Université Mohamed Khider de Biskra
Faculté des Sciences et de la Technologie
Département de Génie Electrique

MÉMOIREDEMASTER

SciencesetTechnologies
Télécommunications
Réseauxettélécommunications

Présenté et soutenupar:

ElguerriKaouthar

KamiriNihal

Le: 26 Juin 2022

Metaheuristic optimization algorithms (genetic, simulated annealing)

Jury:

Mme.	OUARHLENT Saloua	**MAA**	UniversitédeBiskra	**Président**
Mme.	HENDAOUI Mounira	**MCB**	UniversitédeBiskra	**Encadreur**
Mme.	ATAMENA Noura	**MAA**	UniversitédeBiskra	**Examinateur**

Mohamed Khider University of Biskra
Faculty of Sciences and Technology
Department of Electrical Engineering

MASTER'STHESIS

ElectricalEngineeringTelecom
municationsNetworksandcom
munications

Presented and supportedby:

ElguerriKaouthar

Kamiri Nihal

On: June 26, 2022

Metaheuristic optimization algorithms (genetic, simulated annealing)

Jury Members:

Academic year:2021-2022

Mohamed Khider University of Biskra
Faculty of Sciences and Technology
Department of ElectricalEngineering

MASTER’STHESIS

Electrical Engineering
Telecommunications
Networks and communications

Presented and supportedby:

ElguerriKaouthar

Kamiri Nihal

On: June 26, 2022

Metaheuristic optimization algorithms (genetic, simulated annealing)

Presentedby:	**Favorableopinionofthesupervisor:**
ElguerriKaouthar	Mrs.HinndaouiMounira
Kamiri Nihal	

Favorable opinion of the President of Jury:

Mrs.OUARHLENT Saloua

Stamp and signature

Content

List of figures

Abbreviations

GA	Genetic algorithm
TSP	Traveling Salesman Problem
WSN	Wireless Sensor Networks
SA	Simulated Annealing
Optima	Optimum

Dedication 1

I always thought of doing or giving something to my parents as a sign of gratitude for all their efforts, just to see me succeed, and now, the opportunity has come.

To those who gave me life, symbol of beauty, pride and wisdom and patience: my dear dad and my sweet mom.

To my brothers and sisters:

“Ishak, Bouthaina, Salsabil, Tasnim, Mohamed Yakoub”.

To my supervisor.

To all my teachers each with his name.

All my friends .

To my colleagues from the class of 2022.

To the readers of this memory.

Acknowledgement 1

I would first like to thank my god “Allah” who gave me the courage to do this work. First ofall, I would like to express my thanks and my deep gratitude to my supervisor Dr.MouniraHendaoui for his requirement of clarity and rigor which brought me a lot, for the Confidence he showed me during decisive moments.As well as for his support throughout this memory.

My gratitude goes as much to the members of the jury: and, for having done me the honor of examining this thesis. May they find here the expression of my deep respect.

I would like to thank the teachers, librarians and administrators of the Faculty of Electrical Engineering.

Dedication 2

I dedicate this thesis to:

A special dedication to my heroes of life my father and my mother who gave me love and hope for encouraging me in my education.

To my soul my sister “Lina”, and my brothers “Ilyes and Hamada” and my lovely Rita.

To my supervisor.

To all my teachers.

To all my friends .

To my colleagues from the class of 2022.

To the readers of this memory.

Acknowledgement 2

I would first like to thank my god &quot;Allah&quot; who gave me the courage to do this work.

I wish to thank my supervisor Dr. MouniraHendaoui who was more than generous with his expertise and precious time, and for guiding me and encouraging me to do thiswork. A special thanks to the jury member for criticizing my thesis.

I would like to express my sincere gratitude to Mohamed Khider University and all the teachers in the electrical engineering department for all the considerate guidance.

Abstract

I am sure you already heard about the traveling salesman problem or TSP. that belong to NP-hard optimization problems which are difficult to solve using classical mathematical methods. There are many applications for this problem and also many solutions with different performances to find the optimal solution in shortest possible time. Meta-Heuristic algorithms are one of the proposed solutions which are successful in finding the solutions that are very near to the optimal. In this work we applied the genetic algorithm then the simulated annealing then we have compared between them to get the faster and the optimum

The results show that the faster is the genetic algorithm and the the optimum is simulated annealing

Keywords: TSP problem, genetic algorithm, metaheuristic, optimia, optimization algorithm, simulated annealing

ملخص

أنا متأكد من أنك سمعت بالفعل عن مشكلة البائع المتجول أو TSP. التي تنتمي إلى مشكلات تحسين NP-hard والتي يصعب حلها باستخدام الطرق الرياضية التقليدية. هناك العديد من التطبيقات لهذه المشكلة وأيضًا العديد من الحلول ذات الأداء المختلف لإيجاد الحل الأمثل في أقصر وقت ممكن. تعد الخوارزميات الفوقية من الحلول المقترحة الناجحة في إيجاد الحلول القريبة جدًا من الحل الأمثل. في هذا العمل قمنا بتطبيق الخوارزمية الجينية ثم التلدين المحاكى ثم قمنا بالمقارنة بينهما للحصول على الأسرع والأفضل

أظهرت النتائج أنه كلما كانت الخوارزمية الجينية أسرع والأمثل هو التلدين

الكلمات المفتاحية: مشكلة TSP ، الخوارزمية الجينية ، metatheuristic ، الأمثل ، خوارزمية التحسين ، محاكاة التلدين

General Introduction

General introduction

Solving an optimization problem consists of exploring a search space in order to maximize (or minimize) a given function.The relative complexities (in size or structure) of the search space and the function to be optimized lead to the use of radically different resolution methods. As a first approximation, we can say that a deterministic method is suitable for a small and complex search space and that a large search space rather requires a stochastic search method (simulated annealing, genetic algorithm ...). In most cases, an optimization problem is naturally divided into two phases: search for acceptable solutions and then search for the cost-optimal solution among them.

The travelling salesman problem (TSP) is one of the most signifi-cant problems in combinatorial optimization. It is important both as a separate problem and as a part of more complex optimization problems. Also, since it is strongly NP-complete, in practical applications for large-scale problem instances it is only possible to use heuristic optimization algorithms which give approximate solutions. Some of the most successful heuristic algorithms that have been used to solve the TSP and its variants include genetic algorithms, simulated annealing.

In this memory, we compare two metaheuristic optimization algorithms appliedto solving the travelling salesman problem. We focus on two methods: genetic algorithms and simulated annealing and we compare between the results of these two methods.

Chapter I

Optimization algorithms

I.1 Introduction:

Optimization is the process of finding the best solution to a problem. For example, finding the shortest route to the destination, scheduling the hospital staff in the most efficient manner possible, or planning the day's activities to best utilize the available time. In order to solve the real-world optimization problems, the problems are formulated as mathematical functions and optimization deals with minimizing/maximizing the output of that function to find the best solution.

I.2 Machine Learning:

I.2.1 Definition

Since their evolution, humans have been using many types of tools to accomplish various tasks in a simpler way. The creativity of the human brain led to the invention of different machines. These machines made the human life easy by enabling people to meet various life needs, including travelling, industries, and computing. Moreover, Machine learning is the one among them. According to Arthur Samuel Machine learning is defined as the field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel was famous for his checkers-playing program. Machine learning (ML) is used to teach machines how to handle the data more efficiently.

Machine Learning relies on different algorithms to solve data problems. The kind of algorithm employed depends on the kind of problem you wish to solve, the number of variables, the kind of model that would suit it best and so on.(1)

The basic concept of machine learning in data science involves using statistical learning and optimization methods that let computers analyze datasets and identify patterns. The typical supervised machine learning algorithm consists of (roughly) three components

1. A decision process: A recipe of calculations or other steps that takes in the data and returns a “guess” at the kind of pattern in the data your algorithm is looking to find.
2. An error function: A method of measuring how good the guess was by comparing it to known examples (when they are available). Did the decision

process get it right? If not, how do you quantify "how bad" the miss was?

3. An updating or optimization process: Where the algorithm looks at the miss and then updates how the decision process comes to the final decision so that the next time the miss won't be as great.

Figure I 1: the applications of machine learning

I.2.2 Types of Machine Learning:

Many machine learning models are defined by the presence or absence of human influence on raw data whether a reward is offered, specific feedback is given or labels are used

- Supervised learning: The dataset being used has been pre-labeled and classified by users to allow the algorithm to see how accurate its performance is.
- Unsupervised learning: The raw dataset being used is unlabeled and an algorithm identifies patterns and relationships within the data without help from users.
- Semi supervised learning: The dataset contains structured and unstructured data, which guide the algorithm on its way to making independent conclusions. The combination of the two data types in one training dataset allows machine learning algorithms to learn to label unlabeled data.
- Reinforcement learning: The dataset uses a "rewards/punishments" system, offering feedback to the algorithm to learn from its own experiences by trial and error

I.2.2.1 The difference between supervised and unsupervised learning

The main difference between supervised and unsupervised learning: Labeled data

The main distinction between the two approaches is the use of labeled datasets. To put it simply, supervised learning uses labeled input and output data, while an unsupervised learning algorithm does not.

In supervised learning, the algorithm “learns” from the training dataset by iteratively making predictions on the data and adjusting for the correct answer. While supervised learning models tend to be more accurate than unsupervised learning models, they require upfront human intervention to label the data appropriately. For example, a supervised learning model can predict how long your commute will be based on the time of day, weather conditions and so on. But first, you’ll have to train it to know that rainy weather extends the driving time.

Unsupervised learning models, in contrast, work on their own to discover the inherent structure of unlabeled data. Note that they still require some human intervention for validating output variables. For example, an unsupervised learning model can identify that online shopper often purchase groups of products at the same time.

I.2.3 Why Is Machine Learning Important?

Machine learning and data mining, a component of machine learning, are crucial tools in the process to glean insights from massive datasets held by companies and researchers today. There are two main reasons for this:

Scale of data: Companies are faced with massive volumes and varieties of data that need to be processed. Processing power is more efficient and readily available. Models that can be programmed to process data on their own, determine conclusions, and identify patterns are invaluable.

