Ministery of Higher Education and Scientific Research
University of Bordj Bou Arreridj
Faculty of Mathematics and Computer science
Department of operational research

Master's thesis

Presented by :

Bendrimia Alaaedine

A thesis submitted in partial fulfillment of the requirements for the Master degree in operational research

Ant Colony Optimization for Multi objective Combinatorial optimization Problems

This thesis is presented at 21/06/2023 in the presence of examining committee of :

2022/2023

Dedication

I dedicate This work :

To the tow persons who give me all the love and all kindness **my parents ,**

To my **Aunts** who sacrifice to raise me

To my **friends** who with whom I fell fine

To my **brothers** who are my support In life

To **all persons** love me and I love them.

And of course to my supervisor **Saha Adel** who help me al lot to achieve this work

Abstract

In this work, we use a metaheuristic method which depends on swarm intelligence called ant colony optimization to solve the multi objective combinatorial optimization problems. We will focus on the multi-objective travelling salesmen problem.

Key words :

Ant colony optimization, Combinatorial optimization, Multi objective optimization, Multi objective Travelling Salesmen problem.

Résumé

Dans ce travail, nous utilisons une méthode métaheuristique basée sur l'intelligence collective appelée optimisation de colonie de fourmis pour résoudre des problèmes d'optimisation combinatoire multi-objectifs. Nous nous concentrerons le problème du voyageur de commerce multi objectifs.

Mots clés :

Optimisation par colonies de fourmis (OCF), optimisation Multi objectif, Optimisation combinatoire, Problème du voyageur de commerce multi -objectif.

الملخص

ان العمل المنجز يستخدم طريقة تقريبية تعتمد على ذكاء السرب وتسمى التحسين باستعمال مستعمرات النمل من اجل حل مشاكل التحسين المركبة ذات الهدف المتعدد. سنركز على حل مشكلة البائع المتنقل ذو الأهداف المتعددة.

الكلمات الدالة

مشكل البائع المتنقل متعدد الأهداف, مشكل التحسين متعدد الأهداف, خوارزمية مستعمرة النمل

Contents

List of Figures

Glossary

General Introduction **1**

I Combinatorial Optimization **3**

I.1 Introduction 3
I.2 Combinatorial optimization problem 4
I.3 Single-objective combinatorial optimization problem 4
I.4 Multi-objective combinatorial optimization problem 5
I.5 Examples 6
I.6 Conclusion 6

II Multi Objective Optimization **7**

II.1 Introduction 7
. . .
II.2 Multi objective optimization concepts.......... 8
II.3 Multi objective optimization problems 13
II.3.1 Academic applications.......... 13
II.3.1.1 Multi objective combinatorial problems.......... 13
II.3.2 Real-life applications.......... 15
II.4 Conclusion 16

III Methods for Combinatorial Optimization Problem **17**

III.1 Introduction.......... 17
III.2 Resolution methods.......... 18
III.2.1 The exact methods.......... 18
III.2.2 Approximate methods 19
III.2.2.1 Heuristics.......... 19
III.2.2.2 Metaheuristics.......... 19
III.2.2.2.1 Methods with a single-solution.......... 20
III.2.2.2.2 Methods with a population of solutions 20
III.2.2.2.3 Algorithms based on swarm intelligence 20
III.2.2.2.4 Ant colony optimization.......... 21
III.2.3 Hybrid methods.......... 21
III.3 Conclusion.......... 22

IV Ant Colony Optimization **23**

IV.1 Introduction 23
IV.2 From real to artificial ants 23
IV.2.1 Ants' foraging behavior and optimization..........23
IV.2.2 Double bridge experiments..........24

IV.2.3 The comparison between artificial ants and real ants 25
IV.3 Ant colony optimization (ACO) algorithm.. 26
IV.3.1 Steps of ACO algorithm.. 27
IV.3.2 Ant colony optimization algorithms for the traveling salesman problem 28
IV.3.2.1 The significance of the traveling salesman problem (TSP) 28
IV.3.2.2 Tour construction 29
IV.3.2.3 Update of pheromone trails 30
IV.4 Conclusion 31

V Multi Objective Ant Colony Optimization for Multi Objective Travelling Salesmen Problem 32
V.1 Introduction.. 32
V.2 Multi objective travelling salesmen problem 32
V.3 Multi objective ant colony optimization... 33
V.3.1 Variants.. 33
V.3.2 Single colony and single pheromone matrix 33
V.3.3 Single colony and multiple pheromone matrix 34
V.3.4 Multi colony and single pheromone matrix 34
V.3.5 Multi colony and multi pheromone matrix 34
V.4 PACO for multiple objective optimization 35
V.4.1 Algorithm of PACO.. 35
V.5 Conclusion.. 36

VI Implementation 37
VI .1 Introduction..37
VI .2 Steps of execution 37
VI.2.1 Data structure.. 38
VI.2.2 MOACO algorithm..39
VI.2.3 Dominance check algorithm.. 43
VI .3 Conclusion 44

General Conclusion 45

References 46

List of Figures

II.1 Decision space and objective space in MOP 9

II.2 Non dominated solutions in the objective space 10

II.3 Nadir and ideal points in a MOP...11

II.4 Supported and non supported solutions in a MOP. Weak dominance and strict dominance concepts.. 12

II.5 A permutation flow-shop scheduling problem.. 14

III.1 Schema of the Resolution Methods.. 18

IV.1 experimental setup for the double bridge experiment 24

IV.2 the results of the experiment 25

IV.3 A flow describe ACO process 28

IV.4 An ant arriving in city I choses the next city j to move..
....29................

VI.1 objective 1 of data TSP 38

VI.2 objective 2 of data TSP 38

VI.3 resources for MOACO psuedocode 39

VI.4 Prototypes of MOACO parameters 40

VI.5 Prototypes of MOACO process 40

VI.6 Prototypes of MOACO process 41

VI.7 Prototypes of MOACO process... 41

VI.8 Prototypes of MOACO process... 42

VI.9 Prototypes of MOACO process 42

VI.10 prototypes of end MOACO process and visualize solutions 42

VI.11 domiance check algorithm 43

VI.12 parto front in MOP.. 43

Glossary

ACO : Ant colony optimization

FSP : Flow shop problem

MOACO : Multi objective Ant colony optimzation

MOP : Multi objective optimization

MOTSP : Multi objective travelling salesmen problem

QAP : quadratic assignement problem

TSP : Travelling salesmen problem

VRP : vehicle routing problem

General Introduction

Operational research (OR) is a discipline that employs mathematical modeling, statistical analysis, and optimization techniques to assist decision makers in solving complex problems and optimizing outcomes. By utilizing mathematical optimization methods, such as linear programming, integer programming, or dynamic programming, OR helps organizations to find the best possible solutions of intricate problems. These optimization techniques play a crucial role in improving efficiency, resource allocation, and decision-making processes [1].

Optimization techniques find wide applications across various real-life domains. In transportation and logistics, optimization is used for route optimization, vehicle scheduling, and inventory management, leading to cost reduction and improved efficiency, In finance, optimization models aid in portfolio optimization, asset allocation, and risk management [2].

However, most of these optimization problems are combinatorial optimization problems, Combinatorial optimization primarily deals with finding the best solution among a finite set of discrete options. It involves selecting a subset of elements or arranging them in a specific order to optimize a particular objective, such as minimizing costs or maximizing efficiency. the TSP is one of the combinatorial problems which will be the topic of this thesis [3].

In This work. The TSP aims to find the shortest possible route that visits a set of cities and returns to the starting point. In practice, however, there may be additional objectives to consider, [such as minimizing travel time, reducing fuel consumption, or maximizing customer satisfaction]. Multi-objective optimization techniques can handle such scenarios, generating a set of trade-off solutions known as the Pareto frontier, which represents the best compromises between different objectives [4].

Solving the Combinatorial optimization problems, is a challenge due to their discrete nature and the exponentially large search space. Various methods have been developed to tackle these problems .One of the resolution methods in the Bio-inspiration methods which draws inspiration from biological systems to develop innovative solutions. By studying nature's mechanisms and processes, bio-inspired methods aim to solve complex engineering problems. These methods mimic biological structures, behaviors, and evolutionary processes to create efficient and adaptable designs. Examples include swarm intelligence algorithms inspired by ant colonies and genetic algorithms based on natural selection [5].

In order to solve the multi objective (TSP) we will use approximate method which is based on the optimization of ant colonies or (Ant Colony Optimization).

This research looks at how well multi objective ant colony optimization algorithm (MOACO) performs in the case of multi objective travelling salesmen problem (MOTSP).

The Muli objective ant Colony Optimization (MOACO) algorithm encourages exploration of different paths, promoting a diverse set of optimal solutions. This is particularly useful in multi-objective optimization problems where conflicting objectives exist. The MOACO algorithm helps in improving the quality and diversity of solutions obtained, making it valuable for solving complex optimization problems.

The ability of the MOACO algorithm is to detect the pareto front .

The purpose of this dissertation is **applying the ACO to solve the MOTSP .**

The organization of the work will be as follows :

Our work covers various aspects of combinatorial optimization problems. The first chapter introduces and explains these problems. Moving on to the second chapter, we explore multi-objective optimization concepts and their applications in academia and real life. In the third chapter, we discuss how to solve combinatorial optimization problems. The fourth chapter focuses on Ant Colony Optimization (ACO) and its use in solving the Traveling Salesman Problem (TSP). In the fifth chapter, we delve into the Multi-Objective Traveling Salesman Problem and explore variants of the Multi-Objective Ant Colony Optimization (MOACO) approach. We also outline the steps and process of the PACO algorithm. Finally, in the sixth and final chapter, we detail the execution steps of the algorithm and analyze the results. Our work concludes with a summary and general conclusion of our findings.

Combinatorial optimization

I. 1 Introduction

Combinatorial optimization theory has been a focus for researchers and practitioners. This field tackles problems with a large number of possible combinations, aiming to find the best solutions [3].

One of the remarkable aspects of combinatorial optimization problems is their transformational nature. Many real-world problems can be formulated as combinatorial optimization problems, even if they initially have a finite or countably infinite number of alternative solutions [3].

Lawler (1976) defines combinatorial optimization as the mathematical study of finding optimal arrangements, groupings, orderings, or selections of discrete objects. These problems are intriguing because they are easy to describe but notoriously challenging to solve[9].

In practical applications, many of these problems are considered NP-hard. This classification implies that it is widely believed they cannot be solved optimally within a reasonable amount of time using polynomial-based algorithms[14].

The examples of combinatorial optimization problems are diverse and impactful. They include determining the most cost-effective delivery plan, optimizing task assignments, designing efficient routing schemes, sequencing jobs in production lines, and allocating resources in various industries[5].

In this chapter we will discuss the combinatorial optimization in details.

I.2 Combinatorial optimization problem

Combinatorial optimization is a branch of optimization that focuses on solving problems involving discrete decision variables and a finite set of feasible solutions. It deals with finding the best configuration or arrangement of elements from a given set to optimize a certain objective, subject to various constraints. In combinatorial optimization, the emphasis is on exploring the combinatorial structure of the problem to efficiently search for optimal or near-optimal solutions [9].

Combinatorial optimization problems arise in various domains, including operations research, logistics, network design, scheduling, and resource allocation. Prominent examples include the traveling salesman problem, which seeks the shortest route visiting multiple cities, and the knapsack problem, which aims to determine the most valuable combination of items that fit within a limited capacity [9].

A combinatorial problem is a computational problem that involves the exploration and analysis of combinatorial structures or configurations to determine the optimal arrangement, selection, or assignment of discrete objects based on predefined criteria. These problems often deal with discrete entities, such as graphs, permutations, subsets, or combinations, and require finding the most favorable or optimal configuration [3]

I.3 Single-objective combinatorial optimization problem

The formulation of a single-objective combinatorial optimization problem involves defining the decision variables, the objective function, and the constraints. The goal is to find the optimal arrangement or combination of variables that minimizes or maximizes the objective function while satisfying the given constraints.

Decision Variables

These are the variables that determine the solution. They represent the choices or decisions to be made in the problem. For example, in the traveling salesman problem, the decision variables could be the order in which cities are visited [10].

Objective Function

The objective function quantifies the quality or value of a solution. It assigns a numerical value to each possible solution, indicating how well it satisfies the problem's objective. The objective can be either to minimize or maximize a certain quantity. For example, in the knapsack problem, the objective function could be the total value of the selected items [10].

Constraints

Constraints are conditions or limitations that the solution must adhere to. They define the feasible region of the problem, ensuring that the solution satisfies specific requirements. Constraints can include limitations on variables, resource availability, logical conditions, and more. For example, in a scheduling problem, constraints could restrict the assignment of tasks to specific time slots [10].

By formulating the decision variables, objective function, and constraints, the combinatorial optimization problem is defined, and various algorithms and techniques can be applied to search for the optimal solution [10].

The Mathematical formulation is :

$$\begin{cases} \min f(x), \\ \text{Such That}: \\ g_i(x) \le 0, i = 1, \dots, m \\ h_j(x) = 0, j = 1, \dots, p \\ x \in S \subset R^n \end{cases} \quad \text{(I.2)}$$

Where :

- f : is the function to minimize, called cost function or objective function;
- x : represents the vector of optimization variables;
- g_i : the inequality constraints;
- h_j : the equality constraints;
- S : the space of variables (also called search space). Note that the space of variables S indicates what type of variables, namely: real, integer, mixed (real and integer in the same problem), discrete, bounded, in the case of combinatorial optimization we deal with discrete variables

A point x_A is called an admissible point if $x_A \in S$ and if the optimization constraints are satisfied:

$g_i(x_A) \le 0, i = 1,\dots,m$ and $h_j(x_A) = 0, j = 1,\dots,p$

I.4 Multi-objective combinatorial optimization problem

"A multi-objective combinatorial optimization problem is a computational problem in which the objective is to find the optimal arrangement, selection, or assignment of discrete objects or elements from a given set to simultaneously optimize multiple objective functions. These objective functions represent different criteria or measures of performance, and the goal is to find a set of solutions that achieve a trade-off between these objectives, known as the Pareto front or Pareto set." [11]

Solving multi-objective combinatorial problems requires algorithms and techniques that can explore the solution space, identify a diverse set of non-dominated solutions, and provide decision-makers with a range of trade-off options

The Mathematical formulation is:

$$\begin{cases} \text{min or max or the both } f_1, f_2, \dots \dots, f_n \\ \text{Such That} \\ g_i(x) \le 0, i = 1, \dots, m \\ h_j(x) = 0, j = 1, \dots, p \\ x \in S \subset R^n \end{cases} \quad \text{(I.3)}$$

Solving a combinatorial optimization problem requires the study of three points:

1. The definition of all feasible solutions;

2. The expression of the objective to be optimized;

3. The choice of optimization method (exact or approximate) to use.

The first two points relate to the modeling of the problem and the third point to its resolution

I.5 Examples

– Job shop problem.
– Scheduling problems.
– Flow shop problem.
– Travelling salesmen problem.
– vehicule routing problem.

We will take the traveling salesman problem (TSP), which is an example of combinatorial optimization problems and we will look of variant of (TSP) problem which is the (TSP) in the case of multi objective.

