Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 127 (2018) 180–189

Procedia Computer Science

www.elsevier.com/locate/procedia

The First International Conference On Intelligent Computing in Data Sciences

Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning

Chaymaa Lamini[a], Said Benhlima[b], Ali Elbekri[a,b,*]

[a]FSM, Meknes 5000, Morocco
[b]FSM, Meknes 5000, Morocco

Abstract

In this study, an improved crossover operator is suggested, for solving path planning problems using genetic algorithms (GA) in static environment. GA has been widely applied in path optimization problem which consists in finding a valid and feasible path between two positions while avoiding obstacles and optimizing some criteria such as distance (length of the path), safety (the path must be as far as possible from the obstacles) ...etc. Several researches have provided new approaches used GA to produce an optimal path. Crossover operators existing in the literature can generate infeasible paths, most of these methods dont take into account the variable length chromosomes. The proposed crossover operator avoids premature convergence and offers feasible paths with better fitness value than its parents, thus the algorithm converges more rapidly. A new fitness function which takes into account the distance, the safety and the energy, is also suggested. In order to prove the validity of the proposed method, it is applied to many different environments and compared with three studies in the literature. The simulation results show that using GA with the improved crossover operators and the fitness function helps to find optimal solutions compared to other methods.

© 2018 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/). Selection and peer-review under responsibility of International Neural Network Society Morocco Regional Chapter.

Keywords: Genetic algorithm; path planning; crossover operator; navigation; mobile robot;

1. Introduction

An autonomous mobile agent is an intelligent vehicle which is able to perceive the workspace, extract and interpret the significant data coming from its sensors,locate its position in the environment, recognize and plan. The agent must be able to produce a feasible path from a starting point to a target position, and decide how to act to reach that path, and finally the movement control; the agent must has a means of control to regulate its movement in order to complete the planned trajectory [1]. Autonomous mobile agents have become an indispensable asset in industry, transport, agriculture and even in everyday life. For example they are widely used in transportation, cleaning and maintenance, assisting a person with limited mobility...[2]. Thus, to achieve the above cited tasks, the mobile agents must operate in

* Corresponding author. Tel.: +2126-129-66630
E-mail address: lamini.chaymaa@gmail.com

1877-0509 © 2018 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/). Selection and peer-review under responsibility of International Neural Network Society Morocco Regional Chapter.
10.1016/j.procs.2018.01.113

an intelligent and autonomous way, and must be armed with perception, location, planning and navigation capabilities, which allows the mobile agent to make decisions autonomously based on the information extracted by the perception of its environment. In this study we focus on motion planning which is an important task in robotics, that aims at perceiving the environment via sensors, extracting information (obstacle positions , starting position, arrival position) that will help the agent to build its map , locate its position in its environment, then the agent executes a planning method to find the optimal path between the two positions and finally navigate from a starting point to a target point while avoiding obstacles [3]. Path planning involves finding a path between two configurations by optimizing a number of criteria such as distance, energy, safety, and time. The generated path must be efficient (the agent gets to the point quickly) and secure (obstacle avoidance) [2]. Path planning can be either global or local planning. Global path planning consists in finding a reference path. It is determined before the agent begins its navigation by considering the environment as static. The second level (local planning) is based on the global path and data of the environment detected by the sensors in order to form a dynamic occupancy grid. The reference path is chosen in the first time, but it will be changed according to the changes happening in the environment so as to ensure a fast and secure navigation while avoiding fixed and mobile obstacles. Global planning is a very important step in mobile agent navigation, as it is difficult to ensure the convergence toward the target in dynamic environments. Indeed, local planning may be trapped in local minima where it can be difficult to get out without having the global path (reference path) to reach the goal. There are several methods and tools in the literature for the autonomous mobile agents to solve this problem. These methods can be classified according to three approaches; deliberative, reactive and hybrid approaches (deliberative-reactive) [4]. Deliberative approaches operate on static environments, where the input information is provided by the user or from the sensors to produce a valid path before the agent begins its navigation. This type of approach is executed in a succession of actions. The perception involves the understanding of the environment based on a model given by the user or the information detected by the sensors. Then the planning module runs. This module is used to identify the position of different obstacles in the environment using the model found in the perception phase to find an optimal path between two positions. Finally, the agent starts its navigation according to the trajectory found [5] [6]. When planning is local, and the environment is uncertain, the use of a reactive method [7]is a necessity. In this approach the perception module works instantaneously and in real time to perceive the state of the current position of the mobile agent. At every moment the sensor provides the next action to realize, without having a prior knowledge about the location of the fixed and mobile obstacles. This approach differentiates by the absence of the building modules of the environment and planning. The hybrid approach is the combination of both deliberative and reactive approaches to enjoy the benefits of each one.When the deliberative module takes care of the global planning (global level), the solution found will be sent to the local level which is the reactive module to execute it and refine it according to the data it receives from the sensors [1]. In this work, we are interested in the global planning where all the information of the environment is known in advance and before the navigation process, we adopt the deliberative approaches using genetic algorithms (GAs). This paper is organized as follows: the first section will highlight the background of this work by giving a theoretical presentation of genetic algorithms, while the second section will present the proposed path planning algorithm. The last section will then sum up the simulation result and the comparison between our method and those existing in the literature.

