Hybrid Evolutionary Motion Planning via Visibility-Based Repair

Gerry Dozier, Albert Esterline, Abdollah Homaifar, Marwan Bikdash

NASA Center for Autonomous Control Engineering
North Carolina A&T State University
Greensboro, NC 27411, USA
{gvdozier,esterlin,homaifar,bikdash}@ncat.edu

Abstract— **This paper introduces a hybrid evolutionary system for global motion planning within unstructured environments. This hybrid system combines a novel obstacle representation scheme, the concept of evolutionary search and a new concept that we refer to as the visibility-based repair to form a hybrid which quickly transforms infeasible motions into feasible ones. Our hybrid evolutionary system differs from other evolutionary motion planners feasible motions rather than using simulated evolution exclusively as a means of discovering feasible motions, (2) a continuous map of the environment is used rather than a discretized map, and (3) it develops motion plans for moving or mobile destinations. In this paper, we demonstrate the effectiveness of this new hybrid by using two challenging motion planning problems where the target tries to move away from our point robot.**

Keywords—**Motion Planning, Visibility Graphs, Visibility-Based Repair.**

I. Introduction

EVOLUTIONARY Algorithms (EAs) typically solve difficult problems for which traditional search paradigms yield unsatisfactory results. EAs have been successfully applied to a variety of areas such as design optimization, machine learning, constraint satisfaction, and constrained optimization [9]. Recently, there has been a growing number of successful applications of EAs to the area of motion planning [1, 2, 6, 8, 9, 11, 12]. The Motion Planning Problem [5, 7, 10] can be stated as follows. Given an environment $E(R, X, T, O)$ where R represents a point robot, X represents the starting point (or current position of R), T represents the goal or destination point and O represents a set of obstacles, find a collision free (feasible) path from X to T (path planning phase) that R can traverse (navigation phase).

This paper introduces a hybrid evolutionary system for global motion planning within unstructured environments. This new hybrid combines a novel obstacle representation scheme, the concept of evolutionary search and a new concept that we refer to as the visibility-based repair to form a hybrid which quickly transforms infeasible motions into feasible ones. Our hybrid evolutionary system differs from other evolutionary motion planners in that (1) more emphasis is placed on repairing infeasible motions to develop feasible ones rather than using simulated evolution exclusively, (2) a continuous map[1] of the environment is used rather than a discretized map, and (3) it develops motion plans for mobile destinations. In this paper, we demonstrate the effectiveness of this new hybrid by using two challenging motion planning problems where the target tries to move away from our point robot.

[1] [6, 8, 9] also use use continuous maps.

The remainder of the paper is organized as follows. Section 2 provides a brief introduction to the concept of visibility-based search. In Section 3, we present our hybrid evolutionary system called GEPOA (A Global Evolutionary Planning and Obstacle Avoidance system) in detail and, in Section 4, we introduce our test suite. In Section 5, we present our results and conclusions.

II. Visibility-Based Search

A visibility-based search algorithm can be regarded as any search procedure that uses a visibility graph (VG) [2, 5] to aid in the discovery of feasible paths. A VG is a graph, (V, E), where V is the set of all vertices of the obstacles within an environment including the coordinates of the robot (X) and the destination, and E is the set of all edges connecting any two vertices in V that do not pass through any obstacles within the environment.

Figure 1 shows an example of a visibility graph. Notice that X is connected only to the vertices that are reachable by way of a straight-line segment that does not cut or pass through (or violate) any obstacles. Notice also that the vertices which are visible to R at X are connected, in similar fashion, to other vertices that are visible to them (again by way of straight-line segments).

Once a VG has been constructed for a given environment, usually an A^* search algorithm is used to find the shortest path between starting and destination points. One advantage of using a visibility graph is that every motion path from the point of origin to the destination is a feasible path. One disadvantage of using a visibility graph is that it takes $O(n^2)$ time [5] to construct, where n is the number of vertices within the environment. This may be a greater disadvantage when developing feasible paths within dynamic environments.