I.6 Conclusion

In this chapter, we discussed the fundamental concept of optimization and optimization problem then we explained a type of optimization called combinatorial optimization then combinatorial optimization problem, we took a look to the classification of optimization problem and examples of combinatorial optimization problem, our focus is in the (TSP) or travelling salesmen problem, we will disscuss this problem solving but in the case of multi objective, the next chapter we will make view to the main concepts of multi objective optimization and how can we solve the multi objective travelling salesmen problem on the next chapters.

Multi Objective Optimization

II.1 Introduction

The challenge of solving large and complex optimization problems with multiple factors to consider is a common issue faced by various industries. In practical scenarios, these problems rarely involve just one objective, as there are often conflicting objectives that need to be balanced.

To address these challenges, the field of multi-objective optimization has emerged. It has a rich historical background, with roots dating back to the 19th century in economics . Initially applied in economics and management science, multi-objective optimization has gradually found its way into engineering sciences and logistics[5].

In today's world, multi-objective optimization has become a crucial field in science, engineering, and logistics. However, the complexity of these problems has increased significantly, with larger problem sizes, a greater number of objectives, and a wider search space to consider[5] .

To tackle these complex optimization problems effectively, extensive research has been conducted since the late 1980s, leading to the development of advanced techniques known as multi-objective metaheuristics. These techniques play a vital role in finding solutions within a reasonable timeframe for practical applications [5].

In this chapter, we will discuss the notions of multi objective optimization and the features of metaheuristics for solving multi objective problems .

II.2 Multi objective optimization concepts

This section covers the main concepts of multi objective optimization, such as dominance, Pareto optimality, Pareto optimal set, and Pareto front. In these definitions, the minimization of all the objectives is assumed, without loss of generality, this concepts is from **[5].**

Multi objective optimization problem

A multi objective optimization problem can be defined as

$$\mathbf{MOP} = \begin{cases} \boldsymbol{min\, F(x)} = (\boldsymbol{f_1(x), f_2(x), \ldots, f_n(x)}) \\ \boldsymbol{s.t. x \in S} \end{cases} \quad \text{(II.1)}$$

Where n (n ≥ 2) is the number of objectives, $x = (x_1 \ldots x_k)$ is the vector representing the decision variables, and S represents the set of feasible solutions associated with equality and inequality constraints and explicit bounds. $\boldsymbol{F(x) = (f_1(x), f_2(x), \ldots, f_n(x))}$ is the vector of objectives to be optimized.

The search space S represents the decision space or parameter space of the MOP. The space in which the objective vector belongs to is called the objective space. The vector F can be defined as a cost function from the decision space in the objective Space that evaluates the quality of each solution $\boldsymbol{(x_1 \ldots x_k)}$ by assigning an objective vector $\boldsymbol{(y_1 \ldots y_n)}$, which represents the quality of the solution (or fitness) (Fig. II.1).

In the field of multi objective optimization, the decision maker uses it to work in terms of evaluation of a solution on each criterion, and is naturally placed in the objective space.

The set $\boldsymbol{Y = F(S)}$ represents the feasible points in the objective space, and $\boldsymbol{y = F(x) = (y_1, y_2, \ldots, y_n)}$, Where $\boldsymbol{y_i = f_i(x)}$, is a point of the objective space.

In multi-objective optimization, the notion of an "optimal" solution is slightly different from traditional single-objective optimization. Instead of a single optimal solution, we have a set of solutions known as the Pareto optimal set.

The existence of optimal solutions in multi-objective optimization depends on the problem formulation, constraints, and the nature of the objectives. It is important to note that finding all Pareto optimal solutions can be a challenging task, and various algorithms and techniques are used to approximate the Pareto front and identify a diverse set of optimal solutions.

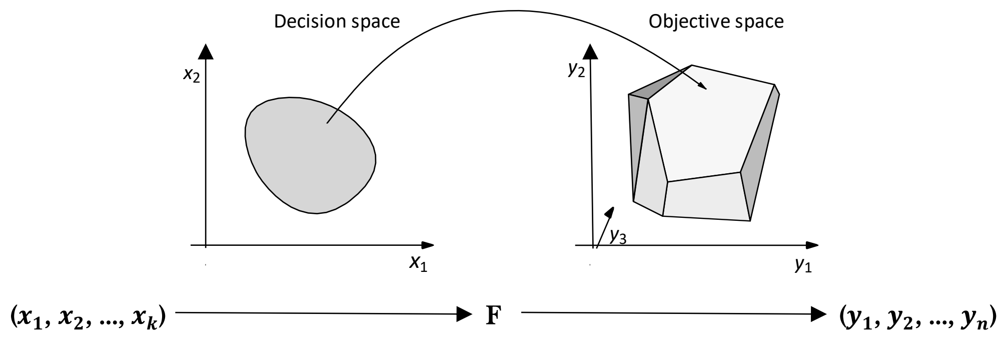


Figure II.1 Decision space and objective space in a MOP [5]

It is not usual to have a solution x^*, associated with a decision variable vector, where $x^*$is optimal for all the objectives:

$$\forall\ x \in S\ ,\quad f_i(x^*)\ \leq\ f_i(x)\quad ,\quad i = 1,2,........,n \tag{II.2}$$

Given that this situation is not usual in real-life MOPs where the criteria are in conflict, other concepts were established to consider optimality. A partial order relation could be defined, known as dominance relation. [5]

Pareto dominance

An objective vector u = $(u_1...u_n)$ is said to dominate v = $(v_1...v_n)$ (denoted by $u \prec v$) if and only if no component of v is smaller than the corresponding component of u and at least one component of u is strictly smaller, that is,

$$\forall\ i \in\{1,......,n\}:\ u_i \leq v_i\ \ \wedge\ \ \exists\ i \in\{1,...,n\}: u_i < v_i$$

The generally used concept is Pareto optimality. Pareto optimality definition comes directly from the dominance concept. The concept was proposed initially by F.Y. Edgeworth in 1881 and extended by W. Pareto in 1896. A Pareto optimal solution denotes that it is impossible to find a solution that improves the performances on a criterion without decreasing the quality of at least another criterion. [5]

Pareto optimality

A solution x^*∈ S is Pareto optimal if for every x ∈ S, $F(x)$ does not dominate $F(x^*)$, that is, $F(x) \nprec\ F(x^*)$.

Graphically, a solution x^* is Pareto optimal if there is no other solution x such that the point $F(x)$ is in the dominance cone of $F(x^*)$ that is the box defined by $F(x)$, with its projections on the axes and the origin (Fig. II.2). In general, searching in a monoobjective problem leads to find a unique global optimal solution. A MOP may have a set of solutions known as the Pareto optimal set. The image of this set in the objective space is denoted as the Pareto front. [5]

Pareto optimal set

For a given MOP (F,S), the Pareto optimal set is defined as
$$P^* = \{\ x \in S\ /\ \nexists\ x' \in S\ ,\ \ F(x') \prec F(x)\ \}.$$

Pareto front

For a given MOP (F,S) and its Pareto optimal set i P^*, the Pareto front is defined as $PF^* = \{ F(x) \ , \ x \ \in P^*\}$.

The Pareto front is the image of the Pareto optimal set in the objective space, obtaining the Pareto front of a MOP is the main goal of multi objective optimization.
However, given that a Pareto front can contain a large number of points, a good approximation of the Pareto front may contain a limited number of Pareto solutions, which should be as close as possible to the exact Pareto front, as well as they should be uniformly spread over the Pareto front. Otherwise, the obtained approximation of the Pareto front would not be very useful to the decision maker who should have a complete information on the Pareto front.

Let us notice that depending on the considered space (decision space or objective space), the number of Pareto solutions may be different. In the objective space, two solutions having the same objective vector will be considered as a single point, whereas they represent two different solutions in the decision space.

The Pareto optimal set in the objective space is called the minimal complete Pareto set, whereas it is the maximal complete Pareto set in the decision space. It is worth to point out that ideally one would like to obtain a solution minimizing all the objectives. Let us suppose that the optimum for each objective function is known, the objective functions being separately optimized

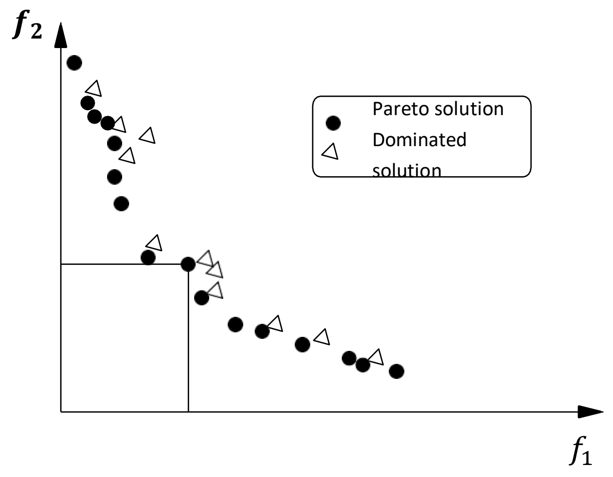


FIGURE II.2 Nondominated solutions in the objective space. [5]

Ideal vector

A point y^*= ($y_1{}^*$,$y_2{}^*$,.....,$y_n{}^*$) is an ideal vector if it minimizes each objective function f_i in $F(x)$, that is, $y_i{}^*$= min ($f_i(x)$), $x \in S$, $i \in [1,n]$.

The ideal vector is generally an utopian solution in the sense that it is not a feasible solution in the decision space. In certain cases, the decision maker defines a reference vector, expressing the goal to reach for each objective. This generalizes the concept of ideal vector. The decision maker may specify some aspiration levels $\overline{z_i}$, $i \in [1,n]$ to attain for each objective function f_i. Aspiration levels represent acceptable or desirable levels in the objective space. A Pareto optimal solution satisfying all aspiration levels is called a satisficing solution.

Reference point

A reference point z^*= [z_1,z_2,.....,z_n] is a vector that defines the aspiration level (or goal) z_i to reach for each objective f_i.

Nadir point

A point y^* = ($y_1{}^*$,$y_2{}^*$,......,$y_n{}^*$) is the nadir point if it maximizes each objective function f_i of F over the Pareto set, that is, $y_i{}^*$ = max ($f_i(x)$), $x \in P_i{}^*$, $i \in [\ 1, n\ ]$.
The ideal and nadir points give some information on the ranges of the Pareto optimal front (Fig. II.3)

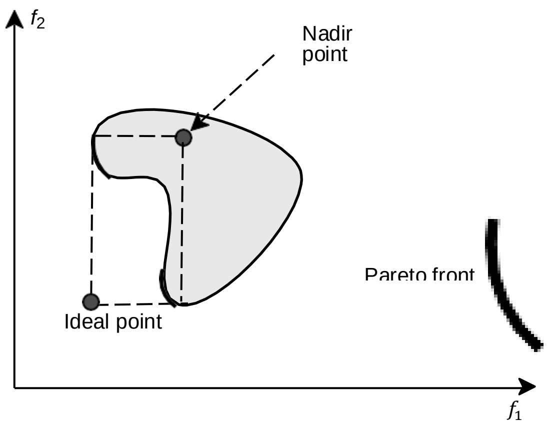


FIGURE II.3 Nadir and ideal points in a MOP [5].

In multi objective optimization, the concept of local minima can be generalized to the locally Pareto optimal solution. This notion is related to the concept of neighborhood, usually applied to S-metaheuristics.

Locally Pareto optimal solution

A solution x is locally Pareto optimal if and only if $\forall\ w \in\ N(x)$, $F(w)$ does not dominate $F(x)$, and $N(x)$ represents the neighborhood of the solution x.

Some Pareto optimal solutions may be obtained by the resolution of the following mathematical program:

$$\boldsymbol{MOP_\lambda}\begin{cases}\boldsymbol{min\ F(x) = \sum_{i=1}^{n} \lambda_i f_i(x)} \\ \boldsymbol{s.t. x \in S}\end{cases} \quad \text{(II.3)}$$

With $\lambda_i \geq 0$ for i=1,......,n, and $\sum_{i=1}^{n} \lambda_i = 1$

Such solutions are known as supported solutions. Supported solutions are generated by the resolution of (MOP_λ) for various values of the weight vector λ.

The complexity of (MOP_λ) is equivalent to the subjacent mono objective optimization problems.

If the subjacent optimization problems are polynomial, it will be relatively easy to generate the supported solutions. Nevertheless, there exists other Pareto optimal solutions that cannot be obtained by the resolution of a (MOP_λ) mathematical program. Indeed, these solutions, known as non-supported solutions, are dominated by convex combinations of supported solutions, that is, points of the convex hull of $Y = F(S)$

Other types of domination definitions exist, such as the concept of weak dominance And strict dominance.

Weak dominance

An objective vector $u = (u_1 ... u_n)$ is said to weakly dominate $v = (v_1 ... v_n)$ (Denoted by $u \leq v$) if all components of u are smaller than or equal to the corresponding components of v, that is, $\forall\ i \in \{ 1,...,n \}$, $u_i \leq v_i$

strict dominance

An objective vector $u = (u_1 ... u_n)$ is said to strictly dominate $v = (v_1 ... v_n)$ (Denoted by $u \prec\prec v$) if all components of u are smaller than the corresponding components of v, that is, $\forall\ i \in \{ 1,...,n \}$, $u_i < v_i$

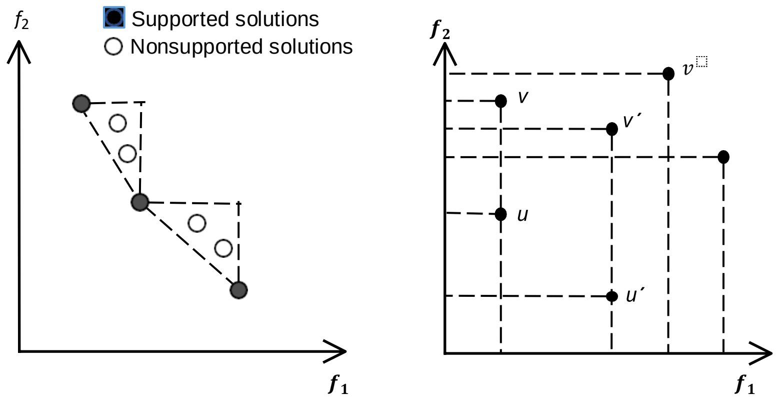


Figure II.4 : Supported and nonsupported solutions in a MOP. Weak dominance and strict dominance concepts. Solution u weakly dominate solution v; solution u′ weakly dominates solution v′; solution u strictly dominates solutions v′ and v′′. [5]

II.3 Multi objective optimization problems

Similar to mono-objective optimization, multi-objective optimization problems (MOPs) can be categorized into two main types based on the type of variables used to encode the solutions. These categories are continuous MOPs, where solutions are represented by real-valued variables, and combinatorial MOPs, where solutions are represented by discrete variables.

Over the last four decades, a significant focus in multi-objective optimization research has been on multi-objective continuous linear programming problems. This emphasis can be attributed to two main factors: the advancements made in mono-objective linear programming within the field of operations research and the relative ease of handling such problems.

