Mobile Robot Path Planning for Complex Dynamic Environments

Mohamed Lamine Tazir, Ouahiba Azouaoui, Mohamed Hazerchi, Mohamed Brahimi

Abstract—The last few years, research in the area of path planning for mobile robots has been focusing on dynamic environments. Most of methods proposed in this topic need to re-plan the remaining path of the robot, every time when new information come in which significantly increase the computation time and make real-time implementation of these methods difficult and sometimes impossible. The proposed approach consists of two different modules (static and dynamic) that combine two navigation methods to exploit prior information of global static map and local information coming from sensors. The robot uses global information about his environment, and plans the optimal path using genetic algorithms (GA) combined with Dijkstra algorithm through static obstacles. The dynamic phase is done while the robot is moving. The algorithm is able to avoid a moving obstacle by wait/Accelerate concept (W/A C), or by producing a new optimal local path using Dijkstra algorithm. A particularly new aspect of the work is to use the prior information about the environment for searching a global path efficiently and the incorporation of Dijkstra algorithm in both off-line and online phases, which leads to a significantly computation time reducing, and increases the accuracy of the global trajectory. The introduction of the accelerate action is also a new aspect. The simulation results confirm the efficiency and effectiveness of this approach in complex environments.

I. INTRODUCTION

Mobile robots are commonly used in many industrial fields. Thus, the research on path planning for mobile robot to avoid the obstacles in its path is very important. The objective of path planning is to find a collision-free route in an environment cluttered with obstacles, from a specified start location to a desired goal destination while satisfying certain optimization criteria such as distance, computation time, delay in the communication and energy consumption [1] [2] [3].

Path planning depends mainly on the environment in which the robot acts. Researchers classify the various methods used to solve this issue according to two factors [4] :

1- The environment type (static or dynamic)
2- The information gathered by the robot about the environment (local or global)

Global methods like visibility graph [5][6], Voronoi diagram method [7], A* [8], Ant colony optimization [9], simulated annealing [10] and genetic algorithms [11][12] require a complete and correct knowledge about the entire environment [11]. Therefore, they work only in static environment. In dynamic environment, complexity and uncertainly increase with the number of dynamic obstacles [13]. Consequently, global methods do not perform well in such environment. On the other hand, in local methods the robot uses only currently sensed information regarding its position to plan his local path. In other words, the path planning is being implemented while the robot is moving. Local methods include the artificial potential field [14] [15], state dominance method [16], dynamic window approach [17]. These methods have been studied extensively in the literature.

In the real world, navigation in a partially known environment is the most often situation. Although, this environment is not dynamic as a whole, it is a static model and inside elements keep changing. In this work, this prior knowledge is used to plan a global path using one of the meta-heuristic that has been often used in the field of motion planning, namely Genetic Algorithms. They are easy to implement, use parallel searches and are a powerful tool for searching in the space of higher dimension [18] [19].

Recently, many approaches [20][4][15][21] have been developed for the robot path planning in a dynamic environment [22]. However, most of these approaches are mainly used in known environment where robots have prior knowledge about mobile obstacles before navigating. In this case, the optimal solution can still be obtained. Unfortunately, this is not always the case in real life. Also, these approaches cost too much time, which may weaken the performance of real-time implementation. According to their simulation results, approaches proposed in [21] [20] need successively nearly 30, 60 seconds to find the first feasible path and not the optimal in an environment that does not exceed 15 static obstacles and 5 moving obstacles. Further, the major drawback of these methods is that they require complete re-planning of the robot path from the point where it is to the goal, each time when new information comes in. This process needs much computation time when dealing with dynamic and large sized environment [4]. Evenly, results of local minima can happen in certain cases [15]. These algorithms cannot reach an ideal solution individually in complex dynamic environment. Therefore, an improved method for path planning in a dynamic environment purpose is really essential.

In this paper, GWAC approach based on the GAs combined with W/A C algorithm for path planning of a mobile robot in a dynamic complex environment is proposed. This method focuses only on vertices of the obstacles rather than focusing on the entire environment. Using GAs, it seeks for an optimal path for the robot and when a moving obstacle is detected with a risk of collision, the second step called dynamic planning is launched. Our approach is simple and performs very quickly. Also, it aims to overcome the major drawbacks of existing works described above, and uses a simple representation of the environment which leads to a significantly computation time reducing and saves a lot of memory space. This appears more clearly in the dynamic phase, during the update of moving obstacle, instead to update all points, only its vertices are updated. It does not require complete re-planning path when new environment changes occur

ML. Tazir is with the Faculty d'ingénierie, Université Pierre et Marie Curie, 75005 Paris, France. tazir.med@gmail.com

O. Azouaoui is with the Centre de développement des technologies avancées, CDTA, 16081 Alger, Algeria. oazouaoui@cdta.dz

M. Hazerchi is with the Département d'Automatique, École Nationale Polytechnique, 16200 Alger, Algeria. hazerchimed@yahoo.com

M. Brahimi is with the École Supérieure d'Informatique, 16270 Alger, Algeria. m_brahimi@esi.dz

978-1-4673-7509-2/15/$31.00 ©2015 IEEE

leading to a computation time reduction. In the online phase, it deals just with dynamic obstacles. More exactly, it takes only moving obstacles entering into its perception zone which allows the robot to navigate in highly cluttered environment.
This paper is organized as follows. First, the model and working hypotheses are introduced in section II. Section III explains in detail the proposed approach, including the application of GAs to find the optimal path, then the application of the W/A C algorithm to avoid dynamic obstacles. Section IV includes simulations to test and validate the approach, and interpretation of results for different environments. Finally, we conclude the manuscript with a summary of results as well as the proposal of some future works.

