2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Mobile Robot Path Planning in Dynamic Environments

Yang Wang, Ian P. W. Sillitoe and David J. Mulvaney, *Member, IEEE*

Abstract— **This paper introduces a genetic algorithm (GA) planner that is able to rapidly determine optimal or near-optimal solutions for mobile robot path planning problems in environments containing moving obstacles. The method restricts the search space to the vertices of the obstacles, obviating the need to search the entire environment as in earlier GA-based approaches. The new approach is able to produce an off-line plan through an environment containing dynamic obstacles, and can also re-calculate the plan on-line to deal with any motion changes encountered. A particularly novel aspect of the work is the incorporation of the selection of robot speed into the GA genes. The results from a number of realistic environments demonstrate that planning changes in robot speed significantly improves the efficiency of movement through the static and moving obstacles.**

I. Introduction

THE mobile robot path planning task is to find a collision-free route, through an environment containing obstacles, from a specified start location to a desired goal destination while satisfying certain optimization criteria [1]. While off-line planning methods are designed to deal with the motion of a robot in environments containing both static and dynamic obstacles, a complementary on-line method is required to deal with changes in expected motions encountered during navigation through the environment.

This paper introduces vertex++, a genetic-based algorithm for path planning in dynamic environments (those in which one or more moving obstacles are present), that has the ability to deal with both static and dynamic constrains simultaneously. Although designed initially as an off-line algorithm, vertex++ is also appropriate for use in on-line planning, where its operation can be triggered in response to changes in the expected movements of the dynamic obstacles. The vertex++ navigation approach is an enhancement of the vertex planning method for static environments described by the authors in [2]. By restricting the planning to obstacle vertices rather than considering than the entire environment, the vertex++ planner is able to significantly reduce the calculation time compared with other GA approaches that have been applied in dynamic environments, in which all points in the environment are considered as potential nodes in a path (regardless of whether they are in free space or within an obstacle). A further novel achievement of the new planning approach is the inclusion of robot speed into the planning process, which takes into account the time at which obstacles are encountered, thereby allowing the consideration of a much greater range of possible avoidance paths.

Traditional mapping techniques, such as grid [3][4], meadow [5], Voronoi diagrams [6] and visibility graphs [7], are not well suited to application in dynamic environments due to the need to reconstruct or repair maps in response to environment changes. To deal with the inherent complexity of path planning tasks, a number of researchers have investigated the application of evolutionary techniques in static environments. Davidor [8] developed a tailored genetic algorithm (GA) with a modified crossover operator to optimize robot trajectories. The Evolutionary Planner/Navigator (EP/N) [9]-[13] has been through a series of revisions to enhance its performance, particularly by introducing additional problem-specific domain knowledge in the form of tailored GA operators that can be brought to bear on the path planning task. Nearchou [14] used the number of vertices produced in visibility graphs to build fixed length chromosomes in which the presence of a vertex within the path is indicated by setting of a bit at the appropriate locus. A reordering operator was applied to enhance performance and the algorithm was capable of determining a near-optimal solution. Cai and Peng [15] developed a fixed-length decimal encoding mechanism to obviate the need for reordering operators and used individuals whose length was fixed to be that of the total number of all obstacle vertices in the environment.

For dynamic environments, the literature records few contributions that discuss robot navigation using GAs or that incorporate the speed of the robot as part of the planning problem. A modified version of EP/N, termed ϑEP/N++, was proposed in [16][17][18] as a decision support system for a ship to voyage without collision on the basis of environmental information obtained from automatic radar plotting aids. The major features of ϑEP/N++ include a time parameter, the variable speed of the ship, and time-varying constrains representing movable ships. The ϑEP/N++ system retains the structure of EP/N and has same set of genetic operators, with the exception of a new addition operator developed to act on the ship's speed. The candidate solutions are obtained from variable-length individuals that represent a

Manuscript received September 15, 2006.

Y. Wang is currently with the Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, LE11 3TU, UK (e-mail: y.wang4@lboro.ac.uk).

I. P. W. Sillitoe is with the Scottish Association for Marine Science, Dunstaffnage Marine Laboratory, Dunbeg, Oban, PA37 1QA, UK (e-mail: ian.sillitoe@sams.ac.uk).

D. J. Mulvaney, is with the Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, LE11 3TU, UK (phone: +44 (0) 1509 227042; e-mail: d.j.mulvaney@lboro.ac.uk).

path through a set of absolute co-ordinates drawn from the environment as a whole. The on-line planning can be activated in response to the motion changes of other ships.

