On-Line Navigational Problem of a Mobile Robot Using Genetic Algorithm

W. N. Abdullah
Department of Computer Science, College of Education- Ibn Al-Haitham,University of Baghdad

Abstract

Manufacturing systems of the future foresee the use of intelligent vehicles, optimizing and navigating. The navigational problem is an important and challenging problem in the field of robotics. The robots often find themselves in a situation where they must find a trajectory to another position in their environment, subject to constraints posed by obstacles and the capabilities of the robot itself. On-line navigation is a set of algorithms that plans and executes a trajectory at the same time.

The system adopted in this research searches for a robot collision-free trajectory in a dynamic environment in which obstacles can move while the robot was moving toward the target. So, the robot must operate in real-time such that the system reacts to unexpected obstacles.

Genetic algorithms that have been used successfully in many search problems are used to solve the on-line navigation problem with less computational cost. The system uses genetic algorithm as a search method for an optimal trajectory.

Key Words

Navigational Problem, On-line Navigational Problem, Genetic algorithm.

Introduction

Planning is an important aspect of the effort to design robots that perform their task with some degree of flexibility and response to the outside world [1]. Navigational problems would involve a workspace that captures all possible actions that could exist [2]. Hence, plans are created by searching through the workspace of possible actions until the sequence necessary to accomplish the task is discovered [1]. A successful action plan allows the robot to complete the task without violating any physical constraints of the robot or task [3].

Many AI planning problems involve navigational planning. In navigational planning problem, a mobile robot has to identify the trajectories between a starting and a goal position within its environment. The robot environment includes many obstacles (portions of the world that are permanently occupied; for example, the walls of a building) and thus finding the shortest trajectory without touching the obstacles in many cases is an extremely complex problem [4].

The object of this paper is to identify the sequence of steps and type of processes necessary to implement a navigational system using genetic algorithm that can generate optimal or near-optimal trajectory according to a given workspace that does not make any prior assumptions about feasible knot points. The rest of this paper is organized as follows: the on-line navigational problem, the genetic algorithm process, the design of the system of planning on-line trajectory using genetic algorithm including the representation of the chromosome, and finally, the results of the system and some concluding remarks are given.

On-Line Navigational Problem

Navigational planning is the science of directing the course of a mobile as it traverses the environment and reaching a destination without getting lost or crashing into any obstacles [5] in the workspace. The requirement that the navigational planning must meet is finding quickly an optimal trajectory due to some imposed criteria for the solution of the problem (which are referred to as trajectory metrics). For example, trajectory length, evaluation of clearance between the robot and obstacles, and others [6, 7].

The on-line navigational reacts in real-time to unforeseen obstacles such as human, and it capable of changing the motion direction while the robot is moving and can adjust the direction commanded by the planner based on sensing readings [8] where there is no prior information about the environment which the robot is supported to travel the problem of finding a trajectory between a start point and a target point.

The initial data are in this case only the coordinates of the two points or equivalently the direction and distance to the target. The problem can be solved only if the robot is equipped with some sensor system which can detect the obstacles around. Examples of such sensors are: touch sensors, range finding sensors (based on triangulation or time of flight techniques), or video cameras [9].

In on-line navigation, the robot can perform a sequence of actions with responding to changes in its environment and would be able to detect and correct errors in its own plans as it executes the navigation [1].

The task of on-line navigation is to navigate the robot to sub-goals generated by the navigator while avoiding collisions with obstacles [8] and reacting to sensory data as quickly as possible while driving towards a sub-goal.

Genetic Algorithm

Genetic algorithms (GAs) are stochastic search algorithms based on evolutionary and biological processes that enable organisms to adapt more to their environment. They are being successfully applied to problem in business, engineering and science [10]. GA works on a set of possible solutions, which is called the population.

A GA encodes a potential solution to a specific problem on a chromosome-like data structure and applies recombination operators to these structures in a manner that preserves critical information. Reproduction opportunities are applied in such a way that those chromosomes representing a better solution to the target problem are given more chances to reproduce than chromosomes with poorer solutions. GA is a promising heuristic approach for locating near-optimal solutions in large search spaces [11].

