People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Chadli Bendjedid University
Faculty of Science and Technology
IT department

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم.

Master Thesis in Computer Science

Presented by

AHLEM MELLOUK

Specialty: Intelligent Computer Systems

Theme

AN EVOLUTIONARY APPROACH TO SEARCHING THE SHORTEST PATH FOR THE MOBILE ROBOT

Defence on: October 2020

Name and First Name	**Grade**		
Dr Ferdenache A	MCB	Univ.of el tarf	President
Dr Benmachiche a	MCB	Univ.of el tarf	Supervisor
Dr Maatallah M	MCB	Univ.of el tarf	Examiner

University Year: 2019/2020

Thanks

My thanks go first to God Almighty for the will, health and patience he gave me during all these years of study.

In particular, I would like to thank Mr. Benmachiche. A, Senior Lecturer in the Department of Computer at Chadli Ben Djedid University, who has mentored me throughout this thesis, for his valuable advice and support. I am grateful for his availability and the trust he has placed in me. My gratitude also to the team of teachers of the Option, **Artificial** Intelligence.

I sincerely thank:

-Dr-ferdenache A, for having honored me to chair the jury

-Dr-maatallah M, for agreeing to be an examiner.

A big thank you to all my family for their concern and concern for me, their encouragement and followed them with patience during the course of my studies. My thanks also go to my colleagues, for the good times we passed together. Finally, I thank all those who helped me from afar or from the meadows.

Thank you all, from the bottom of the heart

Dedications

I dedicate this work:

To my Mother,

To My Brothersand my Sister

To the whole family

And to All my friends

Ahlem

Abstract

For fifteen years, the design of artificial systems has experienced an epistemological revolution. It consisted in overcoming the classic opposition between the organic and inorganic worlds by seeking to endow machines with the capacities of adaptation and autonomy specific to living systems. Evolutionary robotics aims to design machines capable of continuously learning new skills in a continuous, uncontrolled and changing world. This method made it possible to successfully build real robots exhibiting complex reactive behaviors. Based on this observation,

Our work offers a non-exhaustive vision of the research themes associated with the field of mobile robotics, and presents the scientific obstacles that remain to be lifted to lead to the development of an autonomous robot. The autonomy of the latter requires the coordinated achievement of tasks of control and perception of the environment.

Among these, navigation plays a fundamental role in the interaction of the robot with its evolutionary environment. It consts of the determination of trajectories achievable by the robot to follow a pre-established path, while bypassing mobile or fixed obstacles.

To perform this task, our approach is based on the shortest path genetic algorithms. The navigation problem is then modeled in the form of a constraint optimization problem whose fitness function quantifies the difference between the best robot path and the other random paths.

The obstacles are integrated in the form of constraints by penalizing the movement of the robots; the goal being to allow these robots to change position while avoiding obstacles.

Our approach has been implemented and several scenarios have been tested. The results obtained demonstrate the robustness of the method deployed as well as its performance.

Résumé

Depuis quinze ans, la conception de systèmes artificiels a connu une révolution épistémologique. Elle a consisté à surmonter l'opposition classique entre les mondes organique et inorganique en cherchant à doter les machines des capacités d'adaptation et d'autonomie propres aux systèmes vivants.

La robotique évolutive vise à concevoir des machines capables d'apprendre continuellement de nouvelles compétences dans un monde continu, incontrôlé et changeant. Cette méthode a permis de construire avec succès de vrais robots présentant des comportements réactifs complexes. Sur la base de cette observation, nos travaux offrent une vision non exhaustive des thématiques de recherche associées au domaine de la robotique mobile, et présentent les obstacles scientifiques qui restent à lever pour aboutir au développement d'un robot autonome. L'autonomie de ce dernier nécessite la réalisation coordonnée de tâches de contrôle et de perception de l'environnement.

Parmi celles-ci, la navigation joue un rôle fondamental dans l'interaction du robot avec son environnement évolutif.

Il s'agit de la détermination des trajectoires réalisables par le robot pour suivre une trajectoire préétablie, tout en contournant les obstacles mobiles ou fixes.

Pour effectuer cette tâche, notre approche est basée sur les algorithmes génétiques de plus court chemin.

Le problème de navigation est ensuite modélisé sous la forme d'un problème d'optimisation des contraintes dont la fonction de fitness quantifie la différence entre le meilleur chemin du robot et les autres chemins aléatoires.

Les obstacles sont intégrés sous forme de contraintes en pénalisant le mouvement des robots; l'objectif étant de permettre à ces robots de changer de position tout en évitant les obstacles.

Notre approche a été mise en œuvre et plusieurs scénarios ont été testés.

Les résultats obtenus démontrent la robustesse de la méthode déployée ainsi que ses performances.

ملخص

لمدة خمسة عشر عامًا، شهد تصميم النظم الاصطناعية ثورة معرفية، كان يتألف من التغلب على المعارضة الكلاسيكية بين العالمين العضوي وغير العضوي من خلال السعي إلى منح الآلات قدرات التكيف والاستقلالية الخاصة بالنظم الحية. تهدف الروبوتات التطورية إلى تصميم آلات قادرة على التعلم المستمر لمهارات جديدة في عالم مستمر وغير متحكم فيه ومتغير. نجحت هذه الطريقة في بناء روبوتات حقيقية بسلوكيات تفاعلية معقدة.

بناء على هذه الملاحظة ،

يقدم عملنا رؤية غير حصرية لمواضيع البحث المرتبطة بمجال الروبوتات المتنقلة ، ويعرض العقبات العلمية التي لا يزال يتعين التغلب عليها لتؤدي إلى تطوير روبوت مستقل.

تتطلب استقلالية الأخيرة تحقيقًا منسقًا لمهام التحكم وإدراك البيئة.

من بينها ، يلعب التنقل دورًا أساسيًا في تفاعل الروبوت مع بيئته المتطورة.

يتضمن هذا، تحديد المسارات التي يمكن أن يقوم بها الروبوت لمتابعة مسار محدد مسبقًا ، مع تجاوز العوائق المتحركة أو الثابتة.

لأداء هذه المهمة ، يعتمد نهجنا على أقصر الخوارزميات الجينية.

ثم تُصاغ مشكلة التنقل في شكل مشكلة تحسين معوقات تقيس وظيفتها الملائمة الفرق بين أفضل مسار للروبوت والمسارات العشوائية الأخرى.

يتم دمج العقبات في شكل قيود من خلال مراقبة حركة الروبوتات. الهدف هو السماح لهذه الروبوتات بتغيير موقفها مع تجنب العقبات.

تم تنفيذ نهجنا وتم اختبار العديد من السيناريوهات.

أظهرت النتائج التي تم الحصول عليها متانة الطريقة المستخدمة بالإضافة إلى أدائها.

Contents

Chapter 3: Realization and Implementation

Table of Figures

List of tables

Abreviations list

- **AI** : Artificial intelligence
- **APF**:Artificial Potential Fields
- **GA** : Genetic Algorrithm
- **VBA** : Virtual Bee Algorithm
- **BCO** : Bee Colony Optimization
- **BOD** : Dance Bee Optimization
- **ABC** : Artificial Bee Colony
- **RCGA** : Real Coded Genetic Algorithms

General Introduction

Artificial Intelligence (AI) is a branch of computer science where machines are trained and engineered to mimic the human decisive and reactive functions without human intervention.
AI is the implementation of a number of techniques aimed at allowing machines to imitate a form of real intelligence. The need to use artificial intelligence is increasing in all areas.

In 1950; Mathematician Alan Turing developed a test to measure artificial intelligence. The Turing test, thanks to a series of questions, determines whether the answer can be determined by the machine in this way.If the computer's responses are indistinguishable from those of humans, the computer is considered to be artificially intelligent.

The purpose of artificial intelligence is to design systems capable of reproducing the behavior of humans in their reasoning activities. AI sets as its goal the modeling of intelligence taken as a phenomenon (as well as physics, chemistry or biology which aim to model other phenomena).
The scientific field that specializes in studying, designing and implementing "smart machines" is called artificial intelligence.It should first be remembered that the word "machine" does not designate a physical object but rather an automatic system capable of processing information.
With the advancement of computer science and technology, a wider area of research has opened in the area of robotics, which is devoted to machines that perform movements in space. So when we talk about "robot" in artificial intelligence we are referring to a computer program showing some form of intelligence.

Robotics is an important sub-domain of AI; robotics can be seen as an intelligence interconnection of perception, action, as well as the functioning of robots.

Used to maintain dynamic representations of their environment, it allows robots to acquire the ability to communicate in natural language.

Robotics is a very good example of a multidisciplinary field that involves many themes such as mechanics, mechatronics, electronics, automation, computer science or artificial intelligence.

Depending on the authors' area of origin, there are therefore various definitions of the term robot, but they generally revolve around this one: A robot is a machine equipped with capacities of perception, decision and action which allow it to act independently in his environment according to his perception of it.

The navigation strategies allowing a mobile robot to move to reach a goal are extremely diverse, as are the classifications that can be made of them.

In this framework, we call movement planning the problem of the preliminary calculation of the movements necessary for a robot to accomplish a given task.

In its most general form, movement planning is defined in the following way: given a model of the robotic system and its environment, planning a movement consists in calculating the movement that the system must make to reach an objective set a priori.

Movement planning, also called autonomous navigation is defined as movement planning focusing on the specific case of robots or mobile agents.

The problem addressed in this document is that of "autonomous navigation in a dynamic environment".Solving this problem, by exact methods ([1], [2], [3]), is not applicable to complex systems with many degrees of freedom.

The effective methods of motion planning are based on probabilistic algorithms that randomly explore the space of configurations in order to characterize their connectedness. We find a synthesis of these techniques in [4], or in more recent references [5], [6].

One of its main contributions was to design a special algorithmic framework, which allows us to solve problems: called metaheuristics, a meta-heuristic is a generic method for solving combinatorial problems.

Genetic algorithms use these mechanisms to define a metaheuristic for solving combinatorial optimization problems.

The idea is to develop a population of combinations, by selection, crossing, mutation, and the adaptability of a combination being evaluated here by the objective function to be optimized.

Genetic algorithms are evolutionary methods that draw heavily on biological mechanisms linked to the principles of selection and natural evolution.

Originally developed by Holland [7] to meet specific needs in biology, genetic algorithms were quickly adapted to a wide variety of contexts.

In a simple genetic algorithm [8], the search is regulated by three operators which are applied successively. The cooperation phase is governed by a reproduction operator and a combination operator (or “crossover”) while the individual adaptation phase calls for a mutation operator. It is important to emphasize that the concepts that underlie genetic algorithms are extremely simple.

Indeed, they only involve randomly generated numbers and a set of very general probabilistic rules which do not necessarily take into account all the particularities of the problem treated.

In our work, we are interested in certain metaheuristics (genetic algorithms [8]). For this we have proposed algorithms based on these techniquefor solving the problem of finding the shortest path of an autonomous mobile robot.

Various types of GA have been developed to find a near-perfect solution in robot path planning.It has been a challenging area to work on different methods to find the near-optimal outcome. There are various effecting factors such as, the complexity of the environment in which the source and goal are configured for robot navigation; parameters incorporated in achieving an efficient result affect the path planning problem in finding the near optimal path.

The document is composed of three (03) chapters:

The first chapter is devoted to the presentation of mobile robots.A general overview of the field of mobile robotics is taken up to examine the typology of mobile robots. We present a state of the art of navigation in general and cooperative and autonomous navigation in dynamic environments with obstacle avoidance in particular.

In the second,the contribution to the research problem,we present a conceptual study on finding the shortest path for an autonomous robot using genetic algorithms. The environment modeling with the grid method is described, and the method GA is explained individually. We will discuss the application of a genetic planner for the mobile robot to accomplish a specific task, in other words, convergence towards a goal and the avoidance of obstacles.

In the last chapter, the simulation environments are discussed in brief. Several experiments are conducted with a PYTHON development environment to evaluate the performance of ourGA. examples of simulations are provided to highlight the results of the methods proposed for finding the shortest path for an autonomous robot.

Finally, we conclude and summarize the entire thesis work, explaining the conclusion obtained through simulated experiments. The possible scope of the future work of this thesis is mentioned.

Chapter 1:
Navigation Systems in Dynamic Environments

1.1.Introduction:

A mobile robot is a mechanical, electronic and computer system that concretely acts on its environment in order to address an objective that has been particularized to it. This machine is versatile and can adapt to certain distinctions of its operating conditions. It has the role of perception, decision and action. So, the robot should be able to perform various tasks, in many ways, and complete it properly, even if it encounters new situations abruptly. In this study we are interested in autonomous mobile robots.

A robot is said to be autonomous: it is able to choose its actions to reach its goal and if it is able to perform the task correctly and encounters new unexpected situations without human intervention.

In this thesis, we will present the problem of navigation in mobile robots. We determine the different navigation strategies and offer the categories of existing controllers.

Navigation is defined as the process of answering the following three questions [9]:

i) Where am I?

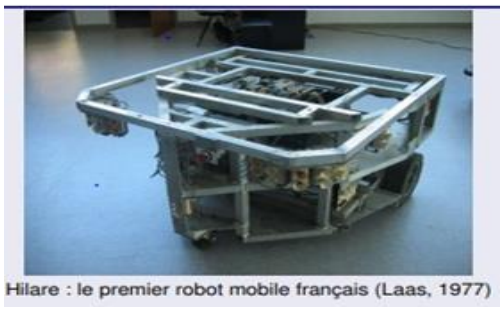
ii) Where are the other places in relation to me?

iii) How can I reach these other places from where I am?

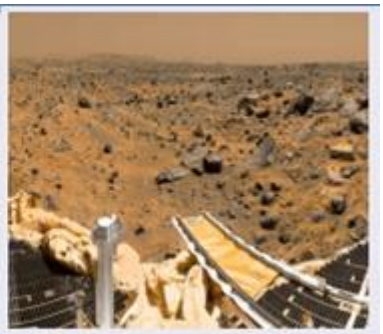
So navigation is [10] the set of techniques and methods for knowing the location (coordinates) of mobile with respect to a reference system, or with respect to a fixed point. Calculate or measure the trajectory to follow to reach another well-known coordinate point by submitting some constraints and criteria that arise from several factors, which usually depend on the characteristics of the robot, the environment, and the type of task to run. Calculate any information corresponding to the movement of this mobile (distance and duration, speed travel time, estimated time of arrival, etc..).

In this thesis, we discuss autonomous navigation techniques for mobile robots in a dynamic environment.

Aspirateur automatique Roomba (iRobot)



Hilare : le premier robot mobile français (Laas, 1977)



Sojourner sur Mars

Figure 1.1: Example of Decent Robots.

1.2.Navigation Strategies:

The navigation strategies allowing a mobile robot to move and to reach a goal are excessively diverse, as are the classifications that can be made of them. We use here a classification established by Trullier and Meyer, which has the advantage of distinguishing strategies with and without internal models.

1.2.1 Artificial Potential Fields (APF):

APF is conceived in order to build a field of potentials on the robot's navigation environment. The value of this field is minimal on the point that the robot must reach and continuously grows as one move away from that point. The obstacles generate a repulsive field of potential for the robot, of value superior to any other point not corresponding to an obstacle of the field potential. And the repulsion field extends around obstacles with intensity inversely proportional to the distance from the obstacle.

The idea is then to ask the robot to move in the direction of the strongest negative potential gradient on the overall potential field obtained.

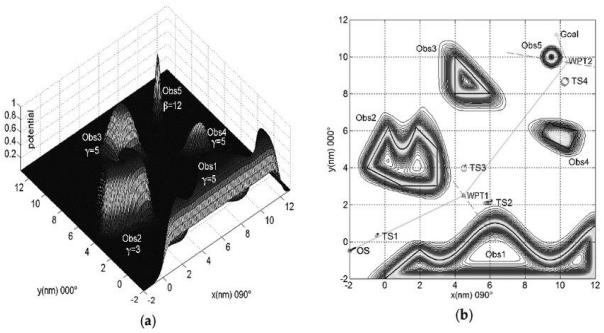


Figure 1.2: Map of potential fields around an obstacle.

In [11], and in [12], the authors were the first to imagine the idea of imaginary forces acting on the robot. These methods have the advantage of being simple to implement, and they were the first to be physically implanted on real robots in 1985 by Brooks [13] and in 1989 by Arkin [14]. Despite the computer equipment still limited at this time, trajectory /control calculations being very fast with this type of approach, it allows the first experiments on relatively slow robots. In [15], Agirrebeitia proposes an extension of the principle of APF methods for robot navigation in a 3D space. The experiment revealed some recurring problems related to the very principle of these methods:

- Local minima causing situations where the robot is trapped (typically the U-shaped trap).

- No passage detected between obstacles close enough.
- Oscillations caused by obstacles and narrow lanes.

In [16], authors in the mathematical field have demonstrated problems of instability of these methods (leading to oscillations), and these problems appear particularly strongly when implementing these methods on "fast" systems.

