Planning Multiple Paths with Evolutionary Speciation

Cem Hocaoglu and Arthur C. Sanderson, Fellow, IEEE

Abstract—This paper demonstrates a new approach to multidimensional path planning that is based on multiresolution path representation, where explicit configuration space computation is not required, and incorporates an evolutionary algorithm for solving the multimodal optimization problem, generating multiple alternative paths simultaneously. The multiresolution path representation reduces the expected search length for the path-planning problem and accordingly reduces the overall computational complexity. Resolution-independent constraints due to obstacle proximity and path length are introduced into the evaluation function. The resulting path-planning system has been evaluated on problems of two, three, four, and six degrees of freedom. The resulting paths are practical, consistent, and have acceptable execution times. The system can be applied for planning paths for mobile robots, assembly, and articulated manipulators. Generation of multiple alternative paths is an example of multimodal search and, in our previous work, a new approach to multimodal function optimization has been developed using a genetic algorithm (GA) with minimal representation size cluster analysis. This multipopulation GA identifies different species and is used as the basis for an evolutionary multipath-planning algorithm which generates multiple alternative paths simultaneously. The multipath algorithm is demonstrated on a number of two-dimensional path-planning problems.

Index Terms—Evolutionary speciation, path planning.

I. INTRODUCTION

CREATING autonomous systems is of great interest in a wide variety of application domains such as manufacturing, space exploration, construction, undersea exploration, and medical surgery. Developing such complex autonomous systems requires research in areas of control, automated reasoning, and perception. Current and future research efforts in this area will continue to strive for increased robustness and flexibility, better reliability, and greater autonomy. The path-planning problem arises in attempts to develop more autonomous robotic systems. This capability is necessary for autonomous robots since it is essential for a robot to accomplish tasks by moving in the real world. This requires the ability to plan a path. The path-planning problem involves determining if a continuous and obstacle-avoiding sequence of positions and orientations of the robot exists from the initial position and orientation of the robot to the goal position and orientation and if so, to specify such a path. Fig. 1 shows a simple two-dimensional (2-D) path-planning example. The path-planning problem is computationally very expensive. In fact, it is an NP-hard problem [1]. An upper bound on the complexity of the \( n \) degree-of-freedom path-planning problem is \( O(n^6) \), which means that the complexity of the path-planning problem grows exponentially with the number of degrees of freedom [1]–[3]. Thus, researchers are actively seeking satisfactory, computationally efficient solutions to the path-planning problem.

A large number of path-planning approaches have been developed, but a number of difficulties remain.

1) Configuration Space Representation and Search Maps: It has been shown that the general path-planning problem is PSPACE-hard [1]. Polynomial-time algorithms exist under certain assumptions and usually for motion planning problems with polygonal objects in 2-D workspaces. Many existing global approaches to path planning assume that a complete representation of configuration space (C-space) has been computed before searching for a path under a search map. Building a search map and searching for a path under such a map contributes to the computational complexity of the existing approaches. Global approaches [4]–[6] are, in general, complete in that if a path exists, it will be found. However, generating the complete C-space is computationally very expensive. In general, the complexity grows exponentially with the number of degrees of freedom of a robot [2], [3], [7]. Local methods such as potential field approaches [8], [9] are more efficient techniques, but they could get trapped in a local minimum.

2) Path Representation: Most of the traditional path planners utilize a fixed resolution path representation. Path resolution refers to the discretization used to represent paths in the algorithm. This may cause problems when dealing with a variety of different environments. For example, more complex environments may require a higher resolution of path representation. On the other hand, it

Fig. 1. 2-D path-planning example. Diagram shows a collision-free path from initial position of the robot to the goal position.
may be more efficient to use a lower resolution of path representation for less complex environments. In general, it is more efficient to adapt the resolution of the path representation with the complexity of the work environment.

3) **Lack of Robustness to Changes:** Most of the traditional planners have only one explicit optimization criterion for the motion planning problem. In general, the optimization criterion is to find the shortest path from the initial location to the goal under the search map structure. Such an objective is only meaningful for special cases where the robot is a point, disc, or a sphere [10]. Planners with a shortest path objective may also implicitly satisfy other optimization goals imposed by the search map as well. For example, roadmap planning approaches utilizing a Voronoi diagram as a search map also have maximum clearances from the obstacles. Artificial potential fields-based path planners can be designed to incorporate more general optimization goals than the shortest path criterion. Relaxation-based path planners attempt to modify a path to meet additional optimization criteria [11]. However, there is no general path-planning approach that can accommodate a variety of different optimization criteria and handle changes in these optimization goals without changing the characteristics of the planner or its search map, especially within a single search engine.