Unexpected findings: Since machine learning algorithms update autonomously, the analytical accuracy improves with each run as it teaches itself from the datasets it analyzes. This iterative nature of learning is unique and valuable because it occurs without human intervention, providing the ability to uncover hidden insights without being specifically programmed to do so.(2)

I.2.4 Machine Learning vs. Deep Learning vs. Neural Networks:

Since deep learning and machine learning tend to be used interchangeably, it is worth noting the nuances between the two. Machine learning, deep learning, and neural networks are all sub-fields of artificial intelligence. However, deep learning is actually a sub-field of machine learning, and neural networks is a sub-field of deep learning.

The way in which deep learning and machine learning differ is in how each algorithm learns. Deep learning automates much of the feature extraction piece of the process, eliminating some of the manual human intervention required and enabling the use of larger data sets.

Classical, or "non-deep", machine learning is more dependent on human intervention to learn. Human experts determine the set of features to understand the differences between data inputs, usually requiring more structured data to learn.

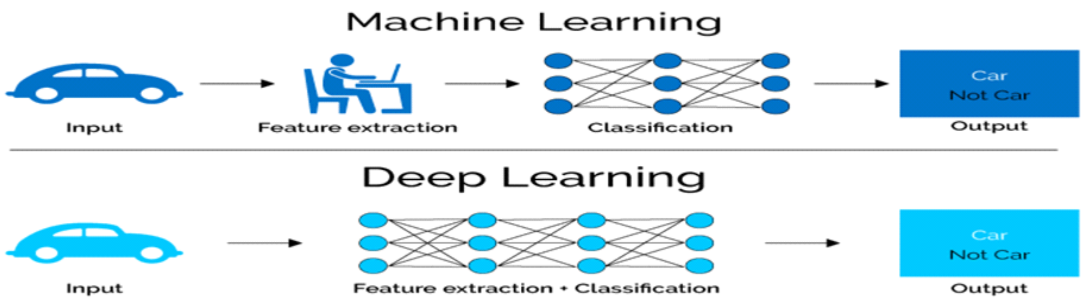


Figure I 2: Machine learning and Deep learning

Neural networks, or artificial neural networks (ANNs), are comprised of a node layer, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. The “deep” in deep learning is just referring to the depth of layers in a neural network. A neural network that consists of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm or a deep neural network. A neural network that only has two or three layers is just a basic neural network. /IBM (3)

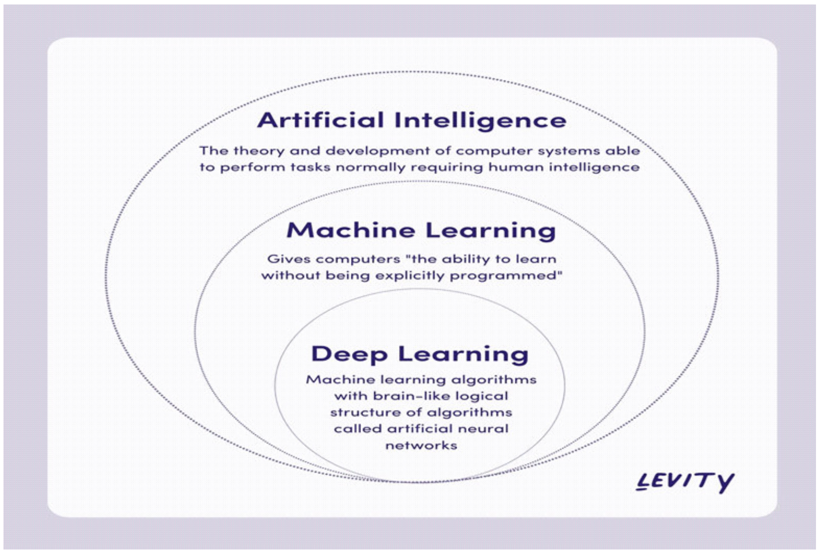

Artificial Intelligence
The theory and development of computer systems able to perform tasks normally requiring human intelligence
Machine Learning
Gives computers "the ability to learn without being explicitly programmed"
Deep Learning
Machine learning algorithms with brain-like logical structure of algorithms called artificial neural networks
LEVITY

Figure I 3: Artificial Intelligence

I.3 Optimization algorithms

I.3.1 Optimization

Optimization is a branch of mathematics seeking to model, analyze and solve analytically or numerically a given problem with the aim of finding the solution that maximizes or minimizes a function to be optimized.

It is the challenging problem that underlies many machine learning algorithms, from fitting logistic regression models to training artificial neural networks. There are perhaps hundreds of popular optimization algorithms, and perhaps tens of algorithms to choose from in popular scientific code libraries. This can make it challenging to know which algorithms to consider for a given optimization problem.(4)

I.3.2 Optimization algorithms:

Optimization refers to a procedure for finding the input parameters or arguments to a function that result in the minimum or maximum output of the function. The most common type of optimization problems encountered in machine learning are continuous function optimization, where the input arguments to the function are real-valued numeric values, e.g. floating point values. The output from the function is also a real-valued evaluation of the input values. We might refer to problems of this type as continuous function optimization, to distinguish from functions that take discrete variables and are referred to as combinatorial optimization problems. There are many different types of optimization algorithms that can be used for continuous function optimization problems, and perhaps just as many ways to group and summarize them.

One approach to grouping optimization algorithms is based on the amount of information available about the target function that is being optimized that, in turn, can be used and harnessed by the optimization algorithm. Generally, the more information that is available about the target function, the easier the function is to optimize if the information can effectively be used in the search. Perhaps the major division in optimization algorithms is whether the objective function can be differentiated at a point or not. That is, whether the first derivative (gradient or slope) of the function can be calculated for a given candidate solution or not. This partitions algorithms into those that can make use of the calculated gradient information and those that do not.Algorithms that use derivative information.(4)

I.3.3 Differentiable Objective Function

A differentiable function is a function where the derivative can be calculated for any given point in the input space.

The derivative of a function for a value is the rate or amount of change in the function at that point. It is often called the slope.

I.3.3.1 First-Order Derivative

Slope or rate of change of an objective function at a given point.

The derivative of the function with more than one input variable (e.g. multivariate inputs) is commonly referred to as the gradient.

I.3.3.2 Gradient

Derivative of a multivariate continuous objective function.

A derivative for a multivariate objective function is a vector, and each element in the vector is called a partial derivative or the rate of change for a given variable at the point assuming all other variables are held constant.

I.3.3.3 Partial Derivative

Element of a derivative of a multivariate objective function.

We can calculate the derivative of the derivative of the objective function that is the rate of change of the rate of change in the objective function. This is called the second derivative.

I.3.3.4 Second-Order Derivative

Rate at which the derivative of the objective function changes.

For a function that takes multiple input variables, this is a matrix and is referred to as the Hessian matrix.

I.3.3.5 Hessian matrix

Second derivative of a function with two or more input variables, Simple differentiable functions can be optimized analytically using calculus. Typically, the objective functions that we are interested in cannot be solved analytically.

Optimization is significantly easier if the gradient of the objective function can be calculated, and as such, there has been a lot more research into optimization algorithms that use the derivative than those that do not.

Some groups of algorithms that use gradient information include:

- Bracketing Algorithms
- Local Descent Algorithms
- First-Order Algorithms
- Second-Order Algorithms

I.3.3.5.1 Bracketing Algorithms

Algorithms are intended for optimization problems with one input variable where the optima is known to exist within a specific range.

Bracketing algorithms are able to efficiently navigate the known range and locate the optima, although they assume only single optima is present (referred to as unimodal objective functions).

Some bracketing algorithms may be able to be used without derivative information if it is not available.

Examples of bracketing algorithms include:

•Fibonacci Search

•Golden Section Search

•Bisection Method

I.3.3.5.2 Local Descent Algorithms

Local descent optimization algorithms are intended for optimization problems with more than one input variable and single global optima (e.g. unimodal objective function).

Perhaps the most common example of a local descent algorithm is the line search algorithm.

•Line Search

There are many variations of the line search (e.g. the Brent-Dekker algorithm), but the procedure generally involves choosing a direction to move in the search space, then performing a bracketing type search in a line or hyperplane in the chosen direction.

This process is repeated until no further improvements can be made.

The limitation is that it is computationally expensive to optimize each directional move in the search space.

I.3.3.5.3 First-Order Algorithms

First-order optimization algorithms explicitly involve using the first derivative (gradient) to choose the direction to move in the search space

The procedures involve first calculating the gradient of the function, then following the gradient in the opposite direction (e.g. downhill to the minimum for minimization problems) using a step size (also called the learning rate).

The step size is a hyperparameter that controls how far to move in the search space, unlike “local descent algorithms” that perform a full line search for each directional move.A step size that is too small results in a search that takes a long time and can get stuck, whereas a step size that is too large will result in zig-zagging or bouncing around the search space, missing the optima completely.

First-order algorithms are generally referred to as gradient descent, with more specific names referring to minor extensions to the procedure, e.g.:

•Gradient Descent

•Momentum

•Adagrad

•RMSProp

•Adam

The gradient descent algorithm also provides the template for the popular stochastic version of the algorithm, named Stochastic Gradient Descent (SGD) that is used to train artificial neural networks (deep learning) models.

The important difference is that the gradient is appropriated rather than calculated directly, using prediction error on training data, such as one sample (stochastic), all examples (batch), or a small subset of training data (mini-batch).

The extensions designed to accelerate the gradient descent algorithm (momentum, etc.) can be and are commonly used with SGD.

•Stochastic Gradient Descent

•Batch Gradient Descent

•Mini-Batch Gradient Descent

I.3.3.5.4 Second-Order Algorithms

Second-order optimization algorithms explicitly involve using the second derivative (Hessian) to choose the direction to move in the search space.

These algorithms are only appropriate for those objective functions where the Hessian matrix can be calculated or approximated.

Examples of second-order optimization algorithms for univariate objective functions include:

•Newton’s Method

•Secant Method

Second-order methods for multivariate objective functions are referred to as Quasi-Newton Methods.