1.2.2 Artificial Neural network:

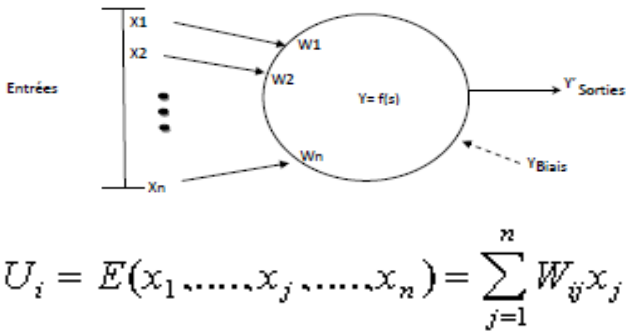
The formal abstraction of the behavior of biological neurons results in artificial neurons [17]. It is an ideal solution to some problems that are highly thought-provoking, or highly complex.This is due to their learning ability. A neuron is a nonlinear function, parameterized, with bounded value. A neuron contains 2 main elements:

- The weights associated with neuron connections.
- An activation function, the input values are multiplied by their corresponding weight and summed to obtain a sum U_i.

$$Ui = E(x1, \dots., xj, \dots. xn) = \Sigma Wij \, x \, j(1)$$

Learning can be understood as a change in the capacity or behavior of an organism brought about by experience. [18] The learning algorithm will formulate the explicit rules that allow it to generalize.

The formulation of the rules is done by the change of the synaptic weights which leads to the change of the behavior of the network; the change is carried out by a set of iteration which makes these networks able to react with new situations based on the experience passed.



$$U_i = E(x_1 \dots\dots x_j \dots\dots x_n) = \sum_{j=1}^{n} W_{ij} x_j$$

Figure 1.3: Artificial Neural network.

The neural understanding of knowledge is different; such knowledge can be in the following form:

The optimal path is found by the synaptic weight vector W, hence a different coding of knowledge. This representation of knowledge is closer to the machine language, which makes the interpretation difficult for the human being in comparison with the other methods of representation of knowledge.

For this reason, the networks of neurons are called a black box. One of the goals of the research is to understand how knowledge is distributed within a neural network and how to extract that knowledge in a comprehensive way from a human being.

The neural network approach is a method for modeling intelligence. These networks are able to solve problems in different areas. Their strong point lies in their learning ability.

However, the approach suffers from a major problem: the method of representing knowledge.

1.2.3 Fuzzy Logic:

Binary logic has the advantage of simplicity but is far removed from the human way of reasoning. If one takes the example of the qualification of the proximity of an obstacle, the fuzzy logic makes it possible to involve notions such as "near enough" or "very far", instead of being limited to a binary definition « obstacle or no obstacle ». This was formalized by Zadeh in 1965 [19].

The principle of a controller based on fuzzy logic comes in 3 phases:

A fuzzification step, which will transform the input variables into fuzzy variables;

A step using a table of rules of behavior, logical rules of the type "if (condition 1) and / or (condition 2) then (action on the outputs)";

A last, the defuzzification step translates the action determined by the rules of behavior in command to send to the actuators.

The fuzzification stage uses fuzzy intervals, which will delimit the space of the input variables in a certain number of fuzzy subsets (for example for proximity, we can have very close (contact), quite close, average distance, far enough and very far); membership functions are then used to define the degree of truth (probability of belonging) of the fuzzy variable as a function of the input quantity.

These functions can be a triangle, Gaussian, etc. Thus for given distance measurement, the membership rule will tell us "there is a 95% chance that the obstacle is close enough, 5% chance of being in contact".

This notion is based on the fact that there are always uncertainties about the sensor measurements and the information available in general.

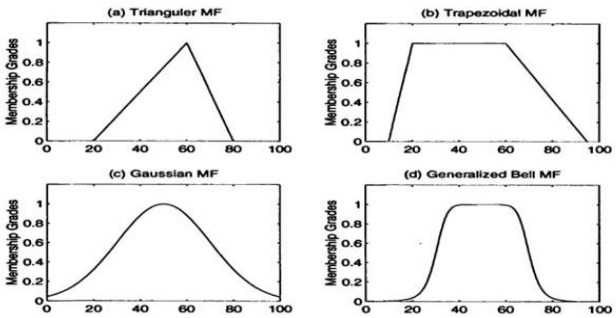


Figure 1.4: Membership functions

The second step is the development of rules of behavior for the robot, following the combination of fuzzy input variables. The rules of behavior table are built manually and are dependent on the experience of the person who will adjust the controller. For a mobile robot, a rule can be: "if there is a fairly close obstacle on the right, you have to turn left and decrease the speed of the robot".

The last step is to transform the behavior obtained by the rules table, in command of the robot.

One method that can be used to do this is that of the centers of gravity, which will consist of making the weighted average of the orders to be applied. The weighting being related to the probabilities of membership of each input variable. A fuzzy controller to allow navigation of a robot-car-type mobile robot with dual steering has been studied in [20]. This controller allows the robot to reach its final position while respecting the kinematic constraints of the robot, but for the moment does not take into account obstacles.

Work is underway to integrate the consideration of the final orientation of the vehicle. In [21], the authors use the right/left symmetry of the logic rules of behavior of the robot to simplify these, and thus, reduce the calculation time.

The problem of these fuzzy logic methods is generally the same as that of the APF methods, the problem of local minima which results in the robot being able to remain trapped in dead ends for example.

A technique for getting out of these traps has been proposed by Xu in [22]. This technique uses virtual targets to take the robot out of the trap into which it has fallen, as soon as it recognizes that it has fallen into a trap.

Another problem that comes out of the fuzzy logic methods is that they are too specialized for a given type of environment, and therefore they suffer from problems of adaptation to different environments. However, there are so-called learning methods that allow the robot to modify itself its rules of behavior as it explores a new environment.

The problem with these methods is that they take a long time to learn more or less long before the robot can navigate effectively [23].

1.2.4 Ant Colony:

Ant colony algorithms are inspired by ant behavior when looking for a path between the sources of food and their colonies. When an ant walks a path it leaves a trace in this way by producing a material called "pheromones". This subject is attractive to other ants, so they follow in the footsteps of pheromones.

The existence of this matter is related to distance, plus the distance is long, the lower the concentration of matter becomes. If the path is short we notice traces strong of this material, as a result, the ants follow the strongest trace which indicates a shorter path.

Ant colony algorithms aim to optimize a path for mobile robots. These algorithms require a group to be realized, they are used for a task that requires a collaborative work of several mobile robots.

The aim of this research approach was to model and develop on a simulator, then a real robot, the way ants [24] move using complex combinations of information, as a vector of integration and visual cues. The goal was concretely to solve the problem of navigation.

As a first step, developed a simulator to evaluate directly inspired navigation strategies ant behavior, the main interest of the simulation is to quickly allow an evaluation of biological hypotheses before moving to a physical implementation. Then, as the simulation does not allow taking into account all the

aspects of the study, by simplifying for example the environment or the physical behavior of the robot, the researchers have developed a robot able to really test the strategies of displacement.

In an environment with obstacles, the robot must first detect and then bypass them. If the robot has already passed a given point, it will be to anticipate the direction to be taken without being disturbed by the large amount of information resulting from the presence of a large number of obstacles. If the robot-ant encounters a very long obstacle, it should not be considered as impassable even if no issue appears on the rangefinder that we use to prevent the robot to hit the obstacles present in its environment. It must go along and around it by taking the direction provided by its integration vector being updated.

They developed first modeling based on the estimation of the vector of integration (direction of the nest, distance) that the ants realize spontaneously from their first outings out of the nest:

- If the robot-ant encounters no obstacle, it will follow its integration vector pointing in a straight line on its starting point;
- If he encounters a close obstacle but occupying a weak angular sector (Figure 2), the robot-ant will follow the free direction "closest" to the integration vector allowing it to bypass the obstacle.

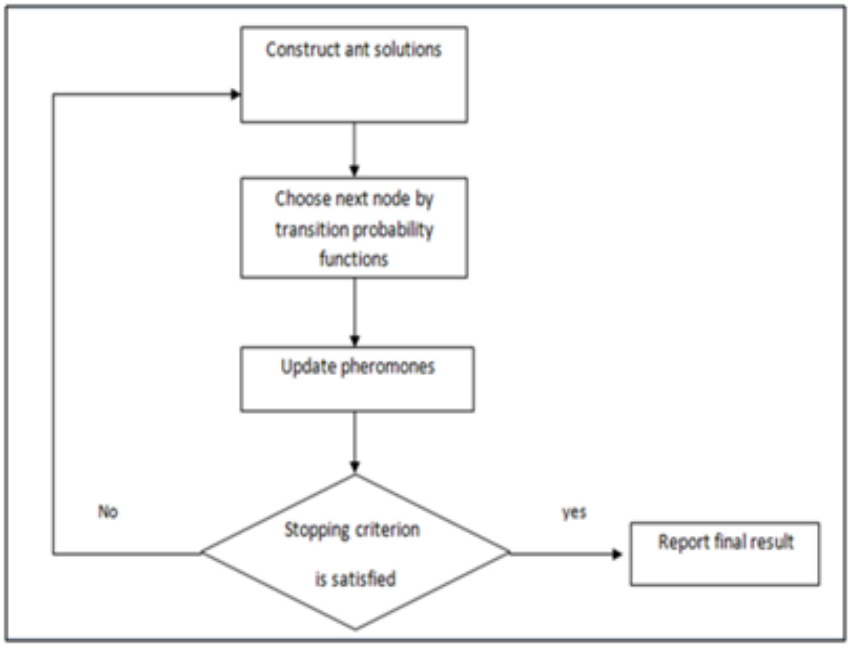


Figure 1.5: Ant Colony

The robot-ant has the ability to memorize couples (Perception, Action) and thus build an elementary representation of space.

The perception may correspond to the value of the integration vector and / or the perceived image, and the action to the movement to be performed (direction, distance to be traveled).

These pairs are, in a way, the equivalents of the local vectors previously mentioned in the insect. In order to minimize the stored information and thus be as realistic as possible, we define the notion of point of choice for any place where the robot-ant must make a decision.

This is the case, for example, for obstacles to avoid. Then, the robot-ant can use these couples (perception, action) by comparing them to the current situation. There is then a gradual transition from routine training to the automatic use of this routine.

The robot-ant is thus making more and more movements fast because once the routine is learned and "automated", it does the economy of the perceptual phase throughout this type of journey. The robot then only uses its sensors at each memorized point of choice in order to recalibrate itself, and not during each elementary movement. The trajectories are more and more smooth and efficient, which corresponds to the behaviors we observed in ants.

Ant Colony algorithms are very robust (they are always efficient, even in case of failure of certain individuals), they are flexible (a colony can adapt to a new environment) and they are fast good solutions as well the work in parallel and the use of heuristic information, among others. But there are many disadvantages, we can take a blocking state can happen, runtime sometimes long and does not apply to all types of problems.

1.2.5 Bee Colony:

If we only use some of the nature or behavior of bees and add some new features, we can design a class of new algorithms. In what follows, we represent some algorithms (the best known), without being exhaustive, based on the behavior of the bees during the foraging.

1.2.5.1 Virtual Bee Algorithm:

This algorithm was developed by XinShe Yang in 2005 [08] for solving numerical optimization problems, this can optimize functions and discrete problems, although only functions with two parameters have been given as examples. The arrangement of the VBA algorithm starts with a virtual bee troop, each bee moves randomly into the search space and in most cases, the search space can be just a 1-D or 2-space -D. The main steps of the bee algorithm virtual functions for optimizing functions are:

- Creation of a population of multi-agents or virtual bees.

Each bee is associated with a solution vector with several parameters to optimize.
Coding of optimization functions (objective functions) and conversion to virtual food (Virtual Food).

- Definition of a criterion to communicate the direction and the distance in a way similar to the physical aptitude of the bees (the dance of the bees).

Update a population of individuals in new positions forvirtual food research, doing a virtual dance to define distance and direction; "the virtual dance of waggle".

After a certain period of evolution, the highest mode, in the number of virtual bees or the intensity / frequency of the bees that make the visit is high, corresponds to the best evaluation.

- Decoding of the results to obtain the solution of the problem.

1.2.5.2 Bee Colony Optimization

This algorithm was introduced by yuce et al in 2013 [25] in order to find the optimal solution for a given difficult combinatorial optimization problem, such as the problem of a commercial traveler, the problem of p-Median, problem routing in optical networks.

Each bee generates a solution to the problem. To build one step in BCO algorithm. There are two alternate stages (step forward and step back) .In each step forward, each artificial bee visits Nsolutions creates a partial solution and then returns to the hive.

The bees gather in the hive and begin the step backward. When all the solutions are completed, the best of them is determined, and it is used to update the best overall solution and like that, an iteration of BCO is accomplished. At this point, all the solutions are removed, and a new iteration is born. Let 'B' be the number of bees in the hive, and "NC" the number of constructive moves forward. When the search starts, all bees are in the hive. The pseudo-code of the BCO algorithm can be described as follows:

- Initialization: an empty solution is assigned to each bee;
- For each bee: at. k = 1

    a. Count the constructive moves forward)

    b. Evaluate all possible steps;

    c. Choose a step;

    d. k = k + 1;

    If k ≤ NC, Go to b.

    Return of all the bees to the hive;
- For each bee evaluate the value of the objective function.
- Each bee decides randomly either to continue its own exploration.
- Recruiter, or become the bee who does the harvest. For each follower, choose a new solution from the recruiters.
- If the solutions are not complete, go to step (b).
- Evaluate all the solutions and find the best one among them.
- If the stop criterion is not checked, go to step (b), otherwise, go to the next step.
- Show the best solution found.

1.2.5.3 Dance Bee Optimization

The BOD (Dance Bee Optimization) algorithm was developed by Laga and Nouioua in 2009 [25] to solve the problem of T-coloring graphs. This algorithm is inspired by the behavior of bees when foraging. The algorithm starts by randomly positioning the n bees in the search space.

After evaluating the fitness features of these bees, the bees with the best fitness (elite bees) are chosen for neighborhood building. In the next step, the algorithm guides the search in the vicinity of the best sites m found by elite bees. Indeed, these are the bees recruited to search around the best sites e, ie. Follow the best dancers, are also recruited bees pursue the other dancers. This recruitment is the key

operation of the BOD algorithm. For each recruited bee (solution), we associate a neighborhood meta-heuristic to search around this solution.

In the end, in each neighborhood, only the best ones, among m bees (solutions) are therefore retained to form the next population. Note that in nature, there is no similar restriction; this restriction is introduced in the algorithm to reduce the number of solutions to explore.

To complete the bee population, the remaining bees are randomly generated. In At the end of each iteration, the colony will consist on the one hand, of m bees representative of each neighborhood (to intensify the search) and on the other hand, of bees assigned randomly (to diversify the search).

These steps are repeated until a predefined stopping criterion (a number of iterations or a stagnation number). The dance is vertical the pollen is in the direction of the sun. The dance is vertical and is directed downwards the pollen is in the opposite direction to that of the sun.

The angle that the plane of the dance makes with the vertical is equal to the angle that the food makes with the Sun in a horizontal plane. The direction of the sun is thus represented by the vertical, seen from below upwards; and the angle of the direction of the spoils with that of the sun is reproduced in relation to the azimuth.

1.2.5.4 Artificial Bee Colony

The ABC (Artificial Bee Colony) algorithm was developed by Karaboga and Basturk in 2007 [26], inspecting the behavior of real bees to find the food source, called nectar, and share information from food sources to others bees in the nest. In this algorithm, artificial bees are defined and classified into three groups: bees (bees that search for food), spectators (bees of observation), and scouts are responsible for finding new foods, (the new nectar source). For each food source, there is only one employing bee. That is, the number of worker bees equals the number of food sources. If a worker bee at a site cannot find the food source, she must be a Scout to randomly search for new food sources. Employing bees share information with bee-goers in a hive so that bee-goers can choose a food source to explore.

Among the advantages of the bee colony method, mention may:

-Very effective in finding optimal solutions.

-overcomes the problem of the local optimum.

-Easy to implement.

-The use of several adjustable parameters.

-Sensitive to extremely difficult problems.

But there are many disadvantages, like most optimization algorithms features of a mechanism of evolution and a mechanism of diversification, increments a counter for solutions that do not improve not to reach a threshold limit, fixing this parameter is a problem in itself, and small values can eliminate a solution before to explore its neighborhood incomplete, while large values may trap the algorithm in minima premises for several cycles.

1.2.6 Genetic Algorithm:

Genetic Algorithms belong to a family algorithm called Evolutionary Algorithms [27]. These algorithms are random optimization techniques inspired by Darwin's theory of evolution, which aim to find an approximate solution at the correct time. Genetic Algorithms use techniques derived from genetics and natural evolution: crosses, mutations, selections, etc ..., they represent a stochastic optimization method of order "0", which means neither continuity nor differentiability is necessary for the smooth running of the method, only the knowledge of the value where the proximity of the function to be optimized is sufficient. So, the effectiveness of a genetic algorithm depends on the good knowledge of the problem to be treated.

Genetic algorithms are the result of research by John Holland and his colleagues and students at the University of Michigan who, as early as 1960 [28], worked on this topic. The novelty introduced by taking into account the crossover operator in addition to the mutations in this operator which allows us to get closer to the optimum of a function by combining the genes contained in the different individuals of the population.

Genetic algorithms are based on the notion of natural selection and apply it to a population of solutions to a given problem.