II. GWAC PLANNER

A. Assumptions

It is widely acknowledged that because of the nature and complexity of general motion planning problem, a variety of simplifications have been adopted in planning literature to reduce this complexity. The proposed approach takes in consideration the following assumptions:

- The robot has a prior knowledge about the static obstacles, and it moves in a two dimensional limited area.
- The search space is represented by vertices of static obstacles.
- Moving obstacles can be assumed to be in almost uniform linear motion (straight line segment and with a uniform velocity).
- The motion parameters of dynamic obstacles, such as velocity and direction, are detected by the robot if the obstacle enters in its perception range at any instant during the navigation process.
- The robot can change its speed and direction at any time.

B. General structure

The proposed algorithm proceeds in two phases: global one based on static obstacles and local phase when moving obstacles are detected (Figure 1).

Global off-line planning: In the static phase, the calculation is based on vertices of stationary obstacles. The approach uses GA to find the optimal path by creating an initial population of feasible and infeasible paths. Once the path is determined, the robot starts moving between stationary obstacles while scanning the environment via its on-board sensors that can detect any changes in the environment.

Local online planning: this dynamic phase is applied when moving obstacles are detected. The acquired information (speed and direction of the obstacle) are used to calculate the possibility of collision. If there is a collision as shown in Fig. 2, two strategies are intended to deal with this situation depending on the velocities of both robot and obstacle:

Wait/accelerate strategy: the robot chooses between two actions, wait until the obstacle passes the intersection point (the algorithm calculates a position and a necessary waiting time, where the robot can safely wait, as Fig. 2.a illustrates) or speed up and pass before the obstacle reaches the intersection point.

Maneuvering around: this strategy aims to adjust the global path by an optimal local one, when the robot is imprisoned by obstacles. It uses the Dijkstra algorithm to generate an optimal local path, based on the surrounded static vertices or inserting a set of new way-points in a region around the obstacle (Fig. 2.b), in order to circumvent the detected obstacle. Then, the robot returns to its global trajectory generated by the global planner. This can happen only when the robot is trapped with obstacle. For example, if an obstacle reaches his goal on the trajectory of the robot. In this case, the robot waits for a period and if the obstacle doesn't change its position, the trajectory is recomputed locally using this strategy. Another scenario is when the robot is in a waiting period and a second obstacle appears with possibility to hit it. Then, it uses this strategy to escape from this situation. In extremely complex environment, both strategies may be used at the same time. The robot plans a local trajectory around the obstacle and applies the wait/accelerate action in order to achieve a safe position on a global trajectory and continues its displacement to the goal.

In case that there is no collision between the robot and the obstacle, the robot uses the path obtained by GAs to achieve its goal. As Fig. 3 shows, the robot calculates from information collected by the sensors that obstacle "T" will not collide with it.

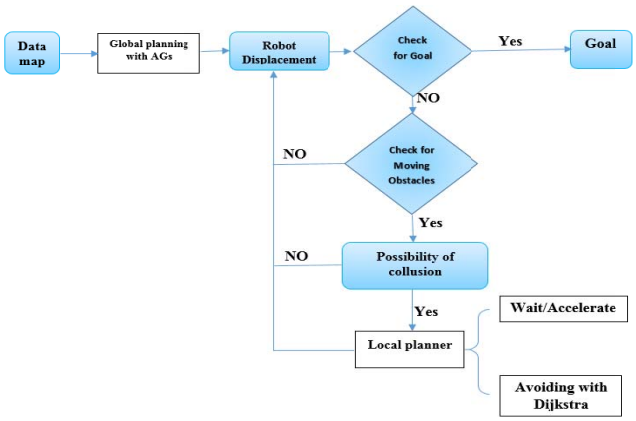
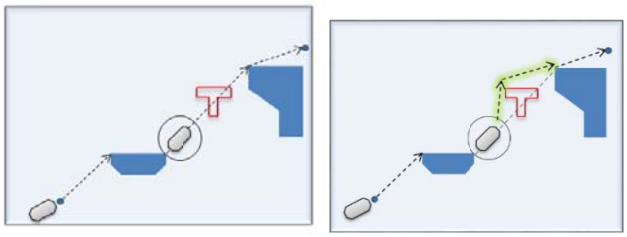


Fig 1. State transformation diagram of GWAC approach



(a): wait strategy (b): maneuvring around the moving obstacle
Fig 2: Scenario of collision between robot and moving obstacle

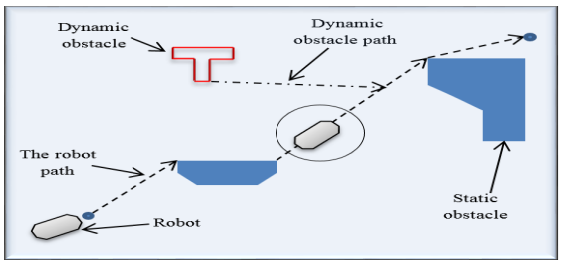


Fig 3. Online planning

C. Environment representation

Graph visibility model is used to represent the environment which is represented by polygons with vertices and segments. The obstacles boundary are formed by their real borders plus predetermined value, that the robot can approach obstacles without collision. The size of the robot is ignored, and therefore the robot is considered as a material point [4][21]. In Fig. 4, black polygons represent static obstacles and red triangles are moving ones, the blue polygons represent obstacle's

enlargement. The robot uses vertices of enlarged obstacle as a search space.