2. GENETIC ALGORITHMS FOR PATH PLANNING

2.1. Genetic Algorithms:

Genetic algorithms (GAs) were originally developed by John Holland in 1960 [8]. GAs are natural inspired algorithms which are based on the concepts of Darwinian evolution that involve an initialization method, fitness function to evaluate each chromosome, natural selection, crossover, and mutation operators [9] [10]. The GAs begins by generating randomly an initial population which represents the possible solutions(chromosomes) to the problem to be optimized [11]. Each chromosome is then evaluated by an adaptation function to determine the quality of every potential solution [12]. After that, genetic operators are used to create the new progeny; selection to choose the parents which will be subjected to the reproduction according to their adaptation values [13]. Later crossover is applied to products new offspring by recombining data from the two parents selected in the previous step [14]. Mutation is used to guarantee the diversity of the population by changing the genetic structure of some individuals according to a mu-

tation rate [15]. This evolutionary cycle is repeated until the stopping criteria is satisfied, which can be the number of generations previously fixed otherwise, or the algorithm could be stopped when the population does not involve quickly enough. The steps of the standard GAs [16] are detailed in **Algorithm** 1.

Algorithm 1 Pseudo-code of the standard genetic algorithm

```
Input : N : Population size; P_c : Crossover rate; P_m : Mutation rate.
Output : Best Chromosome.
t ← 0
Initialize arbitrarily the initial population P(t)
while (not termination condition) do
    Evaluate P(t) using a fitness function
    Select P(t) from P(t − 1)
    Recombine P(t)
    Mutate P(t)
    Evaluate P(t)
    Replace P(t − 1) by P(t)
    t ← t + 1
end
```

A GA is executed the following five components:

- A suitable genetic representation for individual (chromosomes).
- A method to generate the initial population.
- A fitness function to evaluate the quality of each potential solution.
- Genetic operators that modify the genetic composition of parents to produce a new offspring.
- The choice of the values of the various GA parameters (population size, cross over rate, mutation rate, stopping criteria... etc.)

2.2. Proposed approach:

A new GA approach is suggested to solve path planning problems in the 2D environment. The strength of genetic algorithms lies in the cooperative use of different genetic operators (selection, crossover, and mutation). The choice of genetic operators is a crucial step for the convergence of the genetic algorithm.Thus we tried to improve the classic crossover operator. In our approach we propose an improved crossover operator that uses parents with good fitness to increase the quality of progeny as well as parents with low fitness value to explore the research space and ensure diversity among the population.This operator also takes into account variable length chromosomes.

The components of the proposed algorithm are:

- **The Genetic Representation (Encoding of chromosomes):** The application of genetic algorithms to a given problem requires a genetic coding of potential individuals of the initial population. Binary encoding was the most widely used in GAs [8]. Different type of encoding can be used, depending on the problem to be solved. The fitness function and the genetic operators act on this coding and therefore the choice of a coding way is a very important step for the success of the behavior of the GAs [17]. The potential solutions of the path planning problem are paths (Positions set) which connects two known positions (starting point and a target point). The nodes of this path must be valid, i.e. nodes that don't match with any obstacle. In our proposed approach, we have modeled each potential solution by a sequence of ordered positions P (x, y), starting from the first position (X1,Y1), and ending at the target position (X2,Y2), where x and y are the coordinate of each position in the 2D environment. The length of chromosomes (number of intermediate nodes) is variable. Fig 1 shows an example of the path encoding.