Typically, a GA is composed of two main components, which are problem dependent: the *encoding problem* and the *evaluation function*. The *encoding problem* involves generating an encoding scheme to represent the possible solutions to the optimization problem. In this paper, a candidate solution (i.e., a chromosome) is encoded to represent a trajectory from the start position to the goal position. The *evaluation function* measures the quality of a particular solution. Each chromosome is associated with a fitness value, which in this case is the length of the chromosome. For this paper, the smallest fitness value represents the better solution.

Chromosomes evolve through successive iterations, called generations. To create the next generation, new chromosomes, called offspring, are formed by (a) selecting chromosomes to be merged (b) merging two chromosomes from the current population together using a crossover operator and (c) modifying a chromosome using a mutation operator.

Selection is the process of keeping and eliminating chromosomes in the population based on their relative quality or fitness. There are several possible selection strategies. One of them is *Tournament* selection which is adopted in this paper. In *Tournament* selection, *s* chromosomes are taken at random, and the better chromosome is selected from them. The winner of the tournament is the chromosome with the lowest fitness of the *s* tournament competitors, and the winner is inserted into mating pool. The mating pool, being filled with tournament winners, has a lower average fitness than the average population fitness.

Crossover generates valid offspring by combining features of two parent chromosomes. Chromosomes are combined together at a defined crossover rate, which is defined as the ratio of the number of offspring produced in each generation to the population size.

Mutation produces random changes in various chromosomes. Mutation serves the critical role of either replacing the chromosomes lost from the population during the selection process or introducing new chromosomes that were not present in the initial population. The mutation rate controls the rate at which new chromosomes are introduced into the population.

On-Line Navigational Problem Using Genetic Algorithm

The design of the System adopted here is based on On-line navigation. The general steps of this system and the details description of its steps are presented in the following:

Before applying genetic algorithm operations, the robot workspace must be translated in a form that computer programs can deal with. So, two-dimensional array of grids is used for representing the map of the workspace with a given start and goal points and obstacles are represented by rectangular shapes.

Representation and Initialization

The first step in this navigational problem is to set up the initial population. For this purpose, we have taken the sensor information into account and the coordinates obtained from those sensors are used to set up the initial population. In this case, it is assured that all the initial population are feasible, in the sense they are obstacle-free points and the straight line trajectories between the starting and the selected next points are obstacle-free.

The chromosome representation is extremely simple due to that in one genetic iteration, we plan the trajectory up to the selected next point. The data structure to represent the chromosome consists of four fields (X_i , Y_i , X_j ,and Y_j). The first two fields (X_i , Y_i) are the coordinates of the starting point and the second two fields (X_j , Y_j) are the coordinates of one of the 2D points, obtained from the sensor information. For example, the starting point is (30,50), the robot sensing range=10, and the coordinates of some points sensed by the robot are (40,40),(20,60),(40,60). Then, the chromosomes correspond to those point are formed as (30,50,40,40), (30,50,20,60), and (30,50,40,60) respectively. All these chromosomes form the initial population.

Checking the Feasibility

The process of evaluating the feasibility of the trajectory includes two aspects: (a) checking that the trajectory does not fall on an obstacle. To do so, the system should check the feasibility of each trajectory points. The system modifies Digital Differential Analyzer (DDA) algorithm [13] to check each trajectory point if it is feasible or not (b) checking that it is sufficient spaces between obstacles in the environment for the easy movement of the robot through the trajectory (the space should be greater than the robot size).

If any trajectory point is on an obstacle or if there insufficient space between obstacles around the trajectory then this trajectory is infeasible

Genetic Algorithm Operators

To create a sequence of populations that will contain the next points to the current point as time goes on, the main genetic operators (Crossover, Mutation and Selection) are applied iteratively on each population.

The integer crossover operation is employed in this system. The crossover point should be selected such that most of the offsprings generated in this process are feasible. The crossover point is set between the third and the fourth integer for every chromosome [12].

After making crossover between all pairs of population, we will get the new population. For this new population, we will find the feasibility, i.e.; they are reachable from the starting point by the straight line or not. If the trajectory resulted from crossover operation is not feasible, this trajectory will not store in the new population in order not to compete with the other trajectories stored in the mating pool.

The next step of the algorithm is making the mutation. In this process a binary bit is randomly selected on the bit stream of the sensor coordinates and alters that binary bit value, such that the feasibility should not be lost for that chromosome.