This paper describes the operation of the new vertex++ planner, and its performance when applied to a set of simulations of realistic environments is considered in detail.

II. Planning Algorithm

This section explains the internal description of the environment used in the new planner, details the internal structure of the GA and describes the operation of the vertex++ during both off-line and on-line planning.

A. Operating environment and modeling of obstacles

The working environment for the mobile robot consists of a set of stationary obstacles whose shapes either are defined to be, or to be are approximated by, bounding polygons. In addition, the robot movement may be affected by the presence of one or more dynamic obstacles that are also represented by polygons. If the motion parameters (the heading and speed) of those dynamic obstacles remain constant, a safe trajectory for the robot can be generated by the vertex++ planner in an off-line manner. In addition, the path generated off-line can be adaptively revised in response to any changes in the motion characteristics of the dynamic obstacles.

In the off-line planner, it is assumed that complete motion knowledge of the moving obstacles in the observed region is available. In the on-line planner, it is assumed that changes to the motion parameters of the moving obstacles are made available whenever one comes within sensor range. Although no particular sensor type or configuration is specified, it is assumed that in order to allow the robot to be guided so as to avoid any potential collisions with obstacles, there is an adequately large time interval between the detection of obstacle movements and the implementation of newly generated actions. Note that this assumption may be relaxed if guidance is achieved by reactive navigation, such as in [19].

For purposes of planning, the static obstacles are enlarged by a value determined from the minimum distance (herein referred to as the safe distance) that the robot can approach obstacles without collision, to account for the robot dimensions (see Fig. 1 for an example). Such a representation allows the physical dimensions of the mobile robot to be neglected and regarded as a single point.

To model the motion of the dynamic obstacles in the vertex++ planner, the same strategy as in ϑEP/N++ is adopted. In brief, for each obstacle, its motion is described by its trajectory, consisting of a series of one or more segments, each having start and finish co-ordinates between which the heading (defined by the co-ordinates) and the speed of the obstacle are fixed.

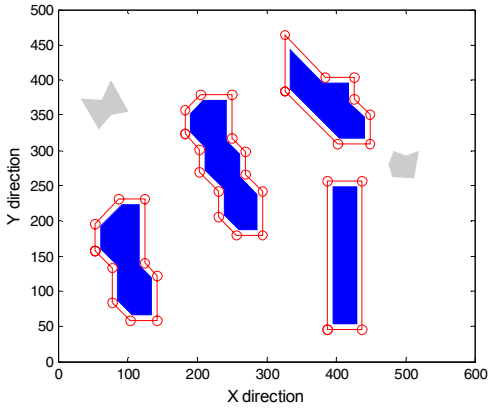

Fig. 1. An example of the environment representation in vertex++ planner. The gray polygons represent exclusion areas surrounding the moving obstacles; the black obstacles are static.

In order to assess the possibility of the robot colliding with dynamic obstacles, the following method has been developed and implemented in vertex++. The first crossing point between the robot path proposed by the planner and the trajectory of a moving obstacle is calculated before examining the possibility of collision. Based on the time t required for the robot to cover the distance from the current position to the first crossing point, the instantaneous location of the moving obstacle can be calculated and, consequently, the exclusion area for this obstacle. If the crossing point falls within this area, a collision would occur between the robot and the moving obstacle. An example of such an occurrence is shown in Fig. 2. Note that the safety margins for the longitudinal and transverse dimensions of the moving obstacle are unlikely to be the same when constructing this area, as the speeds of the robot and the moving obstacle need to be taken into account in addition to the dimensions of the robot. For the problem in Fig. 2, the time t is firstly calculated for the robot to travel from its current location to the crossing point determined from the generated path. The instantaneous location of the moving obstacle after time t can then be calculated according to the motion information relating to the dynamic obstacle, allowing a region to be identified for assessment of feasibility using the algorithm for checking polygon clipping given in [20].

B. Genetic representation

Candidate paths are represented by a chromosome (Fig. 3) consisting of a total number of genes l, where l has a minimum value of two (a path containing only the start and goal points) and maximum value of N+2, where N is the total number of the vertices of all obstacles (both static and dynamic) in the environment. The absolute coordinates of the vertices (x_i, y_i) , i=0,..,l-1, are used directly in the gene representation rather than a reference to one of the N vertices [2]. The robot's speed, s_i in the segment originating from each gene is selected from a set of available discrete speeds. A single bit is also provided in each gene to indicate the feasibility of the path that originates from the gene; if the

path segment connecting two consecutive vertices intersects one or more obstacles, then the infeasibility bit of the gene representing the originating node is assigned 1 to mark this segment as infeasible (it is 0 otherwise).