X1,Y1	X2,Y2	X3,Y3	X4,Y4	.....	Xn,Yn

Fig. 1. Genetic representation of the path(Chromosome)

- **Representation of the environment:** While the path planning process, the mobile robot interacts with its environment to make decisions. Thus, it needs a model of representation of the environment. Several representations of the environment have been used by researchers in mobile robotics, as the occupancy grid, the polygon, the cell tree representation.. etc. In [18], they used the polygon representation to model the environment, since it offers a good approximation of the environment. Adem and Mehmet [15] used the grid-based representation with an orderly numbered grid. Researchers in this study[19] have used the grid map, where the environment is segmented into a set of cells of the same size. In our method, we apply the occupancy grid presentation Fig. 2(a), in which the robot environment is represented by 2D matrix,in which each position (x, y) in the grid has two likely values: 0 for free cells, and 1 for occupied ones Fig.2(b). An appropriate solution is a path (set of positions) from the starting point to the goal point crossing a set of free positions.

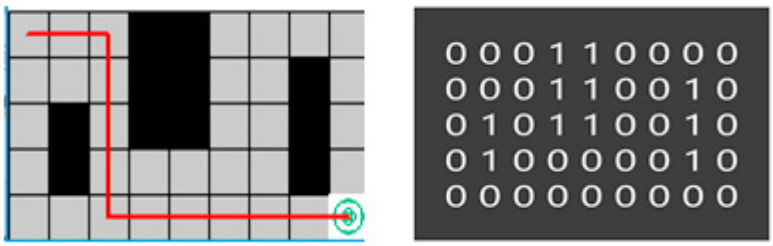


Fig. 2. (a) The occupancy grid based environment of 29x30 dimension; (b) Matrix environment.

- **Initial population:** As GA begins its searching process for the optimal path by acting on the initial population which is as set of potential candidates, the initialization method is a very important step since it alters the efficiency of the GA. Hence the choice of an efficient initial population method enhances the GA search effectiveness [20]. Different methods that exist in the literature are adopted to get the initial path set such as random, key cells, Clearness Based Roadmap Methods(CBRM), random walk, potential field, and greedy approach methods [18] [21][22] [23]. The initial population is usually created randomly in the standard GAs. However, the use of a random process causes invalid solutions, which increases the convergence time of the algorithm. Thus, researchers have proposed new constructive methods that allows having only valid solutions in the initial population [24], They have also used a combination of random and constructive methods to construct the initial population of GA [25]. The size of the population is a parameter that is generally fixed during the execution of the GA, but there are modified versions of GA where the size is dynamic [26]. The choice of the value of this parameter is a main factor which determines the quality of the convergence of the GA [27]. In our approach, we adopted the method proposed in [24] for the creation of the initial population with a fixed population size throughout the algorithm execution. This method is based on a directed acyclic graph (DAG). It explores the environment which is modeled as a grid, and then generates several feasible paths that will constitute the initial population of the GA.
- **Fitness function:** Once the initial population is created, the GA must determine the performance of each individual by using an adaptive function, which assigns to each possible solution a fitness value that reflects its quality. The FF must consider several criteria, such as distance, safety, smoothness..etc.The definition of a suitable fitness function is a crucial task because the GA uses the information generated by this function to choose the individuals for reproduction, mutation, and at the end of the GA process choose the best solution in the final population according to its fitness value [28]. There are four criteria to be taken into account in the proposed fitness function: path length, Safety First Level (SFL), Safety Second Level(SSL) and energy. A path P with N positions is evaluated based on the following function:

$$F(p) = \frac{1}{w_l \cdot l(p) + w_{s1} \cdot s_1(P) + w_{s2} \cdot s_2(P)} - e \tag{1}$$

Where :

- $l_i(p)$: path length, this value is calculated using the Euclidean distance.
- $s_1(p)$: Safety First Level(SFL), This value is calculated as :
  $s_1(p)= \sum_{i=1}^{N-1} s_i$
  s_i receives a penalty of (+1) if an obstacle is in the first security level of the current position $p(x_i, y_i)$ of the mobile robot Fig.3
- $s_1(p)$: Safety Second Level(SSL), This value is calculated as :
  $s_2(p)= \sum_{i=1}^{N-1} s_i$
  s_2 receives a penalty of (+1) if an obstacle is in the second security level of the current position $p(x_i, y_i)$ of the mobile robot Fig. 3
- e : energy, penalty of energy consumption, e receives a penalty of (+1) if the robot turns, and (0) if it continues forward.
- wl, w_{s1}, and w_{s2} are the weights of the length, SFL and SSL properties for a given path p.