The stages of execution of a genetic algorithm:

Initial population: We must choose a random population of n chromosomes, each chromosome here presents an upcoming location of the robot, and we can also change it so that each gene represents a next direction of the robot.

Fitness function: Measure the fitness of each chromosome in the population.

Selection: Create a new population with repetition next steps until the population is complete.

Crossover and mutation: Each pair generates two children, in these operation two chromosomes exchanging one or more parts for data of the new chromosomes. If there is no mixing, the result is an exact copy of the parents.

Mutation means that a gene in a chromosome can substitute for another in a random way.

Stop test: If this criterion is not checked then go to step (2).

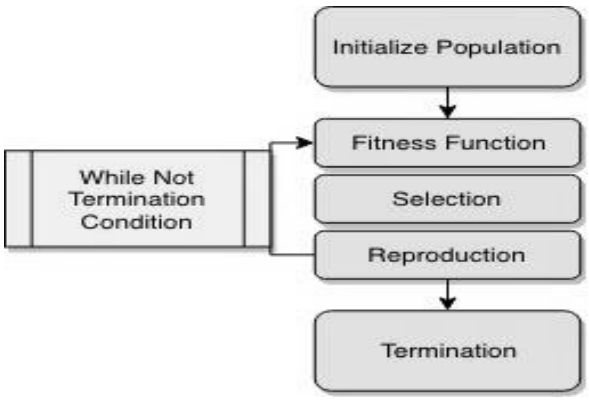


Figure 1.6: Genetic Algorithm.

These algorithms are expensive in terms of calculation volume, in particular at the evaluation function and memory size used.

- The path found is not optimal: The impossibility of being sure that the solution found is the best one even after a significant number of generations. We can only be sure that we are getting closer to the optimal solution (the parameters and the evaluation function).
- No guarantee of convergence of the algorithm or to know the case of non-existence of solution. Algorithms are more often difficult to implement parameters such as population size or mutation rate are sometimes difficult to determine. However, the evolution depends on a number of tests, which further limits the effectiveness of the algorithm. In addition, choosing a good evolution function is also critical. It must, therefore, choose carefully.
- The problem of the local optimums, if in the population at a given time an individual occupies an important place then the population will converge towards this individual and cannot evolve.

This problem is related to the principle of the algorithm itself and has no relation to the environment or the robot used, there are some works and some solutions to remedy this problem. The results obtained through the use of genetic algorithms in the autonomous robot trajectory search domain are not encouraging.

In addition to the time and volume of important calculations, the convergence of this algorithm is not enough what pushes to hybridize this type of algorithm with other algorithms see for example. Compared to conventional optimization algorithms, the genetic algorithm has several strengths such as:

Using only the evaluation of the function: objective without worrying about its nature. Indeed, we did not need any particular property on the function to optimize (continuity, differentiability, convexity, etc.), which gives it more flexibility and a wide range of applications.

GAs uses the coding of parameters, not the settings themselves. Chromosomes have a binary representation. This choice makes them intuitively applicable to all the problems whose solutions can be transposed into binary. The chromosomes are then represented by a chain of bits. This representation is independent of the problem and makes the genetic algorithm all the more robust.

Generating a form of parallelism by working on several points at the same time (population size N) at the place of a single iterated in classical algorithms, GAs work on a population of points instead of one.

The use of probabilistic transition rules (probabilities of crossing and mutation), unlike deterministic algorithms where the transition between two successive iterations is imposed by the structure and nature of the algorithm. This use makes it possible in some situations for genetic algorithms to avoid local optimums and to move towards a global optimum.

1.4 Comparison between Methods

Method	Advantages	Disadvantages
Fuzzy logic	- Translate human experience into a set of rules. - The closest to human reasoning. - It is not heavy in terms of calculation, saving time, and memory space.	- The navigation path is not optimal because of the approximate reasoning method. - Requires the availability of an expert - Robot operation is limited by these rules
Neural networks	- Build a solution in a simple way -The ability to calculate an exact navigation path	-The knowledge representation method.
Ant colony	- Very high adaptability. - Perfect for graph-based problems.	- A blocking state can occur. - Sometimes a long execution time. -Does not apply to all types of problems.
Bees Colony	-Effective in finding the best solutions and easy to implement - Get rid of the local optimum problem - Sensitive to complex problems	-It has a mechanism of evolution and diversification - Big values may trap the algorithm for multiple cycles and it can eliminate a solution before exploitation
Genetic algorithm	- AGs use the encoding of parameters - AGs are working on a population of points - The use of probabilistic transition rules in order to avoid local optimum - Synthesis of aerodynamic shapes, structures and composite materials	-The problem of scheduling industrial networks - Recognition of forms and learning by reducing it - Pattern detection on bioinformatics - Synthesis of electronic circuits

Conclusion

In this chapter, we have presented a state of the art on navigation for mobile robots and the various algorithms available to us to manage it, for the discovery of the shortest paths and distances in an environment. We have also seen the different stages; we also presented the advantages and limitations of each method and found that no navigation approach that denies a solution that eliminates all problems has been encountered. Thus, we see that navigation is a very active field of research and that new methods appear regularly. This review of the literature reveals the importance of using heuristic methods to formulate the motion computation problem as an optimization problem in the interest of satisfying several constraints at the sametime.

Chapter 2:

System Design

2.1. Introduction:

Robots are generally machines including computers, the robot controller, is specific and therefore not portable to another robot. However, there is both industrial and research demand to define a generic model of a robot controller.

In terms of research, the development of applications requiring sharing of the robot's environment with its environment leads to look for reliable and scalable architectures far removed from the fixed architectures of classic robots whose immediate environment is closed to the human operator.

In this context, AI (genetic algorithm) methods for the specification and design of real-time systems should aid in this modeling of the complex system that is a robot controller.

In this chapter, we represent our work which is arranged to:

• A genetic representation of the problem, a coding of solutions used in the form of chromosomes, a mechanism for generating the initial population, a function which makes it possible to evaluate the adaptation of a chromosome to its environment, which offers the possibility of comparing individuals.
Cross-over, mutation operators, and selecting the chromosomes to reproduce. Making it possible to diversify the population over the generations and to explore the state space.

Conceptual modeling is an intuitive approach to designing a database. It is suitable as soon as we need to structure a somewhat complex data set. Another approach, the theory of normalization, is more formal, more rigorous, and offers very fine analytical means to solve problems on specific subsets of the data.

The objection of the conceptual model is to represent the problem using graphical and partially formal representations.

The main characteristics of the conceptual model are:

A simple graphical representation

- High expressive power for a reasonable number of symbols
- Reading accessible to all and therefore a good tool for dialogue between technical and non-technical actors
- Unambiguous formalization and therefore a good detailed specification tool.

2.2. Our Problem:

Our problem is to find the shortest path for an autonomous mobile robot. Where the aim is to search for optimal plans to reach the desired end state such that the cost is minimized. To find optimal solutions to large scale planning problems, we investigate how GAs can be adapted and applied to these optimization problems with intractably large and highly complex search spaces. More specifically, our work aims at answering the following research question:

How a robot can move from a starting point to an ending point in an optimal way and secure?

2.3. The proposed solution:

To answer the research question, the research focuses on adapting already existing and developing new efficient GAs to solve large scale optimization problems of tasks applied to a mobile robot. Moreover, the focus is on adapting and applying an existing GA to assignment problems with huge search spaces to find specific assignments, which are mapped onto a specific region of the solution space. In addition, already existing GAs are adapted to solve path planning problems.

Figure 2.1. Finding the optimal trajectory for a mobile robot

2.3.1. Problem definition

We focus on the following situation: In the initial, state there is a square matrix that represents the environment with obstacles placed randomly.

A robot must reach an arrival point from a start point, the robot has a start position (Xs, Ys) and an arrival position (Xa, Ya), the robot does not know the environment.

We were interested in solving the following problems: Finding the optimal trajectory (the shortest path) and avoiding obstacles.

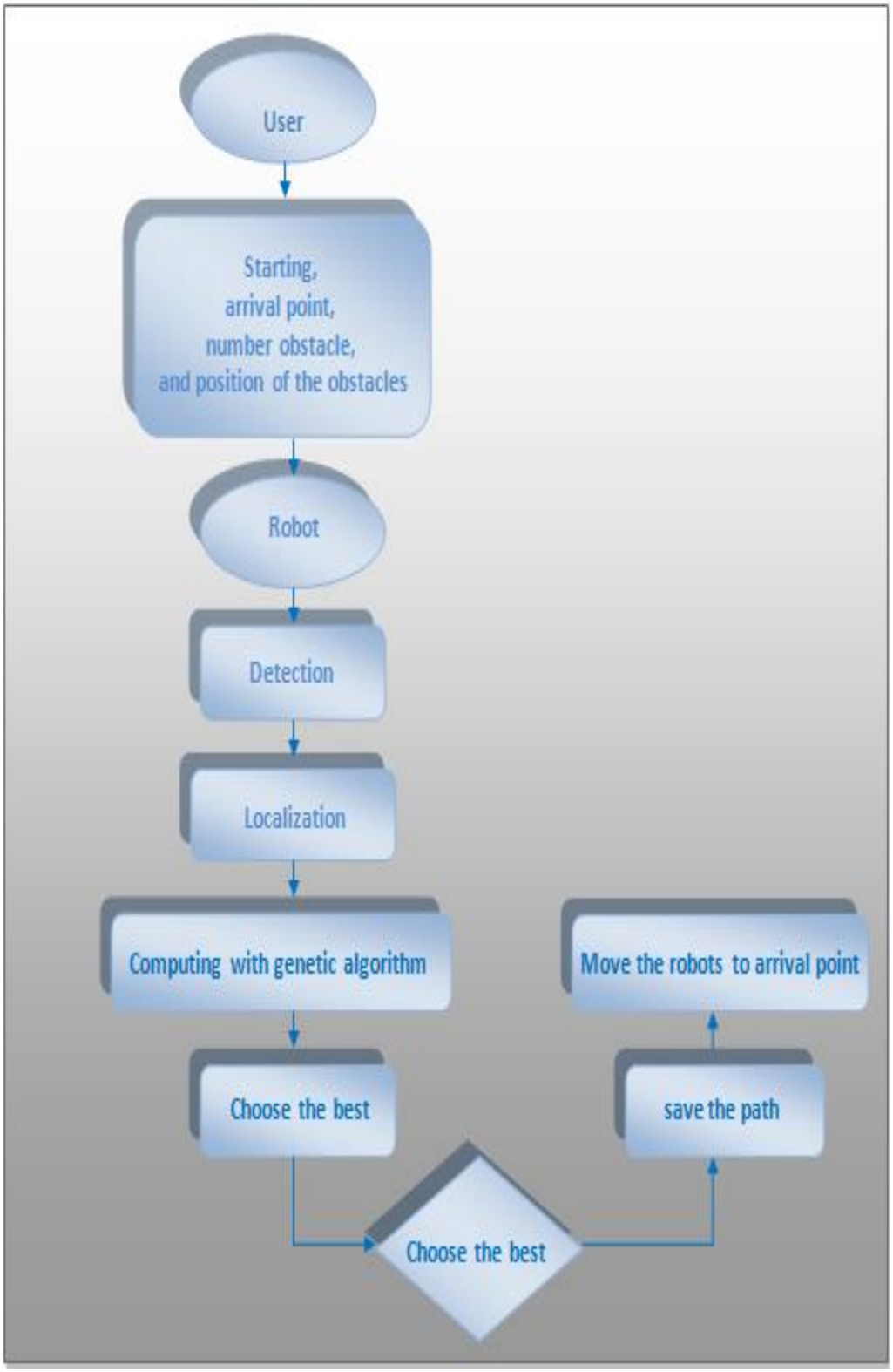


Figure 2.2: General Architecture of our system

2.3.2. Use case diagram:

A use case is a cohesive unit representing functionality visible from the outside. It provides an end-to-end service, with an initiation, an unfolding, and an end, for the actor who initiates it.

Therefore, a use case models a service provided by the system, without imposing the embodiment of this service.

The robot has four main operations: search, move, the avoidance and detection of obstacles:

- Research is a process of identifying goals.

- The movement is activities changes of positions on the instructions received from the controller.
- Obstacle avoidance is also a process of approaching goals.

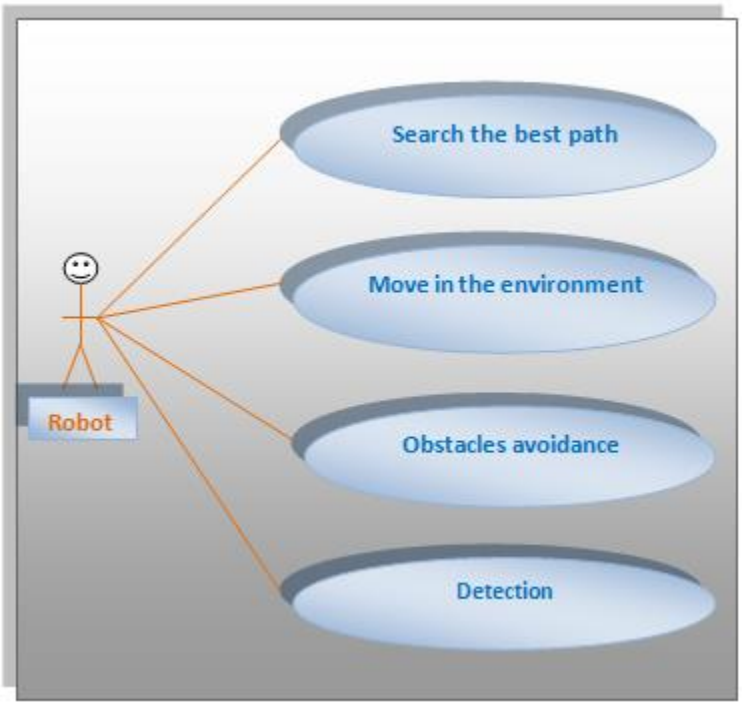


Figure 2.3: Robot use case diagram

In our diagram, we have a single entity that symbolizes our robot as the main actor. the later has 4 central operationss to preform that being:

- The search of the best path: our robot is situated at the starting position in an environment with an ending point to reach, it is then up to it to decide the best and the shortest path
- Move in the environment; with the ending point In record the robot begins to make its movement along the environment
- Obstacles avoidance: obstacles are randomly placed in the environment according to the start and ending point entered, therefore it is essential for the robot to avoid them when they cross its path

The user has two main operations:

- Determine the starting and arrival points.
- Determine the number and position of the obstacles.

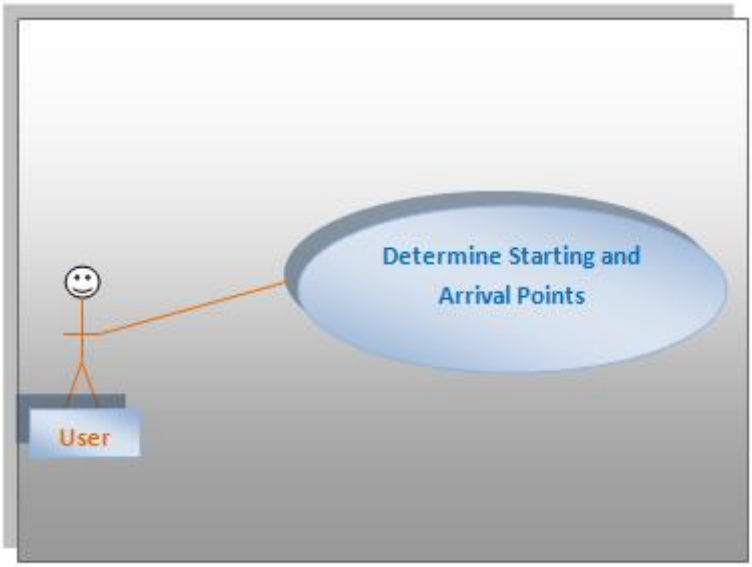


Figure 2.4: User use case diagram

In this use case diagram, the user is the single actor with a sole operation. When starting the program, a window appears requesting for the starting and ending point to determine the path which the robot is required to find

2.3.3. Sequence diagram:

A sequence diagram is a Unified Modeling Language (UML) diagram that represents the sequence of messages between objects during an interaction.

A sequence diagram consists of a group of objects, represented by lifelines, and the messages that these objects exchange during the interaction.

Sequence diagrams represent the sequence of messages transmitted between objects. They can also represent the control structures between objects.

Communication between these objects is modeled by the sequence diagrams in Figure 2.5.
The robot receives the instructions from the trajectory calculation class by the send solutions method then it sends them at each update of position with the selection method. (at a chosen sampling frequency of the new solutions).

The position update receives the 92 solutions; it then performs the calculation of the genetic algorithm (crossover and mutation) with the elitism method. And finally, it sends the result (shortest path) to the robot.

The interactions described in these two diagrams make it possible to deduce a simplified class diagram.