It is not always necessary to construct a complete VG for an environment. Some researchers have experienced a great deal of success with using partial VGs (PVGs) [2]. Figure 2 shows an example of a PVG. An advantage of using a PVG is that a PVG requires less computation. A

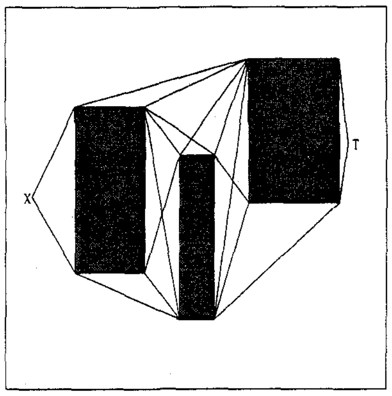


Fig. 1. A Visibility Graph

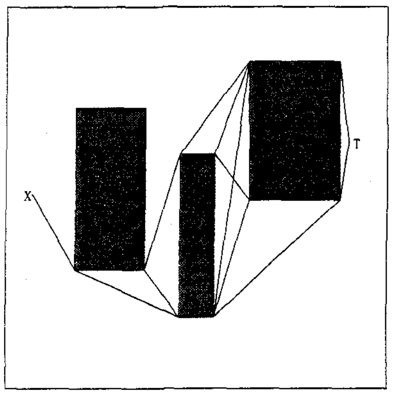


Fig. 2. A Partial Visibility Graph

disadvantage of using a PVG is that the PVG may not contain an optimal path to the destination. However both of these methods, used as is, will perform poorly on path planning problems with dynamic environments.

III. GEPOA

In this section, we discuss seven salient attributes of GEPOA (A Global Evolutionary Planning and Obstacle Avoidance system). These attributes are as follows: the representation of environments, the concept of visibility-based repair, the representation of candidate paths (CPs), the visibility-based repair algorithm, the evaluation function, the selection algorithm, and the evolutionary operators. We conclude this section by providing the parameter settings for four GEPOA hybrids that will be tested with our test suite of two path planning problems.

A. Representation of an Environment

An obstacle within an environment is represented as a set of $\lceil \frac{k_i}{2} \rceil$ intersecting line segments, where k_i represents the number of vertices of obstacle i. Each line segment, $l_{i,j}$, connects two distinct vertices $v(i,j)$ and $v(i, j + \lfloor \frac{k_i}{2} \rfloor)$ (where $j \leq \lceil \frac{k_i}{2} \rceil$) of obstacle i. Associated with each vertex within the environment is a value which represents the number of obstacles which contain it. This value is referred to as the 'containment value' (CV) of a vertex. If a vertex lies along the boundary of an environment its CV is assigned a value of ∞.

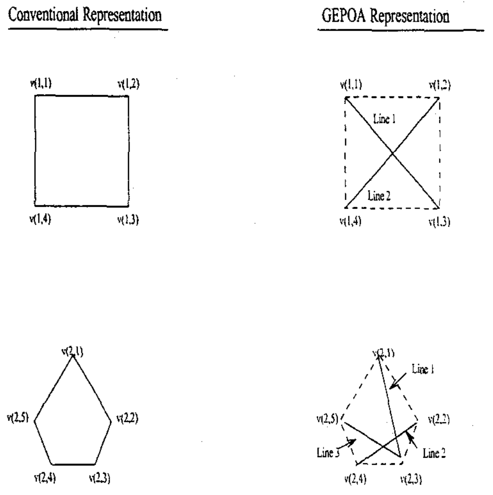


Fig. 3. Obstacle Representation in GEPOA

Figure 3 provides an example of how obstacles are represented in GEPOA. Notice, in Figure 3, that the four sided obstacle (Obstacle 1) is represented by only two lines in GEPOA. Line 1 of Obstacle 1 connects vertices $v(1,1)$ and $v(1,3)$ while Line 2 connects vertices $v(1,2)$ and $v(1,4)$. Obstacle 2 has five sides and is represented in GEPOA using three lines. Line 1 connects $v(2,1)$ and $v(2,3)$, Line 2 connects $v(2,2)$ and $v(2,4)$, and Line 3 connects $v(2,3)$ and $v(2,5)$. Since each of the vertices are contained by only one obstacle, the CV for each vertex is 1.

