Monograph:
Leung, C.H. and Zalzala, A.M.S. (1993) Genetic Based Motion Planning and Evaluation for the B12 Mobile Robotic System. Research Report. ACSE Research Report 493 . Department of Automatic Control and Systems Engineering

GENETIC-BASED MOTION PLANNING AND EVALUATION OF THE B12 MOBILE ROBOTIC SYSTEM

C. H. Leung and A.M.S. Zalzala

Robotics Research Group,
Department of Automatic Control and Systems Engineering,
University of Sheffield

Research Report #493

INTRODUCTION

Motion planning for mobile robots is concerned with providing a feasible and efficient path to accomplish a given task. Although many solutions may exist, a condition for obtaining the best (or near best) option may be imposed by the user, where a criterion in terms of the total distance traversed, energy expended or minimum execution time must be achieved. The planning procedure is made more complicated if the robot has to detect and avoid static or dynamic objects in the workcell.

Nonetheless, when a real-time implementation is considered, a moderate computational complexity is considered for the planning algorithm, hence allowing an efficient on-line execution and interaction with the environment sensors. In this paper, genetic algorithms are investigated as a potential solution for real-time motion planning of mobile robots. Genetic algorithms are theoretically and empirically proven to provide robust search in complex spaces with the advantage of being computationally simple yet powerful in their search for improvement. They had already been shown to be suitable for motion planning of articulated robots where robust convergence was reported [1].

The work reported in this paper investigates a more detailed and sophisticated approach to mobile robot path planning in 3D space involving moving (or disappearing) obstacles, thus accommodating for the concept of intelligent control within a dynamic environment. In addition to reporting the successful results of the genetic planning algorithms, this paper presents an actual implementation utilising the B12 mobile robot, where maps stored through the on-board sonar array are manipulated by the genetic algorithm to present a feasible motion.

WHY GENETIC ALGORITHMS ?

Although different optimisation methods have been developed (e.g. calculus based methods, enumerative schemes, random search algorithms, etc.), these conventional optimisation methods are shown not to be robust enough in certain applications. Despite the fact that genetic algorithms have been applied successfully to optimisation problems [2,3], one central theme of research has always been robustness, the balance between efficiency and efficacy necessary for survival in many different environments. Thus, genetic algorithms differ from standard optimisation and search procedures in a number of important ways [4,5].

Motion planning procedures are rather complicated because of their interaction with the outside world and handling unexpected events. Previous research showed conventional optimisation methods are not robust enough for the job [6] while genetic-based algorithms performed better [1]. Mobile robot motion planning using genetic algorithms had already attracted some attention from the robotics community. Cleghorn [7] used a Symbolics 3670 machine to develop his algorithm while Harvey [8] advocates real-time dynamic neural networks as the medium of evolution. Both papers reported the genetic solution to be a far less computation intensive approach to path planning. One of the drawbacks of the A* algorithm, which is used in many of the existing motion planning programs, is that the size of the search-tree is exponential.

THE DEVELOPED ALGORITHM

Prior to the planning process, a global knowledge of the environment is needed and is translated in the form of a terrain map. Firstly, a set of valid random paths are generated as the initial generation. In order to prevent the robot wandering endlessly inside the workcell, a weighted vector of motion is employed during the path construction phase, that is, the direction of motion from the robot's current position towards the goal has more chance to be chosen than other directions. A fitness value is assigned to each path and the one with the best fitness is stored for future usage. Secondly, pairs of paths are chosen randomly for mating, and those with better fitness will have more chance to be drawn out for mating. The reproduction process repeats until some arbitrary number of generations is reached. It was found that the best path might come from any of the considered generations yet the average fitness value generally improved for successive generations. The structure of the genetic algorithm is described in the following sub-sections.

Space decomposition

In this approach to motion planning, a global model of the environment is considered with knowledge of the 3D space surrounding the mobile and including any obstacles, where collision-free motion is catered for in both static and dynamic cases, as will be discussed later. Amongst various methods, the space decomposition approach is the most used by researchers [9] where the space is divided into blocks and attempts are made to minimise the computation time required for path generation.