Evaluation of Fitness Value

The next task is estimating the fitness of each chromosome of the total present population (both for the initial and new populations). Calculation of the fitness involves finding the sum of the straight line distance from the starting point to the coordinate obtained from the sensor information and the distance from the current point to the goal point.
The formula of fitness function is:

Fitness of chromosome $(X_i,Y_i,X_{j1},Y_{j1})= 1/((x_{j1} - x_i)^2 + (Y_{j1} - Y_i)^2 + (x_g - x_{j1})^2 + (Y_g - Y_{j1})^2)$

Where:
(X_i,Y_i) is the coordinate of the starting point,
(X_{j1},Y_{j1}) is the coordinate of the current point, and
(X_g,Y_g) is the coordinate of the goal point.

Evaluation of Best Fit Chromosome

After finding the fitness value of the chromosomes, we will evaluate the best chromosome, i.e.; for which the fitness is the best. In this case, the best fit chromosome represents the predicted shortest trajectory from the starting point to the goal.

In the first generation itself, we are getting a near-optimal intermediate point to move that third and fourth integer field of the best fit chromosome that will become the next intermediate point to move. Then, we update the starting point with this better point.

Termination Criteria

The whole process of the genetic algorithm, from setting up the initial population, is repeated until the best fit chromosome will have its third and fourth field equal to the x- and y-coordinates of the goal location.
The algorithm is formally presented below.

The Algorithm

```
{$(x_i,y_i)$=starting point; $(x_g,y_g)$=goal point}
  Add-trajectorylist $(x_i,y_i)$
  Repeat
   i) Intialization
      - Get sensor information in all possible directions
        $(x_{j1},y_{j1})$ , $(x_{j2},y_{j2})$ , … , $(x_{jn},y_{jn})$
      - Form chromosomes like $(x_i,y_i,x_j,y_j)$
   ii) Crossover
      - Select crossover point on the third and the fourth fields of the chromosome.
      - Allow crossover between all chromosomes and get new population as
        $(x_i,y_i,x_{j1},y_{j1})$ , $(x_i,y_i,x_{j2},y_{j2})$ , $(x_i,y_i,x_{j1}^{i},y_{j1}^{i})$ , $(x_i,y_i,x_{j2}^{ii},y_{j2}^{ii})$.
```

```
iii) Mutation
      Select a mutation point in bit stream randomly and complement that bit position
      for every chromosome.
  iv) Selection
      - Discard all chromosomes ($X_i,Y_i,X_j,Y_j$) from population whose line segment
        on obstacle region.
      - For all chromosomes in population: find fitness using

           Fitness($x_i,y_i,x_j,y_j$) = $1/ ((x_j - x_i)^2 + (y_j - y_i)^2 + (x_g - x_j)^2 + (y_g - y_j)^2 )$;

           - Identify the best fit chromosome ($x_i,y_i,x_{bf},y_{bf}$);
           - Add-trajectory list ($x_{bf},y_{bf}$)
           - $x_i = x_{bf}$ ; $y_i = y_{bf}$ ;
        End For,
  Until ($x_i = x_g$)) and ($y_i = y_g$);
End.
```

Experimental Results

The system was run on many different cases. These cases are taken from different perspectives: number and distribution of obstacles, size of obstacles, the value of robot sensing range, number of experiments per a workspace, and robot size. As samples, the figures -represented by *case1*, *case2*, and *case3*- show some of the cases of these experiments. Each figure of each case shows the robot location after some generations while some obstacles move randomly within the environment (i.e., in each case, the distribution of obstacles will be changed form figure to another because of the movement of some obstacles). In each case, the values of robot size (Rsize) in square grid unit, robot sensing range (SR) in grid unit, and maximum number of generations at which the robot reaches to the goal (MaxGen) are given. The symbols "S", "G", and "R" refer to start position, goal position, and the location of robot after some generations respectively.

Conclusions

From the previous results, we can conclude the following:

1- The system operates on entire free space and does not make any prior assumptions about feasible knot point of the trajectory.
2- The system can deal with any moving obstacles while avoiding collisions with the robot.
3- At each generation, the robot moves to a new position toward the goal.
4- As the robot ability of sensing the environment (sensing range) becomes bigger, the trajectory points become less and consequently the time of reaching the goal becomes less and vice versa.
5- At each iteration, the planner produces a trajectory which is based on the obstacle most recently sensed positions and assumes that the obstacles are now fixed.
6- The system does not give an acceptable trajectory if the robot's environment contains a very large number of obstacles and there is no sufficient space to move the robot through it.