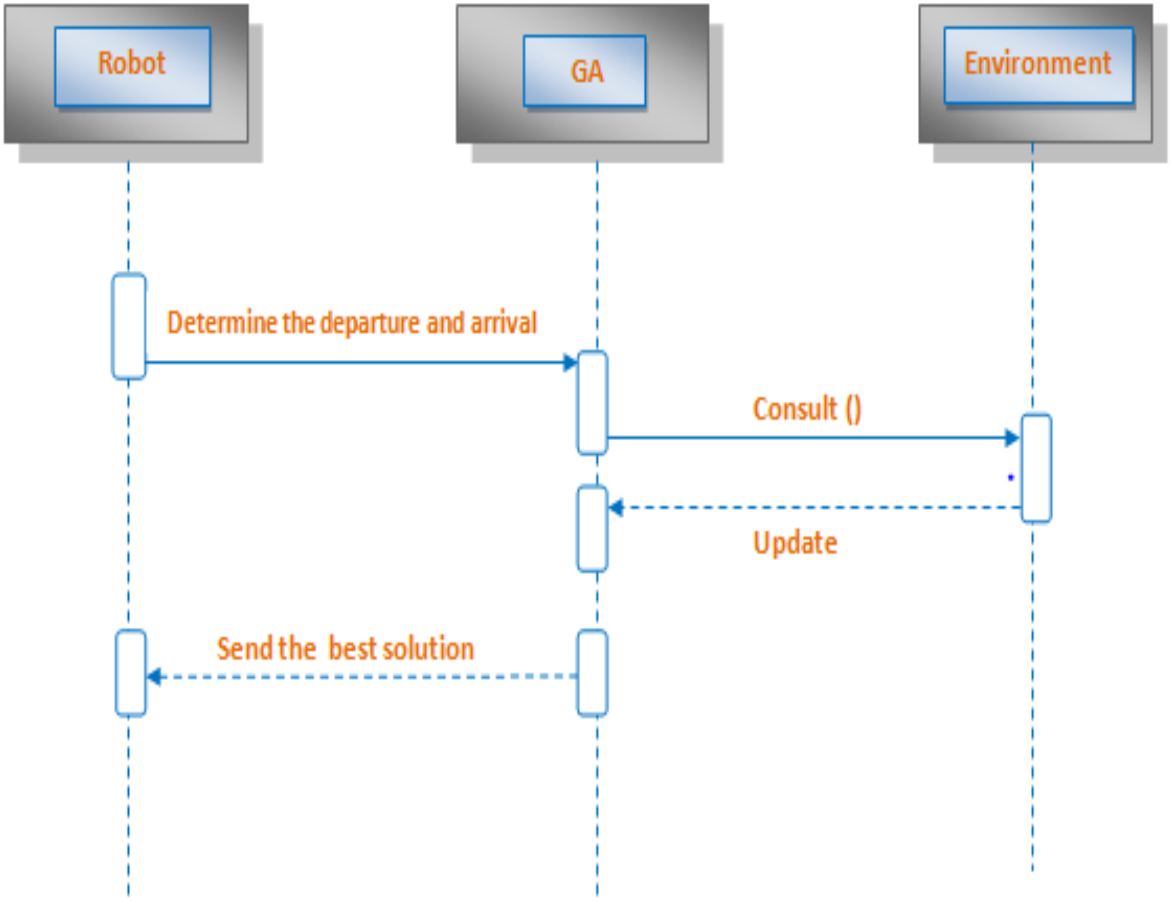


Figure 2.5: Sequence diagram illustrating the robot action

Our process begins when the robots receive the trajectory calculation class, which then it sends over to the genetic algorithm system for analyzation.

The system first consults with the environment in hand before creating a 100 potential path for the robot to take.

Out of those 100 solutions, we save 92% to lessen the options and avoid the chance of destroying the best solution.

With the aid of the elitism method, it then performs the calculation of the genetic algorithm (crossover and mutation).

Ultimately, it transmits the shortest path back to the robot for execution.

2.3.4. The Activity Diagram:

In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions of a process.

Activity diagrams are similar to information processing flowcharts because they show the flows between actions in an activity.

Activity diagrams can, however, also show simultaneous parallel flows and replacement flows.

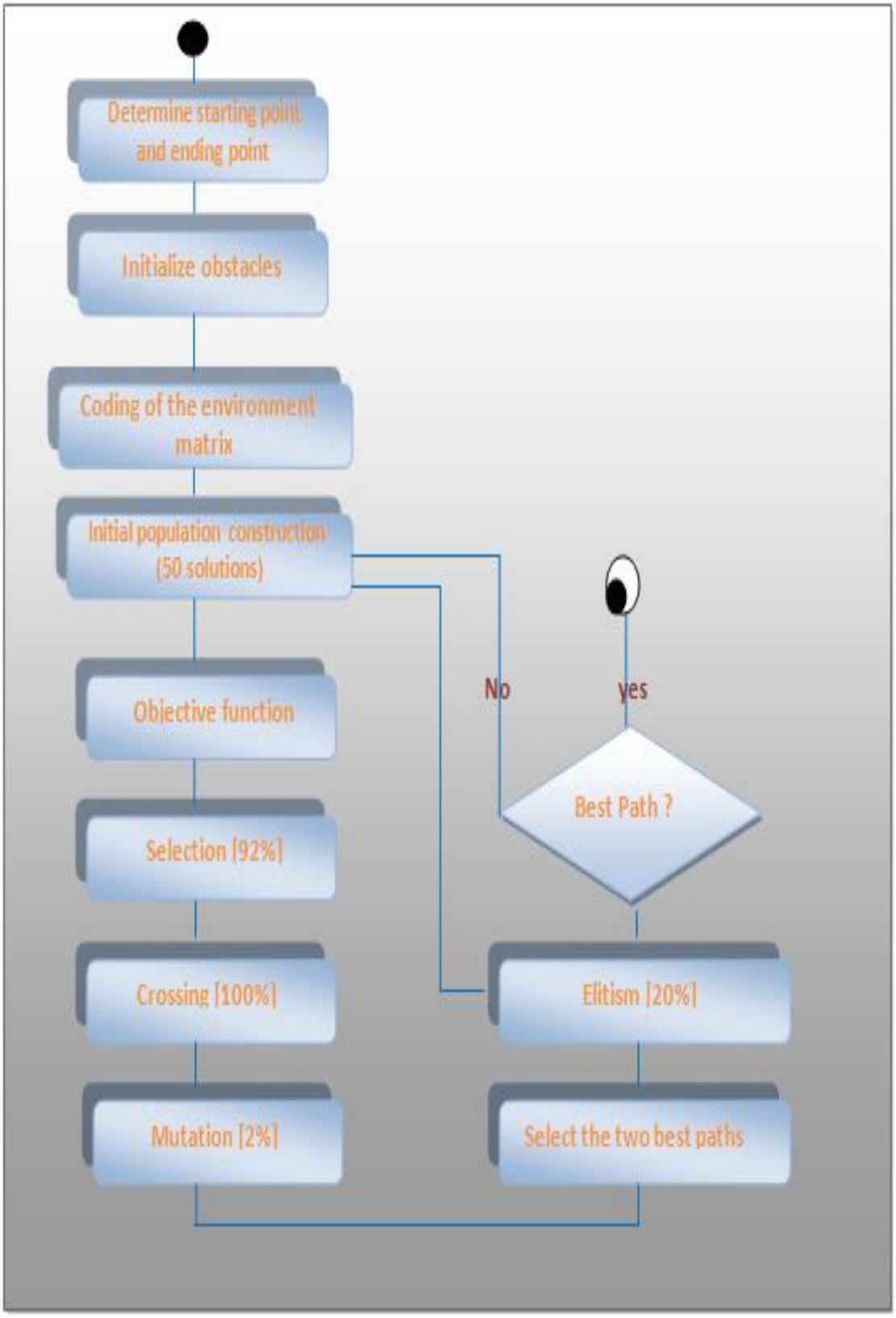


Figure 2.6: Activity Diagram

Commence from the determination of the starting and ending points, the user begins with the coding of the matrix, in initial, it is represented in binary code where 0 is an obstacle placed randomly, and 1 is a free path.

2.3.5. The Class Diagram:

The class diagram illustrates the static structure of the information model, especially the existing objects and their internal structure and their relationships with various other things. A class diagram should not present any information of a temporal nature.

Content: classes, subclasses, attributes and values, methods, links (multiplicity, generalization, composition), categories, and dependency.

The robot class must manage the sequence of trajectory generations and commands so that the robot does not stop between two trajectories.

It has a specific behavior compared to other classes.

UML expresses this behavior by the use of Statecharts which allow representing the parallelism well.

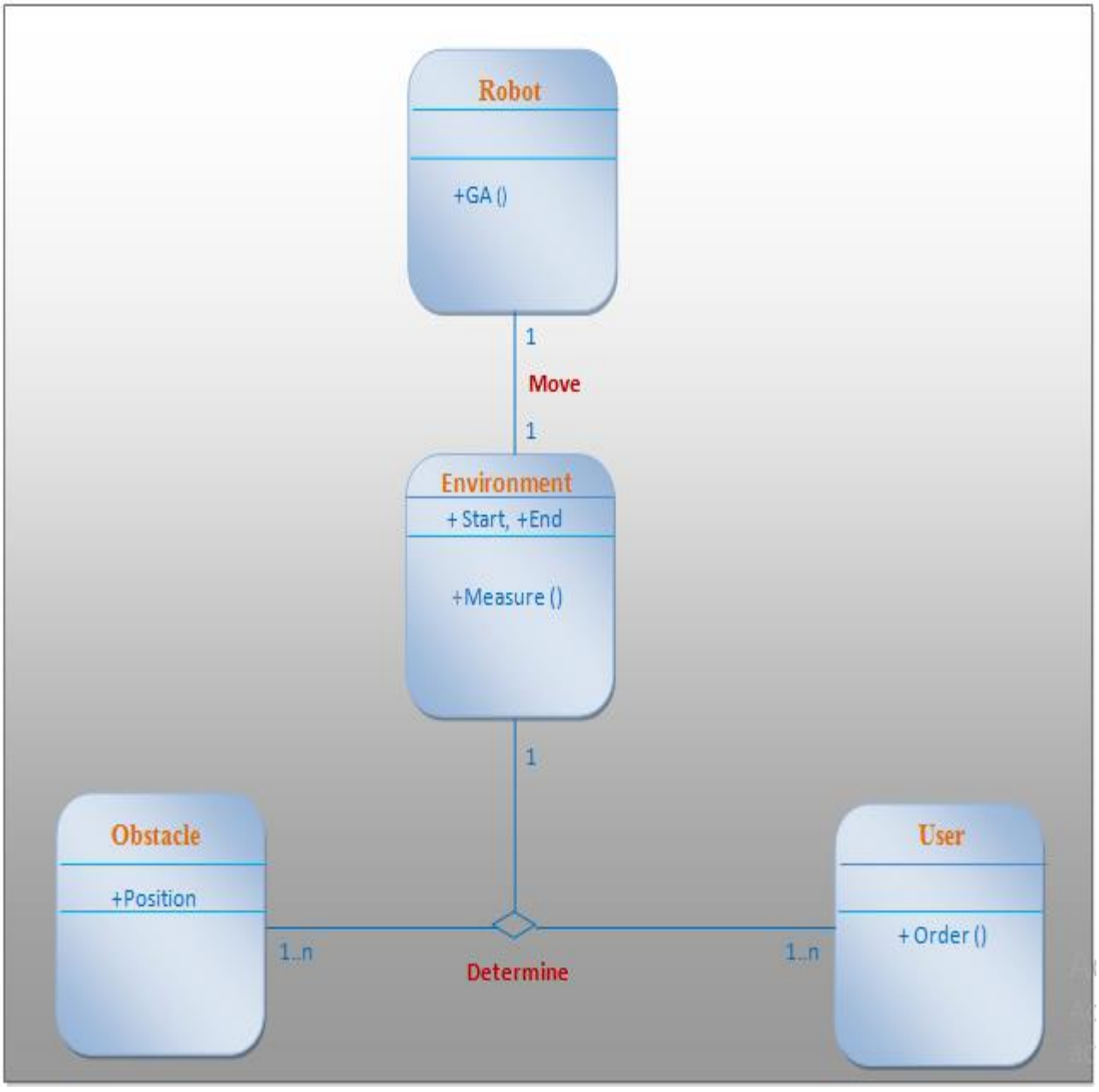


Figure 2.7: Class Diagram

- Robot class it manages all the genetic algorithm operation, which is basically managing the sequence of the trajectory generation and commands to avoid errors and confusion such as malfunctioning and sudden halt

- Environment Class It manages the entry of the starting and ending points, and the measurements of the path chosen.
- Obstacles Class It manages all operations of the obstacles including positions. user class it manages all operation of the user

2.4. GA Architecture:

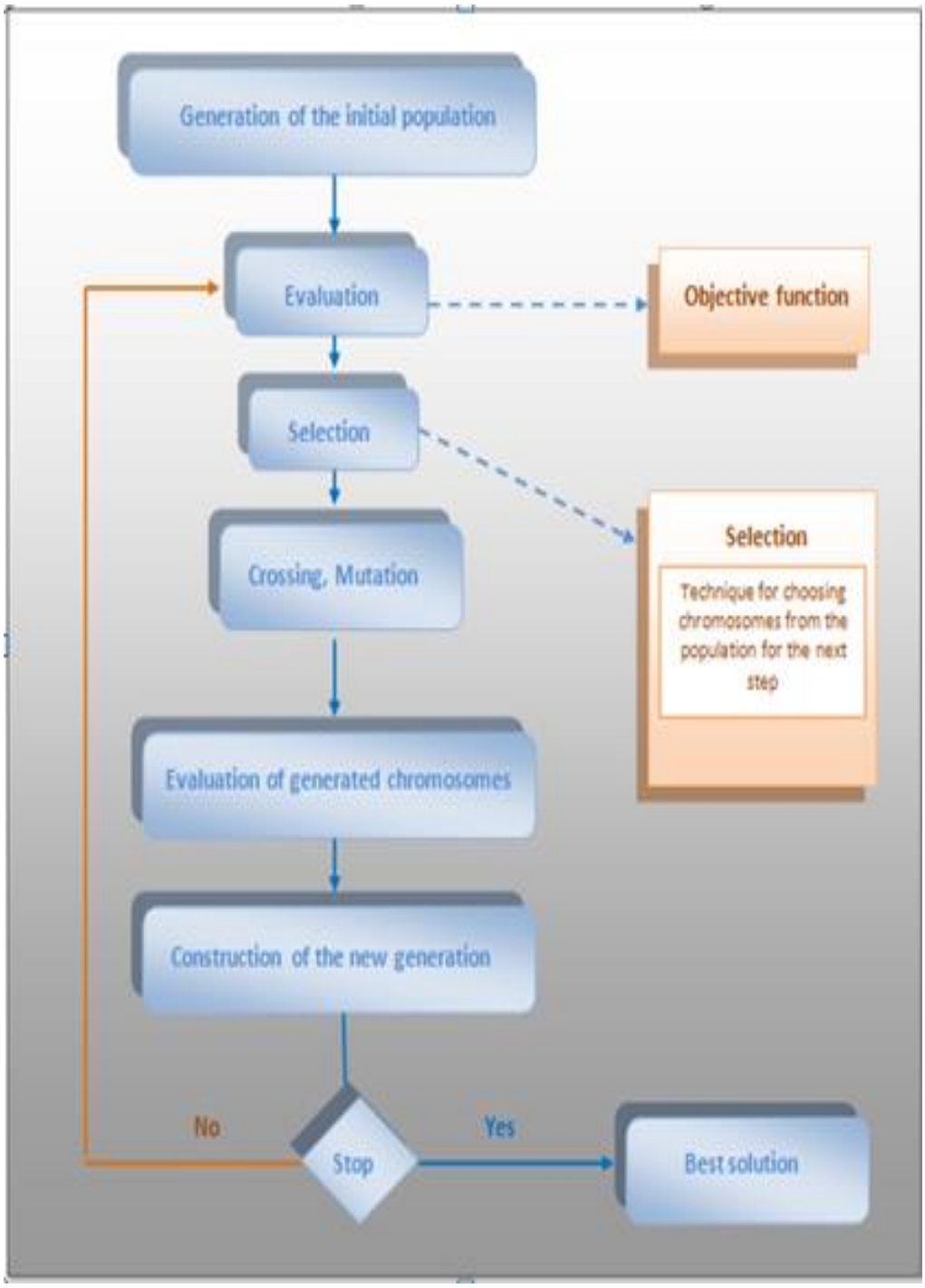


Figure 2.8: General Flowchart of Genetic Algorithm

A genetic algorithm searches for the extrema (s) of a defined function on a data space. To use it, you must have the following five elements:

2.4.1. Coding:

Before describing in detail these different stages of the genetic algorithm, we represent, below, the coding scheme of the individual in this GA.

2.4.2. Representation of the individual:

Coding is a modeling of a solution to a given problem in the form of a sequence of characters called a chromosome where each character, also called a gene, represents a variable or part of the problem. The main task is to choose the content of the genes that facilitate the description of the problem and respect its constraints. It mainly uses two types of encoding: binary encoding, real encoding. In our work we choose the actual coding where each chromosome is a list whose components are the parameters of the optimization process. For example, if we are looking for the optimum of a function of n variables [x1, x2,…, xn], we can simply use a chromosome containing the n variables: [x1, x2,…, xn-1, xn], with this type of coding, chromosomes are faster given the absence of coding and transcoding steps (from real to binary and vice versa). A chromosome is represented as 0-1-2-3 bits strings.