B. Visibility-Based Repair

Visibility-based repair (VBR) is performed as follows. When an obstacle, o_i, lies along a straight-line segment between two nodes P and Q, each line of o_i is checked to see if it is intersected by PQ. If a line of o_i is intersected by PQ then a repair node is created using the following set of rules:

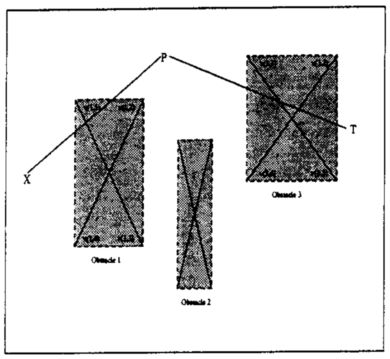
Rule 1: if the CVs of a line's vertices are both equal to one, then the repair node is selected to be a point along an extention [2] of the vertex which is closer to the point of intersection;

Rule 2: if the CVs of a line's vertices are different, then the repair node is selected to be a point just outside of the vertex which has the lower CV;

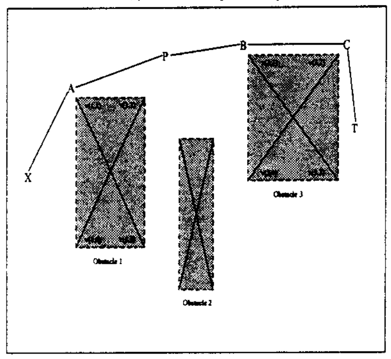
Rule 3: if the CVs of a line's vertices are greater than one and equal, then the repair node is selected to be a point just outside of the vertex which is farther from the point of intersection.

Figure 4 shows an example of how VBR can be used to transform an infeasible path into one that is feasible. In Figure 4a, an infeasible path XPT is shown. The path XPT is infeasible because the line segment XP passes through Obstacle 1 and the line segment PT passes through Obstacle 3. Before proceeding further, notice that

[2] The distance outside of an obstacle at which a repair node is placed is a user specified parameter.



a) Before Visibility-Based Repair



b) After Visibility-Based Repair

Fig. 4. Visibility-Based Repair

each vertex in the environment shown in Figure 4a has a CV of one.

Using VBR, the line segment XP can be repaired to XAP. Since XP intersects Line 1 of Obstacle 1, a repair node corresponding to a point just outside of either v(1,1) or v(1,3) must be selected. By applying Rule 1, Node A, which corresponds to a point just outside vertex v(1,1), is selected as the repair node.

Similarly, the line segment PT can be repaired to $PBCT$. Again Rule 1 must be applied to Line 1 and Line 2 of Obstacle 3. The repair node that results from the intersection of PT and Line 1 is Node B. The repair node that results from the intersection of PT and Line 2 is Node C. Figure 4b shows the result of using VBR on XPT. The repaired, feasible version of XPT is $XAPBCT$.

C. Representation of Candidate Paths

An individual representing a candidate path (CP) contains four fields. The first field is a chromosome which contains a gene corresponding to the cartesian coordinates of each node of the path (where each node of a path is connect by a straight-line segment). The second field is called the seed. The seed of an individual is the gene that will be crossed or mutated to created an offspring. Initially, an individual will have only three genes: the start gene, the seed gene and the destination gene. Repair genes are inserted into the chromosome by the visibility-based repair algorithm each time a straight-line segment of an individual is found to pass through an obstacle. The third field is a value referred to as the violation distance. The 'violation distance' represents the euclidean distance of the CP which cuts through one or more obstacles. The fourth field records the euclidean distance of the path from the start to destination genes.

D. The VBR Algorithm Used by GEPOA

Given a CP, the VBR algorithm used by GEPOA works as follows. Each obstacle within the environment is checked with each straight line segment from the start gene to the destination gene of the CP until a segment is found that passes through the obstacle. The infeasible segment is repaired via VBR and the process is repeated using the next obstacle.