For the genetic algorithm, both free space and obstacles are represented as a collection of blocks of equal size. In addition, the position of a block in 3D space is represented by a 2D map for x and y dimensions while the z altitude is given as discritized contour values prefixed to the particular block, as shown in Figure (1).

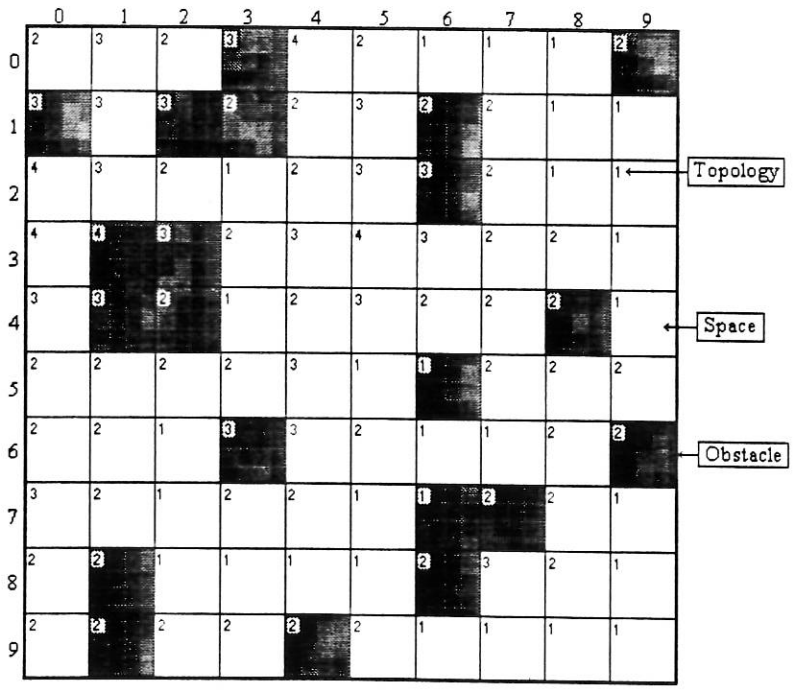


Figure (1): Representation of 3D space

Coding the parameters

The robot path is coded as a string of n number of points represented by their Cartesian co-ordinates as { (x_0,y_0,z_0), (x_1,y_1,z_1), ..., (x_n,y_n,z_n) }, with all values stored in a binary form to be of use in the genetic formulation. Nonetheless, this coding method yields variable-length paths and a proper genetic structure is required to deal with it, in particular while performing crossover.

The objective function

The efficiency of the planned path is calculated based on minimising three factors: the total traversed distance, the energy consumed in climbing slopes and the total time required for executing the motion. Hence, the mobile robot will avoid wandering around the terrain or climbing unnecessary slopes.

The first random generation

To generate a population of paths, a random walk within the bounded decomposed space is used, where all individuals link a start block with an end block while any path including an obstacle is rendered as invalid. Initiating the random walk from the start block is very inefficient for a large grid, while initiating the random walk from both ends and looking for intersection points is somehow more tolerable. Nonetheless, the approach in this work is to present some directional guidance for the path generator. This guidance is presented in the form of limiting transitions from one block to another, as shown in Figure (2). In Figure (2), both N and SE directions are invalid to avoid obstacles while NW and NE diagonal directions are also waived to avoid the obstacle's corner (optional).

Reproduction

To perform reproduction, a roulette wheel is deviced with division sizes proportional to the fitness value of the paths they hold. A replica of the selected path is put in the mating pool.

The efficiency of the mating process is affected by the size of the pool and the existence of replicas. Performing crossover is not straight forward because of the variable-length coding and, more important, since a random crossover would produce a discontinuous motion. Thus, the path pair is checked for points within a certain proximity (coincident, one or two blocks apart). If one is found and is not coincident for both paths, a random segment is generated to connect both points, hence providing the possibility of crossover.