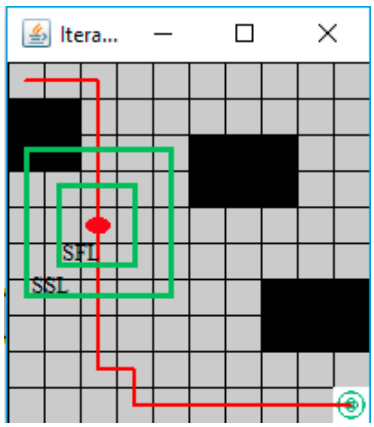


Fig. 3. The SFL and SSL parameters

- **Selection operator:** Selection is a genetic operator used to choose the parents who will survive to produce the next generation. Parents with the best fitness values are more likely to be selected for mating. There are different selection methods that can be used: Elitism Selection, Tournament Selection (TOS), Roulette Wheel Selection (RWS), Stochastic Universal Sampling(SUS), Linear Rank Selection (LRS), Exponential Rank Selection (ERS), and the Truncation Selection (TRS) [28]. The main objective of the selection operator is promoting individuals with high adaptability to be selected for the next generation. The selection pressure is a very important criterion, which strongly influences the performance of GA [13]. If the selection pressure is high the GA converges quickly without exploring all the search space, and a low selection pressure produces random solutions. In our approach; Elitist and (TRS) are used to control the pressure selection; elitist which has high pressure selection is used to keep the fittest solutions throughout generations, and TRS is used to give chance to weak chromosomes to be selected from the last generation for reproduction in the current one, and to avoid the dominance of the best individual.
- **Crossover operator:** After selecting individuals using the selection operator, the crossover is applied. ”Crossover” is a genetic operator that blend the genetic information(genes) of the two selected chromosomes (parents) to yield new chromosomes(offspring/child), to keep the heterogeneity of the population and to boost the fitness value of the candidate solutions. The main idea behind crossover is that new chromosomes inherit the

best characteristic of their parents. Thus, the result is having better child that performs better than its parents. The crossover rate is the probability of performing crossover [29]. Different crossover operators have been introduced; the Partially-Mapped (PMX), the Order Crossover (OX), the Cycle Crossover (CX), and Same Point (SP) crossover. SP seems to be the most used mechanism. [30] Munemoto in his work has used the standard crossover mechanism (SP) crossover which holds two crossover strategies:the one-point and the two-point crossovers that are applied if there are at least two identical genes between the parents. Many improved crossover operators are proposed to solve path planning problems. [19] presents a modified crossover operator to avoid dropping in a local minimum, by choosing two mutual nodes (N1 and N2) from two parents which are selected arbitrarily (P1 and P2), and picking the existing parts before N1; between the two selected nodes N1 and N2; and after the second node N2. The greatest parts are chosen to form a new solution(path). An improved (SP) crossover operator is suggested in this paper [31] called Same Adjacency crossover (SA); where they dont compare each single node but two nonbearing nodes instead to solve path optimization problems. The SA crossover would be improved in our approach. Therefore, an improved Same adjacency crossover operator (ISA) is proposed as shown in algorithm 2.

Algorithm 2 Pseudo-code of the proposed crossover operator

1: **Input :** $P1$: First parent;$P2$: Second Parent ; V_F : Length of P1, V_S : Length of the second parent.
2: **Output :** Fitest two paths.
3: **for** $i = 1$ `to` V_F **do**
4: **for** $k = 1$ `to` V_S **do**
5: **if (isFeasible**$(V_{i+1}, V'_j) =$ **True** and **isFeasible**$(V'_{j+1}, V_i) =$ **True) then**
6: $CrossPointList \leftarrow V_i$
7: $MatchPointList \leftarrow V'_j$
8: **end**
9: **end**
10: **end**
11: **for** $k = 0$ `to` $CrossPointListSize$ **do**
12: $offsprin_1 \leftarrow (V_0, V_1, ..., V_{i-1}, V_i, V'_{j+1}, ..., V'_S)$
13: $offsprin_2 \leftarrow (V'_0, V'_1, ..., V_{j-1}, V'_j, V_{i+1}, ..., V'_F)$
14: $child \leftarrow offsprin1 \cup offsprin_2$
15: **end**
16: sort(child) ▹ Sort childList based on the fitness value.