•Quasi-Newton Method

There are many Quasi-Newton Methods, and they are typically named for the developers of the algorithm, such as:

•Davidson-Fletcher-Powell

•Broyden-Fletcher-Goldfarb-Shanno (BFGS)

•Limited-memory BFGS (L-BFGS)

Now that we are familiar with the so-called classical optimization algorithms, let's look at algorithms used when the objective function is not differentiable.

Algorithms that do not use derivative information:

I.3.4 Non-Differential Objective Function

Optimization algorithms that make use of the derivative of the objective function are fast and efficient.

Nevertheless, there are objective functions where the derivative cannot be calculated, typically because the function is complex for a variety of real-world reasons. Or the derivative can be calculated in some regions of the domain, but not all, or is not a good guide.

Some difficulties on objective functions for the classical algorithms described in the previous section include:

•No analytical description of the function (e.g. simulation).

•Multiple global optima (e.g. multimodal).

•Stochastic function evaluation (e.g. noisy).

•Discontinuous objective function (e.g. regions with invalid solutions).

As such, there are optimization algorithms that do not expect first- or second-order derivatives to be available.

These algorithms are sometimes referred to as black-box optimization algorithms as they assume little or nothing (relative to the classical methods) about the objective function.

A grouping of these algorithms includes:

•Direct Algorithms

•Stochastic Algorithms

•Population Algorithms (4)

I.3.4.1 Direct Algorithms

Direct optimization algorithms are for objective functions for which derivatives cannot be calculated.

The algorithms are deterministic procedures and often assume the objective function has single global optima, e.g. unimodal.

Direct search methods are also typically referred to as a “pattern search” as they may navigate the search space using geometric shapes or decisions, e.g. patterns.

Gradient information is approximated directly (hence the name) from the result of the objective function comparing the relative difference between scores for points in the search space. These direct estimates are then used to choose a direction to move in the search space and triangulate the region of the optima.

Examples of direct search algorithms include:

•Cyclic Coordinate Search

•Powell’s Method

•Hooke-Jeeves Method

•Nelder-Mead Simplex Search

I.3.4.2 Stochastic Algorithms

Stochastic optimization algorithms are algorithms that make use of randomness in the search procedure for objective functions for which derivatives cannot be calculated.

Unlike the deterministic direct search methods, stochastic algorithms typically involve a lot more sampling of the objective function, but are able to handle problems with deceptive local optima.

Stochastic optimization algorithms include:

•Simulated Annealing

•Evolution Strategy

•Cross-Entropy Method

I.3.4.3 Population Algorithms

Population optimization algorithms are stochastic optimization algorithms that maintain a pool (a population) of candidate solutions that together are used to sample, explore, and hone in on an optimum.

Algorithms of this type are intended for more challenging objective problems that may have noisy function evaluations and many global optima (multimodal), and finding a good or good enough solution is challenging or infeasible using other methods.

The pool of candidate solutions adds robustness to the search, increasing the likelihood of overcoming local optima.

Examples of population optimization algorithms include:

•Genetic Algorithm

•Differential Evolution

•Particle Swarm Optimization (4)

I.4 Conclusion

This chapter introduces and summarizes the frequently used optimization methods from the applications of machine learning, and studies their applications in various fields of machine learning. Firstly, we described the machine learning and why we use it then the difference between the machine learning and the Deep Learning and the Neural Networks. Then we described the optimization algorithm. Finally, we discussed about types of optimization algorithm.

Chapter II
Meta-heuristic

II.1 Introduction

Meta-heuristics are the most recent development in approximate search methods for solving complex optimizationvproblems that arise in business, commerce, engineering, industry, and many other areas. The kinds of the metaheuristic method are various. A meta-heuristic guide a subordinate heuristic using concepts derived from artificial intelligence, biological, mathematical, natural and physical sciences to improve their performance. Metaheuristics can be an efficient way to produce acceptable solutions by trial and error to a complex problem in a reasonably practical time. Metaheuristic algorithms are computational intelligence paradigms especially used for sophisticated solving optimization problems

II.2 Definition

A Meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions. Meta-heuristic algorithms are among these approximate techniques which can be used to solve complex problems. Meta-heuristic algorithms have been proposed by researchers to find optimal or near optimal solutions, The Metaheuristic approaches are not guaranteed to find the optimal solution since they evaluate only a subset of the feasible solutions, but they try to explore different areas in the search space in a smart way to get a near-optimal solution in less cost and time. Notable examples of metaheuristics include genetic/evolutionary algorithms, tabu search, simulated annealing, variable neighborhood search, (adaptive) large neighborhood search, and ant colony optimization, although many more exist. (5)

II.3 Metaheuristic optimization

Metaheuristic optimization is the best approach to optimizing such non-convex functions. These algorithms are the best choice because they make no assumptions about how many hills and valleys the function contains. Although this approach aids in finding minimum and maximum values, these correspond to what are called local maxima (near a high hill) and minima (near a low valley). When using metaheuristic optimization to optimize non-convex functions, therefore, remember that these algorithms do not guarantee finding a global maximum (the highest hill) or minimum (lowest valley) because when optimizing non-convex functions, with many hills and valleys, finding the lowest valley or the highest hill is usually not computationally feasible. Despite this, metaheuristic optimization is often the only recourse for real life applications because convex optimization makes the strong assumption of a single hill and/or valley being present in the function. These algorithms often provide solutions that are near a low enough valley or high enough hills. This process is important for optimizing the output of a machine learning model. For example, if you build a model that predicts cement strength for a cement manufacturer, you can use metaheuristic optimization to find the optimal input values that maximize strength. A model like this takes input values corresponding to ingredient quantities in the cement mixture. The optimizer would then be able to find the quantities for each ingredient that maximizes strength. (6)

II.3.1 Properties

- Metaheuristics are strategies that guide the search process.
- The goal is to efficiently explore the search space in order to find near–optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- Metaheuristics are not problem-specific (7)

II.3.2 Classification

There are a wide variety of metaheuristics and a number of properties with respect to which to classify them.

II.3.2.1 Globalsearch

Other global search metaheuristic that are not local search-based are usually population-based metaheuristics. Such metaheuristics include ant colony optimization, evolutionary computation, particle swarm optimization, genetic algorithm, and rider optimization algorithm.(7)

II.3.2.1.1 Population-based metaheuristics

Population-based metaheuristics find good solutions by iteratively selecting and then combining existing solutions from a set, usually called the population. The most important members of this class are evolutionary algorithms (EA) because they mimick the principles of natural evolution. We use the term evolutionary algorithms as an umbrella term to encompass the wide range of metaheuristics based on evolution. This includes genetic algorithms (GA) genetic/evolutionary programming

- Evolutionary algorithms:

Evolutionary algorithms operate on a set or population of solutions and use two mechanisms to search for good solutions: the selection of predominantly high-quality solutions from the population and the recombination of those solutions into new ones, using specialized operators that combine the attributes of two or more solutions. After recombination, new solutions are reinserted into the population, possibly requiring them to satisfy conditions such as feasibility or minimum quality demands, to replace other (usually low-quality) solutions. Operators used in evolutionary algorithms (selection, recombination and reinsertion) almost without exception make heavy use of randomness. A mutation operator that randomly makes a (small) change to a solution after it has been recombined, is also frequently applied. Most evolutionary algorithms iterate the selection, recombination, mutation, and reinsertion phases a number of times, and report the best solution in the population. Evolutionary algorithms generally require some form of "population management" to ensure that the best solutions survive through the various iterations, while at the same time diversity is maintained in the population (8)

II.3.2.2 Local search

Local search algorithm is the hill climbing method which is used to find local optimums. However, hill climbing does not guarantee finding global optimum solutions. Many metaheuristic ideas were proposed to improve local search heuristic in order to find better solutions. Such metaheuristics include simulated annealing, tabu search, iterated local search, variable neighborhood search, and GRASP. These metaheuristics can both be classified as local search-based or global search metaheuristics.(7)

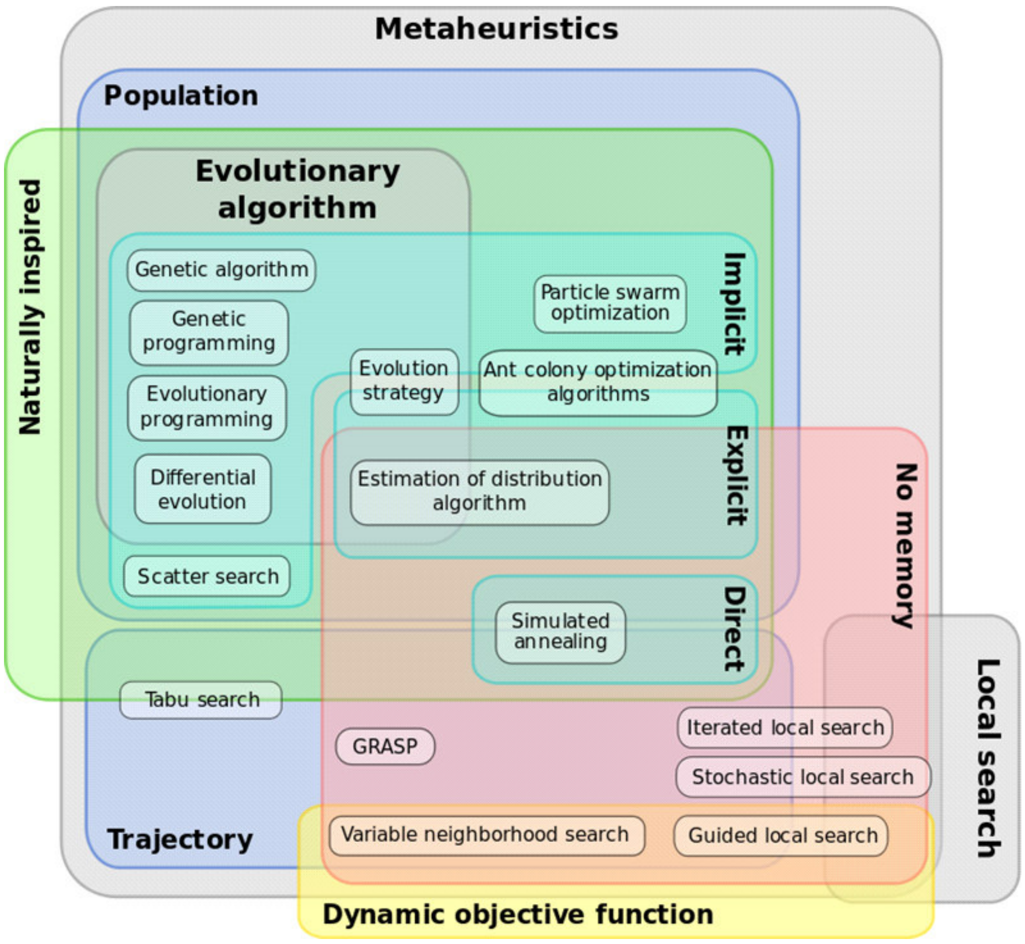


Figure II 1 : classification of metaheuristics

II.3.3 Single-solution vs. population-based

Single solution approaches focus on modifying and improving a single candidate solution; single solution metaheuristics include simulated annealing, iterated local search, variable neighborhood search, and guided local search.