Most of metaheuristics for solving MOPs are designed to deal with continuous MOPs. One of the reasons of this development is the availability of "standard" benchmarks for continuous MOPs. However, since the last decade, there is also a growing interest in solving combinatorial MOPs. Indeed, many real-life and well-known academic problems (e.g., TSP, QAP, VRP, knapsack, scheduling) have been modeled as combinatorial MOPs. In the following sections, some academic examples of MOPs as well as real-life ones are presented, we will take this application from [5]

II.3.1 Academic applications

The majority of the benchmarks used in the comparison of multiobjective metaheuristics were carried out on academic problems. Let us point out that, in many cases, continuous functions are used to perform the first experimentations of a new multiobjective metaheuristic , For combinatorial MOPs, there is a lack of "standard" benchmarks even if recently there is an interest in providing test instances for classical combinatorial MOPs.

II.3.1.1 Multi objective combinatorial problems

Classical combinatorial MOPs that can be classified into two complexity classes:

• **Polynomial problems**: Many multiobjective models of polynomially solvable optimization problems have been tackled: shortest path problems, spanning tree problems, assignment problems, and so on.
• **NP-hard problems**: Most of the multiobjective models concern NP-hard optimization problems: scheduling problems, routing problems, quadratic assignment problems, and so on

An increasing number of combinatorial multiobjective test functions are available in the literature. However, in most of the cases, they are biobjective optimization problems. Globally, there is also a lack of test functions for real-life combinatorial MOPs, especially problems that are subject to many objectives, uncertainty, and dynamicity.

Example Multi objective scheduling problems

The permutation flow-shop scheduling problem (FSP) is one of the most well-known scheduling problems. The problem can be presented as a set of n jobs J_1,J_2,……,J_n to be scheduled on m machines. Machines are critical resources: one machine cannot be assigned to two jobs

Simultaneously. Each job J_i is composed of m consecutive tasks t_{i1},……,t_{im}, where t_{ij} represents the j^{th} task of the job J_i requiring the machine M_j.

To each task t_{ij} is associated a processing time p_{ij}, and to each job J_i a release time r_i and a due date d_i(deadline of the job) are given.

In permutation flow-shop problems, the jobs must be scheduled in the same order on all the machines.

Many objectives may be used in scheduling tasks on different machines. These objectives vary according to the particularities of the treated problem:

C_{max}:	Make span (total completion time): max { C_i $\mid i \in$ [1…..n] }
C :	Mean value of jobs completion time
T_{max}:	Maximum tardiness: max {[max (0 , $C_i - d_i$)] $\mid i \in$ [1…..n] }
T :	Total tardiness: $\sum_{i=1}^{n}[max(0, C_i - d_i)]$
U :	Number of jobs delayed with regard to their due date d_i
F_{max} :	Maximum job flow-time: max {$C_i - r_i \mid i \in$ [1…..n]}
F :	Mean job flow-time

where s_{ij} represents the time at which the task t_{ij} is scheduled and C_i = s_{im} + p_{im} represents the completion time of job J_i .The size of the Pareto front is not very important as the correlation of the two objectives is positively important that restricts the number of Pareto solutions .

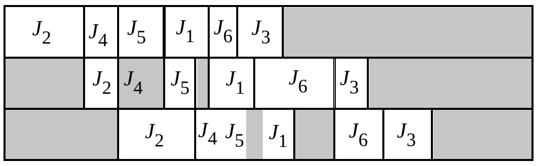


Figure II.5 A permutation flow-shop scheduling problem.[5]

II.3.2 Real-life applications

A huge number of works dealing with MOPs are dedicated to real-life applications .Two key aspects are responsible for this interest: many real-life applications involve various conflicting objectives, and efficient multiobjective metaheuristics have been developed (e.g., evolutionary multiobjective algorithms). Indeed, multiobjective metaheuristics have been applied to real-life applications since 1960. Moreover, several domains were dealt with various multiobjective applications:

- **Engineering design**

In the past 30 years, the development of multi objective metaheuristics has greatly influenced the design of systems in various engineering fields such as mechanics, aeronautics, and chemistry. This approach has proven highly valuable in tackling design challenges, including those related to airplane wings and car engines. The use of multi objective formulation in engineering design problems has been widely successful and well-received [5].

- **Environment and energetics**

In the research literature, there has been significant attention given to the multiobjective modeling of optimization problems related to environmental and energy domains. These include areas such as water distribution management and air quality management. As our environment faces increasing challenges and the availability of energy resources, such as water and nonrenewable fuels like petrol, becomes more limited, this field of study becomes increasingly crucial for addressing future concerns[5].

- **Telecommunications**

In the past decade, the field of telecommunications has emerged as an exciting domain for the application of multiobjective metaheuristics. Various areas within telecommunications, such as antenna design, cellular network design, satellite constellation design, and frequency assignment, have benefited from these techniques. The continuous evolution of network technologies, including sensor networks, ad hoc networks, and cognitive networks, ensures that this domain will continue to be dynamic and full of opportunities for further exploration and optimization[5].

- **Control**

Multiobjective modeling and optimization techniques are widely applied in the field of optimal control design. This field focuses on finding the best controller designs that achieve optimal performance. By considering multiple objectives simultaneously, these approaches help create controllers that effectively balance various performance criteria. The active research and development in this area highlight the significance of using multiobjective modeling and optimization methods in the design of optimal controllers [5].

- **Computational biology and bioinformatics**

Many important problems in computational biology and bioinformatics can be expressed as Multiobjective Optimization Problems (MOPs). The application of multiobjective optimization in this domain is still in its early stages, leaving ample room for further exploration. Various challenging tasks, including classification, feature selection, clustering, association rules, gene regulatory network modeling, phylogenetic inference, sequence and structure alignment, protein identification, protein structure prediction, and molecular docking, require multiobjective optimization approaches to address them effectively. There is significant potential for future research and advancements in this fascinating area [5].

- **Transportation and logistics**

Nowadays, this domain generates a large number of MOP applications (e.g. Containers management, design of grid systems, traced motorway)[5].

Example Multi objective routing problems

Routing problems such as the traveling salesman and the vehicle routing problems are widely studied because of their numerous real-life applications (e.g., logistics, transportation). The most common objectives include minimizing the total distance traveled, the total time required, the total tour cost, and/or the fleet size, and maximizing the quality of the service and/or the collected profit. Numerous other aspects such as balancing of workloads (e.g., time, distance) can be taken into account.

Our works focus on Multi objective travelling salesmen problem.

II.4 Conclusion

In this chapter, we discussed the fundamental concepts of multi objective optimization and then we took academic and real life applications of multi objective problems, in the next chapter we will take a knowledge about the metaheuristics and the resolution methods in general as our study focus on metaheuristics and an algorithm based on swarm intelligence which is the ant colony optimization .

Chapter III

Methods for combinatorial optimization problem

III.1 Introduction

As we discussed earlier, when dealing with combinatorial optimization problems, the main objective is to uncover the optimal solution, also known as the global optimal solution or global optimum. In simple terms, the goal is to find the very best outcome or result that outperforms all other alternatives. This involves thoroughly exploring and evaluating different possibilities in order to identify the solution that excels beyond any other choice [5].

Combinatorial problems in resolution are challenging to solve due to the fact that the number of viable solutions typically increases as the problem size and complexity grow. In simpler terms, as the problem becomes larger and more intricate, the available solution options become increasingly numerous, making it harder to find the best solution [14].

Hence, researchers have proposed resolution methods and invested considerable effort in enhancing their performance in terms of both computation time and the quality of the proposed solutions. In simpler terms, scientists have put forward approaches to solve these problems and dedicated their efforts to make them faster and produce better solutions [5].

Over the years, numerous methods have been proposed to address problems of varying complexity. These methods exhibit a wide range of approaches, each with its own distinct principles, strategies, and performance characteristics. In simpler terms, there is a rich variety of methods available for solving different problems, and they differ significantly in how they operate, the strategies they employ, and their overall effectiveness [14].

Resolution methods can be broadly categorized into two main types: exact (or complete) methods and approximate (or incomplete) methods. Exact methods ensure the completeness of the solution process, meaning they guarantee finding the optimal solution. On the other hand, approximate methods sacrifice completeness to achieve greater efficiency, prioritizing faster computation even if the solution obtained is not guaranteed to be optimal[5].

The combination of techniques from these two classes resulted in a new class known as hybrid methods, which not only achieve an optimal solution but also significantly reduce the computational time required. Another way to express this idea is that the hybridization of these methods brings together the best aspects of both classes, allowing for efficient and effective solutions to be obtained in a shorter period[5].

III.2 Resolution methods

The resolution methods are schematized as follows

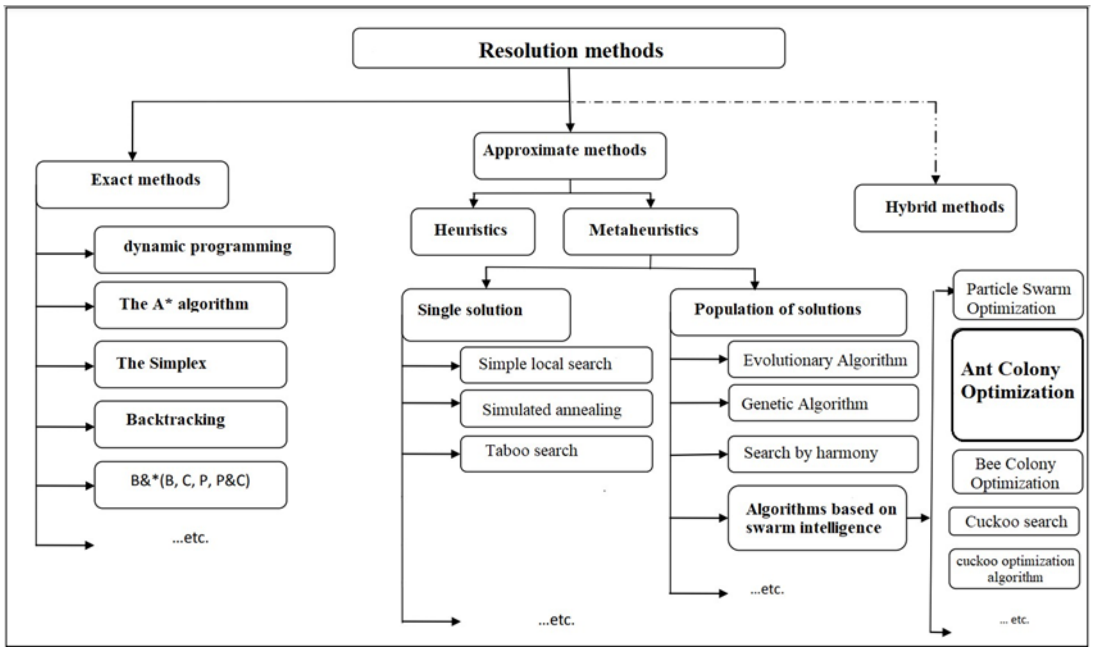


Figure III.1- Schema of the Resolution Methods [12]

III.2.1 The exact methods

Exact methods in computational algorithms aim to explore the entire search space of a problem to find the optimal solution. The search space refers to the set of all possible configurations or combinations that satisfy the problem constraints. The goal of exact methods is to systematically examine this search space and identify the best solution, ensuring optimality[13].

The primary advantage of exact methods is their ability to provide provably optimal solutions. By exhaustively exploring all possibilities or utilizing systematic techniques, these methods guarantee finding the best solution within the given constraints. This makes them particularly valuable when precision and optimality are crucial[13].

However, exact methods also have some disadvantages. One significant drawback is their computational complexity. As the search space grows exponentially with problem size, exploring all possibilities can become computationally infeasible or time-consuming, especially for large-scale problems. The running time of exact methods can quickly become prohibitive, making them less suitable for real-time or time-sensitive applications[13].

Some commonly used exact methods include brute force, dynamic programming, backtracking, branch and bound, and integer linear programming[13].

III.2.2 Approximate methods

Approximate methods for combinatorial optimization are employed when finding an exact optimal solution is computationally infeasible, time-consuming, or when facing large-scale or intractable problems. They are particularly useful in scenarios where time constraints exist, and near-optimal solutions are sufficient. These methods are applied to explore problem domains, conduct preliminary analysis, and handle complex optimization problems with limited resources. Approximate methods strike a balance between solution quality and computational efficiency, providing good-quality solutions within reasonable timeframes[5].

These approaches can be divided into two categories: specific heuristics and metaheuristics. Specific heuristics are designed for solving a particular problem or instance. metaheuristics, on the other hand, are general-purpose algorithms that can be used to tackle almost any optimization problem. They can be seen as higher-level strategies that guide the creation of specialized heuristics for solving specific optimization problems [5].

III.2.2.1 Heuristics

In simpler terms, even if we had incredibly powerful computational abilities, there are certain problems with algorithms that are too complex to be solved within a reasonable timeframe. In such cases, we have to resort to a experimentation approach to find a solution that comes as close to the optimal answer as possible. Since it is impractical to test all possible combinations, we have to make strategic decisions. These decisions, known as heuristics, are highly dependent on the specific situation at hand. The goal of using heuristics is to identify an acceptable approximate solution within a reasonable amount of time, rather than exhaustively testing all potential combinations. Heuristics are employed in algorithms that require analyzing a large number of cases in order to solve problems and make decisions. They help us reduce the overall complexity of the algorithm by prioritizing the situations that are most likely to yield the solution [13].

III.2.2.2 Metaheuristics

In alternative terms, metaheuristics are advanced techniques used to solve complex optimization problems that cannot be effectively addressed by traditional heuristics or standard optimization methods. A metaheuristic is an iterative procedure that guides a subordinate heuristic by intelligently combining various concepts to explore and exploit the search space. It also incorporates learning strategies to organize information and discover efficient solutions that closely approximate the optimal solution. Metaheuristics represent the culmination of extensive research and development in the field of combinatorial optimization[5].

In designing a metaheuristic, two contradictory criteria must be taken into account: exploration of the search space (diversification) and exploitation of the best solutions found (intensification).

The metaheuristics divided into two main categories Population-based search and single-solution based search[5].

III.2.2.2.1 Methods with a single-solution

While solving optimization problems, single-solution based metaheuristics improve a single solution. They could be viewed as “walks” through neighborhoods or search trajectories through the search space of the problem at hand . The walks (or trajectories) are performed by iterative procedures that move from the current solution to another one in the search space. S-metaheuristics show their efficiency in tackling various optimization problems in different domains [5].

III.2.2.2.2 Methods with a population of solutions

In alternative terms, population-based metaheuristics (P-metaheuristics) share common principles and can be seen as a continuous improvement process within a population of solutions. The process starts with the initialization of the population, followed by the generation of a new population of solutions. This new population is then integrated into the existing one using selection procedures. The search process continues until a specific condition, known as the stopping criterion, is satisfied. Several algorithms, such as evolutionary algorithms (EAs), scatter search (SS), estimation of distribution algorithms (EDAs), particle swarm optimization (PSO), Ant colony optimization (ACO), and artificial immune systems (AISs), fall into this category of metaheuristics [5].