For an example of how this repair algorithm works, notice once again Figure 4. When given the path XPT the algorithm works as follows. Obstacle 1 is checked to see if it is violated by segment XP. Since it is, a repair gene (Node A) is generated and Obstacle 2 is then considered. Obstacle 2 is checked to see if it is 'cut' by segment XA. Since it is not 'cut' by segment XA, Obstacle 2 is checked with segment AP then segment PT. Since there are no more segments to inspect, Obstacle 3 is considered. Obstacle 3 is checked to see if it is 'cut' by segments XA, and AP. Finally, Obstacle 3 is checked to see if it is 'cut' by PT. Since it is, two repair genes are generated (Nodes B and C) and the algorithm terminates.

Since this repair algorithm considers an obstacle only once, it is possible for a repair gene to be generated that creates a line segment that 'cuts' through a previously considered obstacle. Therefore, a CP may need to repaired by this VBR algorithm more than once.

E. Evaluation and Selection

The evaluation function computes the euclidean distance of each straight line segment of the path that an individual represents as well as the violation distance. GEPOA uses a modified version of tournament selection, with a tournament size of 2, to select individuals to become parents. The selection process is as follows. Two individuals are randomly selected from the current population. If the violation distances of the two are different then the individual with the smaller violation distance is selected to be a parent. If the violation distances are the same then the individual with the smaller 'overall' distance is selected to be a parent.

F. The Evolutionary Operators

GEPOA uses two operators along with VBR to create and/or refine individuals. [3] The two operators are as follows: (1) a version of Radcliffe's Flat Crossover [4] that we refer to as seed crossover and (2) a version of uniform mutation we refer to as uniform seed mutation.

Seed crossover is as follows. Given two seed genes $s_1 = (x_1, y_1)$ and $s_2 = (x_2, y_2)$, a seed gene for an offspring is created as follows: $s_{off} = (rnd(x_1, x_2) + N(0, 4.0), rnd(y_1, y_2) + N(0, 4.0))$, where rnd is a uniform

[3]The VBR algorithm is applied to parents representing infeasible CPs 50% of the time as well as all newly created offspring.

random number generator and $N(0, 4.0)$ is a gaussian random number with zero mean and a standard deviation of 4.0. The resulting offspring, $\langle X, s_{off}, T\rangle$, has a chromosome containing three genes: a gene corresponding to the node representing the current position of R (X), the seed node, and the destination node. The offspring then undergoes VBR and may have additional repair genes added by the VBR algorithm.

In uniform seed mutation, either the x or y coordinate of a parent is mutated using uniform mutation to create a seed gene for an offspring. A resultant offspring created by seed mutation is similar to one created by seed crossover in that it also has a chromosome containing three genes. Once again the offspring undergoes VBR and may have additional repair genes added by the VBR algorithm.

G. *Attribute Settings for GEPOA Hybrids*

Four GEPOA hybrids with population sizes of 10, 20, 30 and 40 were tested with a test suite of two motion planning problems. These four hybrids differ only in population size. Each hybrid randomly generates its initial population, uses steady-state reproduction [3], uses a seed crossover rate of 0.66 and a uniform seed mutation rate of 0.34.

Every ten generations R is advanced a maximum of one unit along the shortest path to T developed by the hybrid. The amount of advancement is based on the change of direction of R. If no change of direction is needed, R is advanced one unit. If the change of direction is 90° or greater, R is not advanced. After R has been advanced, T is allowed to move a half unit in a direction (north, south, east, or west) furthest away from the current positon of R[4]. This process is repeated until R reaches T.

IV. The Test Suite

Figure 5 shows our test suite of two motion planning problems that will be solved by the four hybrids described above. In each of the test environments, X represents the starting point of R and T represents the mobile destination. Test Environment 1 is an unstructured environment with a variety of four-sided obstacles. In Test Environment 1, X is located at point (1.0,19.0) and T is located at point (19.0,10.0). Test Environment 2 has X located at point (10.0,17.0) and T is located at point (10.0,1.0). In Test Environment 2, R must work its way out of a box while T moves in the opposite direction.