The ISA crossover can be outlined as follows:

- **Step1:** The two parents P1 and P2 are selected randomly **(Line1)** from two lists L1 and L2. L1 contains paths that are selected from the last generation, used elitist selection, which select only paths with hight fitness value. While L2 includes paths that are selected using TS, which pick up paths with different fitness values (hight, medium and low).
- **Step2:** Considering P1 (V1,V2,V3,..,Vi) and P2(V1,V2,V3, .., Vj). If the positions (Vi+1,Vj) are feasible and (Vj+1,Vi) are feasible **(Line 5)**, then the index i is added to the crossPointList, and j is added the matchPointList.
- **Step3:** Instead of choosing randomly and index i from the crossPointList and j from the matchPointList as in [31], a list of potential offsprings is generated (childList), using each index in the crossPointList and its homolog index in the matchPointList as a cross point pair (Vi,Vj), and generate two offsprings (Line 9,10) that would be added to childList **(Line 14)**.
- **Step4:** The childList is ordered, and then the two fittest paths are chosen **(Line 16)**.

• **Mutation operator:** Mutation is a genetic operator which is applied to improve the diversity and to prevent premature convergence of the algorithm. Generally, this operator randomly selects a position (gene) and replaces it with a new gene that is does not exist in the path. Yet, as mentioned in [19] random mutation could generate invalid paths. Even if a solution is valid before the application of the mutation operator, the new gene altered

can contain an obstacle and then it creates an inappropriate path. In this study, we adopt the mutation operator proposed in this paper [15].

3. SIMULATION RESULTS

3.1. Configuration

In this section, we presents the simulation model that we used to evaluate the performance of the proposed GA path planning method in different grid environments. To implement our approach, we used JAVA language, for the grid simulation, we used EJML matrix API. A 2D environment is modelled as a 2D matrix with 0 for free positions and 1 for occupied position as shown in Fig.2(b). We tested the proposed approach on four different maps with different sizes. Our data set is composed of 4 environments as shown in Table 1. The grid maps sizes are (29*30), (13*30), (5*9), and (60*50) cells.

Table 1. 2D experimental environments.

Environments	Rows	Columns
Env01	5	9
Env02	13	30
Env03	29	30
Env04	60	50

3.2. Experimental Result

To test the performance of the suggested method, we ran different genetic algorithms methods used to solve the path planning problems in different environments to compare them with our method Table 3. As the robot has to find an optimal path, that is safe and save the energy, the main evaluation criteria is the number of turns in the optimal paths. For the environment ”Env03”, GA is run one by one using the approaches in Ref.[15], Ref. [19], Ref.[31] and our proposed method. GA is executed 40 times for for each one. Table 2 show the average turns number in the best path obtained, and the average iterations number that each algorithm has done to converge in 40 trials. It is clearly seen in Fig.4 that GA with the proposed Fitness Function and the suggested Crossover finds the optimal path in less iterations compared to other methods, and considerably reduces the number of turns, due to the fitness function that penalizes the solutions with zigzags in the path and favors those which is are characterized by a straight line and which moves away from obstacles thus to obtain a path more optimal as shown in Table.5 .

Table 2. Experimental results in Env03.

Approaches	Number of turns in the best path]	Number of itteration
Ref.[15]	23.6	7.1
Ref. [19]	11.9	5.9
Ref.[31]	23.8	8.8
Our method	7.6	6.2

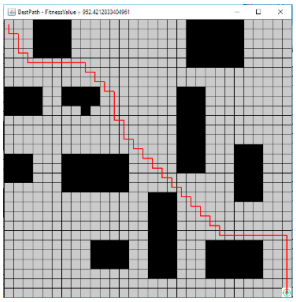
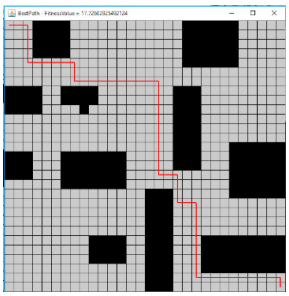
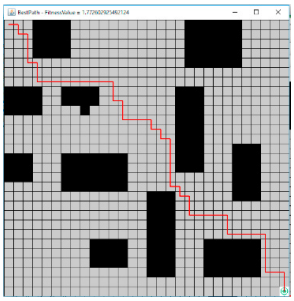
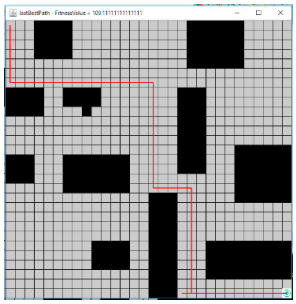
Table 3. Description of different methods used in comparison.