III.2.2.2.3 Algorithms based on swarm intelligence

Swarm intelligence-based algorithms are a class of algorithms that draw inspiration from natural phenomena. These algorithms are specifically designed to emulate the collective behaviors observed in certain species when they solve problems, with the aim of developing powerful metaheuristics for solving various optimization problems[13].

The term "swarm" typically refers to a finite collection of particles or interacting entities. These swarms can encompass diverse groups such as flocks of birds, colonies of ants, colonies of bees, and even immune systems. In bird flocks, the particles are individual birds; in ant colonies, the particles are ants; in bee colonies, the particles are bees; and in the immune system, the particles are specialized cells responsible for recognition and protection[13].

By mimicking the social behavior of particles within these swarms, which exhibit remarkable self-organization capabilities, researchers have proposed several algorithms in recent decades.

Some notable examples include Particle Swarm Optimization (PSO), Artificial Immune Systems (AIS), Artificial Ant Colony Optimization, Artificial Bee Colony Optimization, Cuckoo Search, and Cuckoo Optimization Algorithm. These algorithms leverage the principles of swarm intelligence to address complex optimization problems effectively[13].

III.2.2.2.4 Ant colony optimization

Ant Colony Optimization (ACO) is a metaheuristic approach that employs a colony of artificial ants to find effective solutions for challenging discrete optimization problems. The fundamental aspect of ACO algorithms is cooperation, where computational resources are allocated to a group of relatively simple agents, known as artificial ants. These ants communicate indirectly through a mechanism called stigmergy, which involves indirect communication mediated by the environment itself [14].

The cooperative interaction among the ants leads to the emergence of good solutions to the problem at hand. ACO algorithms are capable of tackling both static and dynamic combinatorial optimization problems. Static problems refer to situations where the problem characteristics are defined once and remain constant throughout the problem-solving process. A classic example of a static problem is the Traveling Salesman Problem (TSP), where the city locations and their relative distances are predetermined and do not change during the problem-solving phase [14].

III.2.3 Hybrid methods

Hybrid methods are becoming popular because they consistently give the best results for many different optimization problems. These methods combine two or more search techniques to take advantage of their strengths and work together effectively.

They leverage the strengths of multiple algorithms to improve solution quality and search efficiency. By integrating algorithms with complementary abilities, such as exploration, exploitation, and constraint handling, hybrid methods provide a flexible and adaptable approach to tackle diverse problem types. They can enhance solution quality, speed up convergence, explore trade-offs between conflicting objectives, leverage problem-specific knowledge, and increase robustness against variations. Hybrid methods offer a powerful and versatile strategy for addressing challenging combinatorial optimization problems [5].

III.3 Conclusion

In this chapter, we explored different approaches to problem-solving, including exact methods, approximate methods, and hybrid methods. However, our primary focus was on approximate methods, particularly metaheuristics, as our work heavily relies on them. Specifically, we will delve into one specific algorithm based on swarm intelligence called ant colony optimization.

Chapter IV

Ant Colony Optimization

IV.1 Introduction

Ant colonies and other social insect societies are characterized by their impressive organizational abilities, despite the simplicity of their individual members. This structured social organization enables them to tackle complex tasks that surpass the capabilities of a single ant. Through cooperation and collective efforts, ants are able to accomplish remarkable feats that would be unattainable for them individually. [14]

The field of "ant algorithms" focuses on studying models inspired by the behavior of real ants and utilizes these models to create innovative algorithms for solving optimization and distributed control problems. By observing and understanding how ants behave in their natural environment, researchers draw inspiration to design new algorithms that can effectively tackle various optimization and control challenges. These ant-inspired algorithms offer fresh approaches and strategies for solving complex problems in fields such as operations research, swarm intelligence, and distributed systems. [14]

The main idea is that the self-organizing principles that allow real ants to work together can be used to coordinate groups of artificial agents to solve computational problems.

Different aspects of ant colony behavior have inspired various types of algorithms. By mimicking the cooperative and adaptable behaviors seen in ant colonies, researchers have created algorithms that help artificial agents collaborate effectively. [14]

These ant-inspired algorithms offer new ways to solve complex problems by coordinating the actions of multiple agents.

We will discuss ant colony optimization in this chapter with details.

IV.2 From real to artificial ants

IV.2.1 Ants' foraging behavior and optimization

Many species of ants possess only rudimentary visual sensory capabilities, indicating limited development in their visual perception abilities, Some types of ants have very weak eyesight or are completely blind, in the early stages of studying ants' behavior, researchers made a key discovery is that most of the communication among ants, both with each other and with their surroundings, relies on chemicals produced by the ants themselves.

These substances are known as pheromones, which distinguishes them from how communication takes place in humans and other higher species is their distinct method of communication. The trail pheromone plays a vital role in the social behavior of specific ant species. It's used by ants to mark pathways on the ground, such as the paths from food sources to the nest. By detecting these pheromone trails, foraging ants can effectively track the route to food that has been previously discovered by their fellow ants.

The fascinating behavior of ants, where they collectively lay and follow chemical trails left by their fellow ants, serves as the fundamental inspiration for Ant Colony Optimization (ACO) [14] .

IV.2.2 Double bridge experiments

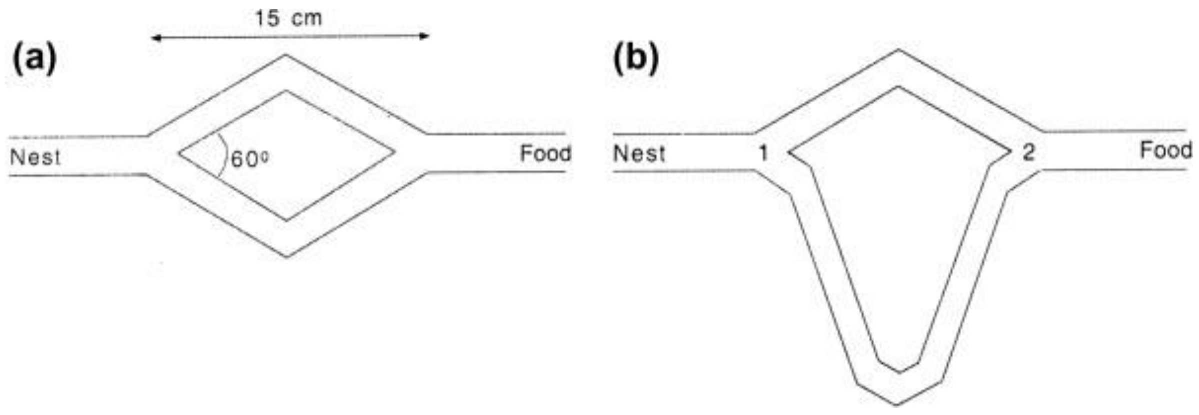


FIGURE IV.1 EXPERIMENTAL SETUP FOR THE DOUBLE BRIDGE EXPERIMENT. (A) BRANCHES HAVE EQUAL LENGTH. (B) BRANCHES HAVE DI¤ERENT LENGTH. MODIfiED FROM GOSS ET AL. (1989). [15]

The pheromone trail-laying and -following behavior of some ant species has been investigated in controlled experiments by several researchers. One particularly brilliant experiment was designed and run by Deneubourg and colleagues, who used a double bridge connecting a nest of ants of the Argentine ant species.

The results of this experiment showed that over time, the ants in experiment tends to choose the shortest path, which is characterized by a high concentration of pheromones. Meanwhile, some ants remain on the longer path, but in a small proportion.

In our case the convergence of the ants' paths to one branch represents the macroscopic collective behavior, which can be explained by the microscopic activity of the ants, that is, by the local interactions among the individuals of the colony. It is also an example of stigmergic communication, ants coordinate their activities, exploiting indirect communication mediated by modifications of the environment in which they move.

The stigmergic communication is the tool to mimic the real ants behaviors to artificial ones when they searching for optimal path to obtain food and back to the nest [14] .

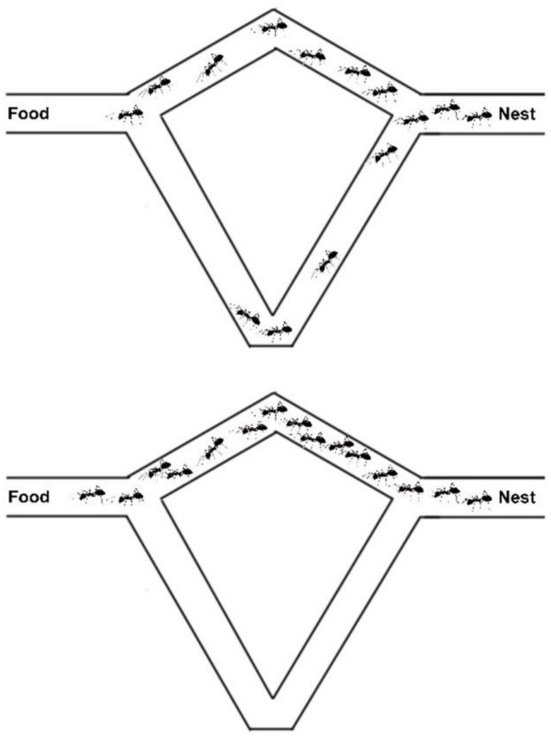


FIGURE IV.2 THE RESULTS OF THE EXPERIMENT [16]

IV.2.3 The comparison between artificial ants and real ants

Aspects	Real Ants	Artificial Ants in Aco
Nature	Living organisms	Algorithmic constructs
Physical Characteristics	Small size, segmented body, multiple legs	No physical presence, exist as virtual entities
Communication	Pheromone trails and direct physical interactions	Updating pheromone trails in the algorithm based on problem-specific rules
Adaptability	Can adapt to environmental changes and learn from experience	Adaptability is achieved through parameter tuning and algorithm design
Decision Making	Decisions are based on instinct and pheromone signals	Decisions are based on probabilistic rules and pheromone trails
Complexity	Complex biological systems	Simplified computational representations
Reproduction	Reproduce and multiply through mating and breeding	Reproduction is not applicable

Table IV.1 : Artificial Ants VS Real Ants[8]

IV.3 Ant colony optimization (ACO) algorithm

Ant Colony Optimization (ACO) is a metaheuristic algorithm inspired by the foraging behavior of ants. It is used to solve optimization problems, especially those related to graph theory and combinatorial optimization. ACO mimics the behavior of ants as they search for the shortest path between their nest and food sources.

The basic idea behind ACO is to simulate the way ants communicate through the deposition and following of pheromone trails. Ants leave pheromone trails on the paths they explore, and these trails attract other ants to follow the paths with stronger pheromone concentrations. In the context of optimization, the pheromone trails represent the accumulated knowledge of the colony about good solutions to the problem.

The ACO algorithm starts by initializing a population of artificial ants. Each ant traverses the problem space by probabilistically selecting the next step based on the pheromone levels and heuristic information. The heuristic information guides the ants to explore promising regions of the problem space. As ants move, they deposit pheromone along their paths, and the pheromone evaporates over time to prevent the system from getting trapped in suboptimal solutions.

The pheromone update rule is a crucial aspect of ACO. When an ant completes its tour or finds a good solution, it updates the pheromone levels on the visited edges based on the quality of the solution. Better solutions lead to higher pheromone concentrations, reinforcing the paths that contribute to those solutions. The pheromone levels are also subject to evaporation, which allows the algorithm to explore new paths and prevents stagnation.

ACO has been successfully applied to various optimization problems, including the traveling salesman problem, vehicle routing problem, scheduling problems, and many more. It has shown its effectiveness in finding high-quality solutions and has become a popular choice in the field of optimization. [14]

Algorithm 1 The ACO procedure [14]

```
1: Initialize parameters
2: For $t$=1 to iteration number do
3:   For $k$=1 to $l$ do
4:     Reapeat until ant k has completed a tour
5:       Select the city j to be visited next
6:       with probability $p_{ij}$ given by (IV.1)
7:     Calculate $L_k$
8:   Update the trail levels according to (IV.3)-(IV.4)
9: End
```

IV.3.1 Steps of ACO algorithm

1. **Initialization:**

Set the parameters such as the number of ants, the number of iterations, pheromone evaporation rate, alpha value, beta value,ect

Initialize the pheromone levels on edges of the problem graph.

2. **Ant Movement:**

Each ant starts from a random initial node.

At each step, an ant probabilistically selects the next node to move to based on a combination of pheromone levels and heuristic information.

The selection is typically guided by a pheromone trail update rule, such as the probability of choosing an edge being proportional to its pheromone level.

3. **Pheromone Update:**

After all ants complete their tours, the pheromone levels on the edges are updated.

Evaporate the existing pheromone trails by applying an evaporation rate to reduce their intensity.

Apply pheromone deposit rule to reinforce the pheromone levels of edges visited by the ants, typically based on the quality of the solution found.

4. **Iteration:**

Repeat steps 2 and 3 for a specified number of iterations or until a termination condition is met.

5. **Termination:**

Terminate the algorithm based on a predefined stopping criterion, such as a maximum number of iterations or reaching a satisfactory solution.

6. **Output:**

After the algorithm terminates, the best solution found by any ant throughout the iterations is returned as the final solution.

The Flow above describe Aco process [17]

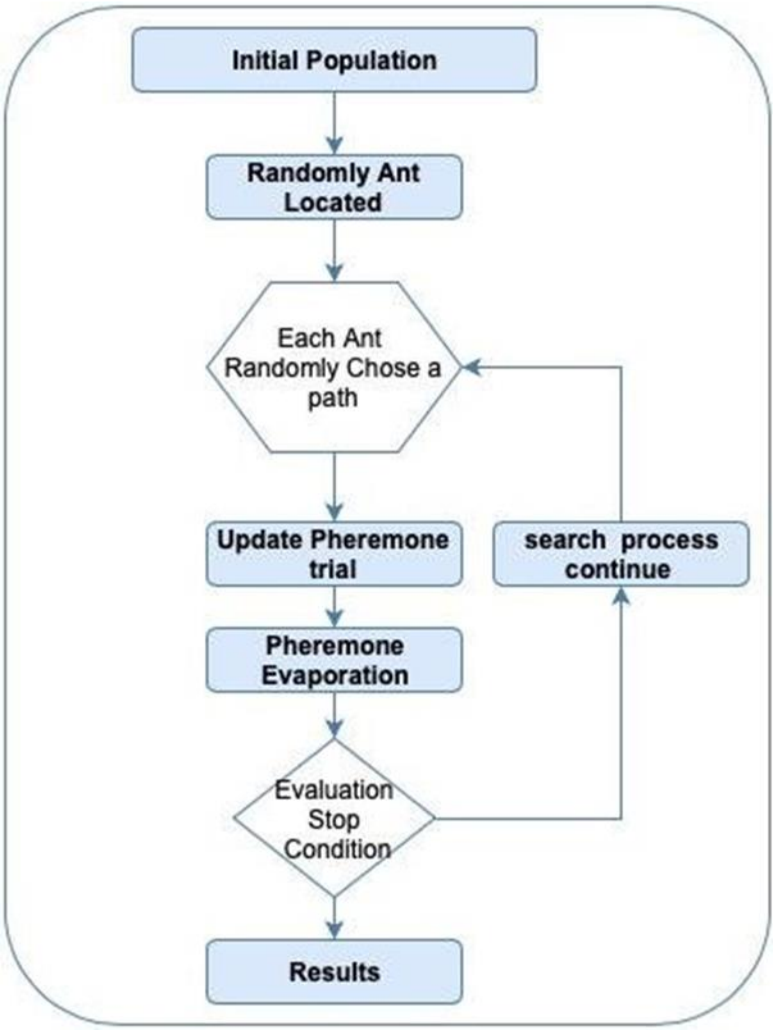


FIGURE IV.3: A FLOW DESCRIBE ACO PROCESS [17]

IV.3.2 Ant colony optimization algorithms for the traveling salesman problem

IV.3.2.1 The significance of the traveling salesman problem (TSP)

The traveling salesman problem (TSP) has been extensively studied in the research community and has received significant attention over the years. It has played a crucial role in the advancement of ant colony optimization (ACO) research. The initial ACO algorithm, known as Ant System, as well as subsequent ACO algorithms, were first tested and evaluated using the TSP.