Population-based approaches maintain and improve multiple candidate solutions, often using population characteristics to guide the search; population-based metaheuristics include evolutionary computation, genetic algorithms, and particle swarm optimization.(7)

II.4 Simulatedannealing

II.4.1 Definition

Simulated annealing is a metaheuristic algorithm which processes only a single solution. In order to use this approach, we must first define the neighborhood of a solution. We de-fined the neighborhood of a current solution as all the solutions that can be reached from the current solution by exactly one application ofthe swap operator. Simulated annealing works as follows. In each iteration a solution from the neighbor-hood of the current solution is randomly chosen. If the new solution is better than the cur-rent one, the current solution is replaced by the new one. If not, it is replaced by the new one with a certain probability which is a function of the number of iterations which passed since the beginning of the optimization and of the difference in objective function values between the two considered solutions. Probability of accepting a solution which is worse than the current one decreases as the algorithm proceeds. In the first stages of optimization simulated annealing behaves more like a random search procedure, and thenit's biased toward accepting only better solutions increases and in the final stages of optimization it operates according to the greedy search principle. (9)

II.4.2 Advantages of Simulated Annealing

- Simulated annealing is easy to code and use.
- It does not rely on restrictive properties of the model and hence is versatile.
- It can deal with noisy data and highly non-linear models.
- Provides optimal solution for many problems and is robust. (10)

II.4.3 Disadvantages of Simulated Annealing

A lot of parameters have to be tuned as it is metaheuristic.

The precision of the numbers used in its implementation has a significant effect on the quality of results.

There is a tradeoff between the quality of result and the time taken for the algorithm to run.(10)

II.5 Travelling Salesman Problem

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem, which is simple to state but very difficult to solve. The problem is to find the shortest tour through a set of N vertices so that each vertex is visited exactly once. This problem is known to be NP-hard, and cannot be solved exactly in polynomial time. TSP can be described as: Given a finite set of n cities and a cost matrix that n It indicates the cost (e.g., distance, money, time) to travel from city i to city i. TSP aims to find the perfect tour to visit each one Exactly once and then eventually return to the starting city. We present an improved hybrid genetic algorithm to solve the two-dimensional Euclidean traveling salesman problem (TSP), in which the crossover operator is enhanced with a local search. The proposed algorithm is expected to obtain higher quality solutions within a reasonable computational time for TSP by perfectly integrating GA and the local search. The elitist choice strategy, the local search crossover operator and the double-bridge random mutation are highlighted, to enhance the convergence and the possibility of escaping from the local optima.

The Simulated Annealing (SA) approach can be practically useful to solve the TSP by using the following steps:

- **Step One:** We need to make the opening list of cities by castling the input list (that is make theorder of visit unsystematic).
- **Step Two:** At all iterations, two cities are exchanged in the list. The cost value is the distancetravelled by the Salesperson for all the tours.
- **Step Three:** If the new length (distance) calculated after the modification, is smaller than thecurrent length (distance), it is preserved.
- **Step Four:** If the new length is longer than the current length, it is preserved with a positiveprobability.
- **Step Five:** We need to bring up-to-date the temperature at every iteration by gradually coolingit down(11)

II.6 Conclusion

Metaheuristic algorithms are computational intelligence paradigms especially used for

Sophisticated solving optimization problems. This chapter aims to review of metaheuristics

algorithms and methods .

We present also the method the simulated annealing and how it works and our main problem

the Travelling salesman.

Chapter III

Genetic algorithm

III.1 Introduction

Genetic algorithms are one of the most prominent global search techniques and are widely used for combinatorial problems based on imitation of processes observed during natural evolution. Genetic algorithms offer a wide range of solutions compared to other accurate algorithms. -Many variations of genetic algorithms have been developed and applied to a wide range of optimization problems, from graph coloring to pattern recognition, discrete systems (such as the traveling salesman problem) to continuous systems and from financial markets to multi-objective engineering optimization, Genetic Algorithms have the ability to deliver a “good-enough” solution “fast-enough”. This makes genetic algorithms attractive for use in solving optimization problems

III.2 Genetic algorithm overview

The genetic algorithms (GAs)belong to the family of evolutionary algorithms, developed by John Holland and his collaborators in the 1960s and 1970s (Holland, 1975; De Jong, 1975), is a model or abstraction of biological evolution based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and ‘survival of the fittest", first clearly stated by Charles Darwin in The Origin of Species. By mimicking this process, genetic algorithms are able to “evolve” solutions to real world problems, if they have been suitably encoded(12)Sometime later. Goldberg released a book in which he summarized the various fields of applications of GAs such as research, optimization and machine learning [Goldberg 1989]. This publication marks the beginning of a growing scientific interest in this new optimization technique. Gen and Cheng [1997] then develop a little more the application of these algorithms to optimization problems. In the same period, a C++ library called Ghalib was developed to serve as support for AG-based programming. This library contains a variety of tools designed to solve optimization problems using GAs.(13)

III.3 Definition of Genetic algorithm

Genetic Algorithm is a Meta-heuristic algorithm that aims to find solutions to NP-hard problems. The basic idea of Genetic Algorithms is to first generate an initial population randomly which consist of individual solution to the problem called

Chromosomes, and then evolve this population after a number of iterations called Generations. During each generation, each chromosome is evaluated, using some measure of fitness. To create the next generation, new chromosomes, called offspring, are formed by either merging two chromosomes from current generation using a crossover operator or modifying a chromosome using a mutation operator. A new generation is formed by selection, according to the fitness values, some of the parents and offspring, and rejecting others so as to keep the population size constant. Fitter chromosomes have higher probabilities of being selected. After several generations, the algorithms converge to the best chromosome, which hopefully represents the optimum or suboptimal solution to the problem.(5)

III.4 The reasons to use GAs

Genetic Algorithms have the ability to deliver a "good-enough" solution "fast-enough". This makes genetic algorithms attractive for use in solving optimization problems.

III.4.1 Solving Difficult Problems

In computer science, there is a large set of problems, which are NP-Hard. What this essentially means is that, even the most powerful computing systems take a very long time (even years!) to solve that problem. In such a scenario, GAs proves to be an efficient tool to provide usable near-optimal solutions in a short amount of time

III.4.2 Getting a Good Solution Fast

Some difficult problems like our problem Travelling Salesman Problem (TSP), have real-world applications like path finding and VLSI Design. Now imagine that you are using your GPS Navigation system, and it takes a few minutes (or even a few hours) to compute the "optimal" path from the source to destination. Delay in such real-world applications is not acceptable and therefore a "good-enough" solution, which is delivered "fast" is what is required

III.4.3 Failure of Gradient Based Methods

Traditional calculus-based methods work by starting at a random point and by moving in the direction of the gradient, till we reach the top of the hill. This technique is efficient and works very well for single-peaked objective functions like the cost function in linear regression. But, in most real-world situations, we have a very complex problem called as landscapes, which are made of many peaks and many valleys, which causes such methods to fail, as they suffer from an inherent tendency of getting stuck at the local optima as shown in the following figure**.** (14)

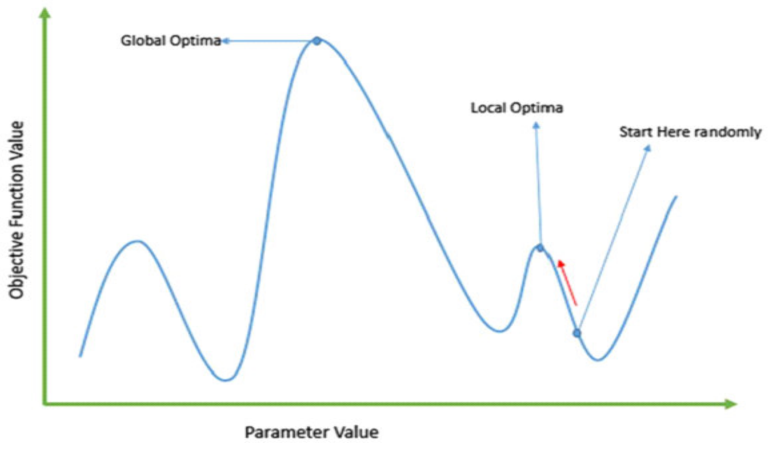


Figure III 1: local and global optima

III.5 Important terms and concepts

Are some of the basic terminologies that can help us to understand genetic algorithms?

- Population: It is a subset of all the possible (encoded) solutions to the given problem.
- Chromosomes: A chromosome is one such solution to the given problem.
- Gene: A gene is one element position of a chromosome.
- Allele: It is the value a gene takes for a particular chromosome.