References

1.Lugar, G. F. and Stubblefield, W. A. (1989), Artificial Intelligence and the Design of Expert Systems. Redwood City, CA: Benjamin /Cummings.
2.LaValle, S.M. (2003): Planning Algorithm. University of Illinois.

3.Farritor, S. and Dubowsky, S. (2002), A Genetic Planning Method and its application to Planetary Exploration. Cambridge, MA 02139 U.S.A.

4.Rich, E. and Knight, K.(1991): artificial intelligence, Mc-Graw Hill, Inc.

5.Michalewicz, Z.(1999), Genetic Algorithms + Data Structures =Evolution Programs. New York: Springer Verlag.

6.Podsedkowski, L.(1999), VERY WELL INFORMED A*SEARCHING ALGORITHM AND ITS APPLICATION FOR NONHOLONOMIC MOBILE ROBOT MOTION PLANNING. International Journal of Science & Technology (1999), 10(2) & 11(1,2), 33-43.

7.Mchenry, M. C.(1998), Slice – Based Path Planning, *Ph.D. Dissertation,* Faculty of the Graduate School, University of Southern California.

8.Kortenkamp, D.; Bonasso R. P., and Murphy, R.(1997), Artificial Intelligence and Mobile Robot: Case Studies of Successful Robot Systems.

9. Lazea, Gh. and Lupu, As. E. (1996), Aspects on path planning for mobile robots.

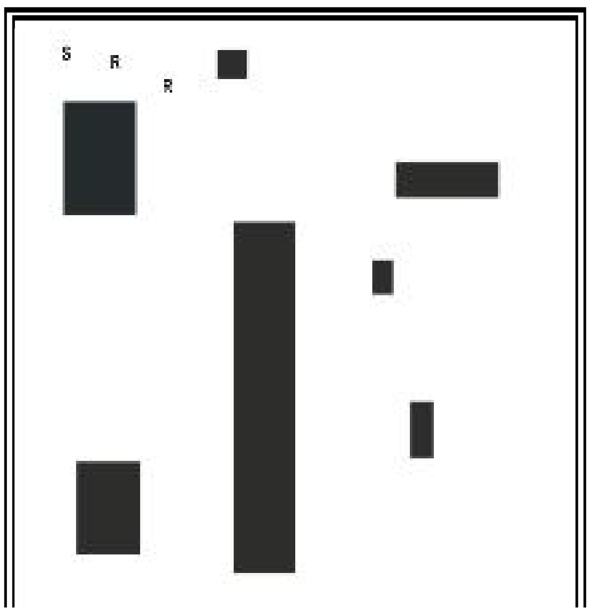
10.Goldberg, D. E. (1994), "Genetic and Evolutionary Algorithms Come of Age", *Communication of the ACM*, 37(3): 113-119, March.

11.Wang, L.; Siegel, H. J.; Roychowdhury, V. P., and Maciejewski, A. A. (1997), "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," *Journal of Parallel and Distributed Computing,* Special Issue on Parallel Evolutionary Computing, 47(1): 8-22, Nov. 25,

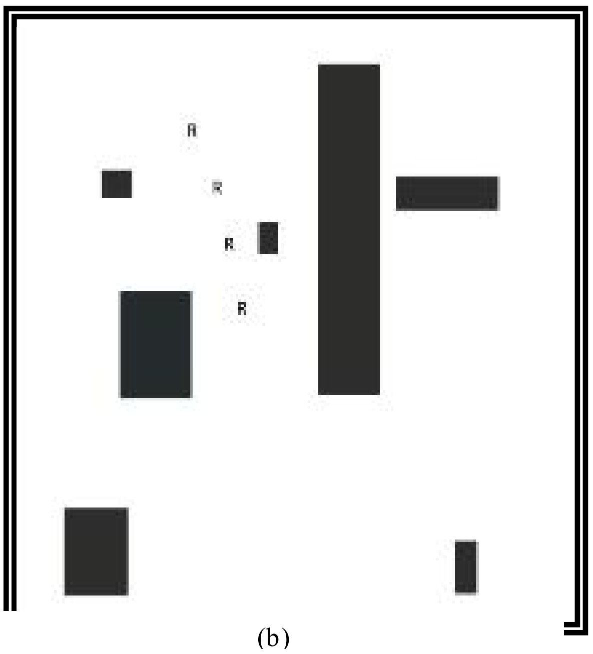
12.Konar A. (2000), Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of The Human Brain.