There are several reasons why the TSP is often chosen as the problem to demonstrate the effectiveness of ACO algorithms. Firstly, the TSP is a well-known and important optimization problem that is classified as NP-hard and is encountered in various practical applications. Secondly, ACO algorithms can be readily applied to the TSP, making it a suitable problem for showcasing their functionality.

Moreover, the TSP is easily understandable, allowing researchers to analyze the algorithm's behavior without being hindered by excessive technical complexities. Additionally, the TSP serves as a standard benchmark for evaluating new algorithmic ideas. Achieving high performance on the TSP is often considered as evidence of the usefulness and effectiveness of novel approaches.

Furthermore, the history of ACO research has shown that the most efficient ACO algorithms for the TSP have also demonstrated exceptional performance for a wide range of other optimization problems. This further highlights the significance of the TSP as a test bed for evaluating and comparing the efficiency and effectiveness of ACO algorithms.

In simpler terms, the TSP is a well-known problem that has received extensive research attention. It serves as an important test case for ACO algorithms due to its significance, suitability for ACO application, understandability, and benchmarking capabilities. Success in solving the TSP often indicates the usefulness and efficiency of ACO algorithms, and the best-performing ACO algorithms for the TSP have shown effectiveness for various other problems as well**,** To enhance the understanding of the ACO algorithm, we will apply it to the Traveling Salesman Problem (TSP) [14] .

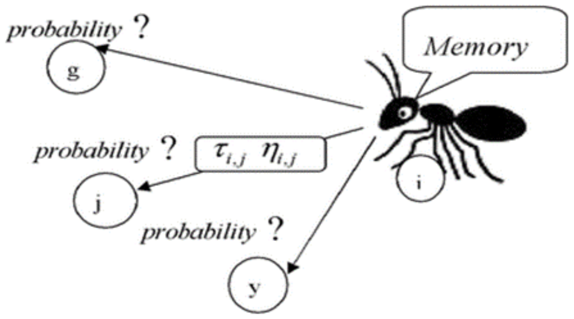


FIGURE IV.4: AN ANT ARRIVING IN CITY I CHOOSES THE NEXT CITY J TO MOVE TO AS A FUNCTION OF THE PHEROMONE VALUES τ_{ij} AND OF THE HEURISTIC VALUESη_{ij} ON THE ARCS CONNECTING CITY I TO THE CITIES J THE ANT HAS NOT VISITED YET. [14]

IV.3.2.2 Tour construction

In the ant system (AS) algorithm, a group of artificial ants simultaneously constructs a tour for the traveling salesman problem (TSP). To begin, the ants are placed on randomly selected cities. During each construction step, an ant (let's call it ant "k") utilizes a probabilistic action choice rule known as the random proportional rule to determine its next city visit.

The random proportional rule involves assigning probabilities to different cities based on certain criteria. Specifically, when ant "k" is currently located at city "i," it calculates the probability of choosing city "j" as its next destination. This probability is determined by a combination of factors, such as the pheromone trail intensity on the path from city "i" to city "j" and the desirability of visiting city "j" based on some heuristic information.

Essentially, the random proportional rule allows ant "k" to make a decision on which city to visit next by considering both the attractiveness of the city based on the pheromone trail and the heuristic knowledge. The higher the probability assigned to a city, the more likely it is for the ant to choose that city as its next stop.

In simpler terms, in the AS algorithm, each ant uses a rule to decide which city to visit next.

This rule takes into account the pheromone trail intensity and some heuristic information to calculate the probability of choosing a particular city. The ant then selects its next city based on these probabilities, guiding the construction of a tour for the TSP.

This process is based on this equation above

$$\boldsymbol{P_{ij}}^{\boldsymbol{k}} = \frac{[\tau_{ij}]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in N_l{}^k} [\tau_{il}]^{\alpha} * [\eta_{il}]^{\beta}}, \text{ if } \boldsymbol{j} \in \boldsymbol{N_l}^{\boldsymbol{k}} \tag{IV.1}$$

In ACO, the probability of an ant choosing a particular node as the next node to visit is calculated relative to the other available nodes. The numerator of the probability formula, $[\boldsymbol{\tau_{ij}}]^{\boldsymbol{\alpha}} * [\boldsymbol{\eta_{ij}}]^{\boldsymbol{\beta}}$, represents the contribution of the specific edge (i, j) to the overall probability. However, to ensure that the probabilities sum up to 1, the contribution of each edge is divided by the sum of the contributions of all available edges.

The denominator in the probability formula, $\sum_{l \in N_l{}^k} [\boldsymbol{\tau_{il}}]^{\boldsymbol{\alpha}} * [\boldsymbol{\eta_{il}}]^{\boldsymbol{\beta}}$, represents the sum of the probabilities for all possible moves from node i. This summation is necessary to ensure that the probabilities of all potential choices from node i sum up to 1, forming a valid probability distribution.

where $\eta_{ij} = 1 / d_{ij}$ is a heuristic value that is available a priori, α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information, and $N_i{}^k$ is the feasible neighborhood of ant k when being at city i, that is, the set of cities that ant k has not visited yet (the probability of choosing a city outside $N_i{}^k$ is 0). By this probabilistic rule, the probability of choosing a particular arc (i, j) increases with the value of the associated pheromone trail τ_{ij} and of the heuristic information value η_{ij}. The role of the parameters α and β is the following. If α = 0, the closest cities are more likely to be selected: this corresponds to a classic stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed over the cities). If β = 0, only pheromone amplification is at work, that is, only pheromone is used, without any heuristic bias. This generally leads to rather poor results and, in particular, for values of α > 1 it leads to the rapid emergence of a stagnation situation[14] .

IV.3.2.3 Update of pheromone trails

After all the ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone value on all arcs by a constant factor, and then adding pheromone on the arcs the ants have crossed in their tours. Pheromone evaporation is implemented by

$$\boldsymbol{\tau_{ij}} \leftarrow (\boldsymbol{1} - \boldsymbol{\rho})\, \boldsymbol{\tau_{ij}}, \forall\, (\boldsymbol{i}, \boldsymbol{j}) \in \boldsymbol{L} \tag{IV.2}$$

Where $0 < \rho \leq 1$ is the pheromone evaporation rate. The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to "forget" bad decisions previously taken. In fact, if an arc is not chosen by the ants, its associated pheromone value decreases exponentially in the number of iterations. After evaporation, all ants deposit pheromone on the arcs they have crossed in their tour:

$$\boldsymbol{\tau_{ij} \leftarrow \tau_{ij} + \sum_1^m \Delta_{ij}{}^k \quad , \forall\, (i,j) \in L} \tag{IV.3}$$

Where $\Delta_{ij}{}^k$ is the amount of pheromone ant k deposits on the arcs it has visited. It is defined as follows:

$$\Delta_{ij}{}^k = \begin{cases} \mathbf{1/C^k}, \textbf{if arc (i, j) belongs To } \mathbf{T^k}; \\ \qquad \mathbf{0}, \textbf{ other wise} \end{cases} \tag{IV.4}$$

Where C^k, the length of the tour T^k built by the $k-th$ ant, is computed as the sum of the lengths of the arcs belonging to T^k. The better an ant's tour is, the more pheromone the arcs belonging to this tour receive. In general, arcs that are used by many ants and which are part of short tours, receive more pheromone and are therefore more likely to be chosen by ants in future iterations of the algorithm. As we said, the relative performance of AS when compared to other metaheuristics tends to decrease dramatically as the size of the test-instance increases. Therefore, a substantial amount of research on ACO has focused on how to improve AS [14].

IV.4 Conclusion

In this chapter, our focus encompassed the broader concept of the ant colony and its significance. We then delved into a comprehensive exploration of the ACO algorithm, providing detailed insights and illustrating its application to the traveling salesman problem (TSP).

Ant Colony Optimization Algorithms for Multi Objective Travelling Salesmen Problem

V.1 Introduction

Multi objective combinatorial optimization have been solved by many algorithms especially ant inspired algorithms, in the case of multi objective a lot of ant inspired algorithms have been proposed to solve multi objective Combinatorial optimization problems especially to multi objective travelling Salesmen problem , there exist many variants to solve MOTSP, the decision maker have to choose one of them according to their interests and results.

V.2 Multi objective travelling salesmen problem

"The multi-objective traveling salesperson problem (MOTSP) is a combinatorial optimization problem that involves finding a set of Pareto-optimal solutions for a traveling salesperson to visit a set of cities, such that multiple conflicting objectives, such as minimizing total distance, minimizing travel time, or maximizing visitation of certain cities, are simultaneously optimized." [20]

The mathematical formulation of MOTSP can be expressed as follows:

Where:

- N be the set of cities, $|N| = n$.
- d_{ij} be the distance between cities i and j.
- x_{ij} be a binary decision variable that takes value 1 if the path includes the edge (i, j), and 0 otherwise.
- $f_k(x)$ be the k-th objective function to be minimized, representing a specific aspect of the problem (e.g., total distance, travel time, cost).

The MOTSP can then be formulated as a multi-objective optimization problem:

Cost function :

Multi-Objective TSP with Distance and Time:

Minimize : $f(x) = [f_1(x), f_2(x)]$

Minimize: $f_1 = \sum(distance(i,j) * x_{ij}) \ (for \ all \ i,j)$

Minimize: $f_2 = \sum(time(i,j) * x_{ij}) \ (for \ all \ i,j)$

Subject to:

Each city must be visited exactly once:

$\sum_{i=1}^{n} x_{ij} = 1$, for $i, j \in N$ and $i \neq j$

$\sum_{j=1}^{n} x_{ij} = 1$, for $i, j \in N$ and $i \neq j$

No subcycles are allowed (eliminating subtours):

$\sum_{\{i \in S, j \in S\}} x_{ij} \leq |S| - 1$, for all non-empty subsets $S \subset N, |S| \geq 2$

Binary decision variable constraints:

$x_{ij} \in \{0, 1\}$, for all $i, j \in N, i \neq j$

V.3 Multi objective ant colony optimization

V.3.1 variants

ACO for multi-objective optimization has various variants. In the Single Colony and Single Pheromone Matrix approach, a single colony of ants explores the solution space, and a single pheromone matrix is updated based on objective values. The Single Colony and Multiple Pheromone Matrices variant uses multiple pheromone matrices to consider multiple objectives simultaneously. In the Multi-Colony and Single Pheromone Matrix approach, multiple colonies explore independently with a shared pheromone matrix. The Multi-Colony and Multiple Pheromone Matrices variant combines multiple colonies with individual pheromone matrices for decentralized exploration. These variants offer different ways to tackle multi-objective optimization problems, balancing exploration and exploitation[23] [26] [27].

V.3.2 Single colony and single pheromone matrix

In the Single Colony and Single Pheromone Matrix variant of Ant Colony Optimization (ACO) for multi-objective optimization, a single colony of ants explores the solution space, and a single pheromone matrix is used to update the solution information. This variant is based on the principle that ants communicate through pheromone trails, which represent the quality of the solutions they find.

During the exploration process, the ants move through the solution space, building and updating the pheromone trails based on the objective values of the solutions they encounter. The pheromone values on the trails guide the subsequent ants in their search for better solutions. Through iteration, the pheromone matrix is updated to reflect the knowledge accumulated by the ants and guide the search towards promising areas of the solution space.

The Single Colony and Single Pheromone Matrix approach offers a straightforward implementation and can effectively explore the solution space. However, it may face challenges when dealing with complex multi-objective problems that require balancing conflicting objectives. The ants may struggle to converge on a diverse set of Pareto-optimal solutions due to the lack of explicit consideration for multiple objectives [23] [25].

V.3.3 Single colony and multiple pheromone matrix

In the Single Colony and Multiple Pheromone Matrices variant of Ant Colony Optimization (ACO) for multi-objective optimization, a single colony of ants explores the solution space, but multiple pheromone matrices are utilized. Each pheromone matrix corresponds to a specific objective, allowing the ants to independently update the pheromone information for each objective.

The ants evaluate the quality of solutions based on multiple objectives and deposit pheromone on the respective matrices accordingly. This approach enables the ants to consider the trade-offs between different objectives and promotes a more comprehensive search of the solution space. The multiple pheromone matrices provide a mechanism to guide the search towards a diverse set of Pareto-optimal solutions.

By using multiple pheromone matrices, the Single Colony and Multiple Pheromone Matrices approach enhances the exploration capabilities of the algorithm and helps in obtaining a well-distributed set of solutions along the Pareto front [24].

V.3.4 Multi colony and single pheromone matrix

In the Multi-Colony and Single Pheromone Matrix variant of Ant Colony Optimization (ACO) for multi-objective optimization, multiple colonies of ants are employed, but only a single pheromone matrix is used to update the solution information.

Each colony operates independently, with its own set of ants exploring the solution space. The ants within each colony build and update pheromone trails based on the objective values of the solutions they encounter. However, the pheromone information is shared among all colonies through periodic information exchange. This exchange allows the colonies to influence each other's exploration, promoting exploration of different regions of the solution space and facilitating knowledge sharing.

By employing multiple colonies, the algorithm benefits from increased diversity and parallel exploration capabilities. Different colonies may converge on different regions of the Pareto front, leading to a more comprehensive search and a wider coverage of the optimal trade-off solutions.

Despite using a single pheromone matrix, the Multi-Colony and Single Pheromone Matrix approach effectively combines the exploration power of multiple colonies, making it suitable for solving multi-objective optimization problems [26].

V.3.5 Multi colony and multi pheromone matrix

In the Multi-Colony and Multiple Pheromone Matrices variant of Ant Colony Optimization (ACO) for multi-objective optimization, multiple colonies of ants are utilized, and each colony has its own set of pheromone matrices.

Each colony operates independently, exploring the solution space and updating its own set of pheromone matrices. The pheromone values are updated based on the quality of solutions encountered by the ants within each colony. Information exchange occurs between colonies periodically, allowing them to share knowledge and coordinate their exploration efforts.

With multiple pheromone matrices, each representing a specific objective, the ants can independently update the pheromone information for each objective. This decentralized exploration allows for a more comprehensive search of the solution space and facilitates the exploration of diverse regions of the Pareto front.