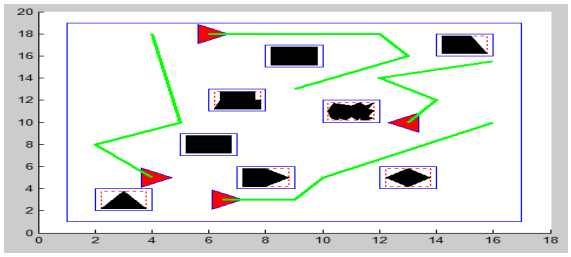


Fig 4. Environment Modeling

D. Creation of the two adjacency matrices 'A' and 'B'

The global path of the robot is constituted by segments that connect two vertices of the search space. Thus, two matrices (A & B) are created to represent all segments that can be part of the robot path. They are square matrices of order N where N is the total number of obstacles vertices in the environment. The rows and columns of these matrices are the obstacles vertices.

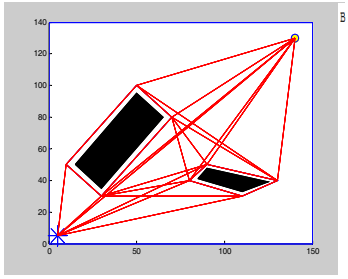
matrix A

It is a distance matrix that contains the distance between vertices Vi and Vj for each position (i,j). The distance is the length of a shortest path connecting the vertices. Instead of telling the exact distance between the vertices, it gives us whether or not two vertices are connected. If there is no connection between two vertices (i.e there is an intersection between the segment [i, j] and an obstacle, this segment is then infeasible), the element (i, j) must contain '-1'.

matrix B

This matrix has the same characteristics as the matrix A, except that in this matrix each element (i,j) contains the number of intersecting obstacles of infeasible segment, otherwise the element (i,j) contains '0 '. This matrix is used to differentiate between infeasible segments.

In Fig. 5, (a) corresponds to the representation of an environment which contains two obstacles, (b) and (c) represent the two adjacency matrices B and A respectively. It may be noted that this environment contains eight vertices plus the two start and goal points. The number of rows of the two matrices is therefore equal to 10.



(a)Representation of all feasible paths in the environment

B =

0	0	1	0	1	1	2	1	0	1
0	0	0	1	1	1	0	2	1	0
1	0	0	0	0	0	0	1	1	0
0	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0
2	0	0	1	1	0	0	0	1	0
1	2	1	0	0	1	0	0	0	1
0	1	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0

(b) Matrix B

A =

-1.0000	64.0312	-1.0000	28.2843	-1.0000	-1.0000	-1.0000	-1.0000	45.2769	-1.0000
64.0312	-1.0000	28.2843	-1.0000	-1.0000	-1.0000	100.0000	-1.0000	-1.0000	94.8683
-1.0000	28.2843	-1.0000	64.0312	41.2311	36.0555	72.1110	-1.0000	-1.0000	86.0233
28.2843	-1.0000	64.0312	-1.0000	50.9902	63.2456	-1.0000	80.0000	35.3553	148.6607
-1.0000	-1.0000	41.2311	50.9902	-1.0000	14.1421	-1.0000	31.6228	82.7647	108.1665
-1.0000	-1.0000	36.0555	63.2456	14.1421	-1.0000	41.2311	-1.0000	96.1769	94.3398
-1.0000	100.0000	72.1110	-1.0000	-1.0000	41.2311	-1.0000	22.3607	-1.0000	90.5539
-1.0000	-1.0000	-1.0000	80.0000	31.6228	-1.0000	22.3607	-1.0000	107.9352	-1.0000
45.2769	-1.0000	-1.0000	35.3553	82.7647	96.1769	-1.0000	107.9352	-1.0000	183.9837
-1.0000	94.8683	86.0233	148.6607	108.1665	94.3398	90.5539	-1.0000	183.9837	-1.0000

(c)Matrix A

Fig 5. Environment modeling

III. PLANNING ALGORITHM

A. Static Planning

1. Genetic representation

Each path is represented by a chromosome with a certain number of intermediate genes (vertices). Fig. 6 shows the structure of a chromosome where the first gene always corresponds to the starting point (number 1) and the last gene to the end point (numbered N +2), where N is the number of vertices of all the static obstacles in the environment. The Intermediates genes represent the vertices of the obstacles in the environment.