V. Results and Conclusions

Each hybrid was run 100 times on each of the test environments. Each run was alotted a maximum of 100 moves. A run was considered successful if the robot reached the target. On each run, the hybrids never failed to find a feasible solution within the initial population. Also, the motion plans evolved by the hybrids allowed the robot to reach the target 100% of the time. This is an indication of the effectiveness of visibility-based repair.

[4]Each time either R or T is moved each individual in the population is re-evaluated.

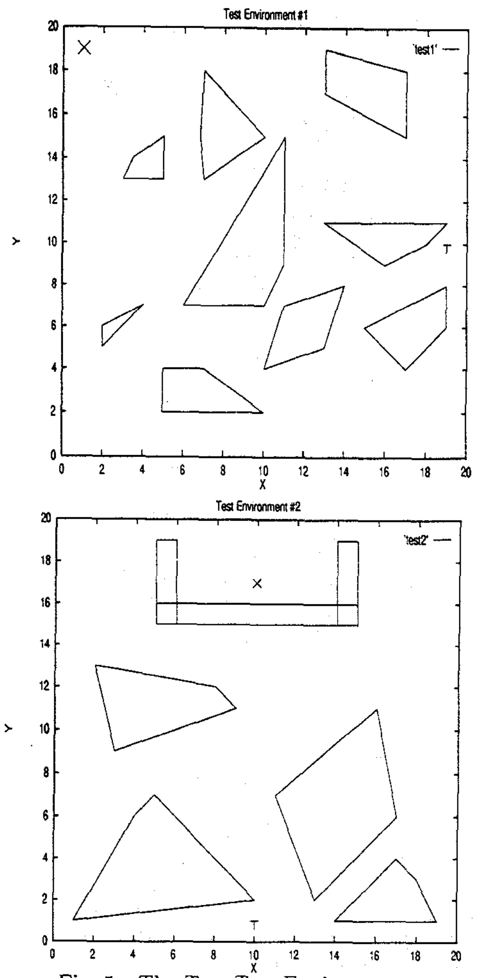


Fig. 5. The Two Test Environments

The results of the performances of the hybrids on both of the test environments are presented in Figures 6 an 7. They are organized into a matrix where the columns (from left to right) correspond to:

- the population size of the hybrid (P),
- the average length of the first feasible solution found ($\overline{Ln.1^{st}}$),
- the standard deviation of the length of the first feasible solution found during each successful run ($\sigma(Ln.1^{st})$),
- the average number of moves needed to reach the destination ($\overline{Moves}$), and
- the standard deviation of the number of moves need to reach a target ($\sigma(Moves)$).

In Figure 6, each of the hybrids develop initial (feasible) paths with identical lengths. The averages of the number of moves needed to reach a target are all fairly close. However, $P = 30$ has the best performance (because it required fewer moves) followed by $P = 20$, $P = 10$, and $P = 40$.

Figure 7 shows the performance of the four hybrids on Test Environment 2. For each column in Figure 7, the numbers decrease as the the population size is increased. However, once again the performances of the hybrid are close. $P = 40$ requires the fewest number of moves to reach its destination.

When viewing the results at first glance one my be lead to conclude that the larger the population size the better

P	$Ln.1^{st}$	$\sigma(Ln.1^{st})$	$Moves$	$\sigma(Moves)$
10	21.147	0.000	31.010	4.997
20	21.147	0.000	30.360	4.344
30	21.147	0.000	29.940	1.618
40	21.147	0.000	31.680	6.665

Fig. 6. Performances on Test 1

P	$Ln.1^{st}$	$\sigma(Ln.1^{st})$	$Moves$	$\sigma(Moves)$
10	26.981	0.991	40.440	7.224
20	26.526	0.693	39.850	6.452
30	26.300	0.458	39.290	5.224
40	26.141	0.411	39.040	5.058

Fig. 7. Performances on Test 2

the performance. However, the results in Figures 6 and 7 favor the performance of $P = 10$. The reason is that, in our current implementation, whenever R or T moves the entire population is re-evaluated (updated). According to recent research [13], this process of updating the entire population is not necessary. One could simply replace the oldest individual each time an offspring is added to the population. Using this method, $P = 10$ and $P = 40$ would create and evaluate exactly 10 individuals between each move of R. Regardless of population size, the results show the advantage of using visibility-based repair along with evolutionary search. Figure 8 provides an example solution for the two problems in our test suite. In Figure 8, the diamonds represent the steps taken by R and the plus signs represent the movement of T.