Most of the traditional off-line planners do not take into account uncertainties. Recently, Hu *et al.* [12] tried to achieve robustness by planning paths with nonzero width. Local planners are in general robust in dealing with small uncertainties such as preventing a collision of the robot with an obstacle, but on the other hand, local methods are also not robust in dealing with large uncertainties such as avoiding a local minimum of the search space.

4) **Lack of Alternative Solutions:** Traditional path planners find a single solution to the path-planning problem based on an optimization criterion such as finding the shortest path. In some situations, it would be useful to have several alternative solutions to the path-planning problem rather than a single path, i.e., in dynamic environments, one or more of these paths may become infeasible, so one of the feasible alternatives can then be chosen.

The multipath evolutionary path planner with multiresolution path representation, which is described in this article, has been motivated by the drawbacks of the existing approaches outlined above and the following observations.

1) Evolutionary computation techniques are proven effective in solving hard problems such as job shop scheduling [13] and the traveling salesman [14]. Randomized search can be very efficient and effective in escaping local minima [15], [16]. Evolutionary approaches look for a solution in parallel, where individual candidates interact through genetic operators to generate possibly better solutions. The evolutionary approaches can also easily be implemented on a massively parallel machine to achieve superlinear speed up with the number of processors [17]. The evolutionary approaches to path planning are not confined to searching for a solution in a search map that may affect the quality of the search.

2) Early applications based on evolutionary algorithms have used standard genetic algorithms (GAs) with a binary string representation as an encoding of the potential solutions to the problem. This type of binary string encoding of the candidate solutions is unnatural for many problems [18]–[21] and in some cases makes GAs a weak method. The power of evolutionary computation can be better utilized by representing a candidate solution with a problem-friendly data structure, i.e., a representation of candidate solutions that fits the problem characteristics and a suitable design of genetic operators possibly incorporating domain-specific knowledge [19]–[21]. Michalewicz [20] calls such an approach an evolution program approach, where standard evolutionary computation techniques are adapted to exploit problem-specific knowledge.

3) One of the distinguishing features of evolutionary computation techniques from other randomized search algorithms is that they facilitate parallel interactions among parallel search actions in a pool of individuals to achieve better performance. For example, crossover combines two individuals to generate possibly more fit individuals.

4) The computational inefficiencies of traditional planners are largely due to the computational expenses in mapping the workspace into C-space and in building a search map for the problem. A path planner could be greatly simplified in both time and space if there were no need for such a C-space mapping and an abstract search map. Abstract search maps may limit the quality of the search by confining the search to be within the search map.

5) The shortest path optimization criterion is only meaningful for special cases and is usually not an ideal optimization goal. In general, an ideal optimization criterion can be very complex and subjective. In traditional planners, the search maps are usually discretized approximations of the original continuous environment so the shortest path found in such a search map may not be the optimum path in the real workspace. In general, the shortest path represents the best path for the model used (the search map).

6) A flexible and efficient path planner that can be applied to a wide variety of planning applications, such as mobile and articulated robots, and assembly trajectory problems in multidimensional spaces is preferable.

7) Since it may be difficult to combine all the important factors into a single evaluation function, a few important criteria can be used in constructing the optimization criteria and generating alternative solutions to the problem. From these selected solutions, one can then choose the best one based on further and more subjective analysis of the problem. Generating alternative solutions may also be important in dynamic environments, where one or more of the alternatives may become infeasible. So, an alternative path can then be chosen from the feasible alternatives.

The multipath evolutionary planner described in this paper incorporates the above ideas by using an evolutionary computation approach. The algorithms developed use the concepts of evolutionary computation with problem-specific representation.
of candidate solutions and genetic operators [22]–[24]. The multipath evolutionary path planner does not require a search map, accommodates different optimization criteria, and allows changes in the optimization criteria without changing the characteristics of the planner. It uses an iterative multiresolution path representation as a basis for the genetic encoding of candidate paths. The multiresolution representation forms the basis for a computationally efficient search technique, where a simple path will be found quickly if one exists. The individual candidates are evaluated with respect to the workspace so that computation of the C-space is not required. Alternative paths are discovered by utilizing minimal representation size principles and speciation techniques in evolutionary algorithms.