Starting gene	gene 1			Arrival gene
1	V_1	...	V_{N+1}	N+2

Fig 6. Structure of a chromosome

2. Structure of the algorithm

The pseudo-code of the static planning algorithm is given in Fig. 7, and steps of its implementation are outlined below:

Planning process with Gas
Start
t ← 0
Enlargement of obstacles
Encoding of obstacles vertices
Generation of the initial population P (t) of N individuals
Evaluation of P (t)
- Generate the probability of application of each operator (Pgc, Pgm, Pgr, Pgi) in each generation.
- Generate the probability of application of each operator (Ppc, Ppm, Ppr, PPi) in each population.

While (the stop condition is not met) **do**
t ← t + 1
- Select K= Ppc* N individuals from P (t) for the crossover
  - crossover
- Select M= Ppm* N individuals from P (t) for the mutation
  - mutation
- Select L= Ppr N individuals from P (t) for the reparation
  - Repair
- Select Q= Ppi* N individuals from P(t) for the inversion
  - inversion

Evaluation of P (t)
Select the best N individuals from the population of N + K individuals (parents and children)
End of While
Pick the best individuals among best individuals of each generation
End

Fig 7. The pseudo-code of the static planning

a. Enlargement of obstacles

In the proposed approach, each obstacle is enclosed in a rectangle or square. It is then amplified by a predetermined value, from the minimal distance with which the robot can approach the obstacle without collision, taking in consideration its physical dimensions, as shown in Fig. 8.

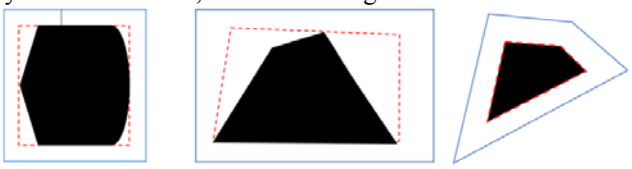


Fig 8. Enlargement of obstacles

b. Coding of obstacles vertices

The vertices of the enlarged obstacles are numbered randomly as shown in Fig. 9. These vertices are used as potential genes for chromosome whose phenotype presents a path between two given locations.

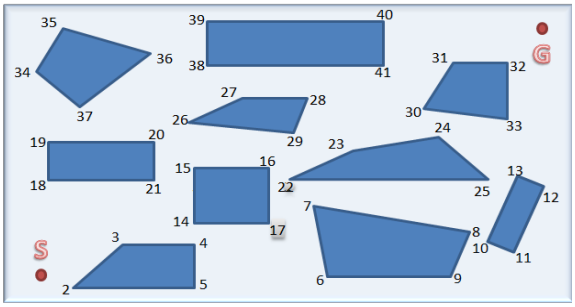


Fig 9. Numbering of vertices

c. Generation of the initial population

In the proposed approach, the selected chromosomes in the population have a variable length where each chromosome represents a set of vertices selected from the graph. Each gene in the chromosome corresponds to a specific vertex in the graph.

All selected vertices represent a potential path for the robot from the start vertex to the destination vertex, where each vertex can belong once at most to this path. We are interested in feasible and infeasible paths to diversify the initial population. In this latter, the chromosomes are produced as follows:

(i) Select a random integer k in the range [2 ... N+1], where k represents the size of the chromosome.
(ii) Select K random integers in the range [2 ... N+1], the integers correspond to chromosome genes. Duplication is prohibited.
(iii) To ensure that there is no repeated vertex in the same chromosome, the algorithm proceeds as follows: for each chosen vertex, the number '1' is put into an array of 'N' boxes (initially filled with zeroes). This number takes the position corresponding to this vertex to indicate that it has been chosen in this chromosome. During the filling process of chromosome, we select vertices from the set that has labels equal to "0" in this array.

d. Population Assessment

In this step, the algorithm evaluates individuals by the fitness function that gives two values for each chromosome. The first indicates whether the path is feasible or not (it takes two values: 0 if the path is feasible and 1 if the path is infeasible), and the second indicates the length of each chromosome.

The fitness function of feasible and infeasible paths

The evaluation function used to define feasible paths is determined by the linear combination of distances. It is given by:

$$E_f = \sum_{i=0}^{l+1} d(V_i, V_{i+1}) \quad (1)$$

Where $d(V_i, V_{i+1})$ represents the Euclidean distance between the vertex V_i and V_{i+1}.

The fitness of infeasible paths function is given also by:

$$E_i = \gamma + \beta \quad (2)$$

The term γ denotes the number of intersecting obstacles along the path and β is the mean number of obstacles intersected by segments. This feature is used to differentiate between the infeasible paths. This method is used in [23] to differentiate between feasible and infeasible paths.

In this evaluation process, the goal is to minimize both fitness functions for feasible and infeasible paths, of respective populations of GAs. In most cases, the obtained populations contain feasible and infeasible paths. A given penalty is added to infeasible path in order to differentiate between them and feasible ones. But, still keep them in the population because, they could become good feasible solutions after certain genetic transformations. So, the worst feasible path is assumed to be better than the better infeasible path. To do so, we add a constant $\boldsymbol{\pi}$ to the fitness of infeasible paths functions to make sure that the value of the fitness function of any given infeasible path is worse than those of feasible paths. This constant must be big enough and is defined as:

$$\boldsymbol{\pi} = (N + 2) * D \quad (3)$$

Where N+2 indicates the maximum number of genes in a chromosome, and D denotes the maximum length of a path segment.

e. The used genetic operators

Five operators are used in the proposed algorithm, three standard genetic operators: crossover, mutation and selection, and two additional operators: repair and inversion. Each operator is applied with two probabilities, one for generations and the other for the individuals in each population. Note the significant difference between our genetic approach and the one of [23]. In this approach [23], only two standard genetic operators (crossover and mutation) are used and an additional operator called repair. The used algorithm is different since only one operator is needed for each generation compared to our approach where all operators are applied at the same time with different probabilities.