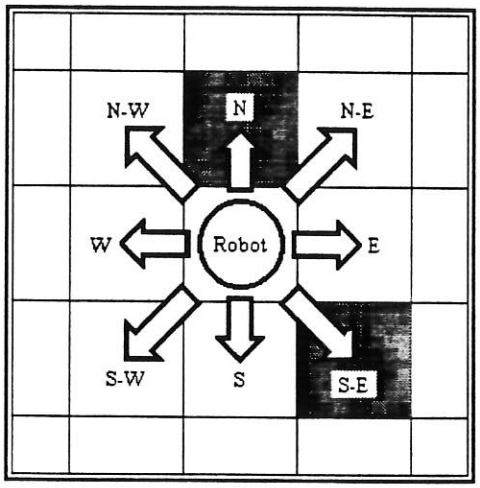


Figure (2): Restrictions on motion

Once again, following a conventional mutation procedure will present motion discontinuities and a special process is devised. For this algorithm, if mutation occurs at a random point along a path, the remainder of the unmutated path between that point and the goal is destroyed and replaced by a randomly generated segment. Consequently, the mutation probability parameter is set as appropriate to control the rate of the procedure.

Production strategy

Enrolling only the best paths increases the average fitness of the next generation. However, the rate of convergence of the algorithm is controlled via a number of production parameters, as shown below. Hence, setting the proper parameters may help to minimise the computation time which is of particular interest for this real-time system.

Permitting a replica in the mating pool provides more than one copy of a good individual and may affect the rate of convergence. The size of the mating pool is varied from 50% to 100% of the population size in order to experiment with keeping the goodness for the next generations. Moderate mutation rate is used so as not to destroy the good individuals within the generation. A large population size provides better information via its individuals but leads to more computations than a smaller size.

PLANNING IN A DYNAMIC ENVIRONMENT

A further difficulty is encountered when a dynamic model of the environment is considered where moving obstacles must be taken into account. If the location and time of occurrence of the moving obstacles is known in advance, a number of terrain maps are generated combining both the stationary and moving obstacles at certain points in time, as shown in Figure (3). Hence, the planning process will be as described above but with the algorithm checking the validation of the path within a terrain map at different time instances and the paths hitting obstacles being discarded. As this is performed in real-time with the processing and data collection alternating, the robustness offered by the genetic structure is very much appreciated.