13.Salmon, R. and Slater, M. (1987), Computer Graphics: Systems & Concepts. Reading, MA: Addison-Wesely.

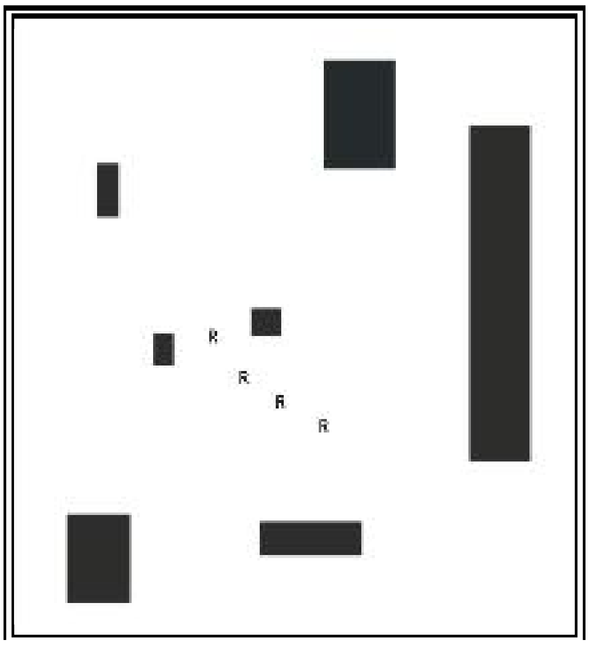
Figures:
Case 1



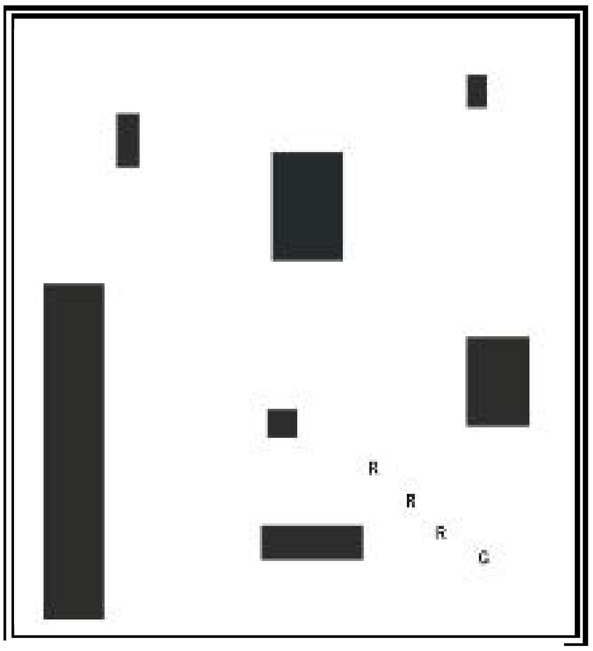
(a)



(b)



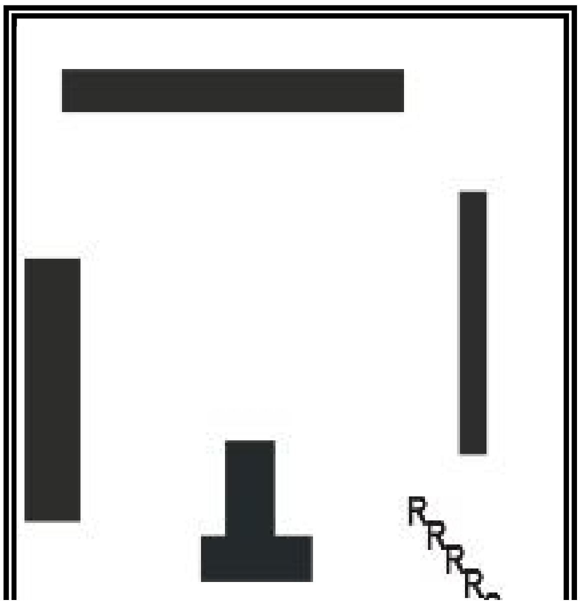
(c)