Crossover

The multi-points crossover is used with two rates [24]:
- One for generations: it indicates the probability that the crossover operator is applied to the generation (applied to how many generations).
- And one for the population: it indicates the probability that the crossover operator is applied to the population (applied to how many chromosomes in the population).

Mutation

The mutation is introduced to diversify the population and explore the search space. For this operator, two rates are also used, one for the generations and the other for the populations as in the crossover operator.

Inversion of genes

The inversion contributes to the adjustment and the ordering of genes (perhaps disordered) of the same chromosome. Like any other operator, it is used with two probabilities (for generations and populations), but too high rate leads to infeasible paths, i.e. the destruction of feasible paths. In our case, we use two types of inversion:

Inversion of two genes

This operator switches between two genes of the same chromosome (one is brought to the position of the other).

Inversion of block of genes

This operator consists to reverse the order of a block of genes in the same chromosome.

Repair

As its name indicates, it is used to repair an infeasible line segment randomly selected from infeasible path, by using

vertices of intersected obstacles as intermediate nodes to overcome it.
An illustration of the repair operator is shown in Fig. 10. The vertices of the broadened obstacle (shown by dashed lines) are used by the operator to find a clear passage around the intersected obstacle. This Figure shows that there are several ways to move the robot from point S to point G without hitting the obstacle. We consider this problem as a local optimization problem and apply the Dijkstra algorithm [5] to find the optimal local path around the intersected obstacle.

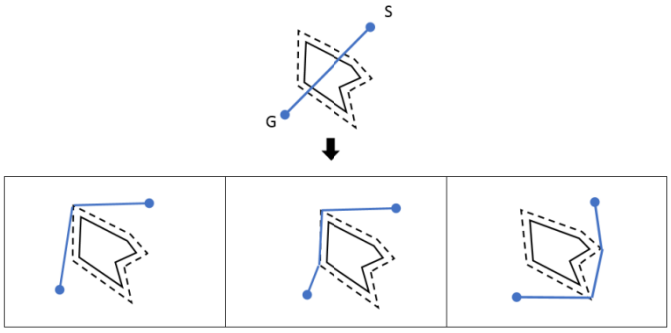


Fig 10. Repair with Dijkstra's algorithm

This operator is used as the other operators with two probabilities for generations and populations. In our approach, we have introduced this operator with a high probability because of its importance and effectiveness. In fact, when the path length is the assessment criterion, repair randomly infeasible segments could contribute to the rapid improvement of the solution. Therefore, the probability of choosing the repair operator takes a great value. After producing a new solution by repairing segments, the new solution will be evaluated using the evaluation function.

Selection

This operator is used for two cases, the first one is the selection of individuals (parents) to be crossed to get new sons and the second is the selection of individuals (chromosomes) to form a new generation from the group that contains individuals of the old population and new sons resulting from the application of the crossover operator. The selection is based on roulette wheel selection [25] where each individual is associated with a portion proportional to his fitness function.

B. Dynamic planning

As stated in the beginning of the paper, the robot uses the path generated by the static planner to move in the environment. The dynamic planner is automatically triggered to handle the situation where a moving obstacle is detected.
The robot uses the information (i.e. position, speed and direction of obstacles) given by the obstacle detection module and its own parameters (position, speed and direction) to calculate the possibility of collision with these obstacles. This process is done in several steps:

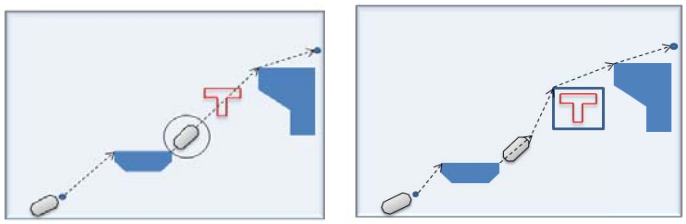
Step 1: the first intersection point between the proposed GAs path and the trajectory of a moving obstacle is calculated before considering the possibility of collision.
Step 2: based on the required time "t" for the robot to traverse the distance from the current position to the first intersection, it is calculated by the following equation:

$$t = d/v \quad (4)$$

Where "v" is the velocity of the robot and "d" represents the distance between the intersection point and the current position which can be calculated by this equation:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (5)$$