Section III gives a description of the representation used for encoding paths. Section IV describes the genetic operators of the evolutionary path-planning algorithm. Section V presents the evaluation function of the path planner. Section VI describes the evolutionary path-planning algorithm. Section VII describes the experimental setup and presents results of the simulations performed for two, three, four, and six degree-of-freedom path-planning problems. Section VIII reports experiments performed on scaling and computational issues. Section IX describes the multipath-planning algorithm based on the evolutionary speciation algorithm and provide experimental results for generating multiple paths for various planning environments. Section X discusses future extensions to the multipath-planning algorithm and draws conclusions from current results.

II. BACKGROUND

A review of the existing approaches for solving path-planning problems is provided in [7]. A survey of existing motion planning algorithms is presented by Schwartz [25] and Sharir [26]. In general, calculus-based methods can become stuck in local minima since they search for a solution locally. Enumerative search techniques are very inefficient as the search space becomes too large to explore. Random search techniques, on the other hand, are probabilistically complete, but may take a long time to find a solution [7]. The proposed technique based on evolutionary computation combines the advantages of global search without the major disadvantages of inefficient enumeration in a large search space. Although as a random search technique it is probabilistic, the use of a multiresolution path representation overcomes traditional disadvantages of random search.

The Euclidean space in which the robot is moving is called a workspace and it is represented as $\mathbb{R}^N$, where $N = 2$ or 3. A configuration of an arbitrary object is a specification of the position of every point in this object relative to a fixed reference frame. A configuration of a robot $\mathbf{R}$ is a specification of the position and orientation of the robot’s reference frame with respect to the world coordinate frame. The C-space of $\mathbf{R}$ is the space of all configurations of $\mathbf{R}$. Thus, each possible configuration of robot $\mathbf{R}$ is represented by a point in the C-space. The path-planning problem becomes equivalent to the motion planning of a point robot in C-space. The C-space obstacles are generated by mapping the original obstacles to the C-space of $\mathbf{R}$. Collision-free space (C-free) refers to the locus of all points in the C-space that represent feasible and collision free configurations.

Namgung and Duffy [27] have recently developed a 2-D collision-free path planner using a linear parametric curve. Objects are mapped into control point space from Euclidean space through intersection checks between path and obstacles. Their paper investigates paths having only a single control point. Barraquand and Latombe [28] describe a potential field-based method called RPP that uses a gradient descent approach to reach the goal configuration. The planner uses random walks to escape from local minima. In [29], a more elaborate potential field-based algorithm is proposed, where a dynamic programming technique is applied to various submanifolds of the C-space. It is slower, but capable of solving harder problems than RPP. When the solution path has to pass through a narrow space, RPP takes a long time to find this narrow region of the space. In [30], a subgoal graph called probabilistic roadmap is developed to overcome this problem. Random configurations are generated and collision-free configurations are kept in a graph. The reachability between subgoals is checked using a simple local planner. The graph is augmented with additional subgoals in the cluttered C-space to increase the connectivity of the graph. The path planning is done by connecting the start and goal configurations to the subgoals on the graph and traversing the graph to connect the two subgoals. Švestka and Overmars [31] describe a learning approach for robot motion planning of two types of carlike robots: normal ones and robots that can only move forward. The motion planning process has been split into two phases: the learning phase and the query phase. In the learning phase, a probabilistic roadmap is constructed incrementally in the C-space. Nodes in this roadmap correspond to randomly chosen configurations in C-free and edges correspond to simple collision free paths between the nodes. In the query phase, the roadmap is used to find paths between different pairs of configurations. The motion planning algorithm is based on their previous work, where the method has been applied to planar articulated robots with many degrees of freedom [32]. Kavraki et al. [30] and Barraquand et al. [33] analyze the random motion planners that they have proposed theoretically. Their scheme consists of randomly sampling the robot’s C-space, keeping the collision-free configurations, and trying to connect them by simple paths. The result of this process is a roadmap where the nodes are the collision-free configurations and the edges are the simple paths. They showed that the randomized planners are probabilistically complete. That is, a path will be found if one exists and if the planner is run long enough. The algorithms will find a solution with high probability ($1 - q$), where $q$ is the failure probability, if the number of collision-free configurations is large enough. It has also been shown that for a family of configurations called expansive spaces [34], the number of collision-free configurations grows as $\log(1/q)$. The randomized planners are applicable to virtually any path-planning problem.