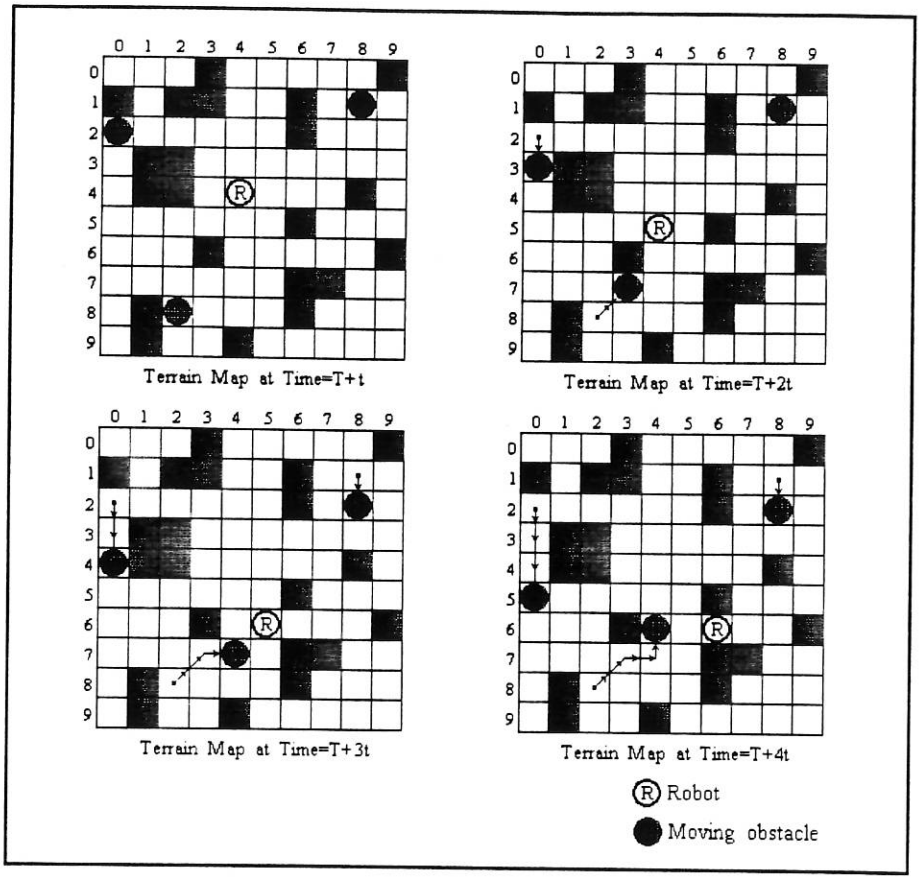
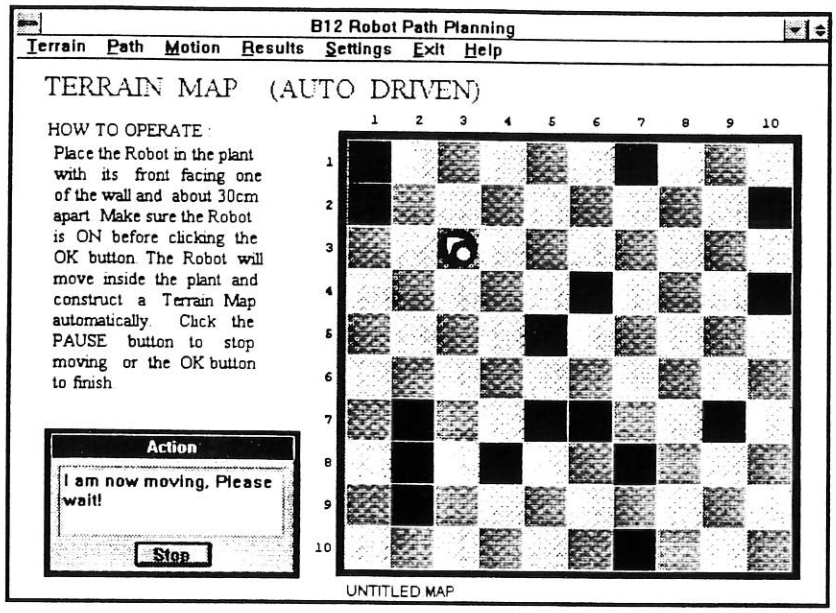


Figure (3): Terrain map at different time intervals

THE REAL-TIME IMPLEMENTATION

The genetic motion planner is implemented for the control of the RWI B12 [10] mobile robot, where the processing power is provided via a 386-based PC. The windows-based software package provides a real-time interface with the B12 where the terrain maps are scanned and updated using the on-board sonar transducers [11,12]. In addition, the application program runs in a multitasking environment, providing for the controller program to execute in the foreground while the communication with the robot is executed in the background. The user interface allows for two modes of operation, a manual mode where the terrain map is specified by the operator, and an automatic mode where mapping the environment and updating the grid depends entirely on the sensory data. In both modes, the genetic algorithm is employed to present a feasible motion which is then down loaded to the robot. Figure (4) shows the interface output for an auto-driven set-up.



Figure(4): Auto-detected terrain map

Detecting obstacles

The B12 moves itself randomly in the environment, detecting any obstacles and building an initial terrain map using the on-board transducers. Since the ultrasound beam angle with respect to the obstacle is known, the location of an obstacle is calculated by adding the angle of the transducer with respect to the robot to the angle of direction that the robot is pointing to with respect to the obstacle. Thus, the x and y co-ordinates of the obstacle are obtained from the distance measured by that transducer. This practical implementation of the system considers a constant height for the z co-ordinate. If the detected obstacle is outside the map then its presence is ignored. However, due to the characteristics of the ultrasonic sensor, false readings may sometimes be obtained [12] and a verification procedure is executed before adding the obstacle onto the map. This verification is accomplished by checking whether any obstacles in the old map lies between the robot and the distance measured. If not, then this new obstacle is added to the map otherwise another firing takes place to double check the detection.