With (x_1, y_1) represent the coordinates of the current position of the robot and (x_2, y_2) represent the coordinates of the intersection point. At this time "t", the location of the mobile obstacle can be calculated, and therefore a safety interval for this obstacle can be obtained. This interval is calculated based on the size of the obstacle and its speed.
Step 3: If the time "t" belongs to this interval, then a collision will occur between the robot and the mobile obstacle. Otherwise, no collision will happen.
Step 4: if there will be a collision between the robot and the obstacle, two strategies (wait/accelerate or Maneuvering around) are intended to deal with this situation. In the first strategy, the approach adopts always the acceleration as an action in order to minimize the global time of the planning, but unfortunately this is not always the case. It depends on the current speed of the robot. The algorithm starts by increasing the speed and calculates the possibility of collision in each case, until a certain threshold. If the speed change exceeds this threshold and the collision still happened, the algorithm chooses the wait action until the obstacle passes the collision point. It is given by looking for a safe position for the robot and calculates a waiting period required for the obstacle to pass the intersection point.
The second strategy is to maneuver around the obstacle if its position is still unchanged and the robot remains trapped, as shown in Fig. 11(a). A local search using Dijkstra algorithm is applied to find a clear passage around this obstacle as shown in Fig. 11(b).
Step 5: In the event there is no collision, the robot can continue his journey following his global path.



(a): the obstacle has reached its Goal which is the intersection point (b): The robot moves around the obstacle.

Fig 11. Dynamic planning

IV. EXPERIMENTS AND RESULTS

In order to test the proposed approach, series of experiments are conducted using MATLAB R2010a running under Windows 7 on a 2.1 GHz Core 2 Duo PC.
The approach is tested in different environments, each environment contains static and dynamic obstacles. In this paper, we show just the simulation results of two environments. The optimization criterion of the genetic planner is both travel time and path length. The given solution is optimal or near optimal. The number of static and dynamic obstacles for the 2 simulation environments is shown in Table I, which is higher than environments proposed in [21][4], this makes these environments more complicated than the two others. The number of vertices of static obstacles for GAs planning is also given in Table I. Static and dynamic obstacles have random shapes (irregular). The first environment simulates simple scenario where the dynamic obstacles appear to the robot simultaneously and they simply move forward in the same direction. The last environment is more complicated where dynamic obstacles do not appear to the robot simultaneously but each obstacle appears at a different

moment and it moves forward or backward. Moreover, the number of dynamic and static obstacles is higher than those of the simple environment. All information on the positions of the static obstacles in the environment are known for the robot before it starts to move.

TABLE I
NUMBER OF OBSTACLES IN THE ENVIRONMENTS

Environment	Number of static obstacles	Number of dynamic obstacles	Number of vertices of static obstacles
1	8	8	32
2	20	7	90

A. The control parameters of the algorithm

This part concerns the GAs planning. Generally, the choice of parameters setting is a critical aspect using genetic algorithms. Unfortunately, there is no formal way to define the appropriate parameters. Traditionally, this is done experimentally. For us, after several tests, we set the control parameters of the algorithm as shown in Table II with the population size equal to 30 and the number of generations equal to 100.

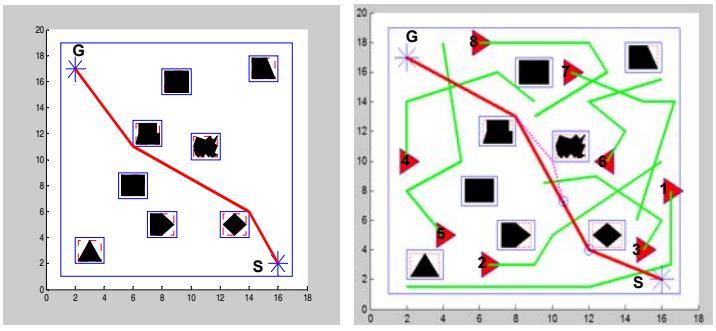
TABLE II
THE CONTROL PARAMETERS OF GAS ALGORITHM

Operator	Probability in generations	Probability in populations
Crossover	0.7	0.6
Mutation	0.4	0.3
Repair	0.85	0.8
Inversion of genes	0.1	0.05
Inversion of block of genes	0.05	0.5

B. Simulation results

1. Environment 1

First, the approach is tested for a simple environment which contains static obstacles in black color, with enlargement in blue and red dynamic obstacles whose trajectories are represented by simple green lines. Positions and velocities of dynamic obstacles are randomly generated. Figure 12(a) shows the optimal path computed by the static planner which is based on the vertices of static obstacles. Figure 12(b) shows that if a moving obstacle is detected with a possibility of collision, position and waiting time are calculated by the dynamic planner.



(a) Static planner (b) dynamic planner

Fig 12. Environment 1

In this environment, three cases of dynamic planner are present. In Fig. 12(b), the robot path's is intersected by five dynamic obstacles among eight. Waiting vector "[0 0.5861 3 0 0]" given by the dynamic planner, indicates that there is no collision between the robot and the first obstacle intersecting the robot path because the robot reaches this intersection point before the obstacle arrives there. However for the second obstacle, there is a collision which brings the robot to calculate a waiting position (represented by the blue circle) and a waiting time equal to 0.5861 seconds (the time required for the obstacle to pass the intersection point). The third obstacle stops squarely on the path of the robot, in this case, the dynamic planner waits a certain period equal to 3 seconds, and then, it calls the second action that is **maneuvering around.** This action uses the Dijkstra algorithm to find a local path which circumvents this obstacle (shown by purple dashed lines in Fig. 12(b)). For the fourth obstacle, the robot estimates its speed and position as well as the remaining speed and position of the obstacle to reach the intersection point. From these information, a collision occurs. The dynamic planner starts then to increase the speed of the robot and calculates if a collision can occur. The speed's mean vector [2/0 2.5/0 2/0 2/0 4.4/0] indicates that with the speed of 4.4m/s, the robot can reach the intersection point before the obstacle reaches its safety interval, therefore, the robot avoids this possibility of collision. For the last obstacle, there is no collision. Table III shows the length of the optimal path given by the genetic planner, the speed of the robot during this path, time needed to find this path and the calculation period of the W/A C planner. The simulation is run ten times, the results in the table are the average results of 10 simulations.