2.4.3. Initial population:

The selection of the initial group of individuals directly controls the vitality of the algorithm. If the position of the optimum in the state space is completely unknown, it is natural to randomly generate individuals by making uniform draws in each of the domains associated with the components of the state space, by ensuring that the individuals produced respect the constraints.

On the other hand, if there is initial information about the problem, then individuals in a particular subdomain are generated in order to accelerate convergence.

In the event that the constraints cannot be managed directly, the constraints are generally included in the criterion to be optimized in the form of penalties.

The main problem in this step is the choice of the size of the population. If the size of the population is too large, the computation time increases and requires significant memory space.

On the other hand, a population of very small size, the solution obtained is not satisfactory. We must therefore find the right compromise.

This mechanism must be capable of producing a non-homogeneous population of individuals which will serve as a base for future generations. The choice of the initial population is important because it can make convergence towards the global optimum more or less rapid. In the event that nothing is known about the problem to be resolved, it is essential that the initial population be spread over the entire research area.

In our case, it is generated randomly and corresponds to the possible paths to reach the point of arrival. Having an acceptable solution suggests considering 100 possible paths in order to reach the destination. To solve the problem of premature convergence (local minimum) the algorithm generates a set of solutions (paths). The initial population is presented by a list of lists to give more user-friendliness because the length of the path found is not always constant.

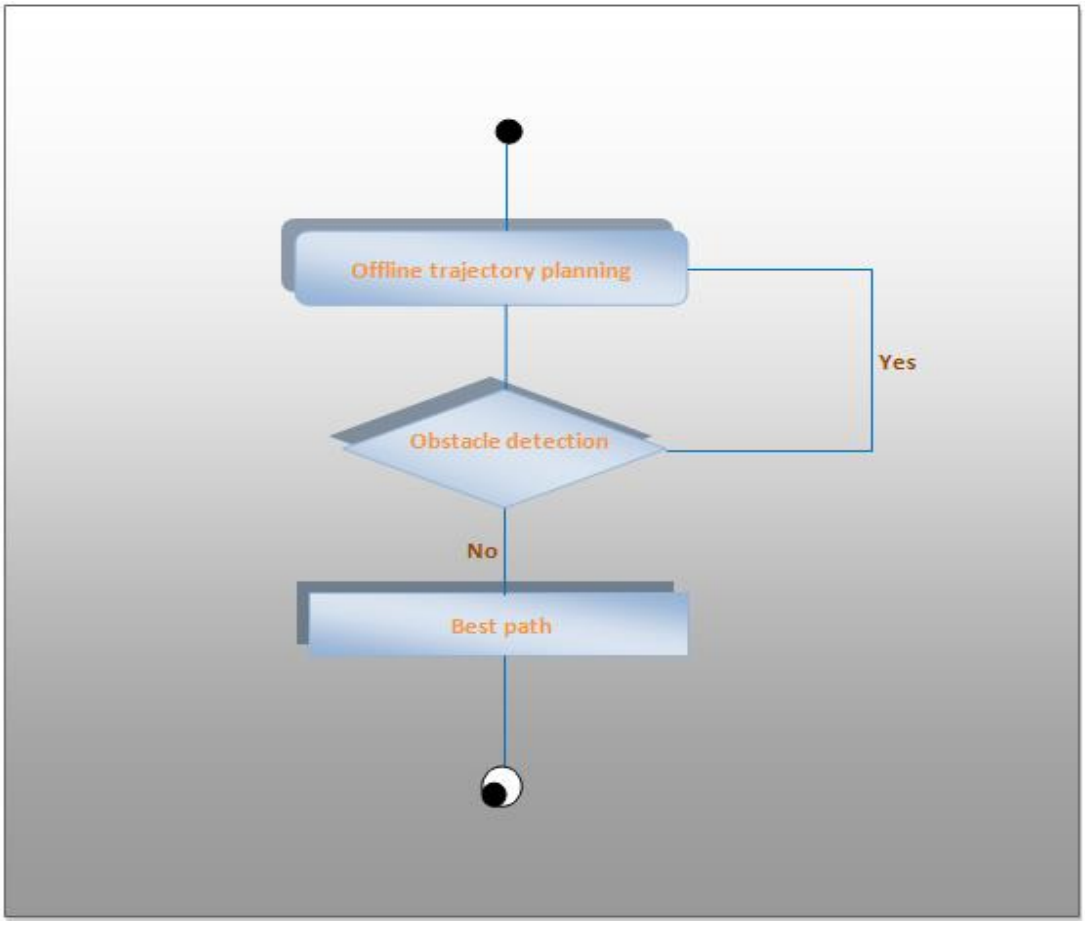


Figure 2.9: The Movement of a Robot to Find Initial Solutions

2.4.4. Fitness function:

During each iteration (generation) the population of paths is evolved to quantify their degree which depends on how each solution (path) is appropriate for the problem. Therefore, an individual's fitness value should be proportional to his ability to survive.

Most path browsers using AGs consider path length as a criterion to minimize the solution.

In this study, we propose an evaluation function which depends on the length of the path. Thus, the objective function for a path is defined as follows.

This function makes it possible to evaluate an individual's ability to survive by assigning him a pod called fitness. The length of each chromosome in the population is calculated so that the shortest ones are retained in the selection step and then modified in the crossing and mutation step.

This takes its values from R + and is called fitness or an individual's evaluation function. This is used to select and reproduce the best individuals in the population.

TheDistance of Manhattan: The standard used in heuristics is the distance of Manhattan. The cost of moving from a space to an adjacent space is D. So the heuristic in our problem must be:

$$H(n) = (abs(Xa - Xs) + abs(Ya - Ys)) \ldots\ldots (1)$$

2.4.4.1. Selection:

Selection makes it possible to identify individuals likely to be crossed in a population. There are several selection techniques.

Unlike other optimization techniques, genetic algorithms do not require any particular hypothesis on the regularity of the objective function.

In particular, the genetic algorithm does not use its successive derivatives, which makes its field of application very broad.

No continuity assumption is required either. However, in practice, genetic algorithms are sensitive to the regularity of the functions they optimize.

The few assumptions required allow us to deal with very complex problems.

The life cycle of individuals is revealed at each generation, out of a population of n individuals, 92% will be kept there, these can reproduce (the ideal selection percentage is between 70-95%). To keep the best individuals of each generation, our algorithm uses an elitism phase: drawing paths according to their length in ascending order and then keeping 20% of the current population to copy them to the new generation.

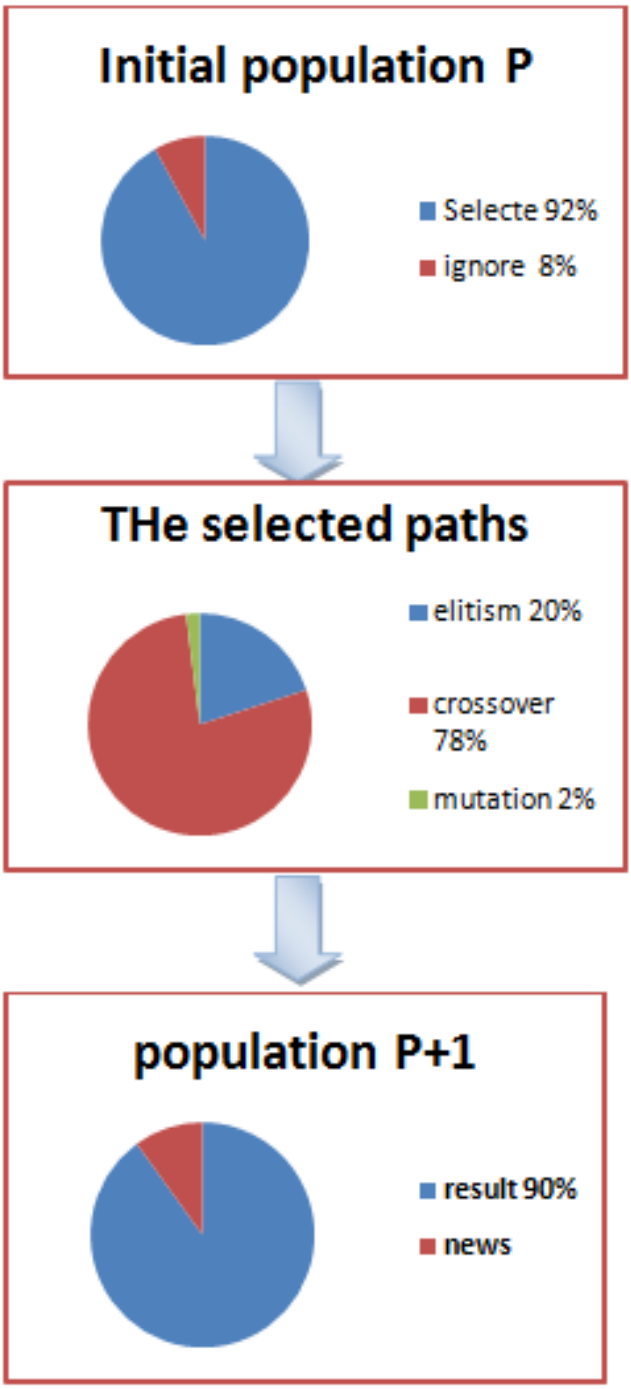


Figure 2.10: The Rate of Each Operation of the Genetic Algorithm

2.4.4.2. Crossing

Crossbreeding enriches the population by manipulating the components of chromosomes. A cross is considered with two parents and generates one or two children. After selecting two individuals, we apply the crossover operator on both parents.

We represent here two most used crossover operators,

1-point crossing consists of dividing each of the two parents into two parts at the same position, chosen at random and copying the lower part from parent to child and filling in the missing genes from the child to from the other parent by maintaining the order of the genes.

The fundamental role of the cross operator is to copy and recombine information (genes) from the two parent chromosomes so as to form two new individuals possessing characteristics derived from both parents.

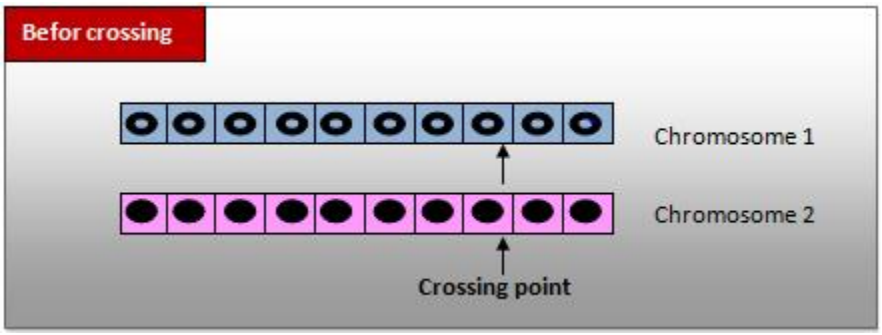
Conventional methods of crossing include crossing with a single point of cut (single point crossover) and one with two points of cut (two point crossover) which is proven to be more efficient.

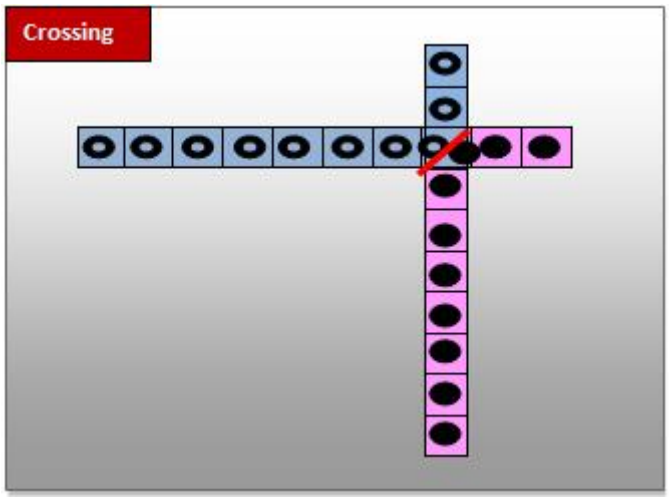
In order to increase the diversity of the population, we use the crossing with a single point, the position of the crossing site is a common position therefore the first child individual is obtained by copying genes from parent1 to the cut point, then completing with the genes of parent 2 up to the last point.

Conversely, we obtain the second child individuals.

We can imagine and test more or less complex crossing operators on a given problem but the effectiveness of these is often intrinsically linked to the problem.

Cross-over aims to enrich the diversity of the population by manipulating the structure of chromosomes. First of all, the meeting points of the two paths are determined, if the two paths do not meet, the son generated will be identical to the father; otherwise, we look for the meeting point which optimizes the journey: the same path as that of the father starting to the meeting point, then the same path as that of the mother from the meeting point at the end of the maternal path. We also define the reproduction of an individual with himself, the son then being identical to the father. This keeps the best parents in the next generation safe.





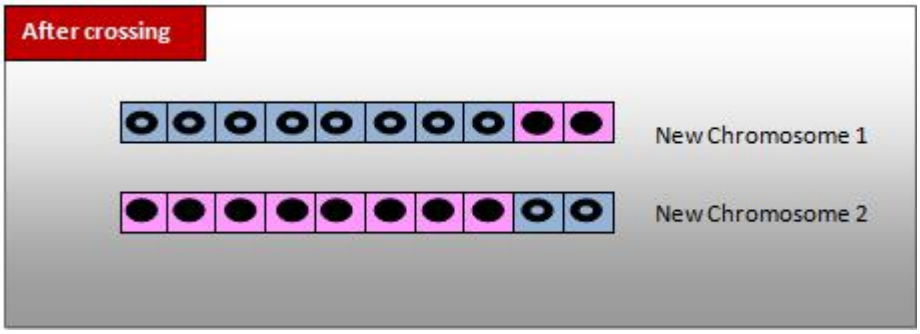


Figure 2.11: Crossing At One Point

2.4.4.3 Mutation:

The mutation provides the necessary for an exploration of space. It allows us to leave the local extremes.

The mutation operation generates recopy "errors", with the aim of creating a new individual who did not exist before. This allows the GA to avoid converging towards a local optimum of the function. Thus, and in order to increase the diversity of the population, the mutation operation is necessary for genetic algorithms despite its low probability.

In a classical genetic algorithm, it is often the random mutation that is used. The latter consists of randomly modifying the value of any gene of any chromosome. However, this random mutation can cause infeasible paths.

This property indicates that the genetic algorithm will be able to reach all points in the state space, without traversing all of them in the resolution process.

Thus in all rigor, the genetic algorithm can converge without crossing, and certain implementations function in this way.

The convergence properties of genetic algorithms are therefore strongly dependent on this operator on the theoretical level.

For discrete problems, the mutation operator usually consists of randomly drawing a gene from the chromosome and replacing it with a random value (see Figure 2.15).

If the notion of distance exists, this value can be chosen in the vicinity of the initial value.

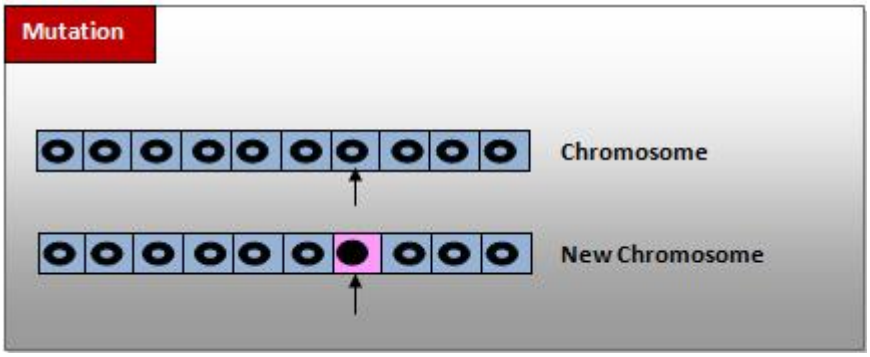


Figure 2.12: Mutation Operator

To simulate the presence of mutations, just after reproduction, the individual generated undergoes mutations randomly. The probability of mutation is 2% (this low rate is justified by the high probability

of falling on false paths or obstacles by mutating the boxes, i.e. the probability of having a broken path is 75%, and 25% for a full path), each of the mutations modifies the path concerned by adding a random number to it, then taking the result of the modulo 4 congruence which represents the 4 directions.

Figure 2.13: The Four Directions

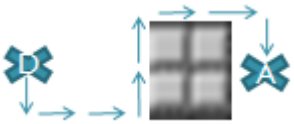


Figure 2.14: Obstacle Avoidance

2.4.4.4. Elitism: Is copying a few better people into the new population. The goal is to prevent the best individuals from being lost after the crossbreed and mutation operators. This method makes it possible to keep, at a given iteration, the best individual found in all the populations generated previously.

2.5. Stop criterion:

The stop test plays a very important role in judging the quality of individuals. There are two stopping criteria:

- stopping after a predetermined number of generations (in our case 100);
- It stops when there is no development in the population or it is no longer sufficiently developed.

Conclusion:

Inthis chapter, we describe our genetic algorithm for navigating a mobile, autonomous robot. A robustness criterion was developed. This criterion consists of finding a solution to good performance which is sensitive to uncertainties. The objective is to find the shortest route to reach the point of arrival.