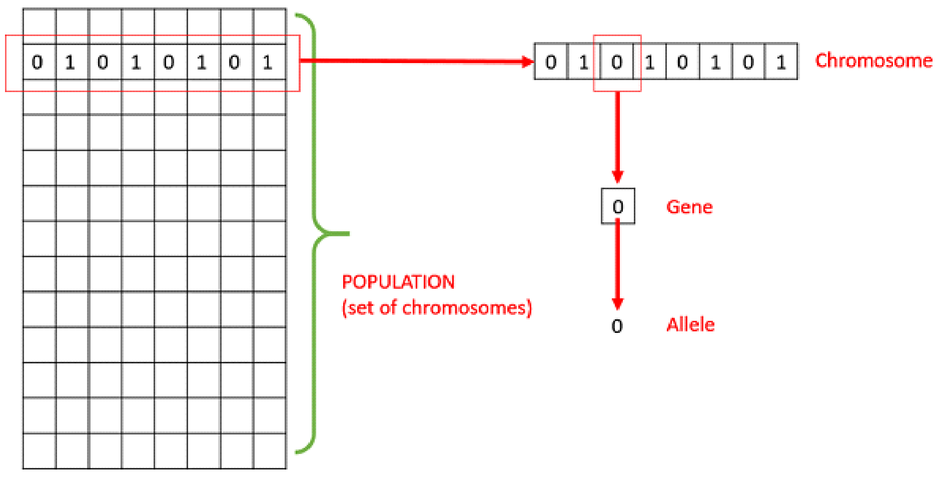


Figure III 2: Terminology of the data structures representing a population of solutions

- Genotype: Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system
- Phenotype: Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations
- Decoding and Encoding: For simple problems, the phenotype and genotype spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation (14)

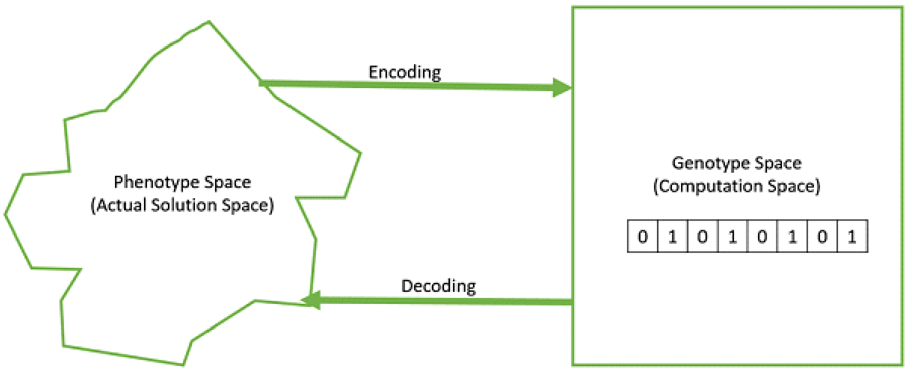


Figure III 3: phenotype and genotype

- Objective function: It is also called a fitness function. Whenever an optimization problem is solved, it is first formulated as a mathematical function that evaluates the quality/fitness of the candidate solution. We usually pass a solution to this function and this function returns the fitness of that solution. Once the fittest/optimal solution is found the process is stopped.
- Genetic operators: genetic operators in genetic algorithms, the best individual's mate to reproduce an offspring that is better than the parents. Genetic operators are used for changing the genetic composition of this next generation. These include crossover, mutation, selection, etc.

III.6 Characteristics of the genetic Search

Broadly speaking, the search performed by a genetic algorithm can be characterized

in the following way

- Genetic algorithms manipulate bit strings or chromosomes encoding useful information about the problem, but they do not manipulate the information as such (no decoding or interpretation).
- Genetic algorithms use the evaluation of a chromosome, as returned by the fitness function, to guide the search. They do not use any other information about the fitness function or the application domain.
- The search is run in parallel from a population of chromosomes.
- The transition from one chromosome to another in the search space is done stochastically(15)

III.7 Principles and Functioning

Since genetic algorithms are designed to simulate a biological process, much of the relevantterminology is borrowed from biology.However, the entities that this terminology refersto in genetic algorithms are much simpler than their biological counterparts [8]. The basiccomponents common to almost all genetic algorithms are:

• A fitness function for optimization

• A population of chromosomes

• Selection of which chromosomes will reproduce

• Crossover to produce next generation of chromosomes

• Random mutation of chromosomes in new generation

Otherwise, the final result is the best chromosome created during the search.

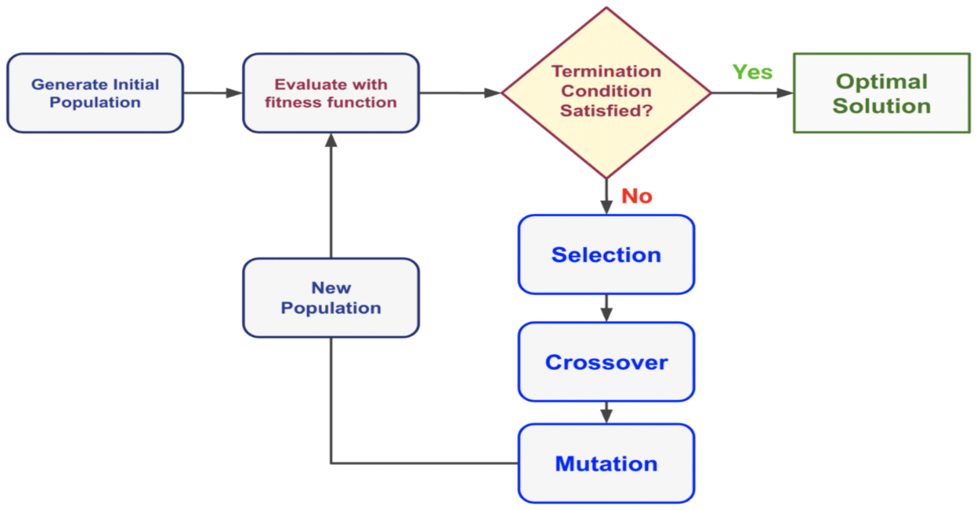

Generate Initial Population
Evaluate with fitness function
Termination Condition Satisfied?
Yes
Optimal Solution
No
Selection
Crossover
Mutation
New Population

Figure III 4: Functioning of GA algorithm

III.7.1 Initialization of Population

A set of "individuals" is called a population, where each individual is characterized by a set of Genes represented in binary (i.e. as 0 or 1). A set of genes represented by a string/sequence is known as a Chromosome. The population with which we start is called the Initial Population.**(16)**

III.7.2 Fitness Function (evaluation)

A Fitness function is a system that determines how fit (the ability of an individual to compete with other individuals) an individual is. It gives a fitness score to each individual which helps quantify the performance**.**This function helps to select the individuals who will be used for reproduction.

A fitness function should possess the following characteristics the fitness function should be sufficiently fast to compute. It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.(16)

III.7.3 Selection

The selection function takes the population and the results of the fitness function to determine who should reproduce. Then selects good chromosomes on the basis of their fitness values and produces a temporary population, namely, the mating pool. This can be achieved by many different schemes, but the most common methods are roulette wheel, ranking, and stochastic binary tournament selection. The selection operator is responsible for the convergence of the algorithm.

III.7.3.1 Roulette wheel Selection

The roulette wheel selection (also known as fitness proportionate selection) is a function used by genetic algorithms for selecting potentially useful solutions for recombination.

The crossover individual probability is computed based on the individual's fitness divided by the sum of all population fitness. The followingis the formula for it:

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$$

Where pi is the probability of each chromosome equals the chromosome frequency divided by the sum of all fitness.(17)

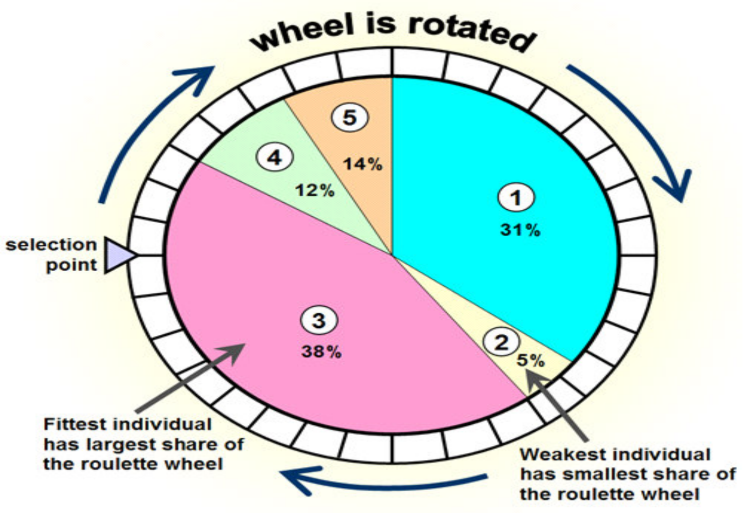

wheel is rotated
selection point
5
14%
4
12%
1
31%
3
38%
2
5%
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

Figure III 5: Roulette wheel selection method

III.7.3.2 Rank selection

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

After this all the chromosomes have a chance to be selected. Rank-based selection schemes can avoid premature convergence. But can be computationally expensive because it sorts the populations based on fitness value. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.(18)

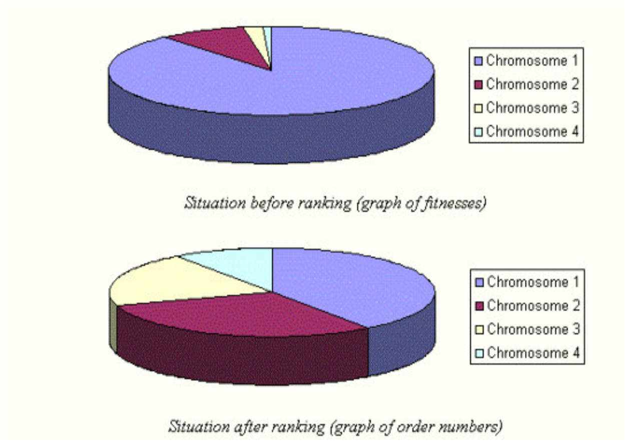


Figure III 6: Rank selection

III.7.4 Reproduction

Generation of offspring happen in 2 ways**:**

III.7.4.1 Crossover Operators

The process of mixing the genes of the pair of individuals chosen to produce a new pair of individuals is called Crossover or Genetic operation. This process is continued to create a new population. Crossover can be performed in different methods

There are 3 major types of crossovers:

III.7.4.1.1One Point Crossover

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs

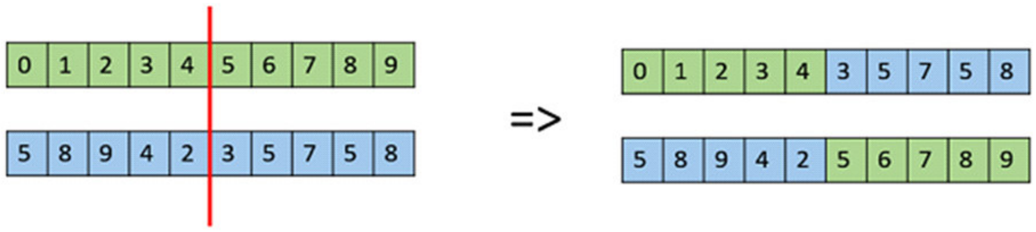


Figure III 7: One Point Crossover

III.7.4.1.2Multi Point Crossover

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.