TABLE III
RESULTS OF THE ENVIRONMENT 1

The search space (number of vertices)	32
Number of dynamic obstacles	8
Numberof static obstacles	8
GA best chromosome	[33 9 27 34]
Robot speed (ms^{-1})	[1.5/0 1.1/0 0.7/0 0.8/0 4.4/0]
Path length given by the genetic planner (m)	158.3405
Waiting Vector(s)	[0 0 0.5861 3 0 0]
Computation time of genetic Planner (s)	0.523856
Global time of waiting (s)	3.586102

2. Environment 2

For this simulation, a more complex environment is tested. More static and dynamic obstacles are added. It contains 20 static obstacles with 90 vertices representing the search space. As shown in Fig. 13, this environment is complicated in terms of research and optimal feasible path. Most of the developed techniques have found their limits to provide an optimal solution or minimize the planning cost (the calculation time, ...) for large scale environments [21][4][15][20]. Since this environment is more complicated, the results given by the genetic planner are not identical each time. However, all the results given by this planner are optimal or near-optimal.

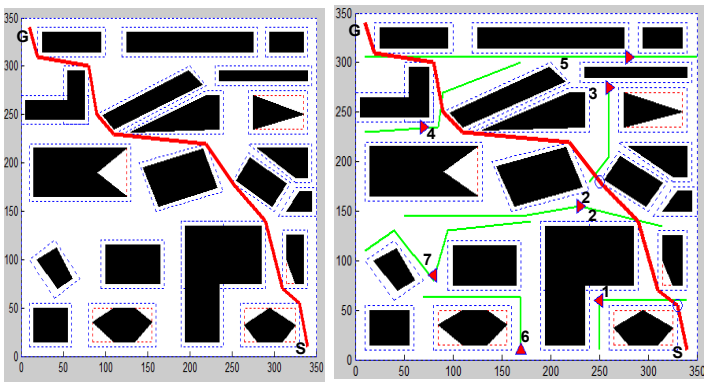


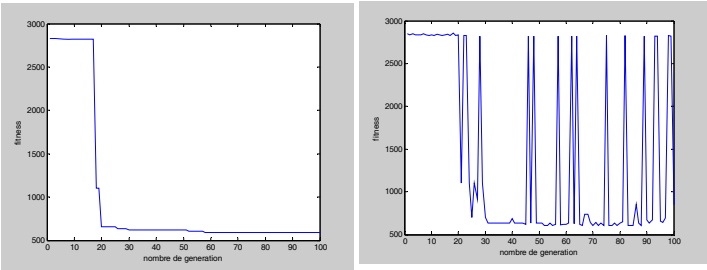
Fig 13. Environment 2

Table IV gives the results of this simulation environment.

Table IV
RESULTS OF THE ENVIRONMENT 2

The search space	90
Nbr dynamic obstacles	7
Nbr static obstacles	20
best chromosome	[91 19 29 15 41 39 61 62 59 77 92]
Robot speed (ms^{-1})	[1.2/0 1.3/0 2.8/0 4.3/0 1.4/0 3.2/0 1.3/0 3.5/0 2/0 1.2/0]
Path length given by GA planner (m)	532.6458
Waiting Vector(s)	[0 0.3152 0 0 1.8450 0 0 0 0 0]
Computation time of genetic Planner (s)	9.018568
Global time of waiting(s)	2.160246

From the result of Table IV, the robot had make two wait periods against the first and the third dynamic obstacles intersecting his trajectory, also it had a two accelerate phases against the second and the fourth dynamic obstacles intersecting his trajectory.



(a) Evolution of the minimum for each generation

(b) Evolution of the mean for each generation

Fig 14. Evolution of mean and minimum for each generation

Figure 14 (a) and (b) show the evolution of the minimum and average in every generation respectively. Initially, the approach has no feasible path. The first Figure shows that the algorithm runs 20 times to have the first feasible path. The optimal trajectory is obtained at generation 57 as shown in the Figure.

V. CONCLUSION

In this study, a new approach to deal with complex dynamic environments is proposed. The approach uses GAs to plan a global path using prior information about the environment. Upon reaching his goal location, its map is changed based on the perceptual information extracted during motion. Then, it uses the W/A C to deal with moving obstacles. Using current information from sensors as a type of local map, this dynamic planner generates one action from (accelerate, wait or coordinate a local path from the robot current position to the next vertex on the global trajectory). The algorithm provides a feasible path from an infeasible one after applying Dijkstra algorithm.