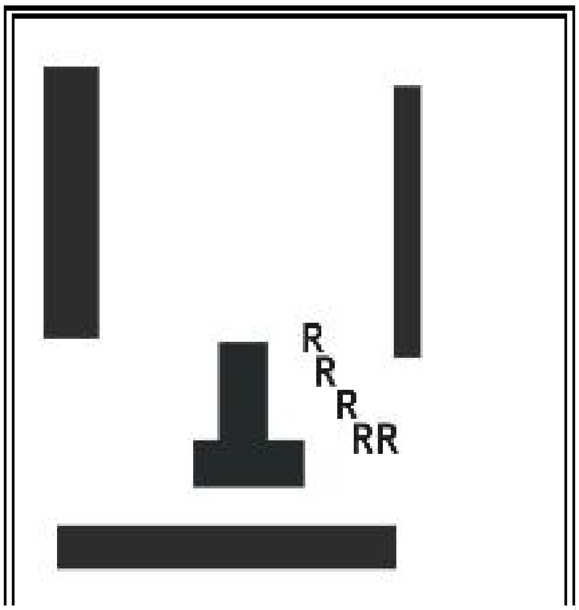
(d)

Rsize = 9 SR= 15 Maxgen = 43

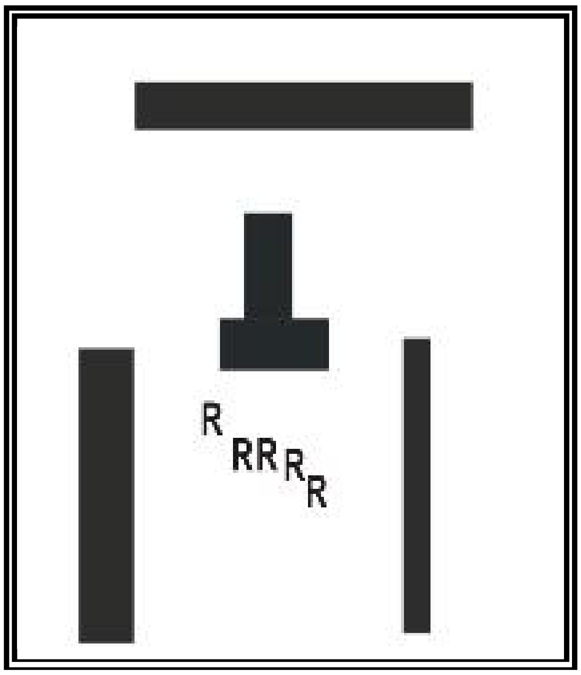
Case 2

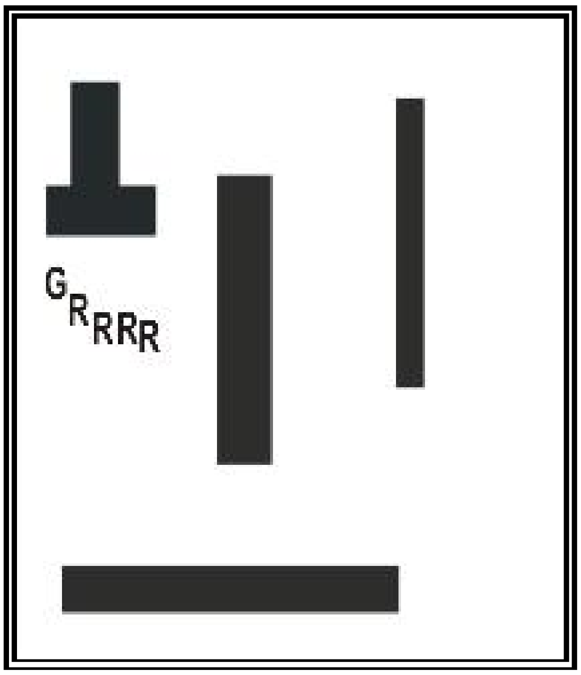


(a)



(b)