The Multi-Colony and Multiple Pheromone Matrices approach promotes both exploration and exploitation by leveraging the diversity of multiple colonies and the ability to consider multiple objectives simultaneously. This variant offers an effective way to tackle multi-objective optimization problems and can lead to a well-distributed set of Pareto-optimal solutions.

Among the previous variants, we have chosen PACO (Population based Ant Colony Optimization) for further elaboration [27].

V.4 PACO for multiple objective optimization

V.4.1 Algorithm of PACO

When applied to multi-objective problems PACO maintains a different pheromone matrix for each objective. For each iteration of the algorithm, where iteration refers to every artificial ant creating a complete solution, a random ant is selected from the population (Q) along with its k closest neighbors to form a sub-population P [18].

At any time Q will contain the complete set of non-dominated solutions found to date. The ants in P are then used to update the individual pheromone matrix for each objective. When available a separate heuristic matrix is used for each objective; in the case of the TSP these heuristic matrices are simply the corresponding edge weights for each individually defined TSP as is the case for most ACO algorithms applied to the TSP [18].

PACO uses an average-rank-weight method to weight the importance of each objective. These weightings (w) are used to bias the solution construction towards satisfying specific objectives over others. Briefly, the average-rank-weight method measures how well each solution in P satisfies each individual objective. Objectives which are better satisfied by the solutions in P relative to the entire population Q are given a higher rank and a subsequently larger weighting [18].

Once the pheromone matrices have been created and the objective weightings defined the transition probabilities are calculated using (1), where h is the total number of objectives. The Ant Colony Systems greedy transition rule is then used to create one or more new solutions [18].

$$p_{ij} = \sum_{d=1}^{h}\left(w_d \cdot \frac{[\tau_{ij}{}^{d}]^{\alpha} . [\eta_{ij}{}^{d}]^{\beta}}{\sum_{l \in N_i{}^k}[\tau_{ij}{}^{d}]^{\alpha} . [\eta_{ij}{}^{d}]^{\beta}}\right) \qquad (V.2)$$

Where :

- h : is the total number of objectives;
- P_{ij} : is the probability of moving from i to j;
- w_d : is the weight of d (d=1 up to d=h);
- ${\tau_{ij}}^{d}$: is the pheromone trace between i and j;
- α : is the pheromone coefficient;
- β : is the heuristic information coefficient.

The steps :

1. Each new solution (s) is evaluated for each objective;
2. It must be tested for dominance against the full population Q before s can be introduced into Q;
3. If all members of Q agree that s is non-dominated, that is, no solution in Q is better in all objectives than s;
4. Then s is placed into Q. If s is placed into Q, s' dominance over Q must be confirmed;
5. The existing solutions in Q are deleted if they are dominated by s.

V.5 Conclusion

In this chapter we discussed the multi objective travelling salesmen problem and the variants of multi objective ant colony optimization, then we took every variant in detail, finally we took a choice of PACO algorithm and explained it .

Chapter VI
Implementation

VI.1 Introduction

In this chapter, we will apply the MOACO algorithm to solve the multi-objective travelling salesmen problem (MOTSP) , we will apply this algorithm by many phases , first we will take a data of bi objective TSP and pass it to the **MOACO** algorithm which give us the solutions of multi objective TSP then we pass it into dominance check algorithm to extract the non-dominated, to apply the MOACO algorithm we use the language MATLAB And C to implements the results.

VI.2 Steps of execution

Unfortunately because of errors in compilation we didn't get the results, in this section we will take a knowledge about approximate results and what we must get in and after compilation.

The diagram of execution steps:

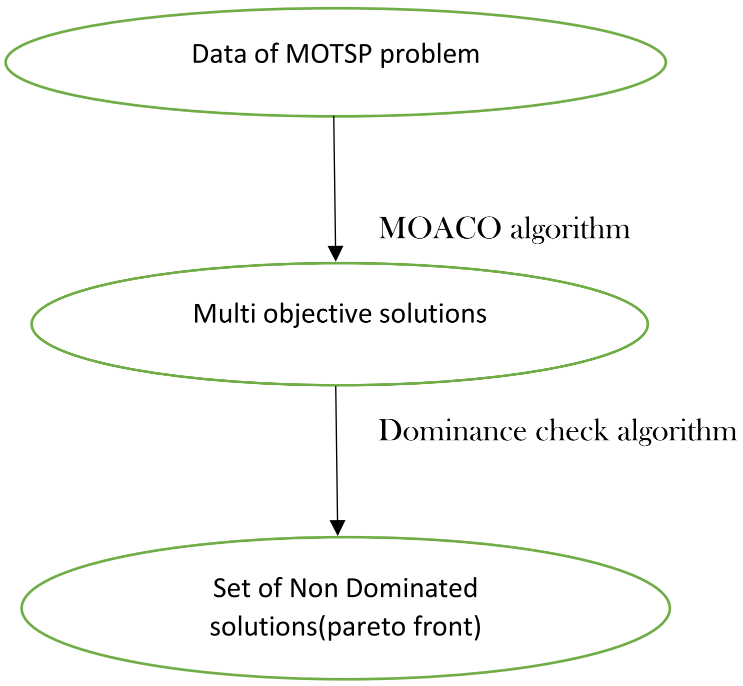
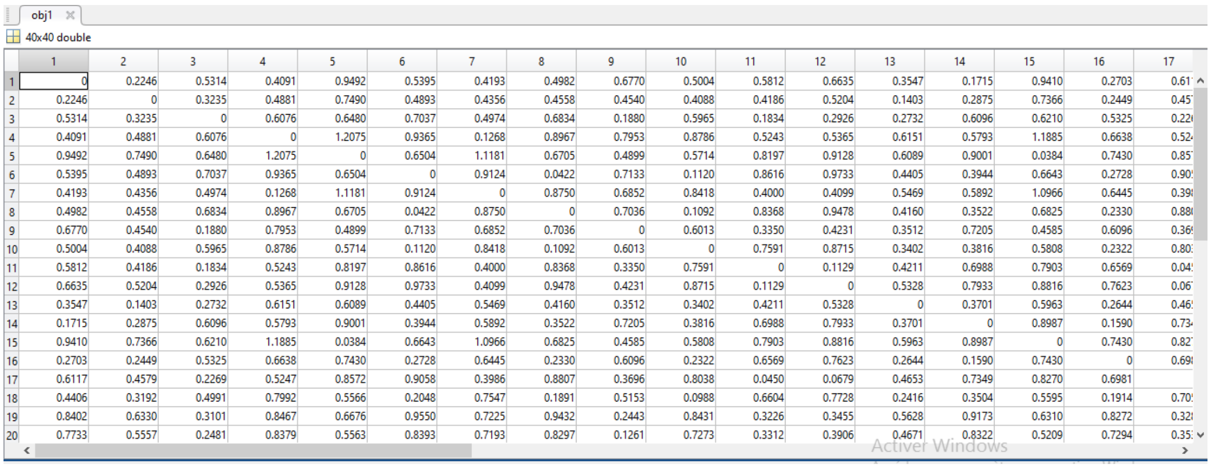

Data of MOTSP problem
MOACO algorithm
Multi objective solutions
Dominance check algorithm
Set of Non Dominated solutions(pareto front)

Diagramm of exucution steps

VI.2.1 Data structure

In the travelling salesmen problem our data is visualize as a square matrix, the matrix contain the distances between each pair of cities, in the case of symmetric TSP we will get symmetric matrix our work focus on the symmetric TSP.

Data is represented by two matrices because we solve a bi objective TSP.

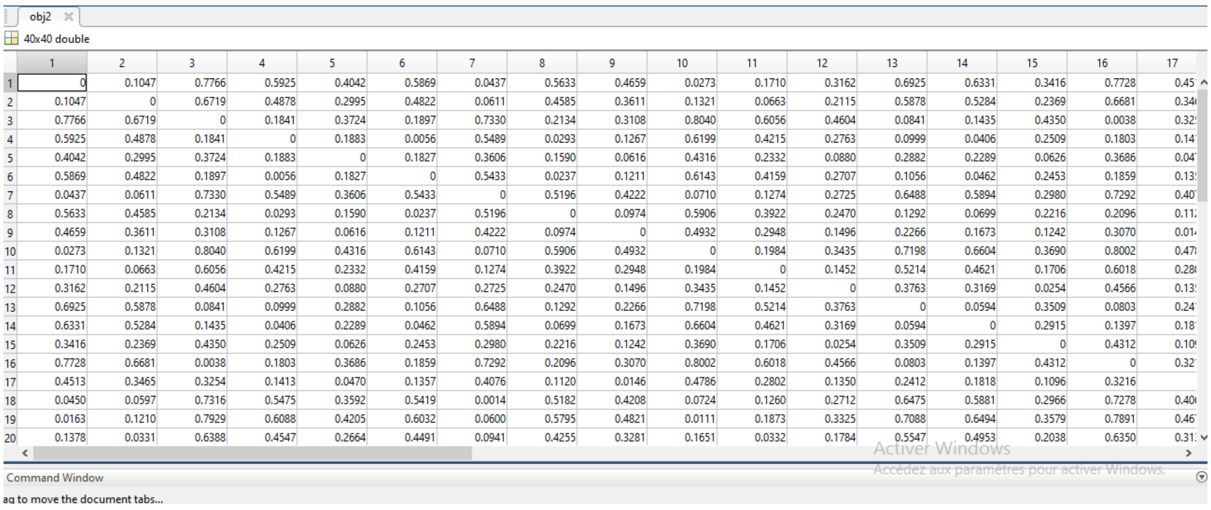


obj1

40x40 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0.2246	0.5314	0.4091	0.9492	0.5395	0.4193	0.4982	0.6770	0.5004	0.5812	0.6635	0.3547	0.1715	0.9410	0.2703	0.61
2	0.2246	0	0.3235	0.4881	0.7490	0.4893	0.4356	0.4558	0.4540	0.4088	0.4186	0.5204	0.1403	0.2875	0.7366	0.2449	0.45
3	0.5314	0.3235	0	0.6076	0.6480	0.7037	0.4974	0.6834	0.1880	0.5965	0.1834	0.2926	0.2732	0.6096	0.6210	0.5325	0.22
4	0.4091	0.4881	0.6076	0	1.2075	0.9365	0.1268	0.8967	0.7953	0.8786	0.5243	0.5365	0.6151	0.5793	1.1885	0.6638	0.52
5	0.9492	0.7490	0.6480	1.2075	0	0.6504	1.1181	0.6705	0.4899	0.5714	0.8197	0.9128	0.6089	0.9001	0.0384	0.7430	0.85
6	0.5395	0.4893	0.7037	0.9365	0.6504	0	0.9124	0.0422	0.7133	0.1120	0.8616	0.9733	0.4405	0.3944	0.6643	0.2728	0.90
7	0.4193	0.4356	0.4974	0.1268	1.1181	0.9124	0	0.8750	0.6852	0.8418	0.4000	0.4099	0.5469	0.5892	1.0966	0.6445	0.39
8	0.4982	0.4558	0.6834	0.8967	0.6705	0.0422	0.8750	0	0.7036	0.1092	0.8368	0.9478	0.4160	0.3522	0.6825	0.2330	0.88
9	0.6770	0.4540	0.1880	0.7953	0.4899	0.7133	0.6852	0.7036	0	0.6013	0.3350	0.4231	0.3512	0.7205	0.4585	0.6096	0.36
10	0.5004	0.4088	0.5965	0.8786	0.5714	0.1120	0.8418	0.1092	0.6013	0	0.7591	0.8715	0.3402	0.3816	0.5808	0.2322	0.80
11	0.5812	0.4186	0.1834	0.5243	0.8197	0.8616	0.4000	0.8368	0.3350	0.7591	0	0.1129	0.4211	0.6988	0.7903	0.6569	0.04
12	0.6635	0.5204	0.2926	0.5365	0.9128	0.9733	0.4099	0.9478	0.4231	0.8715	0.1129	0	0.5328	0.7933	0.8816	0.7623	0.06
13	0.3547	0.1403	0.2732	0.6151	0.6089	0.4405	0.5469	0.4160	0.3512	0.3402	0.4211	0.5328	0	0.3701	0.5963	0.2644	0.46
14	0.1715	0.2875	0.6096	0.5793	0.9001	0.3944	0.5892	0.3522	0.7205	0.3816	0.6988	0.7933	0.3701	0	0.8987	0.1590	0.73
15	0.9410	0.7366	0.6210	1.1885	0.0384	0.6643	1.0966	0.6825	0.4585	0.5808	0.7903	0.8816	0.5963	0.8987	0	0.7430	0.82
16	0.2703	0.2449	0.5325	0.6638	0.7430	0.2728	0.6445	0.2330	0.6096	0.2322	0.6569	0.7623	0.2644	0.1590	0.7430	0	0.69
17	0.6117	0.4579	0.2269	0.5247	0.8572	0.9058	0.3986	0.8807	0.3696	0.8038	0.0450	0.0679	0.4653	0.7349	0.8270	0.6981	
18	0.4406	0.3192	0.4991	0.7992	0.5566	0.2048	0.7547	0.1891	0.5153	0.0988	0.6604	0.7728	0.2416	0.3504	0.5595	0.1914	0.70
19	0.8402	0.6330	0.3101	0.8467	0.6676	0.9550	0.7225	0.9432	0.2443	0.8431	0.3226	0.3455	0.5628	0.9173	0.6310	0.8272	0.32
20	0.7733	0.5557	0.2481	0.8379	0.5563	0.8393	0.7193	0.8297	0.1261	0.7273	0.3312	0.3906	0.4671	0.8322	0.5209	0.7294	0.35