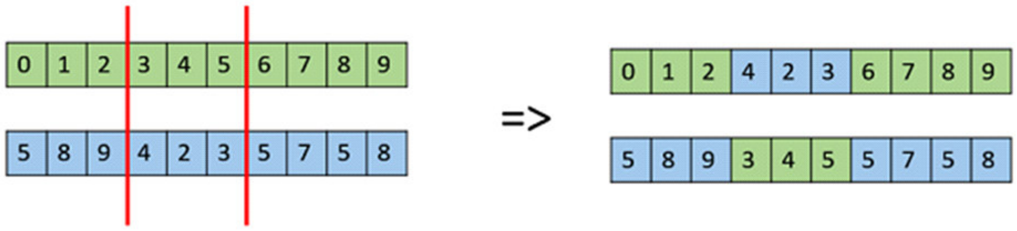


Figure III 8: Multi Point Crossover

III.7.4.1.3Uniform Crossover

In a uniform crossover, we don't divide the chromosome into segments rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent.(14)

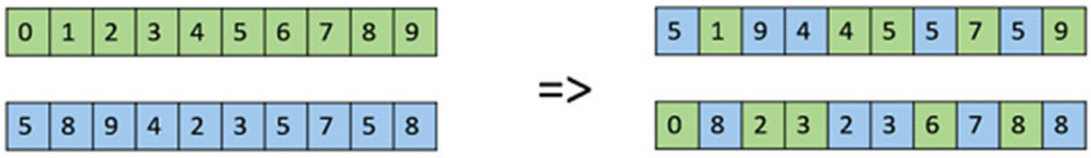


Figure III 9: Uniform Crossover

III.7.4.2 Mutation operators:

This operator adds new genetic information to the new child population. This is achieved by flipping some bits in the chromosome. Mutation solves the problem of local minimum and enhances diversification. The following image shows how mutation is done. (19)There is some of the most commonly used mutation operator:

III.7.4.2.1Bit Flip Mutation

In this bit flip mutation, we select one or more random bits and flip them. This isused for binaryencodedGAs.

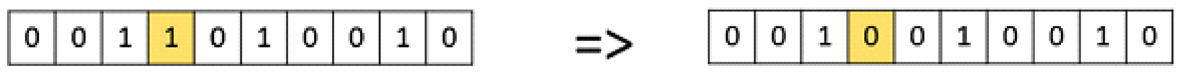


Figure III 10: Bit Flip Mutation

III.7.4.2.2Random Resetting

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

III.7.4.2.3Swap Mutation

In swap mutation, we select two positions on the chromosome at random, and interchange the values. This iscommon in permutation-basedencodings.

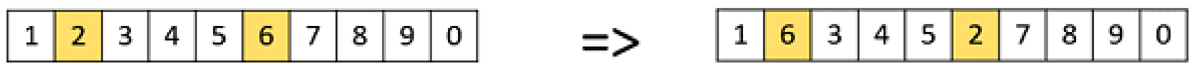


Figure III 11: Swap Mutation

III.7.4.2.4Scramble Mutation

Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.

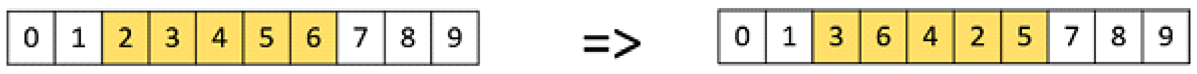


Figure III 12: Scramble Mutation

III.7.4.2.5Inversion Mutation

In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.(14)

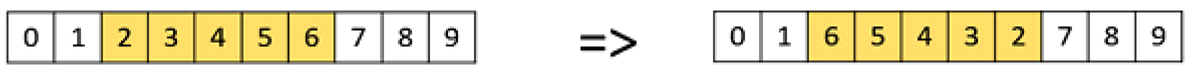


Figure III 13: Inversion Mutation

III.7.5 Termination (Convergence):

The algorithm terminates when a population converges. Convergence here denotes that the individuals no longer have significant difference in their genetic structure. Termination can also occur after a set number of cycles; this normally leads to multiple convergence points.

III.8 Application areas of GA:

Genetic algorithms have a variety of applications, and one of the basic applications of genetic algorithms can be the optimization of problems and solutions. We use optimization for finding the best solution to any problem. Optimization using genetic algorithms can be considered genetic optimization, and there are several benefits of performing optimization using genetic algorithms.

Genetic algorithms are applied in the following fields:

III.8.1 Neural networks:

Neural networks in machine learning are one of the biggest areas where genetic algorithms have been used for optimization. One of the simplest examples of use cases of genetic optimization in neural networks is finding the best fit set of parameters for a neural network. Instead of these, we can find the use of genetic algorithms in neural network pipeline optimization, inheriting qualities of neurons, etc.(20)

III.8.2 Medical science:

In medical science, we can find many examples of use cases of genetic optimization. The generation of a drug to diagnose any disease in the body can have the application of genetic algorithms. In various examples, we find the use of genetic optimization in predictive analysis like RNA structure prediction, operon prediction, and protein prediction, etc. also there are some use cases of genetic optimization in process alignment such as Bioinformatics Multiple Sequence Alignment, Gene expression profiling analysis, Protein folding, etc**.**

III.8.3 Mechanical engineering design:

In many designing procedures of mechanical components, we can also find the application of genetic optimization. They are used to develop parametric aircraft designs. The parameters of the aircraft are modified and upgraded to provide better designs. We can take aircraft wing design as an example where we are required to improve the ratio of lift to drag for a complex wing. This kind of designing problem can be considered as a multidisciplinary problem, the fitness function in genetic optimization can be altered by considering some specific requirement of the design.(20)

III.8.4 Image processing:

There are various works and researches which show the use cases of genetic optimization in various image processing tasks. One of the major tasks related to genetic approach in image processing is image segmentation.Although these genetic optimizations can be utilized in various areas of image analysis to solve complex optimization problems. Using genetic optimization in an integrated manner with image segmentation techniques can make the whole procedure an optimization problem.

III.8.5 Wireless sensor networks:

The wireless sensor network is a network that includes spatially dispersed and dedicated centers to maintain the records about the physical conditions of the environment and pass the record to a central storage system. Some notable parameters are the lifetime of the network and energy consumption for routing which plays key roles in every application. Using the genetic algorithms in WSN we can simulate the sensors and also a fitness function from GA can be used to optimize, and customize all the operational stages of WSNs. (20)

III.8.6 Transport

Genetic algorithms are used in the traveling salesman. This is one of the most common combinatorial optimization problems in real life that can be solved using genetic optimization. The main motive of this problem is to find an optimal way to be covered by the salesman, in a given map with the routes and distance between two points. If genetic algorithms are used in finding the best route structure, we don’t get the solution only once after each iteration, we can generate offspring solutions that can inherit the qualities of parent solutions. TSP has a variety of applications like planning, logistics, and manufacturing.

III.9 Advantages of Gas

GAs has various advantages which have made them immensely popular:

- Is faster and more efficient as compared to the traditional methods.
- Have very good parallel capabilities.
- It can optimize various problems such as discrete functions, multi-objective problems, and continuous functions
- Always gets an answer to the problem, which gets better over the time.
- A genetic algorithm does not need derivative information.
- Useful when the search space is very large and there are a large number of parameters involved. (19)

III.10 Limitations of Gas

Like any technique, GAs also suffers from a few limitations.Theseinclude:

- GA is not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.

III.11 Mathematical formulation

The traveling salesman problem consists in finding the shortest path connecting n given points and passing once and only once by each point and it returns to the starting point. The dot can represent a city, a country, or a warehouse, etc. For a set of n points, there exists in total $n!$ possible paths. The starting point does not change the length of the path, we can choose it arbitrarily, so we have ($-$ 1)! different paths. Finally, each path can be traversed in both directions and the two possibilities have the same length, so we can divide this number by two. For example, if we call the points, a, b, c, d, the paths $abcd$, $bcda$, $cdab$, $dabc$ $adcb$, $dcba$, $cbad$, $badc$ all have the same length, only the starting point and the direction of the course changes. So we have (-1)! 2 candidate paths to consider.

Let $G = (V, A)$ be a graph such that $V = \{ 1, \dots , v\ n\}$ the set of vertices, $n = |V|$ the number of vertices of the graph G, each vertex modeling a city where vi characterized by the coordinates x and y and d the distance matrix of size ($n \times n$), such that $d(i, j)$ defines the distance between the two vertices vi and vj. (G) The set of arcs, models the cost of travel between two cities.

The objective is to find a tour of the minimum total length, the length of which is the sum of the costs of each arc in the tour.

The search space for the TSP is a set of permutations of n cities and the optimal solution is a permutation that gives the minimum cost of the tour.