EXPERIMENTAL RESULTS

The test program is written in C++ for Windows which provides a menu-driven package for setting the genetic parameters, B12 and the interface signals in addition to a real-time display of the motion environment and further analysis screens.

Experiments are conducted considering a terrain of 10x10 as the workspace, a map is created where blocks (0,0) and (9,9) are assigned as the start (S) and end (E) blocks, respectively. The mobile is required to move from S to E while avoiding any static pre-defined obstacles or wandering new objects.

For the static environment, the B12 is instructed to ignore any new objects in the workspace. However, for the dynamic environment, the B12 is tested by introducing three objects moving at constant speed with known directions. The computation time

required by the real-time system when navigating through the dynamic world is much more intensive (a ratio of 4:1) since continuous data acquisition and detection is needed in addition to re-computing the planning procedure. Nonetheless, the modest computational power of a 386 PC may be increased to achieve a better performance.

Considering a fixed number of generations of 25 the best paths in every generation can be stored and displayed, as shown in Figures (5) and (6) with the final results presented in Figure (7). The planning procedure is displayed on-line in a graphical form (see Figure (8)) which aids the operator to detect any functional problems, while the motion's statistical data is stored for further analysis by the user, as indicated in Figure (9).

Test results

In testing the genetic algorithm, different parameters give different rates of convergence as well as in which generation the best path occur. The following parameters were found to be the most effective to achieve an efficient run:

1. Population size : 40-50
2. Mating pool size : 40-50 % of the Population Size
3. Replica : allowed
4. Mutation rate : 0.01-0.05