Chen and Hwang [35] present a dynamic graph search algorithm called SANDROS for motion planning of manipulators, rigid objects, and multiple rigid objects. It uses a hierarchical nonuniform multiresolution best-first strategy to find a heuristically short motion in the C-space. The SANDROS planner
Fig. 2. (a) Genetic representation of an encoded path. \( \Delta d_1 \)'s represent \((D-1)\)-dimensional points. Numbers in the nodes of the binary tree represent the order in which the nodes are processed (post-order traversal of the binary tree). (b) Decoded set of path vertices and their sequence. Numbers in the nodes represent the sequence in which path vertices are ordered along the path (in-order traversal of the binary tree). Leftmost leaf node is the starting point and the rightmost leaf node is the goal point. Path sequence is \((S, P_2, P_3, P_2, P_4, P_5, P_6, P_7, G)\).

is resolution complete and its computation time depends on the problem difficulty as measured by the solution-path complexity. At first, distance computations are performed to determine whether a given point is in C-free. Then, a two-level hierarchical planning method is used to reduce memory requirements. Since it is difficult to store all the collision-free points, both a global and local planner is used. The global planner keeps track of reachable, unreachable, and potentially reachable portions of the C-space and the local planner checks the reachability of a portion of space from a point. The global planning module is used first to search promising portions of the C-space at a coarse resolution. If a solution is not found at the coarse level, it increases the resolution to finer levels in promising portions of the C-space. A set of subgoals is maintained to be used by the robot as guidelines in moving to the goal configuration. Post processing of a solution path is required since it typically has many noise-like kinks. The algorithm is efficient for problems with low degrees of freedom \((<10)\), but SANDROS' performance may still be exponential in \(\eta\).

Recently, there have been a few path planners developed using evolutionary algorithms [21], [36]–[40]. Early evolutionary planners [37], [38] used traditional GAs without utilizing domain-specific knowledge. Most recent evolutionary approaches [21], [41] focused on 2-D mobile robot path-planning problems. Learning-based approaches using GAs for tracking and local navigation are presented in [42], [43]. Bessiere et al. [40] describe a path planner with parallel implementation for a robot moving in a dynamic environment. They use a fixed length encoding of paths based on Manhattan motions. Xiao et al. [21] present an evolutionary planner/navigator (EP/N) for path planning and navigation. They utilize two GAs: one for off-line planning and the other for on-line planning. Their algorithm incorporates domain-specific knowledge, can accommodate different optimization criteria, and can self-tune its performance for different environments by adjusting the probabilities of its genetic operators. EP/N has been developed for a point robot in 2-D workspaces. Chen and Zalzala [41] apply GAs to search for near-optimal paths for a mobile robot with a distance-safety criteria and solve the multicriteria optimization problem. They use a grid-by-cell decomposition to represent the environment and two numerical grid potential fields for the goal and obstacles. The algorithm developed is mainly for 2-D problems.

In this paper, we describe an application of multiresolution representation and evolutionary computation to multipath planning and multidimensional path planning and demonstrate a variety of tasks with up to six degrees of freedom.

III. MULTIRESOLUTION PATH REPRESENTATION

This section describes the iterative multiresolution path representation that is used for encoding candidate solutions to the path-planning problem [22], [23]. An iterative multiresolution path representation can reduce the expected search length for the path-planning problem. If a successful path is found at a low level of resolution (early in the search hierarchy), then further expansion of that portion of the path search is not necessary. This advantage is mapped into the encoded search space and adjusts the path accordingly.

Each path is represented by a hierarchically ordered set of \((D-1)\)-dimensional vectors, \(\Delta d_i\) for a \(D\)-dimensional path-
planning problem. The \( \Delta d_i \)'s are ordered in a binary tree as shown in Fig. 2(a). Each \( \Delta d_i \) uniquely defines a point in the \((D-1)\)-dimensional space defined by the orthogonal bisector of two \( D \)-dimensional path vertices. In two dimensions, the space defined by the orthogonal bisector of two path vertices is a line, in three dimensions, it is a plane, etc. Thus, each \( \Delta d_i \) represents an intermediate path vertex. Given the start and goal points, Fig. 4 shows the construction of the path using \( \Delta d_i \)'s given by the preorder traversal of the binary tree shown in Fig. 2(a). This construction process provides the intermediate path vertices in a binary tree as shown in Fig. 2(b). The start point is appended as the left child of the leftmost node and the goal point is appended as the right child of the rightmost node of the binary tree of path vertices. Given two path vertices and a \( \Delta d_i \), the intermediate path vertex is computed using a modified Gram–Schmidt orthogonalization process \[44\], \[45\]. Fig. 3 sets up the problem in two dimensions. \( P_s \) and \( P_g \) represent the two path vertices and the computed intermediate path vertex is represented by \( P_i \). After computing a unit direction vector (first basis vector), \( d_1 \), \( D-1 \) \( D \)-dimensional unit vectors are generated using the modified Gram–Schmidt orthogonalization process. The computed set of basis vectors gives us an orthonormal transformation matrix \( \text{Trans} \). The intermediate path vertex \( P_i \) is then given by