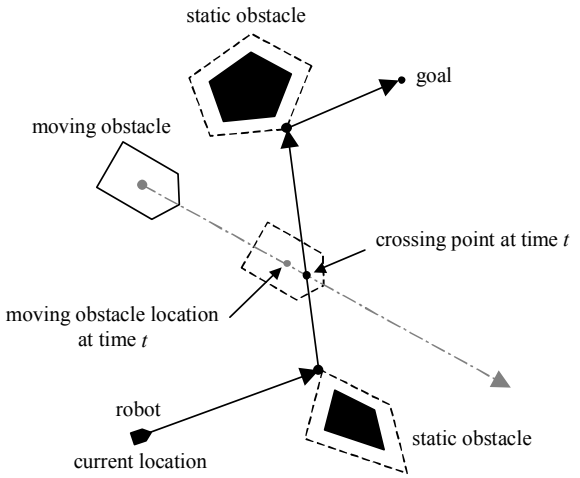

Fig. 2. Evaluation of the possibility of collision with a moving obstacle. Note that the intermediate nodes of the generated path illustrated are vertices of the enlarged static obstacles.

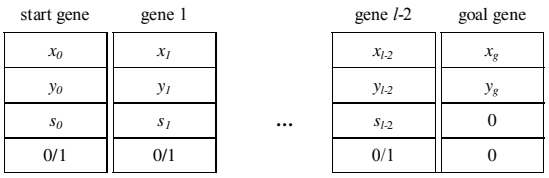

Fig. 3. The structure of a chromosome representing a path from the start to the goal point. The l-2 intermediate genes are vertices of obstacles in the environment. Each gene contains information relating to the vertex's co-ordinates, as well as the robot speed and infeasibility of the segment originating from that gene.

The initial population is generated by randomly choosing for each individual both its length (in the range 2 to N+2) and the coordinates of the vertices contained therein, with the constraints that no vertex is repeated in a individual and that the first and last genes are always the start and goal points respectively. The speed for each gene is selected randomly from a set of discrete speeds.

C. Evaluation functions

Separate evaluation functions are applied to assess the quality of the feasible and infeasible paths. Two parameters, path length and travel time, are considered in the evaluation function E_f for feasible paths, which is given by

$$E_f = w_d \sum_{i=0}^{l-2} d(V_i, V_{i+1}) + w_t \sum_{i=0}^{l-2} t(V_i, V_{i+1}), \qquad (1)$$

where w_d and w_t are the weights for path length and travel time respectively, $d(V_i,V_{i+1})$ denotes the distance between the pair of vertices and $t(V_i,V_{i+1})$ represents the time needed to cover each segment from vertex V_i to V_{i+1} which can be calculated by

$$t(V_i, V_{i+1}) = d(V_i, V_{i+1}) / s_i, \qquad (2)$$

where s_i denotes the speed of the robot when travelling from V_i to V_{i+1}. The infeasible paths are evaluated by the function E_i by considering the deepness of an infeasible path's intersections with obstacles and is given by

$$E_i = \mu + \eta, \qquad (3)$$

where μ denotes the number of obstacle intersections in the path and η is the mean number of intersections in the infeasible segments. Given the two evaluation functions, the aim of the optimization process in the vertex++ planning approach is to minimize the values of E_f and E_i for their respective populations.

When the population contains both feasible and infeasible paths, all infeasible paths are assumed to be no better than the worst feasible path. A sufficiently large constant C is added to the costs for the infeasible paths to ensure the evaluation values of any given infeasible path is worse than the values for all feasible paths. C is defined to be

$$C = (N+2)D, \qquad (4)$$

where N+2 indicates the maximum possible number of genes in an individual and D denotes the maximum length of a path segment (for example, this would be the diagonal in a rectangular environment).