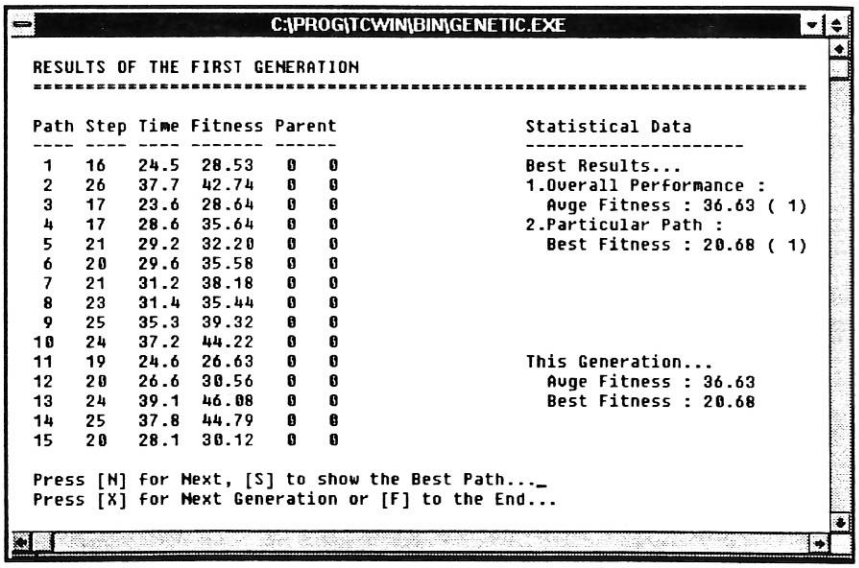


Figure (5): A test run during a generation

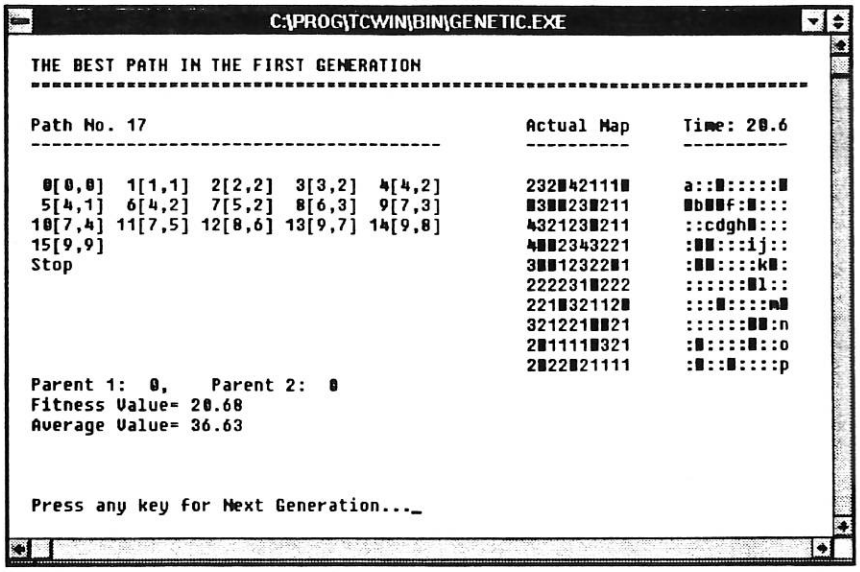


Figure (6): Results of one generation

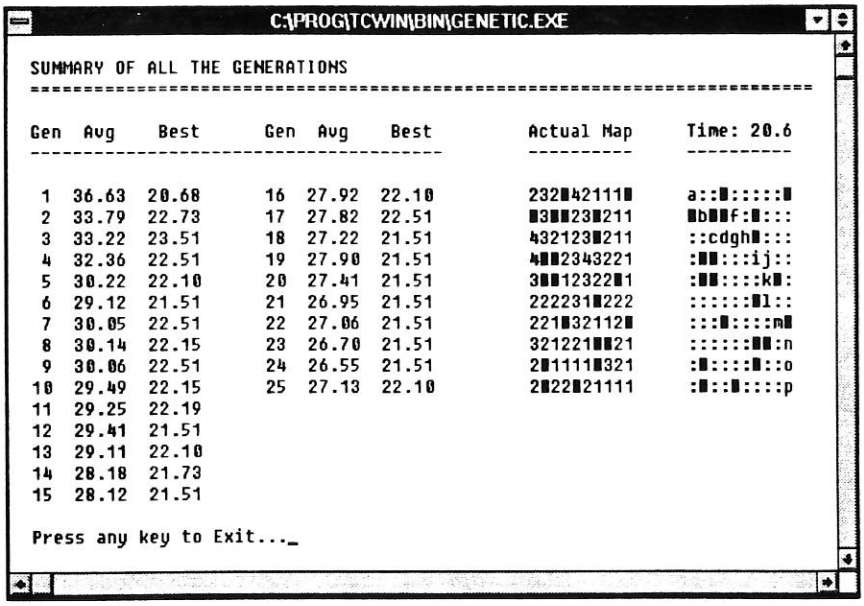


Figure (7): The motion planning results

DISCUSSIONS

This section shows the results of the genetic planning algorithm using different production strategies. The mutation rate, population size, size of mating pool and the permission of replica is changed to detect the best structure.

Mutation rate: The program is run with three different values of 0.005, 0.1, and 0.5. It was noticed that the higher the mutation probability is, the slower the rate of convergence becomes although a better feasible path is obtained. Since the system operates in real-time, the proper mutation rate must be chosen for the successful operation of the algorithm, which may require a trade-off between execution speed and motion cost.

Size of mating pool: This is varied between 15 and 40 with an increment of 5 at each of six experimental runs. The best convergence was obtained with a size of 20 which is about 40% of the population size.

Permission of replica: A better convergence of the genetic algorithm was obtained when replica was permitted.

Population size: The population size is increased from 10 to 50 with an increment of 10 at each experimental test. It was noticed that the total execution time increased by about 70% when the population size was doubled. However, the best rate of convergence and the occurrence of the best path is associated with the run with the largest population size.