We have presented the different diagrams: diagram of cases of initiation, sequence diagrams, activity diagram and class diagram. These diagrams made our work easier to understand the problem; to obtain a solution of the problem of movement of the mobile robot in spaces with obstacles has been proposed.

The simulation results presented give promising solutions for the autonomous control of the mobile robot especially in less complicated environments. In all cases, the robot is able to achieve its objective by avoiding obstacles.

The establishment of a genetic control system requires experience for a better design of this type of controllers.

That is why we have proposed the genetic algorithm to solve this problem (coding, initialization of solutions, selection, crossing, mutation and elitism).

In the next chapter we will present the tools and languages used in our application.

Chapter 3:
Realization and Implementation

3.1. Introduction:

In order to show the autonomous control of the mobile robot, we used an application developed under PYTHON which is an intelligent system with several functions used to simulate the behavior of the robot in obstacle avoidance to reach the target and to assess the performance of the techniques used.

Optimization issues are becoming more complex and the rapid development of technology has made the use of an evolutionary approach increasingly necessary.

In addition, the cost / performance ratio in parallel IT systems continues to decrease. The evolutionary approach is used in the design and implementation of meta-heuristics to accelerate research, and to improve the quality of the solutions obtained, to improve robustness and solve problems on a large scale.

In this thesis, our research is done on the problem of finding the shortest path of an autonomous mobile robot.

We have proposed a new evolutionary approach for solving the problem of finding the shortest path for an autonomous mobile robot, based on the combination of genetic algorithms.

3.2. Tools and working environments:

3.2.1. Software environment:

Python is the most widely used open source programming language for IT professionals. This language has propelled itself to the top of infrastructure management, data analysis or software development. Indeed, among its qualities, [29] Python notably allows developers to focus on what they do rather than on the way they do it.

Python offers several useful features for beginners and experts alike.

At first, it is easy to learn and use.Its features are few, which allows you to create programs quickly and with little effort. In addition, its syntax is designed to be readable and direct. Another advantage of Python is its popularity [29].

This language works on all major operating systems and computer platforms. In addition, even if it is clearly not the fastest language, [30] it compensates for its slowness with its versatility.

Finally, even if it is mainly used for scripting and automation, this language is also used to create professional quality software. Whether it's applications or web services, Python is used by a large number of developers to create software [29].

Figure 3.1: Python Program [image]

3.3. Description of the different modules

3.3.1. The environment:

The navigation space needs, in one form or another, a computer representation. There are two main types of maps: occupation grids and graphs. In the first case, the environment is squared iboxes.

In the initial state the matrix representing the environment with obstacles placed randomly, the black boxes which are obstacles carry the value 0 and the white boxes which are free boxes, carry the value 1 (Figure 3.2).

1	1	1	0	1	1	1	1	1	1	1	0	1
1	1	1	1	1	0	0	1	1	1	1	1	1
1	S	1	1	0	1	0	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	1	0	1	1	1	1
0	1	0	1	1	1	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	0	1	0	1	0	E	1	0	1
1	1	0	0	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	0	1	1

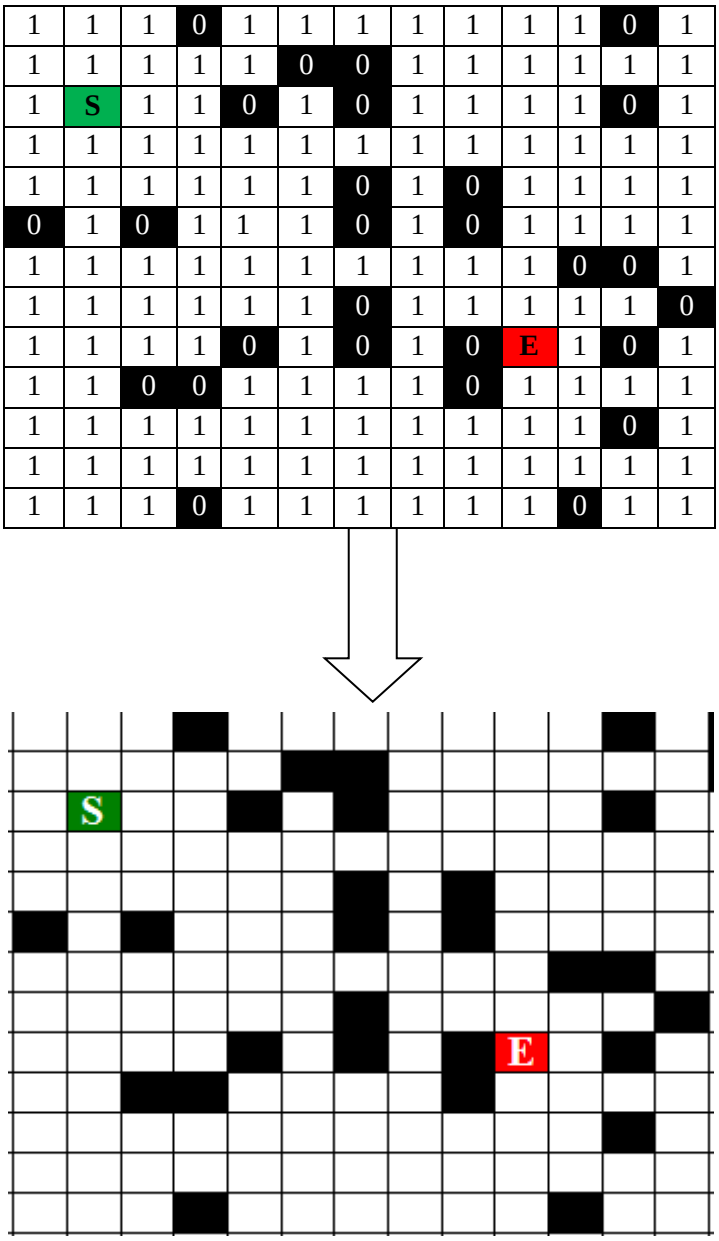


Figure 3.2: Diagram representing the passage from a Boolean matrix to a navigation environment.

3.2. Initialization of matrix:

We have a square matrix, with a starting point and ending point which are random and dynamic and random obstacles. We have a button to enter and determine the starting point and arriving.

In our work we proposed a matrix of 30 * 30 and PDepart and PArrive random.

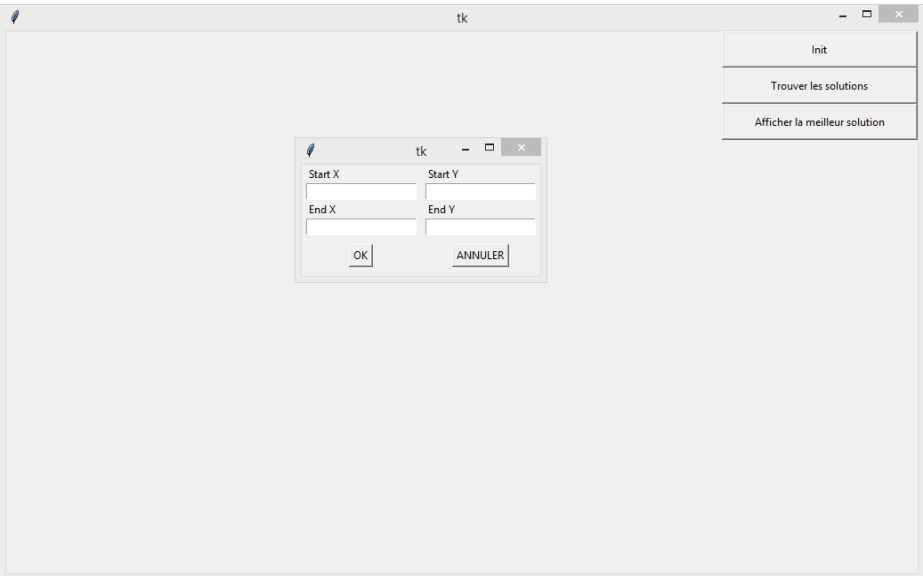


Figure 3.3: Interface of Our Application

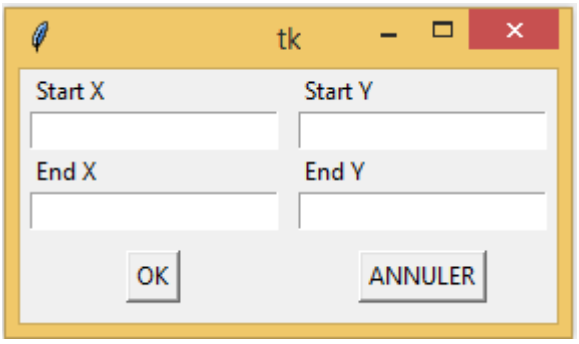


Figure 3.4: A Window to Enter the Point of Departure and Arrival

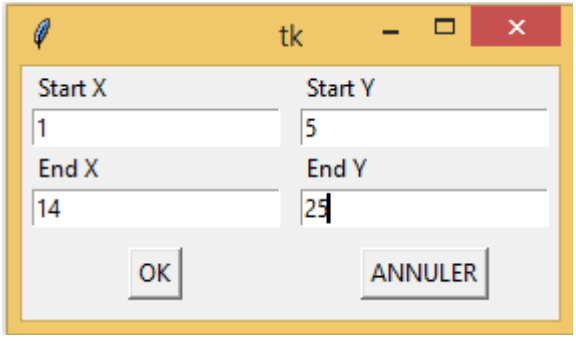


Figure 3.5: Enter the Coordinates of the Point of Departure and Arrival

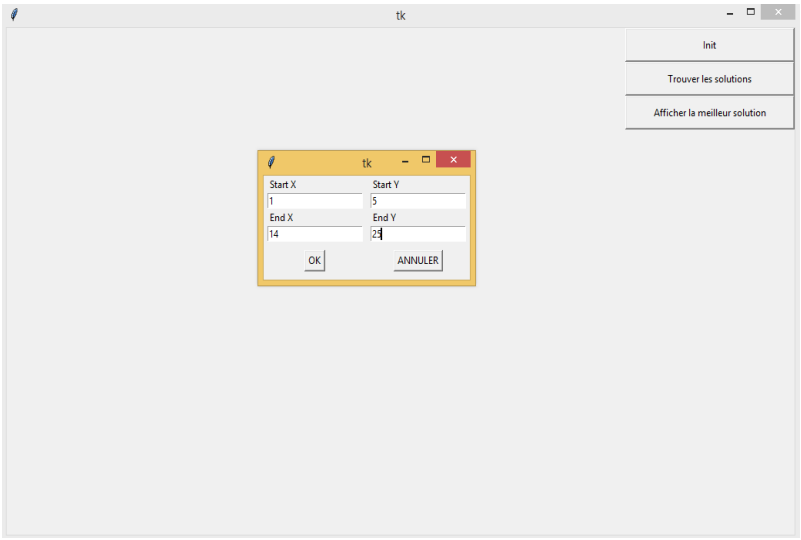


Figure 3.6: interface and window to enter data

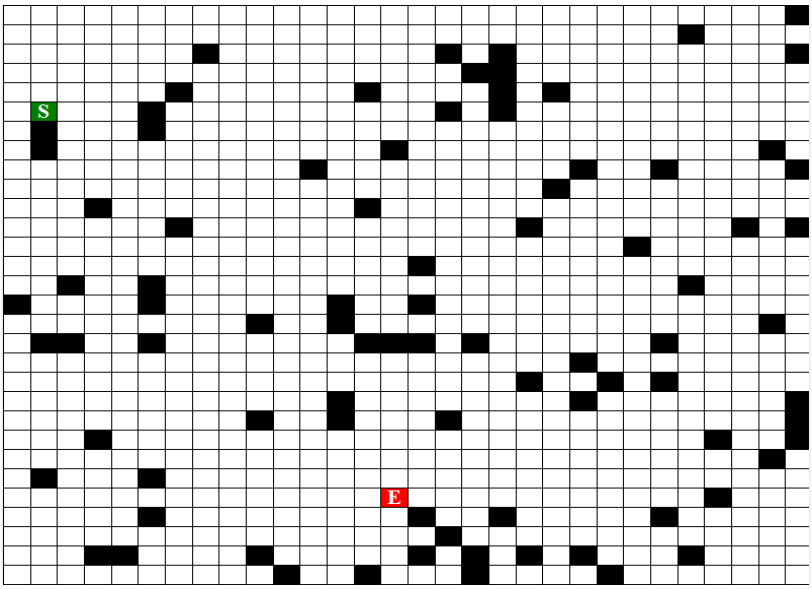


Figure 3.7: determine the starting point and the ending point.

3.3.3. Robot move:

For the implementation of a genetic algorithm of the shortest path, we were led to choose a modeling of our paths, to which we could apply these general principles. For this, we have modeled a path in the following form:

Path = {X1, X2, X3…, Xn}, where i's the robot's number, n is the length of the path and X an integer ϵ {0, 1, 2, 3}.

3.3.4. The paths

The path is a series of boxes where each box contains a pair representing the coordinates of the box (the location in the environment modeled by the matrix t) and an integer between 0 and 3, which indicates a moving the robot. The table is a series of movements that the mobile robot will make starting from the

starting point, hoping to reach the point of arrival. The length of the path is a performance indicator of the individual (fitness), that is to say the number of trips he must make to reach the finish (Table 3.1).

Figure 3.8: The Four Directions

Travel cost	0	0	0	0	0	0	0	1	1	1	1	0	3	3	0	1	1	1	1	2	.	.	.	0	1
The direction	→	→	→	→	→	→	→	↓	↓	↓	↓	→	↑	↑	→	↓	↓	↓	↓	←	·	·	·	→	↓

Table 3.1: Diagram representing a path found between 2 points

Python code:

```
# 0=right 1=down 2=left 3=up
if IsObstacle(currentY+1,currentX):
    directions.remove(0)
if IsObstacle(currentY,currentX+1):
    directions.remove(1)
if IsObstacle(currentY-1,currentX):
    directions.remove(2)
if IsObstacle(currentY,currentX-1):
    directions.remove(3)
if len(directions)>1:
    if prevDir>=0:
```

Figure 3.9: Python Code of Direction

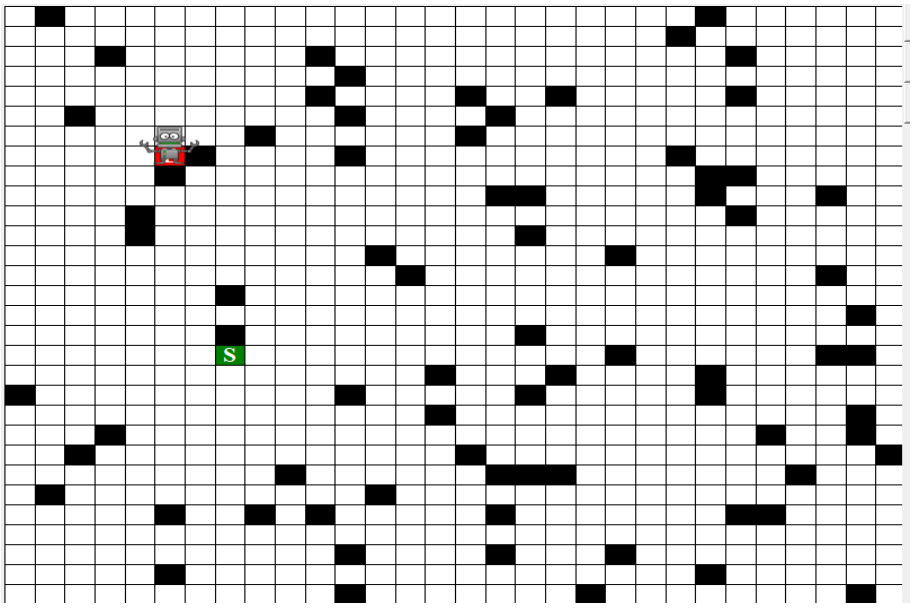


Figure 3.10: A Matrix that Represents the Navigation Environment

3.3.5. The proposed genetic algorithm

3.3.5.1. Initial population

This mechanism must be capable of producing a population of non-homogeneous individuals which will serve as a base for future generations.

The choice of the initial population is important because it can make convergence more or less rapid towards the global optimum.

In the event that nothing is known about the problem to be resolved, it is essential that the initial population be spread over the entire research area.

The population is generated randomly and corresponds to the possible routes to reach the point of arrival. Having an acceptable solution suggests considering 100 possible paths in order to reach the destination.

To solve the problem of premature convergence (local minimum) the algorithm generates a set of solutions (paths), for each generation.

The initial population is represented by a list of lists for more flexibility because the length of the path found is not always constant (Table 3.2).

Path 1	0	1	1	1	1	1	1	1	1	1	0	0	.	.	.	.	x
Path 1	0	0	0	0	0	0	0	1	1	1	1	1	.	.	.	.	x
.																	
.																	
.																	
Path 1	3	3	0	0	0	0	0	1	1	0	2	2	.	.	.	.	x

Table 3.2: Diagram Representing the Set of Paths Found Between 2 Points

Where n is the number of paths in each generation and x is a variable that designates the length of the path.

```
def initSolutions():
    startSolutions=[]
    x=0
    while x<100:
        p=findpath()
        if not p==[]:
            startSolutions.append(p)
            x=x+1
    return startSolutions
```

Figure 3.11: Python Code of Initial Population.