D. Genetic operators and their selection

Three of the total of four genetic operators are the same as those used in the vertex planner in [2], namely, crossover, mutation and repair. The fourth operator, termed 'speedmutation', has been introduced in this work in order to mutate the robot speed indicated in a gene and it is selected with a small probability. In order to keep the number of system parameters to a minimum, the selection of an operator from the four available is made randomly at each generation rather than being based on predefined probabilities. The crossover operation is performed by a conventional one-point operator, following which individuals are examined for repeated vertices, and those replicated vertices of lower locus are removed in order to eliminate circular paths. When the mutation operator is selected, only one bit is modified in the chosen individual. Mutation is inhibited if the replacement genes are already be present in the individual. Note that the mutation rate will depend on the length of the individuals; for example if the average length of the individuals is 10 bits for a certain planning task, then, on average, only 1 bit will be mutated in every fourth generation (there being four operators), giving a mutation rate of 0.025. The repair operator adjusts a randomly selected infeasible segment of an infeasible path, so that it circumnavigates all obstacles previously intersected, as illustrated in Fig. 4.

E. Evolutionary process

As a steady-state GA has been adopted, only one or (following crossover) a single pair of individuals is different in consecutive generations. The generational operation

begins with the random selection of a genetic operator and a quadratic ranking scheme is used to retain the constant selection differential after evaluation. The parent (or parents for the crossover operation) that are involved in the genetic operation are determined by a roulette wheel whose slots are sized in proportion to the fitness as scaled by a ranking technique. To form a new generation, the newly generated offspring replace the worst individual (or pair of individuals if crossover is applied) in terms of fitness in the existing population. The evolutionary process continues until a termination condition is satisfied, which can be defined to be a number of generations specified by the user or determined by monitoring against a specified performance criterion. When the evolution terminates, the best individual is selected as the path planning solution.

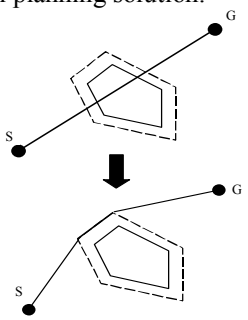

Fig. 4. An illustration of the repair operator that uses the vertices of the enlarged obstacle (shown dashed) to determine a feasible path around the obstacles.

F. On-line planning

On-line planning is triggered automatically to adapt to changes in the movement characteristics of the dynamic obstacles that have occurred since the off-line plan was computed. On-line planning is instigated only when such changes are detected within range of the robot's sensors, otherwise the robot continues to follow the previously-planned trajectory. Information gathered from the robot's sensors with regard to the motion changes of obstacles is supplied to the vertex++ planner which then uses the current state of the robot as the start configuration for its on-line evolutionary planning, and evolves a new path for the robot. The on-line planning algorithm is the same as that used in off-line planning, but with the additional assumption that the planning time is relatively short compared with that needed for the robot to implement motion changes to avoid collision with dynamic obstacles.

III. EXPERIMENTS AND RESULTS

A series of experiments involving the vertex++ planner was conducted in Matlab 7.2 [21] running under Windows 2000 on a 2.8GHz Pentium P4 system. Four simulated environments were conducted in which the on-line planner was required to determine a path through both static and dynamic obstacles, the latter making a number of motion parameter changes. The paths were separately optimized for both travel time and path length. The number of obstacles present in each of the test environments is summarized in Table I.

TABLE I
THE NUMBERS OF OBSTACLES IN THE FOUR TEST ENVIRONMENTS.

environment	number of static obstacles	number of dynamic obstacles
1	4	2
2	5	3
3	9	4
4	14	5

To provide realistic challenges to the planner, the four environments were designed to reflect a representative range of applications in which mobile robots may be expected to operate. Simple trajectories for the dynamic obstacles were designed for the first two environments, with the obstacles simply traveling to and fro between two specified locations. More complex paths for the dynamic obstacles were defined for the remaining two environments, involving speed changes and movement between a series of nodes. For off-line planning, the information regarding the motions of the dynamic obstacles is assumed to be completely known for the four environments before planning. Apart from the robot speed and the optimizing criteria, all parameters remained unchanged throughout the set of experiments and they are listed in Table II.