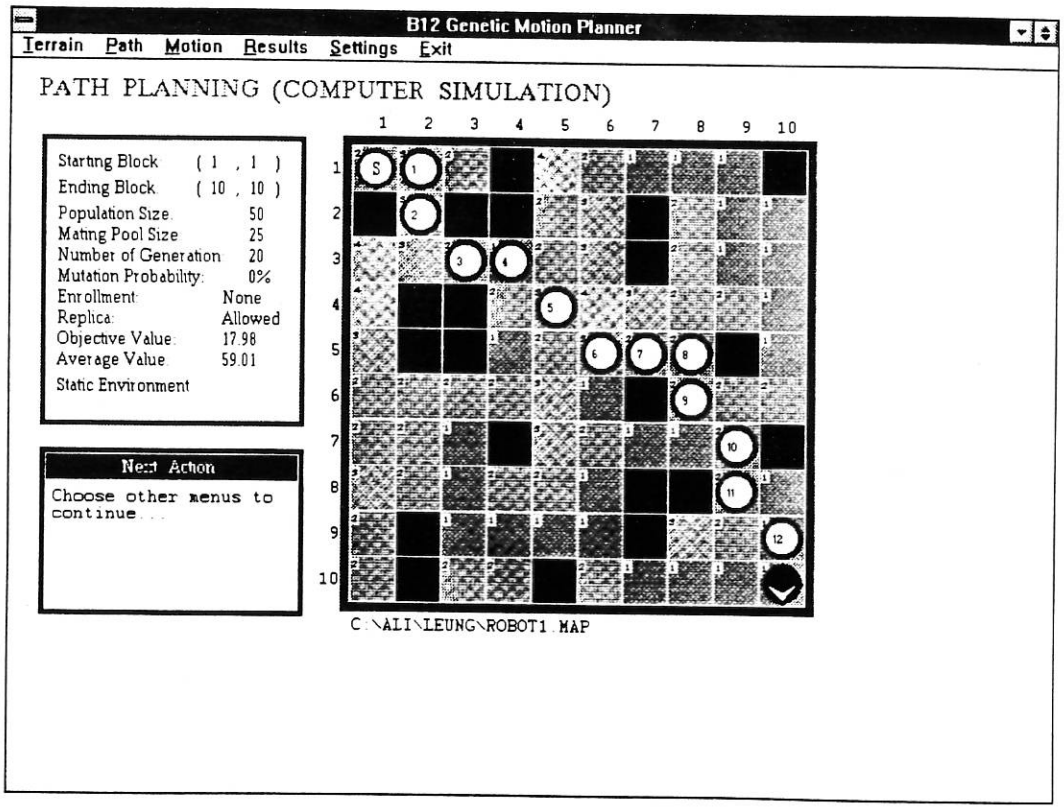


Figure (8): Planning in a static environment

[2] Jong K. D., "Adaptive system design: a genetic approach", *Trans. IEEE on Systems, Man and Cybernetics*, pp.566-574, vol. SMC-10, 1980.
[3] Grefenstette J., "Optimization of control parameters for genetic algorithms", *Trans. IEEE on Systems, Man and Cybernetics*, pp.122-128, vol. SMC-16, 1986.
[4] Goldbery D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, 1989.
[5] Holland J. H., *Adaptation in Natural and Artificial Systems*, Ann Arbor, The University of Michigan Press, 1975.
[6] Sahar, G. and Hollerbach, J.M., "Planning of minimum-time trajectory for robot arms", In *Int. J. Robotics Research*, Vol. 5, No. 3, pp. 91-100, 1986.
[7] Cleghorn T. F., Baffes P. T. and Wang L., "Robot Path Planning Using A Genetic Algorithm", *Proc. 2nd Annual Workshop on Space Operations, Automation and Robotics*, pp. 383-390, 1988.
[8] Harvey I. and Husbands P., "Evolutionary robotics", IEE Colloquium on Genetic Algorithms for Control and Systems Engineering, pp. 6/1-4, May 1992.
[9] Fujimura K., *Motion Planning in Dynamic Environments*, Springer-Verlag, 1991.
[10] RWI Inc., *B12 Mobile Robot Base-Guide to operation (ver 2.1)*, 1993.
[11] RWI.Inc., *G96 Sonar Board-Guide to operation (ver 1.1)*, 1993.
[12] Leonard J. J. & Durrant-Whyte H. F., *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, 1992.