3.3.5.2. Fitness

This takes positive whole values and is called fitness or an individual's evaluation function.

```
def fitness(p):
    global PDepart
    global PArrive
    mindist=abs(PDepart[0]-PArrive[0])+abs(PDepart[1]-PArrive[1])
    if len(p)==0:
        return 0
    return mindist/len(p)
```

Figure 3.12: Python Code of Initial Population.

3.3.5.3. The selection operator

This operator is responsible for defining which individuals (paths) from population P will be duplicated in the new population P 'and which will serve as parents (application of the crossing operator). Let n be the number of individuals in the population P, we must select n / 3 (the crossing operator allows us to return to n individuals).

The elitist method This method consists in selecting the n individuals we need for the new generation by taking the n best individuals from the population P after having sorted it in an increasing manner according to the fitness of its individuals (length of the paths); indeed, the pressure of selection is too strong, the variance zero and diversity non-existent, at least the little diversity that there could be will not result from selection but rather from crossing and mutations.

3.3.5.4. Crossover and mutation operators

They allow to diversify the population over generations and to explore the state space. The crossover operator resets the genes of existing chromosomes in the population.

Crossbreeding is planned with two parents and generates two children. This involves exchanging the parents' genes to give children who have combined properties.

Although random, this exchange of information offers genetic algorithms some of their power: sometimes "good" genes from a parent replace the "bad" genes from a other and create sons better suited to parents.

At the start, the meeting points of the two paths are determined.

- If the two paths do not meet, the son generated will be identical to the father.
- Otherwise, we are looking for the meeting point that optimizes the journey: identical path to that of the father from the start to the meeting point, then identical path to that of the mother from the meeting point to the end of the maternal journey.

We also define the reproduction of an individual with himself, the son then being identical to the father. This keeps the best parents in the next generation safe.

Since randomness plays a large role, we can see that if in the majority of cases, convergence towards an ideal solution occurs very quickly (less than 100 generations), there are cases where this convergence is much slower, even if the route to be taken is actually quite short (for example the last case in the table above).

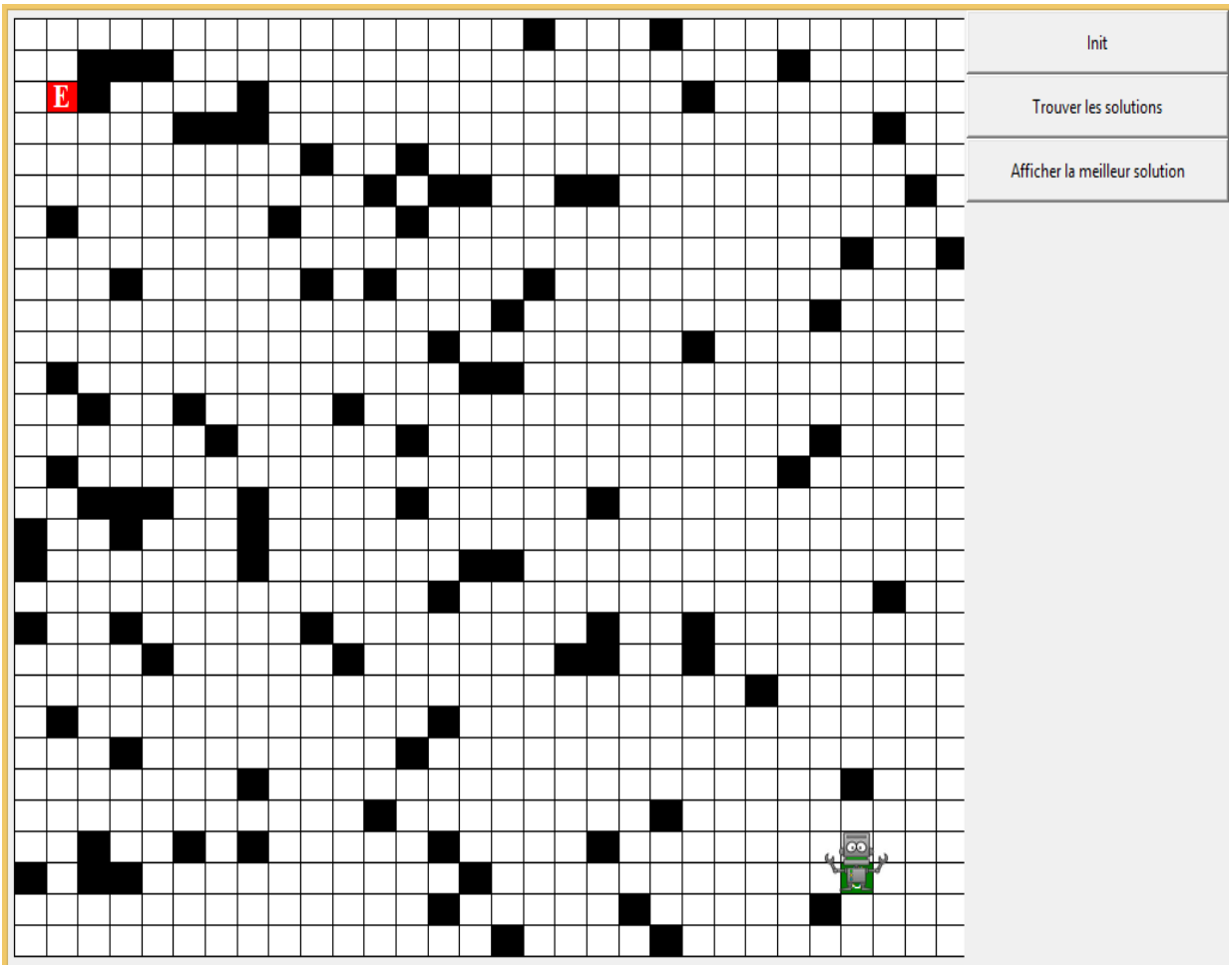
There are cases where 100 generations are not enough to obtain an optimal solution.

The probability of mutation is 2% (this low rate is justified by the high probability of falling on false paths or obstacles by mutating the boxes, that is to say, the probability of having a broken path is 75%, and 25% for a complete path), each of the mutations modifies the path concerned by adding a random number to it, then taking the result of the modulo 4 congruence which represents the 4 directions.

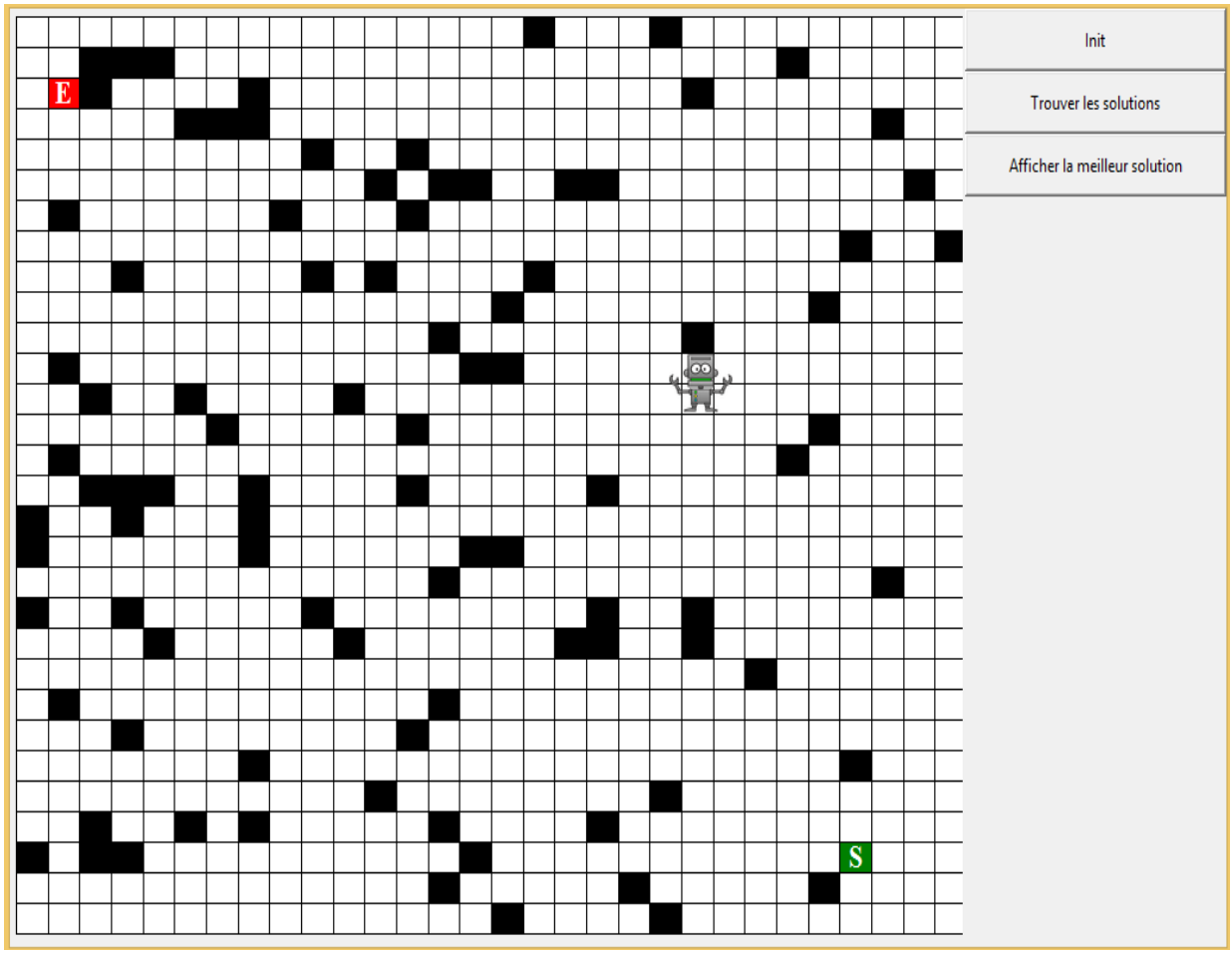
3.3.5.5. Use case:

Once the environment is established, the agents take their positions, in other words, each robot represents a thread and has a starting point and an ending point, these points can be determined randomly or by the user.

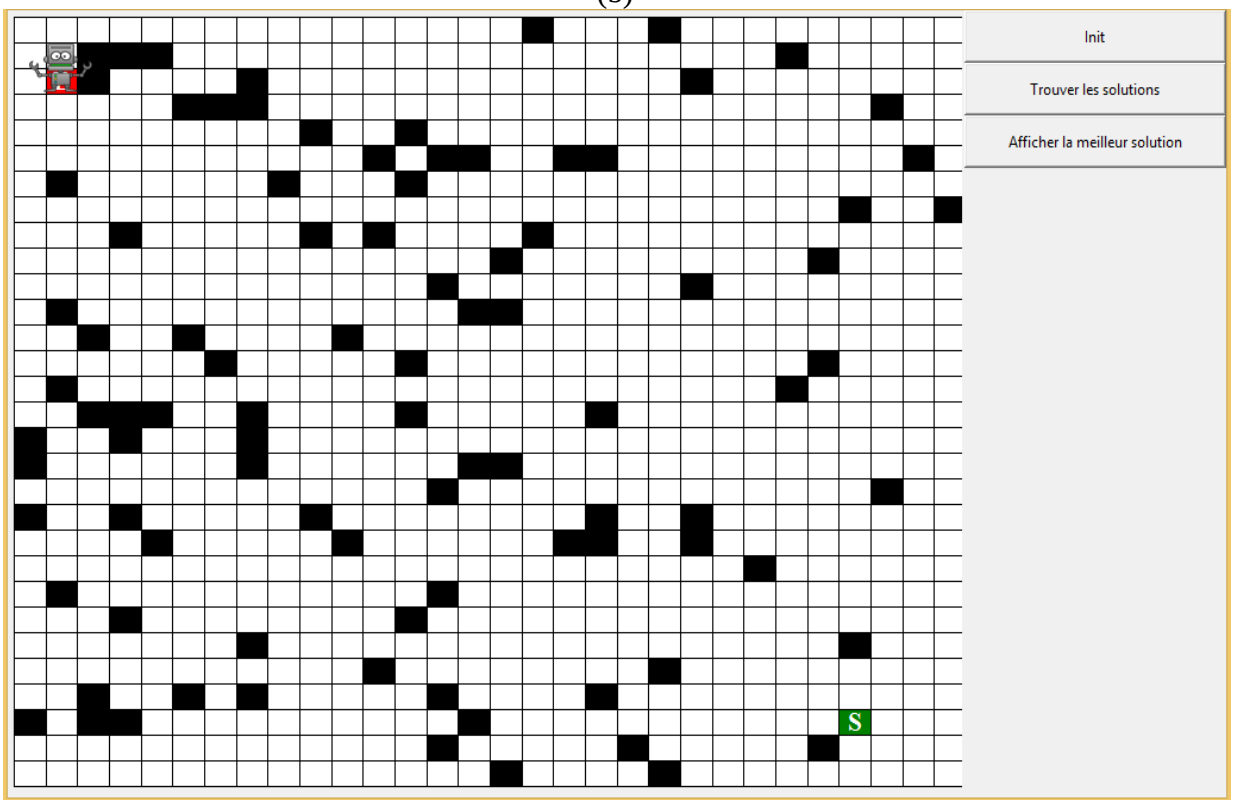
At first, the system provides a random set of obstacles; the user can add or delete them as needed, once the environment is configured, the user can then launch the execution (Figure 3.13).



(a)

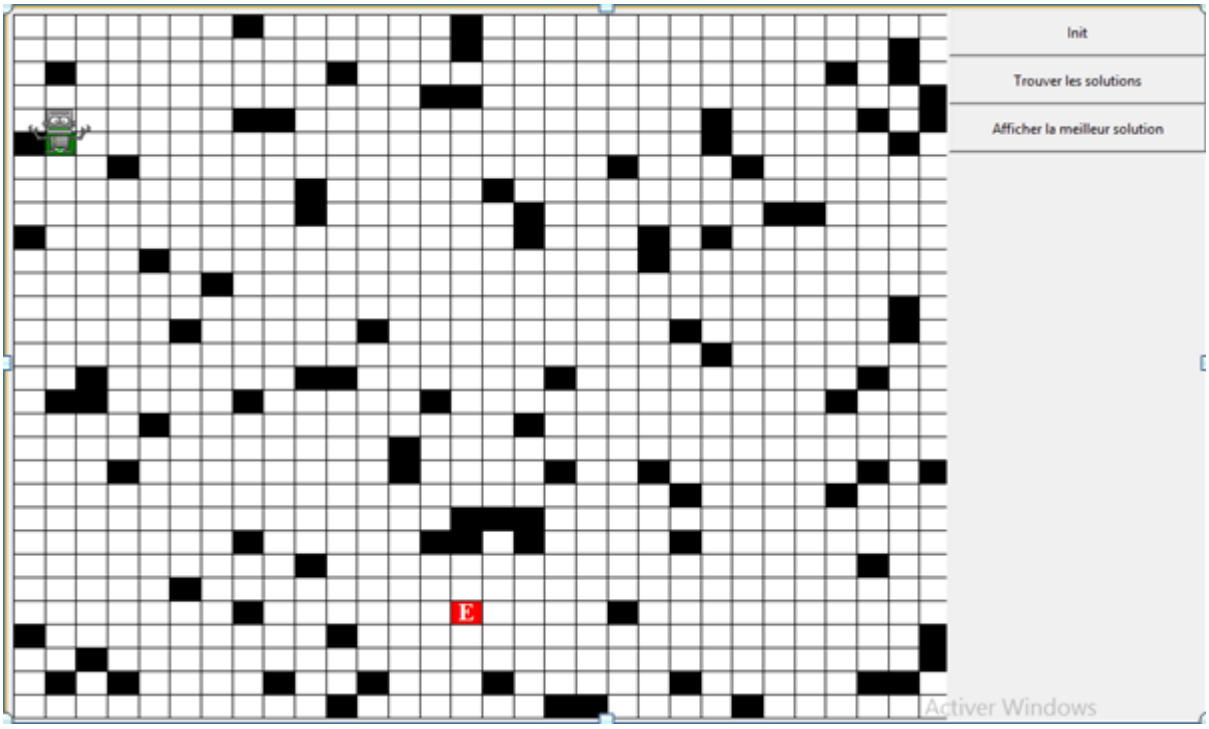


(b)