The obtained simulation results in complex and dynamic environments are, in most cases, sufficient (target met, low execution time, and optimal trajectories in most cases). These results clearly illustrate the first main contribution of the proposed approach which is able to generate for the robot, a free path without collision with obstacles in a dynamic complex environment. Furthermore, the approach gives improved performance compared to GA based approach in [21] in term of computation time to obtain the optimal trajectory. In this work, we introduced some new contributions as:

- The introduction of the Dijkstra algorithm in the repair operator to find the local optimal path around the intersected obstacle.
- Introduction of the accelerate action.
- The combination of W/A C algorithm with GAs for path planning in dynamic environments.

As a future work, the implementation of this approach on a real robot will be conducted. In addition, this work can be extended to multiple robots with unknown dynamic environments.

REFERENCES

[1] Y. Hu and S. X. Yang, "A Knowledge Based Genetic A1gorithm for Path Planning of a Mobile Robot," in *IEEE international Conference on Robotics & Automation*, 2004, pp. 4350–4355.

[2] X. Hu, C. Xie, and Q. XU, "Robot Path Planning Based on Artificial Immune Network," in *International Conference on Robotics and Biomimetics*, 2007. pp. 1053–1057.

[3] S. K. Rahul, "Design And Implementation Of Path Planning Algorithm For Wheeled Mobile Robot In A Known Dynamic," *IJRET: International Journal of Research in Engineering and Technology*, pp. 967–970, 2013.

[4] H. Miao, "A Multi-Operator Based Simulated Annealing Approach For Robot Navigation in Uncertain Environments," *Journal of Computer Science*, no. 4, pp. 50–61, 2009.

[5] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[6] S. S. Ge and F. L. Lewis, *Autonomous Mobile Robots : Sensing, Control, Decision Making and Applications*. Taylor & Francis Group, 2006, p. 698.

[7] Y. Ho and J. Liu, "Smoothing Voronoi-based Obstacle-avoiding Path by Length-minimizing Composite Bezier Curve," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2009, pp. 463 –468.

[8] P. kumar Das, A. Konar, and R. Laishram, "Path Planning of Mobile Robot in Unknown Environment," in *Special Issue of IJCCT for International Conference [ACCTA-10],*, 2010, vol. 1, no. 2, pp. 3–5.

[9] W. E. N. Zhi-qiang and C. A. I. Zi-xing, "Global path planning approach based on ant colony optimization algorithm," *Journal of Central South University of Technology*, vol. 13, no. 6, pp. 707–712, 2006.

[10] Y.-J. Ho and J.-S. Liu, "Simulated annealing based algorithm for smooth robot path planning with different kinematic constraints," in *ACM Symposium on Applied Computing - SAC '10*, 2010, p. 1277.

[11] P. Shamsinejad, M. Saraee, and F. Sheikholeslam, "A new path planner for autonomous mobile robots based on genetic algorithm," in *3rd International Conference on Computer Science and Information Technology*, 2010, pp. 115–120.

[12] 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, Nov. 2012.

[13] M. Abdel, K. Jaradat, M. Al-rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, 2011.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots.," *The International Journal of Robotics Research*, pp. 90 –98, 1986.

[15] L. Huang, "Velocity planning for a mobile robot to track a moving target a potential field approach" *Robotics and Autonomous Systems*, pp. 55-63,2009.

[16] J. P. Gonzalez, A. Dornbush, and M. Likhachev, "Using State Dominance for Path Planning in Dynamic Environments with Moving Obstacles," in *IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA*, 2012.

[17] M. A. Mansor and A. S. Morris, "Path Planning in Unknown Environment With Obstacles Using Virtual Window," *Intelligent and Robotic Systems*, vol. 1, no. 24, pp. 235–251, 1999.

[18] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," in *4th Student Conference on Research and Development*, 2006, no. June, pp. 183–188.

[19] D. Gallardo, O. Colomina, F. Flórez, and R. Rizo, "A Genetic Algorithm for Robust Motion Planning," *IEEE*, pp. 1–8, 1998.

[20] N. Buniyamin, N. Sariff, W. N. W. A. J, and Z. Mohamad, "Robot global path planning overview and a variation of ant colony system algorithm," *International Journal Of Mathematics And Computers In Simulation*, vol. 5, no. 1, 2011.

[21] Y. Wang, I. P. W. Sillitoe, and D. J. Mulvaney, "Mobile Robot Path Planning in Dynamic Environments," in *IEEE International Conference on Robotics and Automation*, 2007, vol. 44, no. April, pp. 71–76.

[22] T. P. Fries, "Navigation of Autonomous Robots Using Genetic Algorithms," in *Computational Intelligence, Man-Machine Systems And Cybernetics*, 2005, vol. 2005, pp. 192–197.

[23] Y. Wang, D. Mulvaney, and I. Sillitoe, "Genetic-based Mobile Robot Path Planning using Vertex Heuristics," in *IEEE International Conference on Robotics and Automation*, 2006.

[24] J. H. Holland., "Adaptation in Natural and Artificial Systems.," *The University of Michigan Press, Ann Arbor*, 1975.

[25] S.N.Sivanandam and S.N.Deepa, *Introduction to Genetic Algorithms*. Springer-Verlag Berlin Heidelberg, 2008, p. 453.