,» *Annals of Operations Research*, vol. 33, p. 17–71, 1991.

[34] A. Mirzaian, «Lagrangian relaxation for the star-star concentrator location problem: Approximation algorithm and bounds,» *Networks* , vol. 15, p. 1–20, 1985.

[35] H. Pirkul, «Efficient algorithms for the capacitated concentrator location problem,» *Computers & Operations Research*, vol. 14, p. 197– 208, 1987.

[36] A. Caprara, M. Fischetti et D. Maio, «Exact and approximate algorithms for the index selection problem in physical database design,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, n° %16, p. 955–967, 1995.

[37] T. Stéphane, «Data mining et statistique décisionnelle : l'intelligence,» *Editions Technip*, 2012.

[38] O. GUEMRI, Proposition de solutions pour l'optimisation, Département d'Informatique Laboratoire d'Informatique d'Oran (LIO) éd., Thèse du Doctorat en LMD , 2017.

[39] A. H. Land et A. G. Doig, «An automatic method of solving discrete programming problems,» *In: Econometrica: Journal of the Econometric Society* , p. 497– 520, 1960.

[40] J. Clausen, *Branch and Bound Algorithms-Principles and Examples,* Copenhagen, Denmark: Department of Computer Science, University of Copenhagen, 1999, p. 1–30.

[41] E. A. Feigenbaum et J. Feldman, *Computers and thought*, New York,NY, USA: McGraw-Hill, 1963.

[42] M. G. C. RESENDE et C. C. RIBEIRO, «Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications,» chez *Handbook of Metaheuristics*, 2010, pp. 283-319.

[43] H. Osman et G. Laporte, « Metaheuristics: a bibliography,» *Annals of Operations Research*, vol. 63, pp. 513-623, 1996.

[44] S. Voß, S. Martello, I. Osman et C. Roucairol, *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[45] S. Kirkpatrick, C. Gelatt et M. Vecchi, «Optimization by Simulated Annealing,» *Science, new series,* p. 671 – 680 , 1983.

[46] F. Glover, «Future paths for integer programming and links to artificial intelligence,» *Computers and Operations Research,,* vol. 13, pp. 533-549, 1986.

[47] T. Feo et M. Resende, «A probabilistic heuristic for a computationally difficult set covering problem,» *Operations Research Letters,* vol. 8, p. 67–71, 1989.

[48] N Mladenovi´c et P. Hansen, «Variable neighborhood search,» *Computers & Operations Research*, vol. 24, p. 1097–1100, 1997.

[49] N. Mladenovi´c, P. Hansen et J. M. Pérez, « Variable neighbourhood search: Methods and applications,» *Annals of Operations Research*, vol. 175, p. 367–407, 2010.

[50] R. Kammarti, *Evolutionary approaches for the resolution of static 1-PDPTW and dynamic,* Doctoral thesis, Ecole Centrale de Lille, France, 2006.

[51] Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press: Ann Arbor, 1975.

[52] N. M. Razali et J. Geraghty, A genetic algorithm performance with different selection strategies, 2 éd., vol. 2, Proceedings of the World Congress on Engineering , 2011.

[53] J. Kennedy et R. Eberhart, «Particle swarm optimization,» *IEEE International Conference on Neural Network*, p. 1942–1948, 1995.

[54] M. Dorigo, G. D. Caro et L. Gambardella, «Ant algorithms for discrete optimization,» *Artificial Life*, vol. 5, n° %12, p. 137–172, 1999.

[55] G. Talbi, «Combining metaheuristics with mathematical programming, constraint programming and machine learning,» *Annals of Operations Research*, vol. 240, n° %11, p. 171–215, 2016.

[56] A. E. SAMROUT, Hybridization of Multicriteria Metaheuristic Optimization Methods, UNIVERSITE DE TECHNOLOGIE DE TROYES, 2018.

[57] J. Puchinger et G. R. Raidl, «Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification,» *Springer*, p. 41, 2005.

[58] E.-G. Talbi, «Metaheuristics: from design to implementation,» *John Wiley & Sons*, vol. 74, 2009.

[59] E. Alba, G. Luque et S. Nesmachnow, «Parallel metaheuristics: recent advances and new trends,» *International Transactions in Operational Research*, vol. 20, n° %11, p. 1–48, 2013.

[60] W. Bożejko, J. Pempera et C. Smutnicki, «Parallel tabu search algorithm for the hybrid flow shop problem,» *Computers & Industrial Engineering* , vol. 65, n° %13 , p. 466–474, 2013.

[61] M. Hijaze et D. Corne, «An investigation of topologies and migration schemes for asynchronous distributed evolutionary algorithms,» *Nature & Biologically*

Inspired Computing, 2009. NaBIC 2009. World Congress on. IEEE. , p. 636– 641, 2009.

[62] Y.-L. Chang, K.-S. Chen, B. Huang et W.-Y. Chang, «A parallel simulated annealing approach to band selection for high-dimensional remote sensing images,» *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 4, n° %13, p. 579–590, 2011.

[63] T. O. Ting, X.-S. Yang, S. Cheng et K. Huang, «Hybrid metaheuristic algorithms: past, present, and future",» *Recent Advances in Swarm Intelligence and Evolutionary Computation. Springer,* p. 71–83, 2015.

[64] P. Shelokar, P. Siarry, V. Jayaraman et B. Kulkarni, «Particle swarm and ant colony algorithms hybridized for improved continuous optimization,» *Applied mathematics and computation,* vol. 188, n° %11 , p. 129–142, 2007.

[65] D. R. Fernandes, C. Rocha, D. Aloise, G. M. Ribeiro, E. M. Santos et A. Silva, «A simple and effective genetic algorithm for the two-stage capacitated facility location problem,» *Computers & Industrial Engineering,* vol. 75, p. 200–208, 2014.

[66] R. R. Louzada, G. R. Mauri et G. M. Ribeiro, «M´etodo heurístico híbrido para resoluç˜ao do problema de localizaçao ˜ de facilidades capacitadas em dois níveis,» *Simposio ´ Brasileiro de Pesquisa Operacional - SBPO,* p. 2460–2471, 2016.

[67] P. H. González, G. S. d. J. Augusto, G. R. Mauri, G. Ribeiro et L. G. Simonetti, «Grasp híbrido para resoluçao ˜ do problema de localizaç˜ ao de facilidades capacitadas em dois níveis.,» *Simposio ´ Brasileiro de Pesquisa Operacional - SBPO,* p. 1–11, 2019.

[68] P. Guo, W. Cheng et Y. Wang, «Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems,» *Expert Systems with Applications,* vol. 71, pp. 57-68, 2017.

[69] I. Dumitrescu et T. Stützle, «Combinations of local search and exact algorithms,» *Springer,* p. 211–223, 2003.

[70] E.-G. Talbi, «Metaheuristics: from design to implementation,» *John Wiley & Sons*, vol. 74, 2009.

[71] I. Litvinchev et L. Ozuna, «Lagrangian Bounds and a Heuristic for the Two-Stage Capacitated Facility Location Problem,» *International Journal of Energy Optimization and Engineering*, vol. 1, pp. 59-71, 2012.