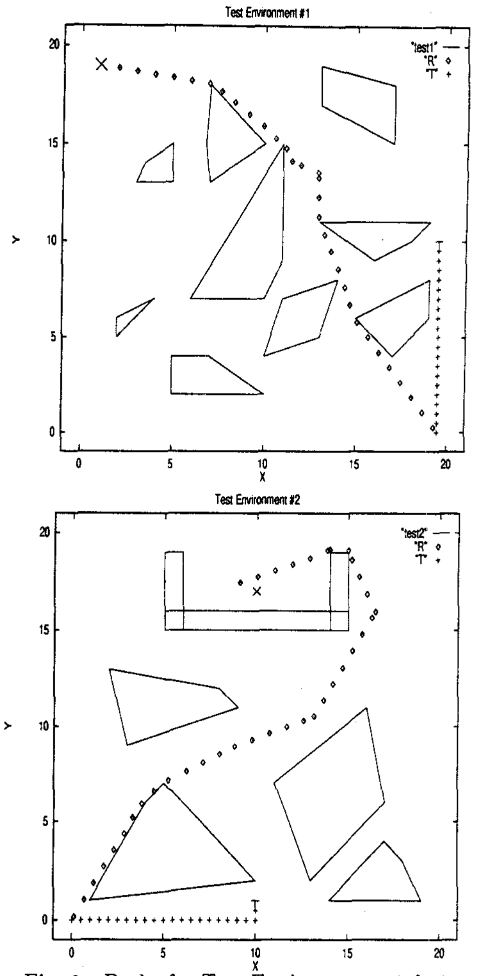


Fig. 8. Paths for Test Environments 1 & 2

Acknowledgments

This work is funded by grants from the NASA Autonomous Control Engineering Center under grant number ACE-48146. The authors wish to thank them for their financial support.

References

[1] Ashiru, I. and Czarnecki, C. (1995). Optimal Motion Planning for Mobile Robots Using Genetic Algorithms, *Proceedings of the 1995 IEEE/IAS International Conference on Industrial Automation and Control*, pp. 306-313.

[2] Chang, T.-Y., Kuo, S-W., and Hsu, J. Y.-J. (1994). A Two-Phase Navigation System for Mobile Robots in Dynamic Environments, *Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pp. 297-300.

[3] Davis, Lawrence. (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 115 Fifth Avenue New York, NY 10003.

[4] Eshelman, L. J. and Shaffer, J. D. (1993). Real-Coded Genetic Algorithms and Interval-Schemata, in *Foundations of Genetic Algorithms II*, pp. 187-202, ed. L. Darrell Whitley, Morgan Kaufman Publishers.

[5] Fujimura, K. (1991). *Motion Planning in Dynamic Environments*, Springer-Verlag.

[6] Hocaoglu, C. and Sanderson, A. (1996). Planning Multi-Paths Using Speciation in Genetic Algorithms, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 378-383.

[7] Latombe, J.C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers.

[8] Lin, H.-S., Xiao, J., and Michalewicz, Z. (1994). Evolutionary Algorithm for Path Planning in Mobile Robot Environment, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 211-216.

[9] Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd Edition, Springer-Verlag.

[10] Michalewicz, Z., and Xiao, J. (1995). Path Evaluation in the Evolutionary Planner/Navigator,*Proceedings of the IEEE/SOFT International Workshop on Biologically Inspired Evolutionary Systems*, pp. 45-52.

[11] Xiao, J., Michalewicz, Z., and Zhang, L. (1996). Evolutionary Planner/Navigator: Operator Performance and Self-Tuning,*Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 366-371.

[12] Shibata, T., and Fukuda, T. (1993). Robot Motion Planning by Genetic Algorithm with Fuzzy Critic, *The 8th IEEE International Symposium on Intelligent Control*, pp. 565-570.

[13] Vavak, F. and Fogarty, T.C. (1996). Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 192-195.