TABLE II
SYSTEM PARAMETERS FOR VERTEX++ PLANNER. NOTE THAT MUTATION ACTS ON ONLY ONE GENE TO ALTER EITHER THE VERTEX OR THE SPEED OF THE SELECTED SEGMENT AND THAT THE VALUE OF THE SAFE DISTANCE IS DETERMINED FROM THE MINIMUM DISTANCE THAT THE ROBOT CAN APPROACH OBSTACLES WITHOUT COLLISION, TAKING INTO ACCOUNT ITS PHYSICAL DIMENSIONS.

population size	mutation rate (for node)	mutation rate (for speed)	repair rate	safe distance (m)
30	one gene	one gene	one infeasible segment	1

The experiments in this paper concentrate on the results obtained from the on-line planning, although clearly, in all cases, the off-line planning was always performed prior to the robot leaving its start position. For clarity, the results presented show only a limited number of paths; these being illustrative of a much larger set of experiments with these environments. The motions of the dynamic obstacles in the four environments were changed (in both trajectory and speed) after the robot had followed the paths generated off-line for a specified time duration. Table III shows an example set of such obstacle speeds used in the generation of the paths shown in Fig. 5 and Fig. 6. In the experiments, when a change in an obstacle's trajectory or speed was detected by the robot's sensors, a new plan was generated using the updated configuration. Only those portions of the robot paths that were followed after the execution of the on-line planner are plotted in the figures. Fig. 5 reports the trajectories planned with travel time as the evaluation criterion for the feasible paths, whereas Fig. 6 shows the

paths evolved on the basis of minimizing path length.

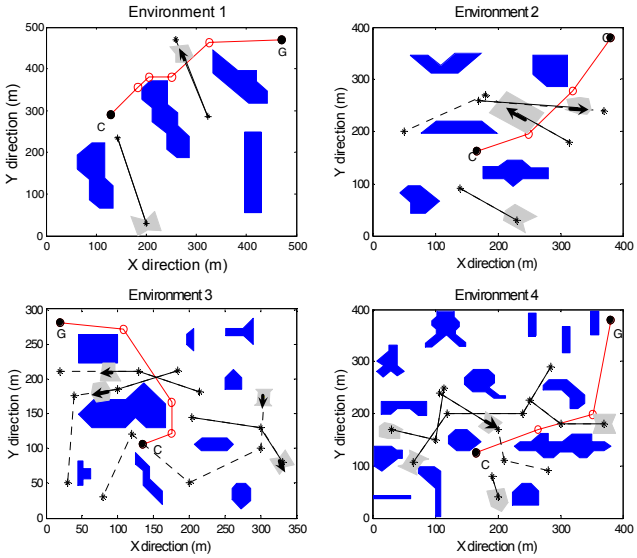

Fig. 5. The paths planned with the goal of minimizing travel time. Motion parameter changes of the obstacles (detected by the robot when positioned at the points marked 'C') occurred after 400, 380, 300, and 320 seconds for the four environments.

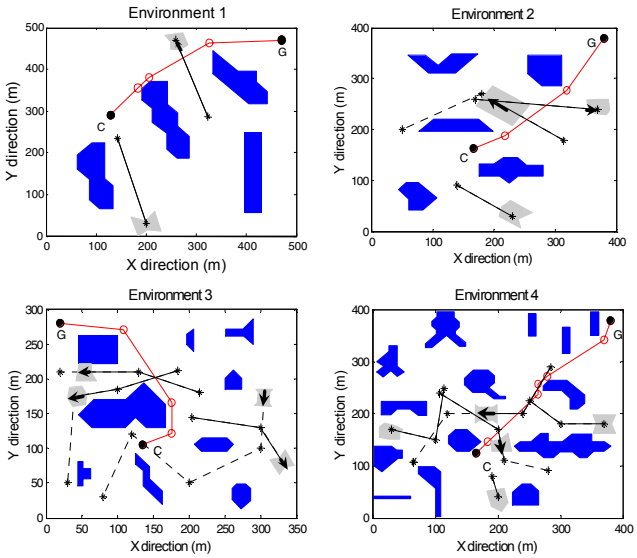

Fig. 6. The paths planned with the goal of minimizing path length. Motion parameter changes of the obstacles occurred as in the results shown for minimizing travel time.

In Fig. 5, the path generated for environment 1 when optimizing for travel time is modified in order to avoid the moving obstacle, but not when optimizing for path length (Fig. 6). For environment 2, the re-planned path when minimizing path length is in fact the same as that planned before the obstacle changed trajectory, whereas, on careful examination, it can be seen that the trajectory when minimizing for travel time has required minor modifications to the choice of intermediate nodes. The paths presented in both Fig. 5 and Fig. 6 for environment 3 include, as intermediate nodes, the same vertices in same order. However, neither path for environment 4 is the same as that generated off-line, with the planned robot movements being updated by the on-line planner to adapt to the modified motions of the dynamic obstacles. In general, it can be seen that, as a result of different optimization criteria, the trajectories produced when optimizing path length are generally shorter than those obtained when minimizing for travel time.