The concept of calculating distances between cities $i\ et$ $(i + 1)$ is calculated under the Euclidean distance rule using the following equation:

$d(T[i], T[i + 1]) = \sqrt{}((xi - xi$+1)2 + ($yi - yi$+1)2)

Such that, $[i]$ is a permutation on the set $\{1, 2, . . . , n\}$

$T = (T[1], T[2] , T[3], \dots , T[n], T[1])$

Then, f the objective function to calculate the cost of each solution of the problem given by the following formula:

$f = d(T[n], T[1]) + \sum d(T[i], T[i + 1])\ n{-}1\ i{=}1$

From (1), (2) et (3) the simple mathematical formula of the traveling salesman problem is given by:

$Min \{ \ (T) , T = (T[1], T[2], T[3] \ldots , T[n], T[1])\}$

III.12 Conclusion

The GA is a probabilistic solution to optimize the problems that are modeled on a genetic evaluation process in biologically and are focused as an effective algorithm to finda global optimum solution for many types of problems. The GA is used in different

Artificial intelligence applications like object-oriented systems, robotics, and futuristicemerging technologies.

We began this chapter by introducing what a genetic algorithm is, which have the ability to deliver a “good-enough” solution “fast-enough” and how it works, We also mentioned the basic components of this algorithm,and the application areas of GA which is TSP and the adventages then the inconvinients of the GA.

Chapter IV

Implementation

IV.1 Introduction

The objective of this chapter is to present the tools (software, languages, libraries and data used in my system).in our work is based on the comparison of two programs that aim to solve the problem of salesman. In the first program we have integrated genetic algorithm to solve the problem. The second one has integrated Meta heuristic algorithm giving each of these two programs a result with the aim of having better traffic with a reduced distance.

IV.2 Development environment and tools

IV.2.1 Google Colaboratory

Colaboratory, shortened to "Colab", allows us to write and run Python code in your browser. Colaboratory is a Google research project created to help spread machine learning education and research. It is a Jupyter notebook environment that requires no configuration to use and runs entirely on the cloud.

It offers the following advantages:

- No configuration required
- Free access to GPUs
- Easy sharing

Figure IV 1: Logo Google Colab

IV.2.2 Language used

In our project we used the Python language.

IV.2.2.1 Python

Is an interpreted, cross-paradigm, cross-platform programming language. It promotes structured, functional and object-oriented imperative programming. It has strong dynamic typing, automatic memory management by garbage collection and an exception handling system; it is thus similar to Perl, Ruby, Scheme, Small talk and Tcl. The Python language is placed under a free license close to the BSD 5 license and works on most computer platforms, from smartphones to central computers, from Windows to Unix with in particular GNU/Linux via macOS, or even Android, iOS, and can also be translated into Java or .NET. It is designed to optimize programmer productivity by offering high-level tools and an easy-to-use syntax.

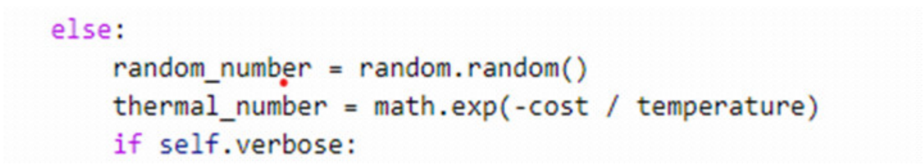
Figure IV 2: logo of python

IV.2.3 Library uses:

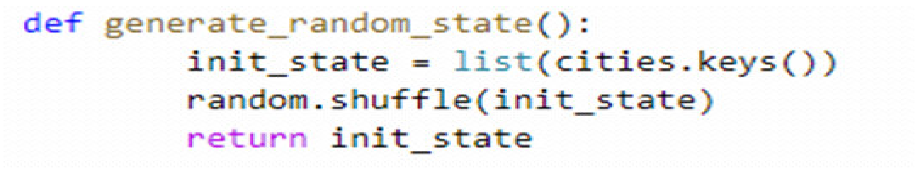
IV.2.3.1 Random

This module implements pseudo-random number generators for different distributions. Forintegers; there is a uniform selection from a range. For sequences, there is a uniform selection of a random element, a function to generate a random permutation of a list in place, and a function for random sampling without replacement.

Some uses of random :

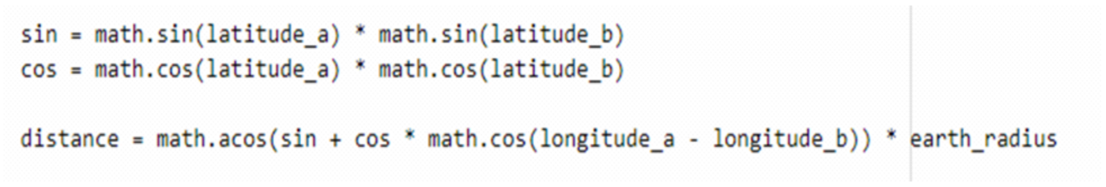


```
else:
    random_number = random.random()
    thermal_number = math.exp(-cost / temperature)
    if self.verbose:
```



```
def generate_random_state():
        init_state = list(cities.keys())
        random.shuffle(init_state)
        return init_state
```

Math,it provides access to math functions arithmetic and logarithmic and exponential…etc. Whatever uses the math library:



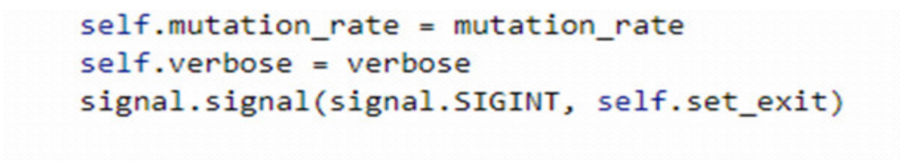
```
sin = math.sin(latitude_a) * math.sin(latitude_b)
cos = math.cos(latitude_a) * math.cos(latitude_b)

distance = math.acos(sin + cos * math.cos(longitude_a - longitude_b)) * earth_radius
```

IV.2.3.2 Signal

The signal.signal () function allows defining custom handlers to be executed when a signal is received. A small number of default handlers are installed: SIGPIPE is ignored (so write errors on pipes and sockets can be reported as ordinary Python exceptions) and SIGINT is translated into a KeyboardInterrupt exception if the parent process has not changed it.

A handler for a particular signal, once set, remains installed until it is explicitly reset (Python emulates the BSD style interface regardless of the underlying implementation), with the exception of the handler for SIGCHLD, which follows the underlying implementation.

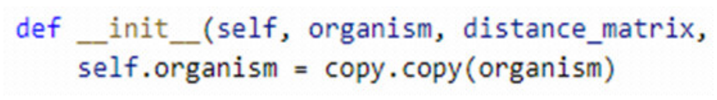


```
self.mutation_rate = mutation_rate
self.verbose = verbose
signal.signal(signal.SIGINT, self.set_exit)
```

IV.2.3.3 Copy

Assignment statements in Python do not copy objects, they create links between the target and the object. For collections that are mutable or contain mutable elements, sometimes a copy is needed, so that you can modify one copy without modifying the other.

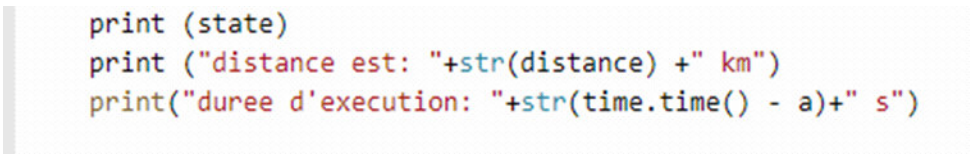
Whatever use



```
def __init__(self, organism, distance_matrix,
    self.organism = copy.copy(organism)
```

IV.2.3.4 Time

This module provides various time-related functions. We used the library for the function time.time () which returns the time in seconds since epoch as a floating point number. The specific epoch date and treatment of leap seconds depends on the platform.



```
print (state)
print ("distance est: "+str(distance) +" km")
print("duree d'execution: "+str(time.time() - a)+" s")
```

IV.3 Program architecture

IV.3.1 Genetic algorithm :

The first is a population-based approach that deals with several solutions at once. They maintain and improve several solutions, and choose the best result.

The methodconsists of:

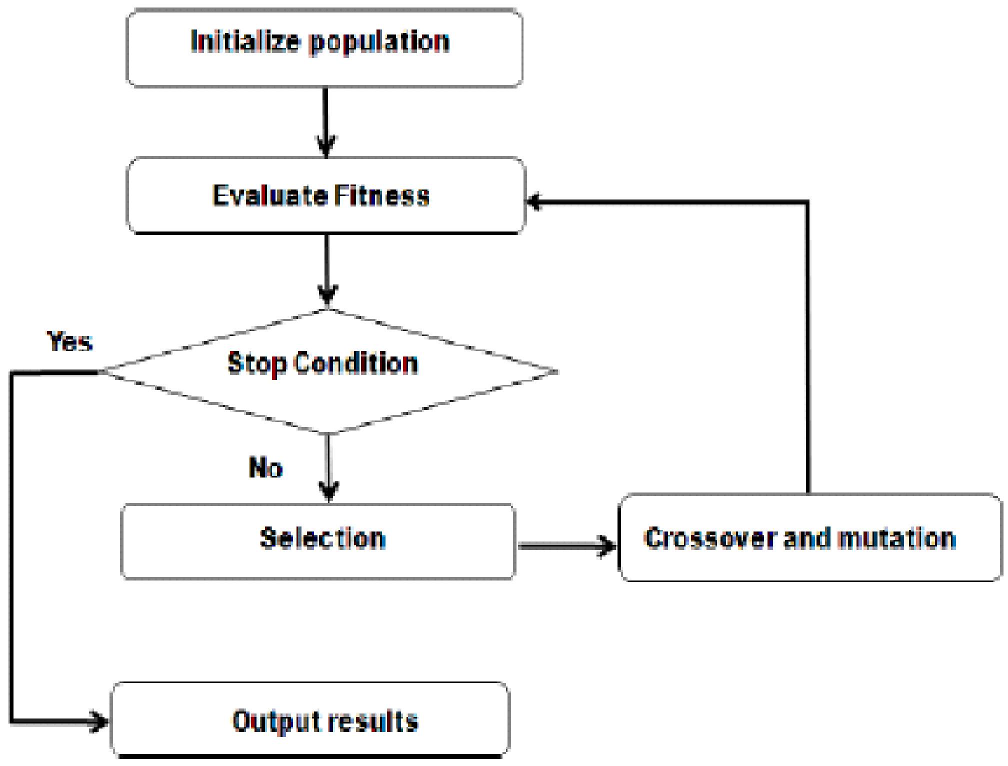


Figure IV 3: algorithm of gentic algorithm

• Creation of an initial population

• Evaluation of the "fitness" (quality) of individuals

• Selection of parents

• Creation of new individuals: crossover + mutation

• Addition of new individuals in the population

• Return to the second step the parameters chosen for the resolution of the TSP with AG

• After having the number of cities and population.