Rsize = 20   SR= 6   Maxgen = 80

مجلة ابن الهيثم للعلوم الصرفة والتطبيقية المجلد23 (2) 2010

مشكلة التجول المباشر للإنسان الآلي باستخدام الخوارزمية الجينية

واثق نجاح عبدالله

قسم علوم الحاسبات، كلية التربية – إبن الهيثم، جامعة بغداد

الخلاصة

تتوقع أنظمة التصنيع المستقبلية إستخدام العربات الذكية للتحسين والتجول، إنّ مشكلة التجول مشكلة مهمة وصعبة في حقل علم الإنسان الآلي. الأنسان الآلي يجد نفسه في أغلب الأحيان في حالة يجب أن يجد فيها مسارا" إلى الموقع الآخر في بيئته، تلك الحالة تكون خاضعة لقيود تتمثل بالحواجز داخل البيئة وقابليات الإنسان الآلي نفسه. التجول المباشر هو مجموعة الخوارزميات التي تخطّط المسار وتنفّذه في الوقت نفسه.

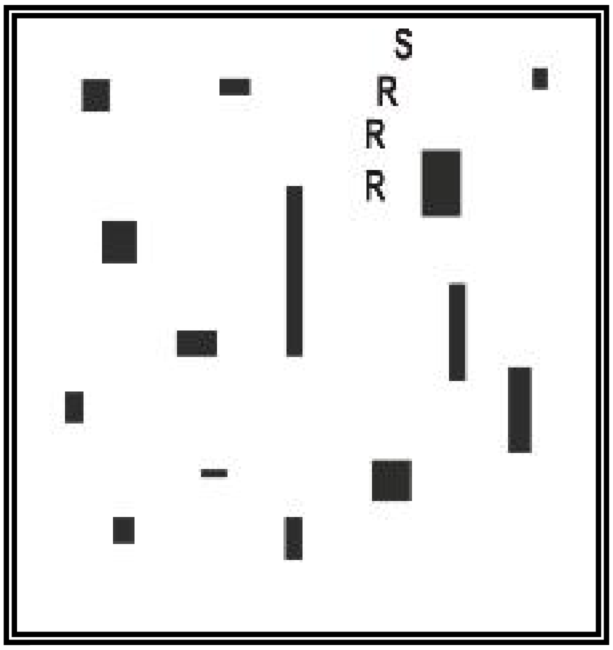
النظام المقترح في هذا البحث يحاول ايجاد مسار للروبوت خالي من التصادم في بيئة ديناميكية– وهي البيئة التي يمكن ان تتحرك الحواجز فيها بينما يواصل الروبوت حركته بأتجاه الهدف، لذا فالروبوت يجب أن يعمل في الوقت الحقيقي بحيث يكون النظام قادرا على التعامل مع حركة الحواجز غير المعروفة مسبقا.

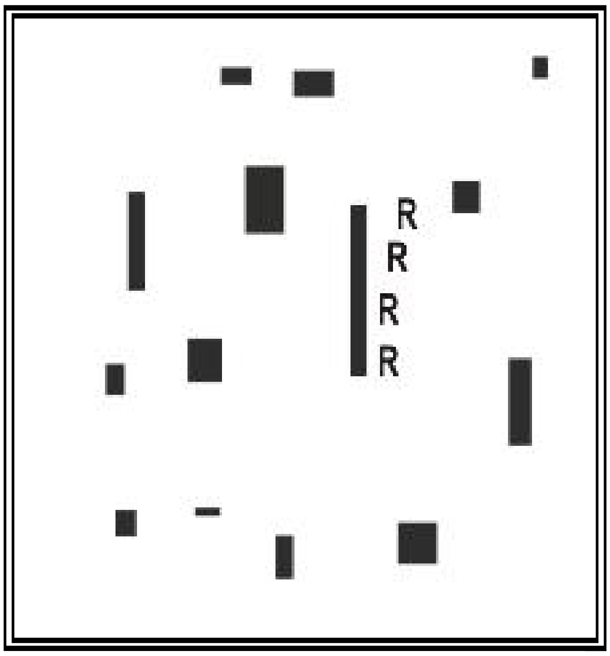
الخوارزمية الجينية – التي أثبتت نجاحها في كثير من مشاكل البحث– أستخدمت لحل مشكلة (التجول المباشر) بكلفة حساب قليلة. أستخدم نظام الخوارزمية الجينية طريقة بحث عن أفضل مسار.

(c) (d)

Rsize = 20 **S R= 6** **Maxgen = 80**

Case 3





(a) (b)