TABLE III
EXAMPLE MODIFIED SPEED PARAMETERS GENERATED FOR EACH MOVING OBSTACLE PATH SEGMENT.

environment	obstacle 1	obstacle 2	obstacle 3	obstacle 4	obstacle 5
1	0.1	0.55	-	-	-
2	0.4, 0.1	0.2	0.5	-	-
3	0.3, 0.5	1.1, 0.4	0.7, 0.4, 0.6	1.2, 0.5, 0.4, 0.8	-
4	0.9	1.4, 0.7	1.1, 0.2, 0.5	0.3, 1.1	1.6, 0.5, 0.6

The quantitative results shown in Table IV for the experiments in all four environments, demonstrate that the maximum speed (0.7 ms^{-1}) is the one most frequently chosen from the set available when the optimization goal is travel time, whereas lower speeds were selected more frequently when the path length is the optimization criteria. Additionally, the number of generations and execution time to find the first feasible path, as well as the execution time for 1000 generations in the on-line process are generally less than those found for off-line planning [2]. This can be explained by the fact that the length of the individuals is generally shorter when performing on-line planning, simply because the robot has advanced closer to the goal and there will likely be fewer intermediate vertices. Table IV also shows that when optimizing for travel time, although the maximum speed was frequently chosen, occasionally a faster path could be obtained by reducing speed to more efficiently avoid a dynamic obstacle. In contrast, optimizing for the minimum path length necessitated more frequently reversion to lower speeds and consequently these paths were likely to consume significantly less power to drive the robot to its destination.

The rapid convergence in the first phase when conversion from infeasible paths to feasible paths takes place, can be attributed to the repair operator that was developed following experimental observation. The mutation operator is less effective in the initial evolution stages as relatively few bits are mutated. As only one gene in the population is modified when the mutation operator is applied, its effect is diluted in the earlier generations' populations due to the greater prevalence of longer individuals. The effective increase in the mutation rate in the later evolutionary generations effectively promotes population diversity (and so exploration into virgin areas of search space) and inhibits premature convergence. The quality of final solution produced in the second phase appears to be highly dependent on the quality of the initial individuals that are supplied following the operations of the first phase. If the initial supply for the second evolution phase is sparse (in that all the necessary building-blocks for the global optimal solution

are not present), the process may be led into a local minimum. The higher mutation rate apparent in later evolutionary steps promotes diversity by modifying the inherited building blocks, however the mutation is not sufficiently dominant in the process to necessarily avoid the GA becoming trapped in a local minimum. The vertex++ planning algorithm has been demonstrated as being capable of generating an optimal or near-optimal path for the robot in a relatively short time compared with 9EP/N++ which is also able to operate in environments containing dynamic obstacles. The current implementation has been carried out in Matlab and a significant improvement (reducing the calculation time by a factor of five to ten times) is to be expected when executed in a compiled language such as C.

TABLE IV
EXPERIMENTAL RESULTS FOR THE ON-LINE PLANNING.

environment		cost in terms of		generations to find first feasible path	execution time to find first feasible path (s)	execution time for 1000 generations (s)	planned robot speed (ms^{-1})
		path length (m)	travel time (s)				
optimize for travel time	1	419	606	64	5.63	29.9	0.7, 0.6, 0.7, 0.7, 0.7
	2	317	546	54	5.41	33.6	0.4, 0.7, 0.7
	3	300	428	48	10.4	47.9	0.7, 0.7, 0.7, 0.7
	4	385	551	84	28.7	106	0.7, 0.7, 0.7
optimize for path length	1	408	1198	12	2.72	25.9	0.7, 0.3, 0.3, 0.3
	2	311	990	60	5.11	33.6	0.4, 0.3, 0.3
	3	300	512	45	7.56	51.3	0.7, 0.3, 0.7, 0.7
	4	346	1061	75	29.4	108	0.6, 0.3, 0.3, 0.5, 0.3, 0.3

IV. CONCLUSIONS

This paper has described the restriction of the search space of the GA planner to include only the vertices of obstacles and has resulted in a significant reduction in planning time, permitting not only rapid off-line planning, but also on-line planning on a timescale suitable for real-time implementation in most practical circumstances.