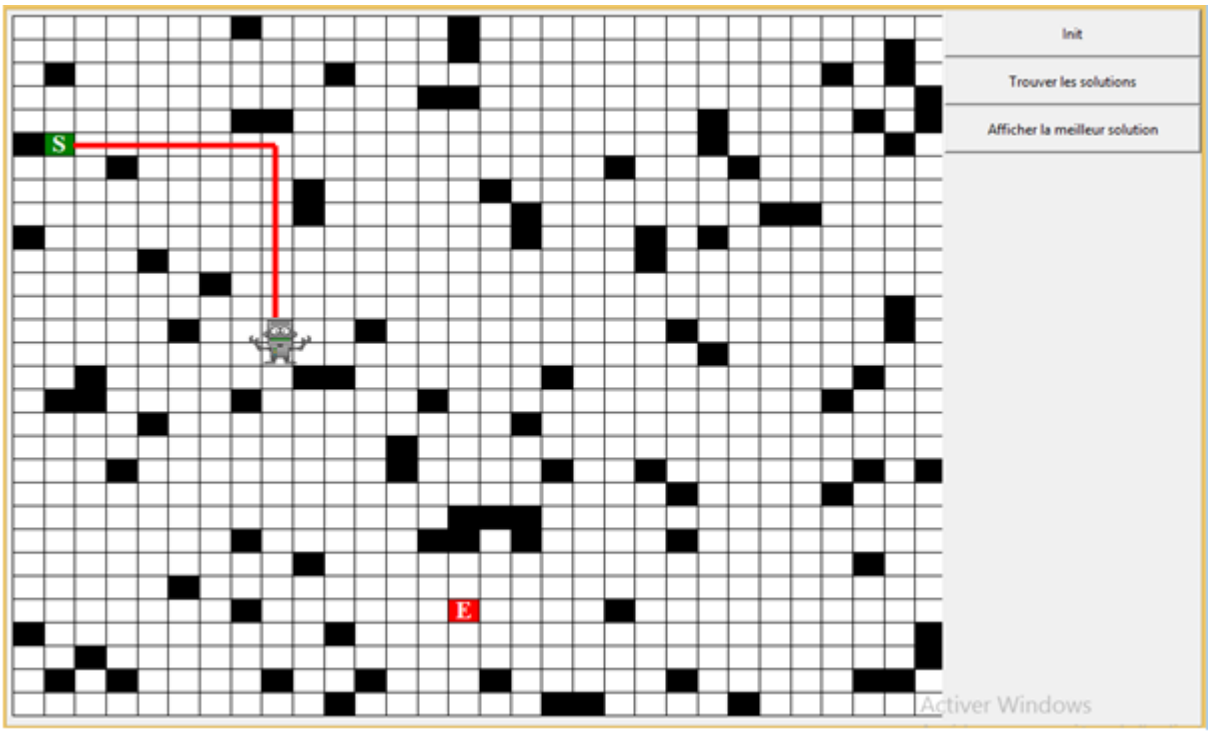


(c)

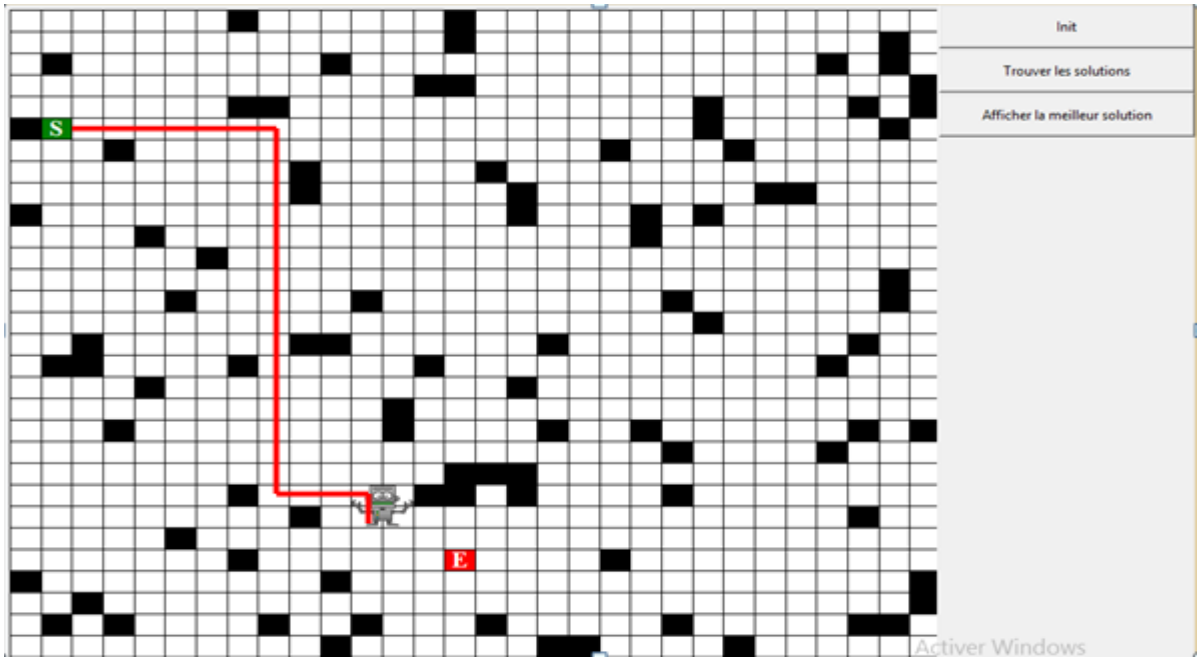
Figure 3.13:a, b and c: software interface


Init
Trouver les solutions
Afficher la meilleur solution
E
Activer Windows

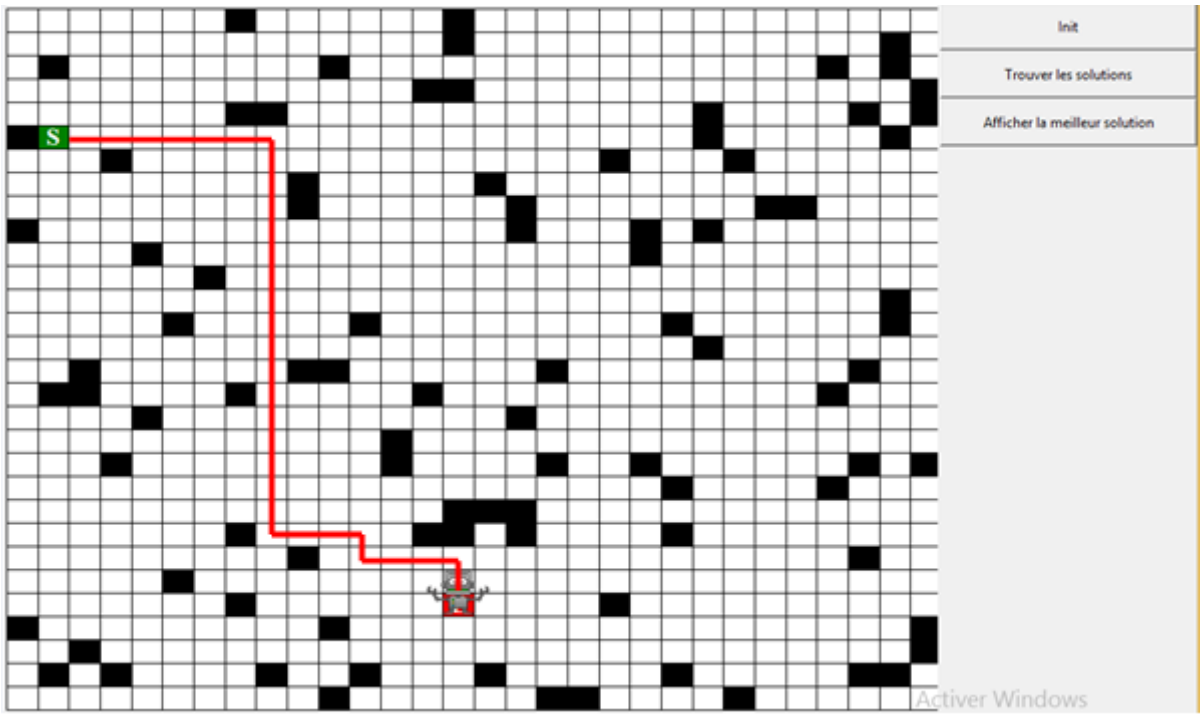
(a)


Init
Trouver les solutions
Afficher la meilleur solution
S
E
Activer Windows

(b)



(c)



(d)

Figure 3.14.a.b.c and d: the trajectory of robot

3.5. Comparison between different methods:

	Environment	Starting	Ending	Obstacles	Distance	path length	convergence rate
AG	[30*30]	1, 5	14, 25	90	33	37	89,19%
CF	[30*30]	1, 5	14, 25	90	33	40	82,50%
CA	[30*30]	1, 5	14, 25	90	33	42	78,57%

Table 3.3: Comparison of the Results of the Genetic Algorithm (Ga), Colony of Formi (Cf), Colony of Bees (Ca)

3.5.1 Discussion

The diversity of work and views has enriched the navigation domain. This makes it possible to identify a set of criteria to ensure the quality of such a solution.And to show the performance of the genetic algorithm to solve the shortest path problem, we made a small comparison between two other methods: ant colony [31] and bee colony [32], and we noticed that the rate of convergence in genetic algorithm is more interesting than other methods.

For example in an environment which is composed of a square matrix of 30 * 30, (starting point (1.5), ending point (14.25), 90 obstacles and distance equals 33).

The genetic algorithm was found to be the most efficient at the shortest path problem compared to others with a convergence rate equal to 89.19% (CF 82.50% and CA 78.57%).

GAs has several advantages which make it possible to use them in multiple fields but they also have disadvantages which impose restrictions on their use:

-The probability of crossing and mutation sometimes makes it possible to avoid falling into a local optimum and to move towards the global optimum.

-In general, GAs accelerates the course of a vast set of solutions. And bad solutions are not taken into account; they are eliminated so as not to affect the optimal solution. - High efficiency to solve all the optimization problems that can be described with chromosome coding.

- GAs deal with very complex problems in a very effective way and making it easy to find the right solutions, this characteristic has helped a lot in areas with a large number of parameters and whose response time must be reduced (obtaining good solutions in just a few iterations).

But:

- Finding the right solutions for problems with a large number of parameters is not always assured by GA because it is very slow.

- GA can converge towards the local optimums and not find a global optimum.

- This is an artificial intelligence technique, so the genetic algorithm cannot always ensure the same results in terms of time. This property limits their use in real-time applications.

Conclusion:

In this chapter, we have tried to present in a simple way the different stages through which we went.

A new evolutionary approach has been shown to find the shortest of an autonomous mobile robot.

This approach is able to guide the robot to move autonomously. The proposed approach improves the performance of navigation on several levels, especially concerning the initialization of the population.

We also made a combination of several methods allows us not only to cite the disadvantages of one method or another, but also to implement high level know-how for a robot.

In general, solutions based on this approach allow the resolution of our problem.

General Conclusion:

The problem of autonomous navigation of a mobile robot in natural environments has aroused growing interest in recent years, Roboticists seek to gradually increase the degree of autonomy of their robots, until reaching complete autonomy and suitable for very long missions.

The fully autonomous movement of a mobile robot in dynamic environments is a problem that is still difficult to solve.

It requires the implementation of functionalities enabling the perception / decision / action cycle to be carried out. To be able to cope with the wide variety of situations that the robot can during navigation.

In a general way, by this study we highlighted the interest of the intelligent system not only for the autonomous control of mobile robot, we used the PYTHON.

The tools provided by the python allow modeling of phenomena that can in some sense approach human reasoning.

The implementation of the genetic algorithm shows that the objective has been achieved by avoiding obstacles.

We have found that genetic algorithms are not necessarily the best; they can be optimized for solving problems of the same type.

Several perspectives are possible following our work. So, it would be interesting to introduce "dynamics" into our study of autonomous navigation, taking into account mobile obstacles, which will raise particularly delicate problems.

This will allow us to detect a greater quantity and variety of obstacles in order to avoid any disruption of navigation due to an unforeseen change in the environment.

References:

[1] Schwartz, J, T.,Sharir M.. On the Piano Movers' Problem : I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers. Communications on Pure and Applied Mathematics, vol. 36, pages 345–398, 1983a.

[2] Schwartz, J. T. Schwartz & M. Sharir. On the Piano Movers' Problem : II. General Techniques for Computing Topological Properties of Algebraic Manifolds. Advanced applied Mathematics, vol. 4, pages 298–351, 1983b.

[3] Schwartz 1983c J. T. Schwartz & M. Sharir. On the Piano Movers' Problem : III. Coordinating the Motion of Several Independent Bodies. International Journal of Robotics Research, vol. 2, no 3, pages 97–140, 1983.

[4] Latombe 1991 J.-C. Latombe. Robot motion planning. Kluwer, Boston, MA, 1991.

[5] LaValle 2006 S. M. LaValle. Planning algorithms. Cambridge University Press, Cambridge, U.K., 2006.

[6] Choset 2005 H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki & S. Thrun. Principles of robot motion : Theory, algorithms, and implementations. MIT Press, Cambridge, MA, 2005.

[7] Holland J.H., 1975. Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor

[8] Goldberg D.E., 1987. Genetic algorithms in search, optimization, and machine learning, Addison Wesley

[9]Hachour, O., 2008. Path planning of Autonomous Mobile Robot, *International Journal of Systems* Applications, Engineering & Development, vol. 2, no.4, pp.178-190.

[10] Latombe, J, C., 1991. Robot Motion Planning, *Kluwer Academic Publishers*, Norwell.

[11] Khatib, O., 1990. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, IEEE International Conference on Robotics and Automation, St. Louis, Missouri, pp.500-505, March 25-28.

[12] Andrews, J, R., Hogan, N., 1983. Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator, *Control of Manufacturing Processes and Robotic Systems*, ASME, Boston, pp. 243-251.

[13] Brooks, R, A., 1986.A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1,pp. 14-23.

[14] Arkin, R, C., 1989. Motor Schema-Based Mobile Robot Navigation, the International Journal of Robotics Research, pp. 92-112.

[15] Adachi, Y., Saito, H., Matsumoto, Y., Ogasawara, T., 2003. Memory-based navigation using data sequence of laser range finder, Computational Intelligence in Robotics and Automation, Proceedings IEEE International symposium, Vol 1, pp. 479 – 484.

[16] Lettvin, J, Y., Maturana, H, R., McCulloch, W, H., 1959. What the Frog's Eye Tells the Frog's Brain, *Proceedings of The Institute of Radio Engineers*, New York, Vol. 47, No. 11, pp. 1940-51.

[17] Lebedev, D, V., Steil, J, J., Ritter, H, J., 2005. The dynamic wave expansion neural network model for robot motion planning in time-varying environments, *Neural Networks*, Volume 18, Issue 3, pp. 267-285.

[18] Belker, T., Schulz, D., 2002. Local Action Planning for Mobile Robot Collision Avoidance, intelligent Robots and System, *IEEE/RSJ International Conference on Volume 1*, 30, Page(s):601 - 606 vol.1.

[19] Zadeh, L., 1965.Fuzzy sets", *Information and Control*, 8(3):338- 353.

[20] Ouadah, N., Azouaoui, O., Hamerlain, M., 2005. Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type voiture, *Troisième Congrès francophone*, Majecstic, Rennes, France.

[21] Chatterjee, R., Matsuno, F., 2001. Use of single side reflex for autonomous navigation of mobile robots in unknown environments", Robotics and Autonomous Systems, Volume 35, Issue 2, pp 77-96.

[22] Xu, W, L., 1999. A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviourbased mobile robot, Institute of Technology and Engineering, College of Sciences, Massey University, Palmerston North, New Zealand.

[23] Mellouk, A., and Benmachiche, A., 2019. "A Survey on Navigation Systems in Dynamic Environments", In Proceedings of ACM ICIST conference, Hammamet, Tunisia - 27-28 December (ICIST '2020).

[24] Lucic, c., Teodorovic, D., 2001.Optimisation des colonies d'abeilles (BCO) Université de Belgrade, Faculingénieur des transports et du traficg, Vojvode Stepe 305 11000 Belgrade, Serbeia dusan@sf.bg.ac.rs

[25] Yuce, B., Packianather, M., Mastrocinque, E.,TruongPham, D., and Lambiase, A., 2013. Honey Bees Inspired Optimization Method: The Bees Algorithm, *Insects 2013*, 4, 646-662; doi:10.3390/insects4040646.

[26] Karaboga, D ., Basturk , B., 2007. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, *International Fuzzy Systems Association World Congress, IFSA 2007: Foundations of Fuzzy Logic and Soft Computing pp 789-798*

[27] Benmachiche, A., Bouhadada, T., Laskri M, T., Zendi, A. 2016. A dynamic navigation for autonomous mobiles robots, *Intelligent Decision Technologies*, ISSN 1875-8843 (E), 10 (1).

[28] XinShe, Y., 2005. Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK xy227@eng.cam.ac.uk

[29] Gérard, S., 2012. Appredre A Programmer Avec Python 3. Page(s) : 449.

[30] Fernad, E., Marcela, M., Bernard, P., Thierry, S.2005. A1- tenir compte du caractére évolutif des TIC. C2IMES.

[31] Hdimi, K.., 2019. Approches bio-inspirées pour la coordination de robots dans un milieu hostile : colonie des fourmis, Master Académique SII, université chadli Bendejdid el Tarf.

[32] Belkaid, N., 2019. Approches bio-inspirées pour la coordination de robots : colonie d'abeilles, Master Académique SII, université chadli Bendejdid el Tarf.

.

ICSENT'2019

8th International Conference on Software Engineering and New Technologies

Certificate of Attendance

Awarded to

Ahlam Mellouk

For attending:

8th International Conference on Software Engineering and New Technologies **"ICSENT'2019"**
Hammamet, Tunisia - 28 – 30 December 2019

and presenting the paper entitled: **A survey of methods for mobile robot navigation in dynamic environments**

Authors: **Ahlam Mellouk and Abdelmajid Benmachiche**

ICSENT'2019 General Chair