• A chromosome of N genes a solution = path taken: coding used is a real coding (each gene of the chromosome=city number).

• Evaluation function: minimization of the distance traveled.

• Crossover operator.

• Mutation operator.

• Stop test number of generations.

• Last adds an item to the end of the list or array.

IV.3.2 Simulated annealing algorithm:

The second method is based on local search.

It consists of:

• Simulated annealing is an iterative algorithm that uses the Metropolis criterion exp (-ΔE/T)

• We have 2 nested loops.

• The first loop to lower the temperature (thermodynamic stability): change the value of the temperature whose goal is to converge towards the global optimum (global minima for the case of the minimization problem).

• The second loop to choose the best neighbor of the current solution, using the Metropolis criterion.

• The stopping criterion will depend on the time (the number of iterations) and the degradation of the solution (Δf), evaluated as the difference of the cost function of the previous solution and of the new selected solution.

• At each degradation, the algorithm stops with a probability depending on this degradation and on the time (number of iterations) of the algorithm.

• The greater the degradation, the lower the probability of continuing.

• The higher this number, the lower the probability of continuing.

• PThe temperature makes it possible to control the acceptance of the solutions or not by the calculation of the criterion of Métropolis exp (-Δf/T).

• The value of the temperature parameter varies during the search for iterations.

• At the beginning Temperature is large.

• It gradually decreases to reach the value 0.

• Test of the probability of accepting the solution or not.

• Choose a Stop Test. arameters of the simulated annealing algorithm

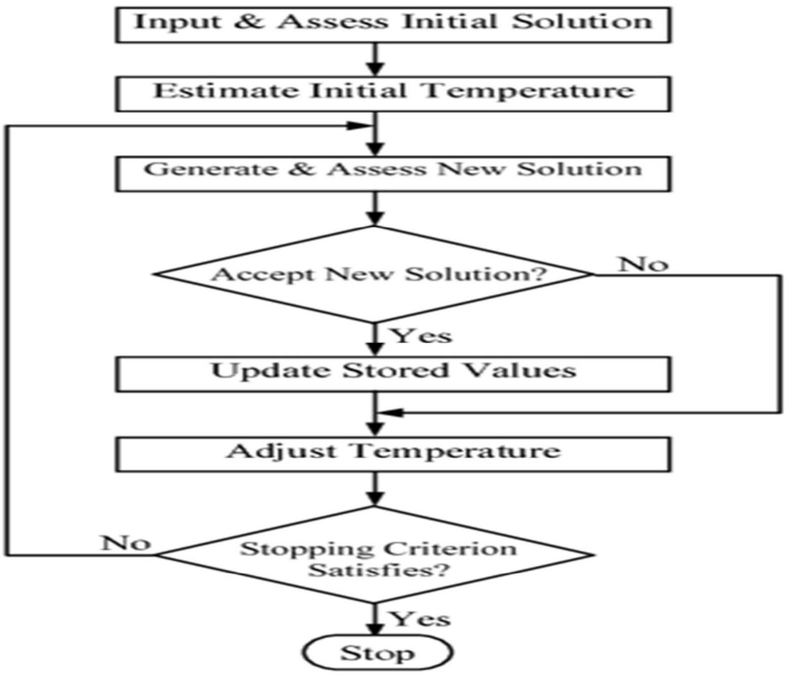

Input & Assess Initial Solution
Estimate Initial Temperature
Generate & Assess New Solution
Accept New Solution?
No
Yes
Update Stored Values
Adjust Temperature
No
Stopping Criterion Satisfies?
Yes
Stop

Figure IV 4: simulated annealing algorithme

IV.3.3 Data declaration phase:

This phase consists of;

• Declaration the list of cities that are defined as the capture, longitudes and latitudes are from Google Maps.

• Calculates distance for each 2 city.

• Selected distances.

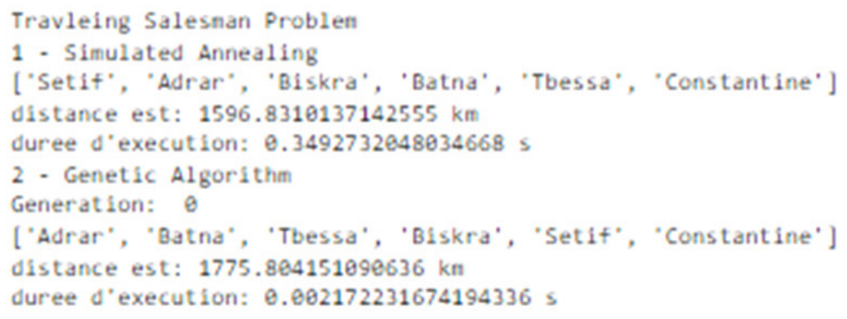
IV.4 Final phase (execution and problem result)

This phase consists of:

• Import time for calculation of execution time and comparison of the two phases.

• Import of methods and display of results.

IV.5 Results :



```
Travleing Salesman Problem
1 - Simulated Annealing
['Setif', 'Adrar', 'Biskra', 'Batna', 'Tbessa', 'Constantine']
distance est: 1596.8310137142555 km
duree d'execution: 0.3492732048034668 s
2 - Genetic Algorithm
Generation:  0
['Adrar', 'Batna', 'Tbessa', 'Biskra', 'Setif', 'Constantine']
distance est: 1775.804151090636 km
duree d'execution: 0.00217223167419433б s
```

Capture shows the final display obtained that we see:

• The list of cities is ordered for each method.

• Distance from traffic.

• The duration of execution.

From the summary results we conclude that simulated annealing

IV.6 Conclusion

In this chapter we present language python and the google collabotoryenvironment where we had implemented the execution of TSP with the genetic algorithm and the SA method.

We found that the GA is the faster but the SA is the optimum.

General Conclusion

General conclusion:

In this thesis, we studied a combinatorial optimization problem: the problem of traveling salesman (TSP) we solved this problem with using two algorithms of metaheuristic (genetic algorithms, simulated annealing) we have compared the results of this two to know who is faster and optima and inorder to get better results.

We started by studying the basic concepts of machine learning and its types then we presented Optimization algorithms.

We also presented metaheuristic algorithms, its classification, applications ...etc.

After looking at the problem and solving it using the genetic algorithm andthe simulated annealing in the fourth chapter, wefound that simulated annealing is better than genetic algorithm.

For future work, we suggest that new students try to find new models to solve the problem of the traveling salesman, so that they are easy and fast and give better results than the models mentioned in this thesis.

References

1. Mahesh, Batta. Machine Learning Algorithms - A Review. (2018).

2. What Is Machine Learning (ML)? s.l. : Berkeley School of Information, June 26, 2020.

3. Machine Learning. s.l. : IBM Cloud Education, 15 jan 2020.

4. Brownlee, Jason. machine learning mastery. December 23, 2020.

5. Gamal Abd El-Nasser A. Said, Abeer M. Mahmoud,El-Sayed M. El-Horbaty. A Comparative Study of Meta-heuristic Algorithms. s.l. : International Journal of Advanced Computer Science and Applications, 1, 2014.

6. A guide to metaheuristic optimization for machine learning models in python . pierre, sadrach. s.l. : buit in , 2022.

7. Christian blum, Andrea Roli . Metaheuristics in Combinatorial Optimization: Overviewand Conceptual Comparison. s.l. : ACM Computing Surveys, 2003.

8. Metaheuristics. s.l. : Scholarpedia, 2015.

9. (https://www.researchgate.net/publication/312889331_Choice_of_best_possible_metaheuristic_algorithm_for_the_travelling_salesman_problem_with_limited_computational_time_Quality_uncertainty_and_speed). Marek Antosiewicz, Grzegorz Koloch, Bogumił Kamiński. s.l. : Journal of Theoretical and Applied Computer Science, January 2013.

10. Simulated Annealing. Babu, Sanjana. s.l. : OpenGenus IQ © 2022 All rights reserved .

11. . Solution to the travelling salesperson problem using simulated annealing algorithm .M.A Rufai ,R.M. Alabison .A.Abidemi and E.J.Dansu . s.l. : Electronic Journal of Mathematical Analysis and Applications, 2017.

12. An Overview of Genetic Algorithms. David Beasley, David R. Bully, Ralph R. Martin.

13. / A closer look at AI: Genetic algorithms January . s.l. : brightrion , 5th, 2017.

14. Genetic Algorithms - Quick Guide. s.l. : tutorialspoint., 2022.

15. Genetic algorithms for the traveling salesman problem. Potvin, Jean-Yves. s.l. : Centre de Recherche sur les Transports,Universitd de Montrgal.

16. Reinforcement Learning vs Genetic Algorithm — AI for Simulations. Neelarghya. jul 26, 2021.

17. Roulette wheel selection in paython s.l. : rocreguant.

18. How to perform rank based selection in a genetic algorithm? s.l. : Stack Overflow .

19. The Basics of Genetic Algorithms in Machine Learning. Muthee, Arthur. s.l. : EngEd Community, 26 may 2021.

20. 10 real-life applications of Genetic Optimization. Verma, Yugesh. s.l. : developers corner , JANUARY 21, 2022.

21. Gamal Abd El-Nasser A. Said, Abeer M. Mahmoud,El-Sayed M. El-Horbaty. A Comparative Study of Meta-heuristic Algorithmsfor Solving Quadratic Assignment Problem. s.l. : International Journal of Advanced Computer Science and Application, 2014.

22. Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed . Marek Antosiewicz, Grzegorz Koloch, Bogumił Kamiński. s.l. : Journal of Theoretical and Applied Computer Science , 2013.

23. simulated annealing applied to the traveling salesman problem . P.Karlsson. 2002.