The work has also introduced a novel GA-based navigation system that is able to vary the speed of the robot through evolutionary progress, allowing a greater number of alternative paths to be considered. Where travel time is crucial it was found that, in the environments considered, full speed can be applied in most segments of the trajectory. However, environments could easily be conceived in which operating at full speed throughout the trajectory would result in a much extended path due to the robot's circumnavigation of dynamic obstacles. If minimizing the path length is more important, for example to conserve energy, lower speeds can be planned and this is likely to result in a shorter overall path.

REFERENCES

[1] C.-K. Yap, "Algorithmic motion planning," in *Advances in Robotics vol. 1: Algorithmic and Geometric Aspects of Robotics*, J. T. Schwartz and C.-K. Yap, Eds. Hillsdale, New Jersey: Lawrence Erlbaum, 1987.

[2] Y. Wang, D. J. Mulvaney and I. P. W. Sillitoe, "Genetic-based mobile robot path planning using vertex heuristics," *2006 IEEE Conf. Cybernetics and Intelligent Systems*, pp.463-468.

[3] D. W. Payton, J. K. Rosenblatt and D. M. Keirsey, "Grid-based mapping for autonomous mobile robot," *Robotics and Autonomous Systems*, vol. 11, no. 1, pp. 13–21, 1993.

[4] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," *2005 International Conference on Automated Planning and Scheduling*, pp. 262-271.

[5] R. C. Arkin, "Navigational path planning for a vision-based mobile robot," *Robotica*, vol. 7, pp. 49-63, 1989.

[6] C. O'Dunlaing, and C.-K. Yap, "A retraction method for planning the motion of a disc," *Journal of Algorithms*, vol. 6, pp. 104–111, 1982.

[7] H. Mitchell, "An algorithmic approach to some problems in terrain navigation," *Artificial Intelligence*. vol. 37, pp. 171–201, 1988.

[8] Y. Davidor, *Genetic algorithms and robotics: a heuristic strategy for optimization*. Singapore: World Scientific Publishing, 1991.

[9] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 18–28, 1997.

[10] H. Lin, J. Xiao and Z. Michalewicz, "Evolutionary navigator for a mobile robot," *1994 IEEE Conf. Robotics and Automation*, pp. 2199–2204.

[11] J. Xiao, Z. Michalewicz and L. Zhang, "Evolutionary planner/navigator: operator performance and self-tuning," *1996 IEEE Conf. Evolutionary Computation*, pp. 366–371.

[12] K. Trojanowski, Z. Michalewicz and J. Xiao, "Adding memory to the evolutionary planner/navigator," *1997 IEEE Conf. Evolutionary Computation*, pp. 483–487.

[13] J. Xiao, "Evolutionary planner/navigator in a mobile robot environment," in *Handbook of Evolutionary Computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds, New York: Oxford University Press and Institute of Physics Publishing, 1997.

[14] A. C. Nearchou, "Path planning of a mobile robot using genetic heuristics," *Robotica*, vol. 16, pp. 575–588, 1998.

[15] Z. Cai and Z. Peng, "The application of a novel encoding mechanism in path planning for a mobile robot," *Robot*, vol. 23, pp. 230–233, 1997.

[16] R. Smierzchalski, and Z. Michalewicz, "Modeling of ship trajectory in collision situations by an evolutionary algorithm," *IEEE Trans Evolutionary Computation*, vol. 4, pp. 227–241, 2000.

[17] R. Smierzchalski and Z. Michalewicz, "Adaptive modeling of a ship trajectory in collision situations at sea," *1998 IEEE Conf. Evolutionary Computation*, pp. 342-347, 1998.

[18] R. Smierzchalski and Z. Michalewicz, "Path planning in dynamic environments", in *Innovations in Machine Intelligence and Robot Perception*, S. Patnaik, L. C. Jain, G. Tzafestas and V. Bannore, Eds, Berlin: Springer-Verlag, 2005.

[19] D. J. Mulvaney, Y. Wang, I. P. W. Sillitoe, "Waypoint-based mobile robot navigation," *2006 IEEE World Congress on Intelligent Control and Automation*, pp. 9063-9067.

[20] T. Pavlidid, *Polygon clipping, Algorithms for Graphics and Image Processing*. Rockville, MD: Computer Science Press, 1982, ch. 15.

[21] Matlab product family, Available: http://www.mathworks.com/.