Figure VI.1 **Distance Matrix of data TSP [21]**



obj2

40x40 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0.1047	0.7766	0.5925	0.4042	0.5869	0.0437	0.5633	0.4659	0.0273	0.1710	0.3162	0.6925	0.6331	0.3416	0.7728	0.45
2	0.1047	0	0.6719	0.4878	0.2995	0.4822	0.0611	0.4585	0.3611	0.1321	0.0663	0.2115	0.5878	0.5284	0.2369	0.6681	0.34
3	0.7766	0.6719	0	0.1841	0.3724	0.1897	0.7330	0.2134	0.3108	0.8040	0.6056	0.4604	0.0841	0.1435	0.4350	0.0038	0.32
4	0.5925	0.4878	0.1841	0	0.1883	0.0056	0.5489	0.0293	0.1267	0.6199	0.4215	0.2763	0.0999	0.0406	0.2509	0.1803	0.14
5	0.4042	0.2995	0.3724	0.1883	0	0.1827	0.3606	0.1590	0.0616	0.4316	0.2332	0.0880	0.2882	0.2289	0.0626	0.3686	0.04
6	0.5869	0.4822	0.1897	0.0056	0.1827	0	0.5433	0.0237	0.1211	0.6143	0.4159	0.2707	0.1056	0.0462	0.2453	0.1859	0.13
7	0.0437	0.0611	0.7330	0.5489	0.3606	0.5433	0	0.5196	0.4222	0.0710	0.1274	0.2725	0.6488	0.5894	0.2980	0.7292	0.40
8	0.5633	0.4585	0.2134	0.0293	0.1590	0.0237	0.5196	0	0.0974	0.5906	0.3922	0.2470	0.1292	0.0699	0.2216	0.2096	0.11
9	0.4659	0.3611	0.3108	0.1267	0.0616	0.1211	0.4222	0.0974	0	0.4932	0.2948	0.1496	0.2266	0.1673	0.1242	0.3070	0.01
10	0.0273	0.1321	0.8040	0.6199	0.4316	0.6143	0.0710	0.5906	0.4932	0	0.1984	0.3435	0.7198	0.6604	0.3690	0.8002	0.47
11	0.1710	0.0663	0.6056	0.4215	0.2332	0.4159	0.1274	0.3922	0.2948	0.1984	0	0.1452	0.5214	0.4621	0.1706	0.6018	0.28
12	0.3162	0.2115	0.4604	0.2763	0.0880	0.2707	0.2725	0.2470	0.1496	0.3435	0.1452	0	0.3763	0.3169	0.0254	0.4566	0.13
13	0.6925	0.5878	0.0841	0.0999	0.2882	0.1056	0.6488	0.1292	0.2266	0.7198	0.5214	0.3763	0	0.0594	0.3509	0.0803	0.24
14	0.6331	0.5284	0.1435	0.0406	0.2289	0.0462	0.5894	0.0699	0.1673	0.6604	0.4621	0.3169	0.0594	0	0.2915	0.1397	0.18
15	0.3416	0.2369	0.4350	0.2509	0.0626	0.2453	0.2980	0.2216	0.1242	0.3690	0.1706	0.0254	0.3509	0.2915	0	0.4312	0.10
16	0.7728	0.6681	0.0038	0.1803	0.3686	0.1859	0.7292	0.2096	0.3070	0.8002	0.6018	0.4566	0.0803	0.1397	0.4312	0	0.32
17	0.4513	0.3465	0.3254	0.1413	0.0470	0.1357	0.4076	0.1120	0.0146	0.4786	0.2802	0.1350	0.2412	0.1818	0.1096	0.3216	
18	0.0450	0.0597	0.7316	0.5475	0.3592	0.5419	0.0014	0.5182	0.4208	0.0724	0.1260	0.2712	0.6475	0.5881	0.2966	0.7278	0.40
19	0.0163	0.1210	0.7929	0.6088	0.4205	0.6032	0.0600	0.5795	0.4821	0.0111	0.1873	0.3325	0.7088	0.6494	0.3579	0.7891	0.46
20	0.1378	0.0331	0.6388	0.4547	0.2664	0.4491	0.0941	0.4255	0.3281	0.1651	0.0332	0.1784	0.5547	0.4953	0.2038	0.6350	0.31

Command Window

ag to move the document tabs...

Figure VI.2 Cost Matrix of data TSP [21]

VI.2.2 MOACO algorithm

The role of this algorithm is to give solutions of the multi objective ACO problem according to the given data, our goal is to look on Pareto front which represent the non-dominated solutions which are the best solutions we can get, in this area the dominance check step playing a vital importance to identify the Pareto front.

The solutions are a set of two outputs as we have two objective.

This algorithm is taken from [22]

The algorithm is below:

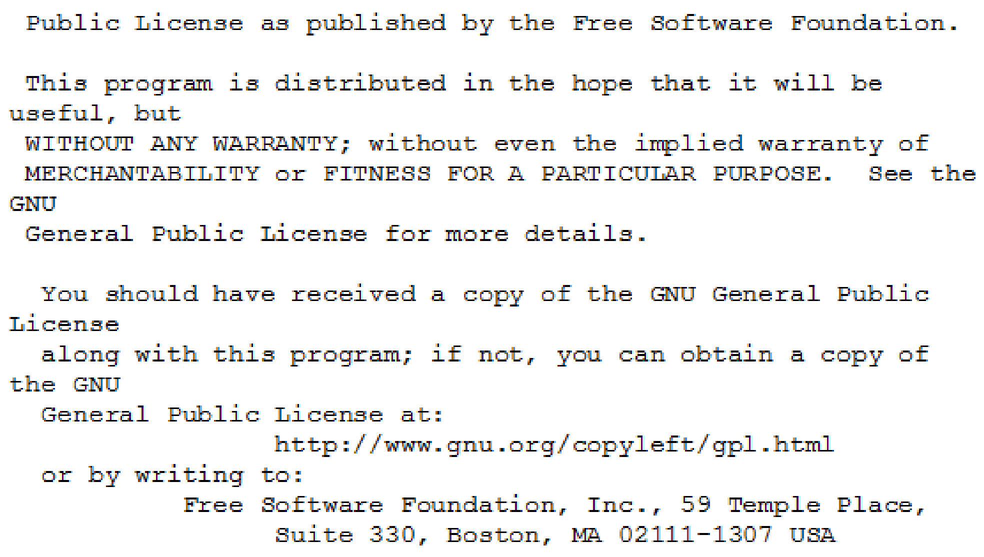
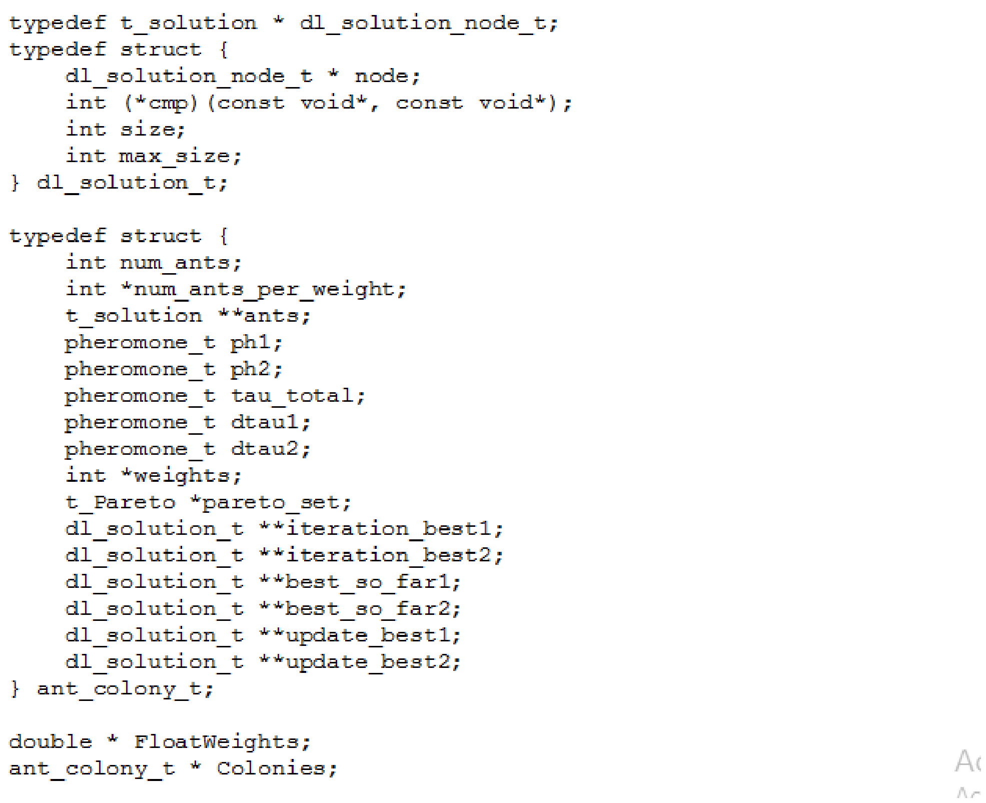


FIGURE VI.3 reources for MOACO psuedocode

The first part of this algorithm is the initialization of parameters such as the pheromone matrices, the heuristic matrices, alpha (the coeficient of pheromone value concentration),

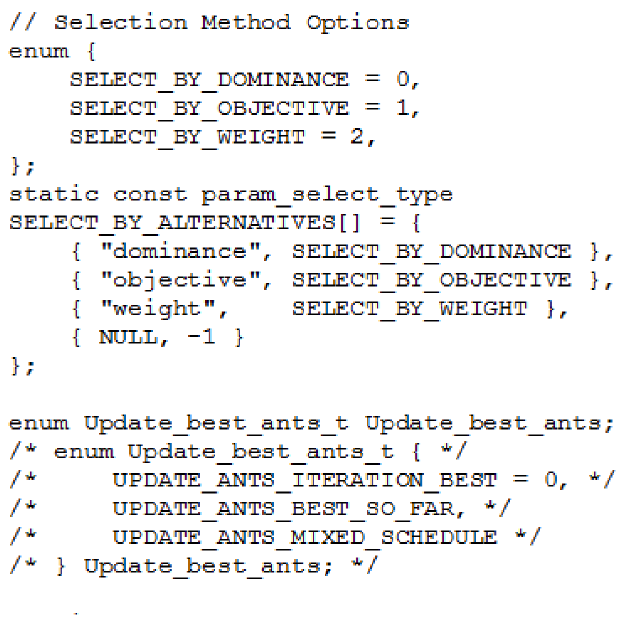
Beta (the coeficient of heuristic value information), number of iterations, number of ants we use in a single colony, number of colonies, the weight for each objective , number of objetives



```
typedef t_solution * dl_solution_node_t;
typedef struct {
    dl_solution_node_t * node;
    int (*cmp)(const void*, const void*);
    int size;
    int max_size;
} dl_solution_t;

typedef struct {
    int num_ants;
    int *num_ants_per_weight;
    t_solution **ants;
    pheromone_t ph1;
    pheromone_t ph2;
    pheromone_t tau_total;
    pheromone_t dtau1;
    pheromone_t dtau2;
    int *weights;
    t_Pareto *pareto_set;
    dl_solution_t **iteration_best1;
    dl_solution_t **iteration_best2;
    dl_solution_t **best_so_far1;
    dl_solution_t **best_so_far2;
    dl_solution_t **update_best1;
    dl_solution_t **update_best2;
} ant_colony_t;

double * FloatWeights;
ant_colony_t * Colonies;
```

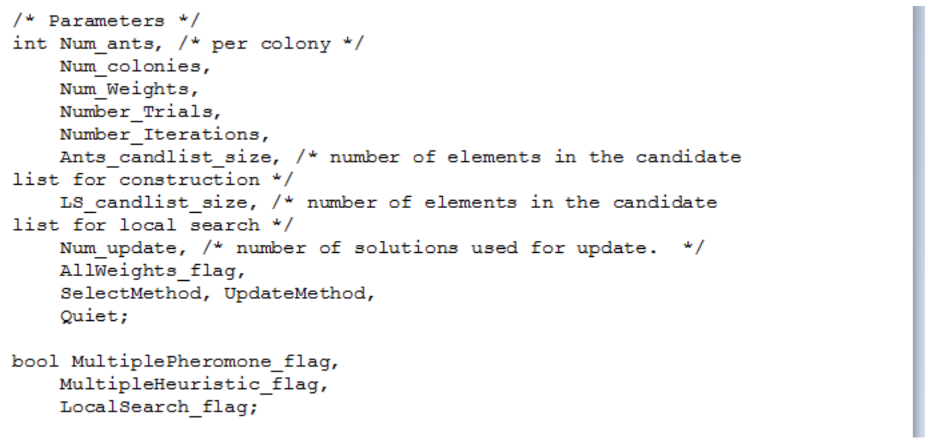
Figure VI.4 prototypes of MOACO parameters

The second part is the part of algorithm procees which contains of updating pheromones, selection the next city for each ant by the probability ,local search , pareto dominance selection , selection solutions generated by ants, selection the best solutions



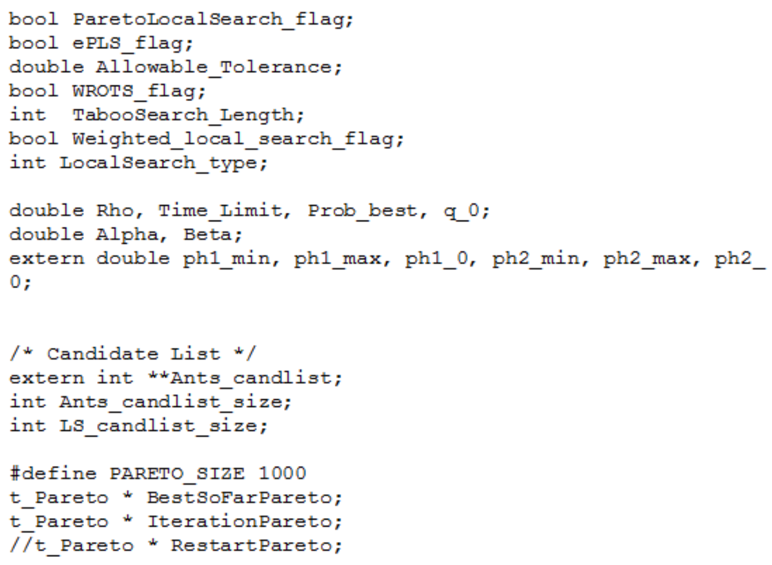
```
// Selection Method Options
enum {
     SELECT_BY_DOMINANCE = 0,
     SELECT_BY_OBJECTIVE = 1,
     SELECT_BY_WEIGHT = 2,
};
static const param_select_type
SELECT_BY_ALTERNATIVES[] = {
     { "dominance", SELECT_BY_DOMINANCE },
     { "objective", SELECT_BY_OBJECTIVE },
     { "weight",    SELECT_BY_WEIGHT },
     { NULL, -1 }
};

enum Update_best_ants_t Update_best_ants;
/* enum Update_best_ants_t { */
/*      UPDATE_ANTS_ITERATION_BEST = 0, */
/*      UPDATE_ANTS_BEST_SO_FAR, */
/*      UPDATE_ANTS_MIXED_SCHEDULE */
/* } Update_best_ants; */
```

Figure VI.5 prototypes of MOACO process



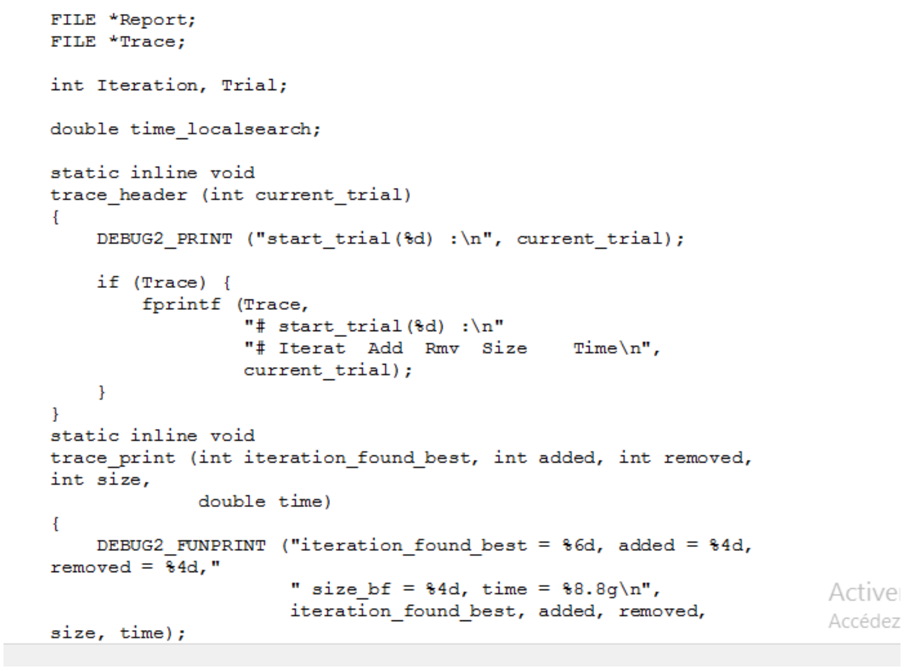
```
/* Parameters */
int Num_ants, /* per colony */
    Num_colonies,
    Num_Weights,
    Number_Trials,
    Number_Iterations,
    Ants_candlist_size, /* number of elements in the candidate
list for construction */
    LS_candlist_size, /* number of elements in the candidate
list for local search */
    Num_update, /* number of solutions used for update.  */
    AllWeights_flag,
    SelectMethod, UpdateMethod,
    Quiet;

bool MultiplePheromone_flag,
    MultipleHeuristic_flag,
    LocalSearch_flag;
```