Approaches		Description
Ref.[15]	Initial Population	Both randomly generated and feasible initial population
	Fitness Function	$F = \begin{cases} \sum_{i=0}^{n-1} d(p(i), p(i+1)) & For\ feasible\ path \\ \sum_{i=0}^{n-1} d(p(i), p(i1)) + penalite & For\ infeasible\ path \end{cases}$
	Selection Method	Rank based Selection
	Crossover Method	Single Point Crossover
	Mutation	Modified mutation
Ref.[19]	Initial Population	Generate randomly the initial population (set of feasible paths) using the greedy approach based on Euclidean distance
	Fitness Function	Path length.
	Selection Method	Elitist Selection.
	Crossover Method	Modified Crossover.
	Mutation	Random mutation: The mutation is performed by randomly choosing a cell from the individual and trying to replace it with one of its neighboring cells of the grid map.
Ref.[31]	Initial Population	Random search
	Fitness Function	$\sum_{i=1}^{n-1} C0 - COST(V(i) - V(i+1))$
.		
	Selection Method	Roulette Wheel Selection.
	Crossover Method	Same Adjacency Crossover.
	Mutation	Randomly select a node called mutation node from the route and generate a path from the termination node by a process similar to the chromosome initialization.

Table 4. Number of turns in the optimal path found using different approaches and the proposed method in several environment.

Environments	Ref.[15]	Ref. [19]	Ref.[31]	OUR METHOD
Env01	2	2	4	2
Env02	13	8	11	3
Env03	24	21	17	4
Env04	52	32	43	29

Table 5. Optimal path found by different methods in Env03.

Ref.[15]	Ref. [19]	Ref.[31]	Our method
			

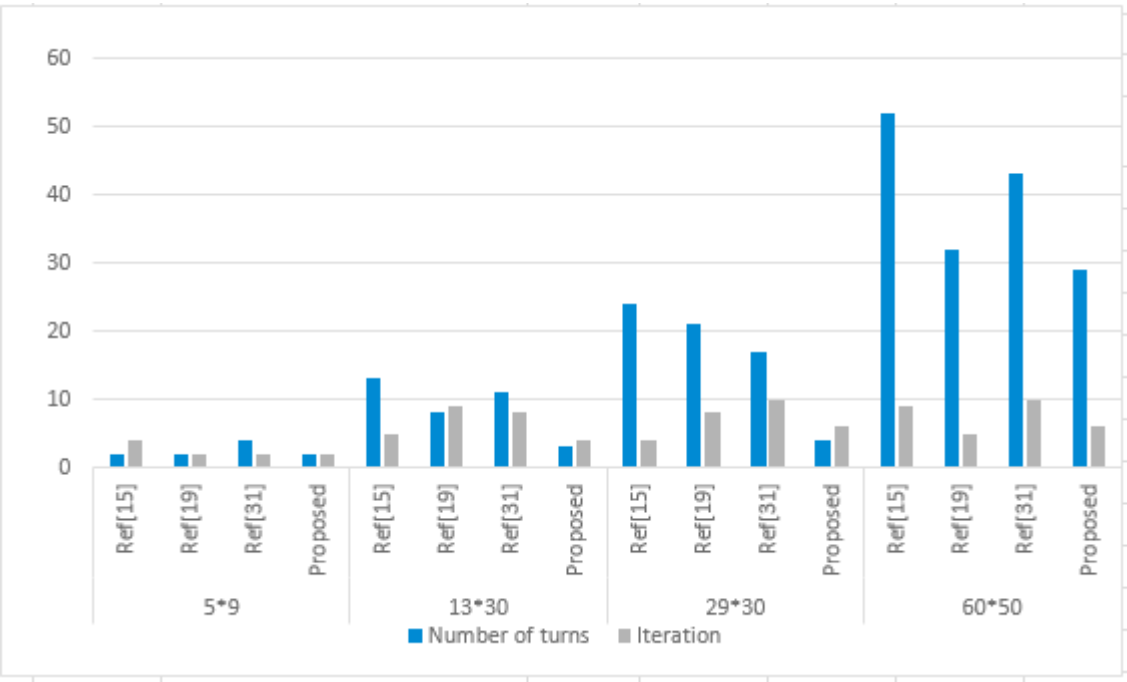


Fig. 4. Comparison of number-turns and number-iterations convergences in different environments

4. Conclusion

In this paper, a fitness function for GAs is proposed, that optimizes the energy consumed by the mobile robot, by reducing the number of turns in its path to reach the goal. An Improved Same Adjacency Crossover is also suggested for the GAs to solve the path planning problem. . In order to validate our postulate, it was applied to several environments with different sizes and compared with existing GA approaches.The results proves that the proposed GA method finds the optimal path. The average turns values and the average iterations numbers of the proposed method are more optimal comparing to other methods.

References

[1] D. Nakhaeinia, S. Tang, S. M. Noor, and O. Motlagh, "A review of control architectures for autonomous navigation of mobile robots," *International Journal of Physical Sciences*, vol. 6, no. 2, pp. 169–174, 2011.

[2] O. Hachour, "Path planning of autonomous mobile robot," *International journal of systems applications, engineering & development*, vol. 2, no. 4, pp. 178–190, 2008.

[3] S. B. C. Lamini, Y. Fathi, "A fuzzy path planning system based on a collaborative reinforcement learning," *International Review of Automatic Control (I.RE.A.CO.)*, vol. 10, no. 2, 2017.

[4] D. L. Tangni, "Asservissement dun système de navigation autonime par réseaux de neurones," Ph.D. dissertation, Université du Québec en Abitibi-Témiscamingue, 2015.

[5] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 138–145.

[6] G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," *Autonomous robot vehicles*, vol. 1, pp. 420–443, 1990.

[7] C. Ye and D. Wang, "A novel behavior fusion method for the navigation of mobile robots," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 5. IEEE, 2000, pp. 3526–3531.

[8] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[9] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[10] A. Chentoufi, A. El Fatmi, A. Bekri, S. Benhlima, and M. Sabbane, "Genetic algorithms and dynamic weighted sum method for rna alignment," in *2017 Intelligent Systems and Computer Vision (ISCV)*. IEEE, 2017, pp. 1–5.

[11] T. Tometzki and S. Engell, "Systematic initialization techniques for hybrid evolutionary algorithms for solving two-stage stochastic mixed-integer programs," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 196–214, 2011.

[12] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 345–370, 2009.

[13] M. Jebari, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, no. 4, pp. 333–344, 2013.

[14] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," *International Journal of Intelligent Systems*, vol. 18, no. 3, pp. 309–338, 2003.

[15] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.

[16] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[17] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2000, vol. 1.

[18] U. of Guelph. School of Engineering and A. Elshamli, "Mobile robots path planning optimization in static and dynamic environments," Ph.D. dissertation, University of Guelph, 2004.

[19] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, "Global path planning for mobile robots in large-scale grid environments using genetic algorithms," in *Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on*. IEEE, 2013, pp. 1–8.

[20] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[21] M. Davoodi, F. Panahi, A. Mohades, and S. N. Hashemi, "Multi-objective path planning in discrete space," *Applied Soft Computing*, vol. 13, no. 1, pp. 709–720, 2013.

[22] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4350–4355.

[23] M. Naderan-Tahan and M. T. Manzuri-Shalmani, "Efficient and safe path planning for a mobile robot using genetic algorithm," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 2091–2097.

[24] Y.-Q. Miao, A. M. Khamis, F. Karray, and M. S. Kamel, "A novel approach to path planning for autonomous mobile robots," *Control and Intelligent Systems*, vol. 39, no. 4, p. 235, 2011.

[25] J. Lee and D.-W. Kim, "An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph," *Information Sciences*, vol. 332, pp. 1–18, 2016.

[26] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, and B. Bouzouia, "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control," *Robotics and Autonomous Systems*, vol. 89, pp. 95–109, 2017.

[27] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evolutionary Computation*, vol. 7, no. 3, pp. 231–253, 1999.

[28] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 565–588, 2001.

[29] T. Back, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 57–62.

[30] M. Munetomo, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 3. IEEE, 1998, pp. 2774–2779.

[31] Z. Qiongbing and D. Lixin, "A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems," *Expert Systems With Applications*, vol. 60, pp. 183–189, 2016.