Figure VI.6 prototypes of MOACO process



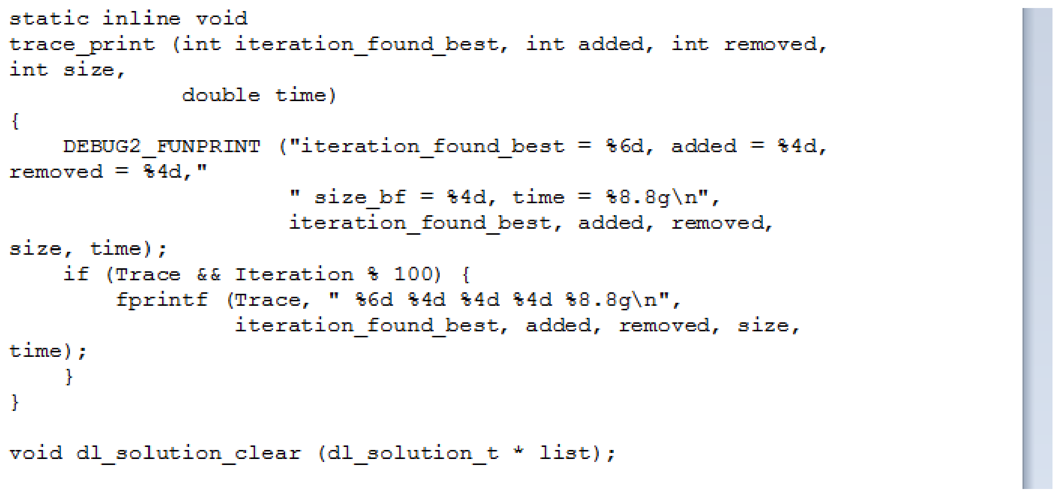
```
bool ParetoLocalSearch_flag;
bool ePLS_flag;
double Allowable_Tolerance;
bool WROTS_flag;
int  TabooSearch_Length;
bool Weighted_local_search_flag;
int LocalSearch_type;

double Rho, Time_Limit, Prob_best, q_0;
double Alpha, Beta;
extern double ph1_min, ph1_max, ph1_0, ph2_min, ph2_max, ph2_
0;


/* Candidate List */
extern int **Ants_candlist;
int Ants_candlist_size;
int LS_candlist_size;

#define PARETO_SIZE 1000
t_Pareto * BestSoFarPareto;
t_Pareto * IterationPareto;
//t_Pareto * RestartPareto;
```

Figure VI.7 prototypes of MOACO process



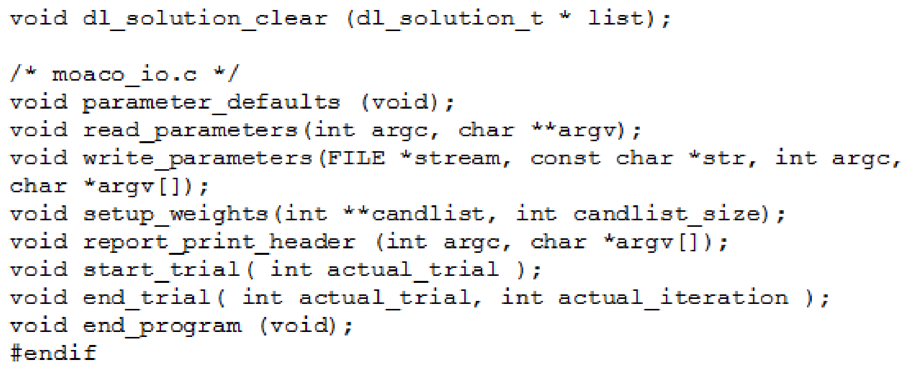
```
FILE *Report;
FILE *Trace;

int Iteration, Trial;

double time_localsearch;

static inline void
trace_header (int current_trial)
{
    DEBUG2_PRINT ("start_trial(%d) :\n", current_trial);

    if (Trace) {
        fprintf (Trace,
                 "# start_trial(%d) :\n"
                 "# Iterat  Add  Rmv  Size    Time\n",
                 current_trial);
    }
}
static inline void
trace_print (int iteration_found_best, int added, int removed,
int size,
             double time)
{
    DEBUG2_FUNPRINT ("iteration_found_best = %6d, added = %4d,
removed = %4d,"
                     " size_bf = %4d, time = %8.8g\n",
                     iteration_found_best, added, removed,
size, time);
```

Figure VI.8 prototypes of MOACO process



```
static inline void
trace_print (int iteration_found_best, int added, int removed,
int size,
             double time)
{
    DEBUG2_FUNPRINT ("iteration_found_best = %6d, added = %4d,
removed = %4d,"
                     " size_bf = %4d, time = %8.8g\n",
                     iteration_found_best, added, removed,
size, time);
    if (Trace && Iteration % 100) {
        fprintf (Trace, " %6d %4d %4d %4d %8.8g\n",
                 iteration_found_best, added, removed, size,
time);
    }
}

void dl_solution_clear (dl_solution_t * list);
```

Figure VI.9 prototypes of MOACO process



```
void dl_solution_clear (dl_solution_t * list);

/* moaco_io.c */
void parameter_defaults (void);
void read_parameters(int argc, char **argv);
void write_parameters(FILE *stream, const char *str, int argc,
char *argv[]);
void setup_weights(int **candlist, int candlist_size);
void report_print_header (int argc, char *argv[]);
void start_trial( int actual_trial );
void end_trial( int actual_trial, int actual_iteration );
void end_program (void);
#endif
```

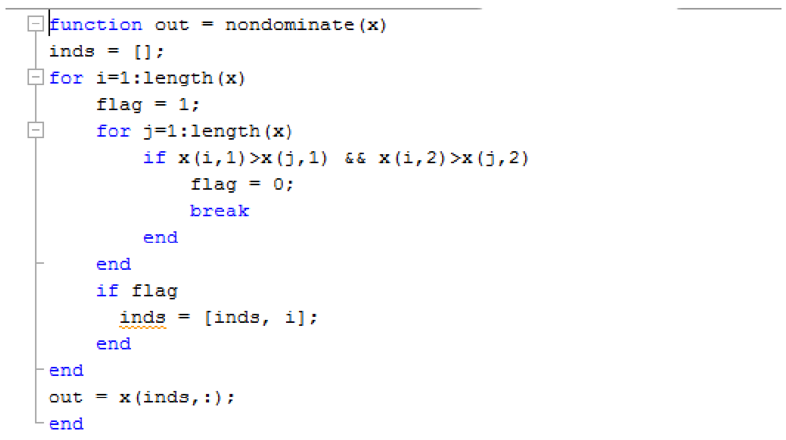
Figure VI.10 prototypes of end MOACO process and visualize solutions

The solutions of MOACO is a set of solutions because we are in multi objective problem, we will pass this solutions to dominance check algorithm to get the non dominated solutions.

VI.2.3 Dominance check algorithm

This algorithm provide the non dominated solutions and identify pareto front

The algorithm is below:



```
function out = nondominate(x)
inds = [];
for i=1:length(x)
    flag = 1;
    for j=1:length(x)
        if x(i,1)>x(j,1) && x(i,2)>x(j,2)
            flag = 0;
            break
        end
    end
    if flag
      inds = [inds, i];
    end
end
out = x(inds,:);
end
```

FIGURE VI.11 domiance check algorithm [21]

The results of MOTSP is visualized in the objective space by the pareto front (Figure VI.12)

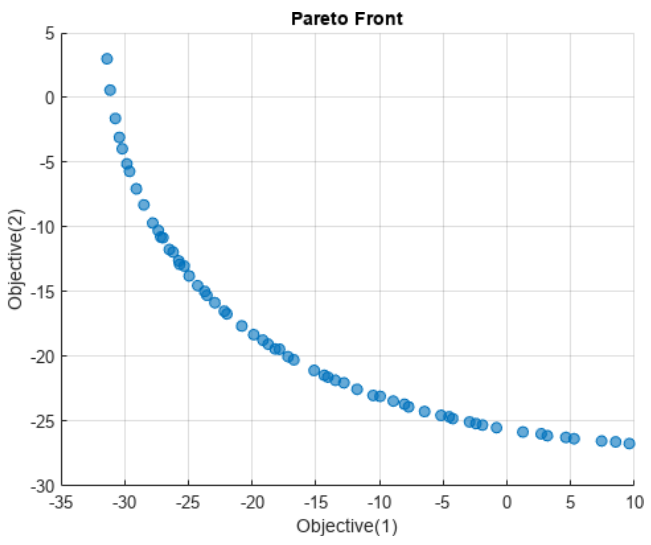


Figure VI.12 parto front in MOP

VI.3 Conclusion

In this chapter we discussed the execution steps , unfortunately because of errors in compilation we didn’t get the results, this research have been studied by many researchers and they get good results when applying MOACO for multi objective Travelling salesmen problem.

General Conclusion

In this work, we are exploring the concepts of optimization and combinatorial optimization. We are discussing various approaches to solving combinatorial optimization problems, with a particular focus on approximate methods. Additionally, we are delving into multi-objective optimization and its practical applications, as well as the utilization of metaheuristics for solving multi-objective problems. The ant colony optimization algorithm is being introduced and its role in solving optimization problems is being explained. Furthermore, we are examining the MOACO algorithm and one of its variants, PACO, which are metaheuristic approaches for tackling combinatorial optimization with multiple objectives.

The PACO algorithm for the multi-objective TSP is being presented in this work. Previous studies have confirmed the quality of the solutions obtained using this algorithm. Unfortunately, in our research, we encountered errors while trying to compile and implement the algorithm. Successful implementation requires a coding background to adjust the parameters of the pseudocode, as well as a background in operational research to fully comprehend the research findings.

Future research is likely to focus on the pheromone update and use of historical data , the PACO algorithm is expected to be used in variety of future projects, Including :

- PACO combined with local search methods to improve the quality of solutions.

References

[1] Winston, Wayne L. Operations research: applications and algorithms. Cengage Learning, 2022.

[2] Deb, Kalyanmoy. Optimization for engineering design: Algorithms and examples. PHI Learning Pvt. Ltd., 2012.

[3] Papadimitriou, Christos H., and Kenneth Steiglitz. Combinatorial optimization: algorithms and complexity. Courier Corporation, 1998.

[4] NIJKAMP, P. "MULTIOBJECTIVE PROGRAMMING AND PLANNING-COHON, JL." (1980): 477-478.

[5] Talbi, El-Ghazali. Metaheuristics: from design to implementation. John Wiley & Sons, 2009.

[6] Bazaraa, Mokhtar S., John J. Jarvis, and Hanif D. Sherali. Linear programming and network flows. John Wiley & Sons, 2011

[7] Collette, Yann, and Patrick Siarry. Optimisation multiobjectif: Algorithmes. Editions Eyrolles, 2011.

[8] personal synthesis

[9] Lawler, Eugene L., et al. "The traveling salesman problem: a guided tour of combinatorial optimization." The Journal of the Operational Research Society 37.5 (1986): 535.

[10] Lawler, Eugene L. Combinatorial optimization: networks and matroids. Courier Corporation, 2001.

[11] Deb, Kalyanmoy. "Multi-objective optimisation using evolutionary algorithms: an introduction." Multi-objective evolutionary optimisation for product design and manufacturing. London: Springer London, 2011. 3-34.

[12] Labed.said. Methodes bio-inpirances hybrids pour la resolution de problemes complexes. University of constatntine ,(2013).

[13] Deffas Zineb. Single-solution parallel metaheuristics for solving the quad problem on grid computing (magister). University of oum bouaghi, (2010).

[14] Marco Dorigo and Thomas Stützle. Ant Colony Optimization The MIT Press Cambridge, Massachusetts London, England (2004).

[15] Atabati, Morteza, Kobra Zarei, and Azam Borhani. "Ant colony optimization as a descriptor selection in QSPR modeling: estimation of the λmax of anthraquinones-based dyes." *Journal of Saudi Chemical Society* 20 (2016): S547-S551.

[16] Gonzales Martínez, Rolando. "Balancing Input-Output tables with Bayesian slave-raiding ants." Statistical Journal of the IAOS 33.4 (2017): 943-949.

[17] Almufti, S., R. Marqas, and V. Ashqi. "Taxonomy of bio-inspired optimization algorithms." Journal Of Advanced Computer Science & Technology 8.2 (2019): 23.

[18] Angus, Daniel. "Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem." 2007 IEEE Symposium on computational intelligence in multi-criteria decision-making. IEEE, 2007.

[19] Angus, Daniel. "Niching for ant colony optimisation." Biologically-Inspired Optimisation Methods: Parallel Algorithms, Systems and Applications. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. 165-188.

[20] Osaba, E., Carballedo, R., Del Ser, J., & Quintián, H. Evolutionary Multi-Objective Optimization in Uncertain Environments: Issues and Algorithms. Springer (2017).

[21] Evolutionary multi-objective optimization platform https://github.com/BIMK/PlatEMO. 06/06/2023 23:04.

[22]https://github.com/oscarfmdc/antSystem/blob/master/multiObjective/moaco/trunk/moaco.h. 06/06/2023 22:45.

[23] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." IEEE transactions on systems, man, and cybernetics, part b (cybernetics) 26.1 (1996): 29-41.

[24] Colorni, Alberto, Marco Dorigo, and Vittorio Maniezzo. "Distributed optimization by ant colonies." Proceedings of the first European conference on artificial life. Vol. 142. 1991.

[25] Dorigo, Marco, and Luca Maria Gambardella. "Ant colonies for the travelling salesman problem." biosystems 43.2 (1997): 73-81.

[26] Lv, Qiang, Xiaoyan Xia, and Peide Qian. "A parallel aco approach based on one pheromone matrix." International workshop on ant colony optimization and swarm intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[27] Mo, Yadong, Xiaoming You, and Sheng Liu. "Multi-colony ant optimization with dynamic collaborative mechanism and cooperative game." Complex & Intelligent Systems 8.6 (2022): 4679-4696.