\[
P_i = P_c + \text{Trans} \times \Delta d_i
\]

where \( \Delta d_i = [0,0, \Delta d_2, \cdots, \Delta d_{D-1}]^T \). The sequence of path vertices is obtained by an in-order traversal of the path vertex tree. The sequence of path vertices defined by Fig. 2(b) is \((S, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, G)\).

**IV. EVOLUTIONARY OPERATORS**

The use of a multiresolution representation with variable-length encoding introduces interesting opportunities for evolutionary operators with respect to the binary tree structures. The evolutionary path planner utilizes one crossover operator and six mutation operators. These operators manipulate the genetic material in the encoded path representation of intermediate path vertices. The characteristics of these operators are outlined below.

1) **Swap-Subtree Crossover:** Recombines two paths to reproduce two new ones. A node is picked randomly from each parent and the subtrees represented by each node are swapped as shown in Fig. 5(a). The binary trees can have different numbers of nodes and the subtrees swapped may have different topologies and may belong to different portions of their corresponding parent tree (see Fig. 5).

2) **Perturb_1 Mutator:** Randomly picks a node in the binary tree representation and perturbs the contents of the node by a small amount. Once a feasible path is found by the evolutionary process, the operator probabilities are adjusted such that the probability of selecting this operator for mutation is higher. This operator allows for fine tuning paths for clearances and optimality. The example in Fig. 5 shows a possible effect of this evolutionary operator on an example path.

3) **Peturb_2 Mutator:** Perturbs the contents of a node in the binary tree encoding of a path by a large amount. It randomly selects a node in the binary tree representation. At the early stages of the evolutionary process, this mutation operator has a high probability of being selected so that infeasible path nodes and segments can be perturbed to feasible regions. Once a feasible path is found, the probability of selecting this operator is reduced. The example in Fig. 5 shows a possible effect of this evolutionary operator on an example path.

4) **Fix Mutator:** Attempts to fix an infeasible path by: a) by perturbing infeasible nodes [see Fig. 5 (Fix Mutator: Part a)] of a path and b) inserting a new intermediate node for infeasible segments of a path as shown in Fig. 5 (Fix Mutator: Part b). The probability of selecting this operator at the early stages of the evolutionary process is high and is reduced to a low value after feasible paths are found.

5) **Swap-Node Mutator:** Swaps two randomly picked nodes in a binary tree representation of a path (see Fig. 5).

6) **Insert-Node Mutator:** Creates a new intermediate path node by inserting a leaf node to the binary tree. Fig. 5 shows a possible impact of this operator on an example path.

7) **Flip Mutator:** Changes the sign of the contents of a randomly picked node (see Fig. 5).
V. EVOLUTIONARY EVALUATION FUNCTION

Let a path $p$ be represented by an ordered set of nodes joining the starting node and the goal node be denoted by $n_{i,j}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$. The cost function $C(p)$ is defined by

$$C(p) = K_1 \cdot D(p) + K_2 \cdot \sum_{i=1}^{N} \sum_{j=1}^{M} U_{rep}(n_{i,j}) + K_3 \cdot R(p)$$

where $D(p)$ represents the traveling distance and the second term in the evaluation function represents the repulsive potential of the path. The third term $R(p)$ represents the ratio of the number of infeasible nodes to the number of feasible nodes. $K_1, K_2, K_3$ are positive weighting factors among the three criteria. In the experiments performed, the three criteria have equal weight ($K_1 = K_2 = K_3 = 1.0$). Fig. 6 demonstrates how the evaluation function components behave for various situa-
Fig. 6. Components of the evolutionary evaluation function. (a) Behavior of the path length component of the evaluation function. This component of the evaluation function prefers shorter paths. In this case, path(2) is preferred over path(1). (b) Repulsive potential component of the evaluation function prefers paths that are not in close proximity to the obstacles. In this case, path(1) is preferred over path(2). (c) Paths that make fewer collisions with obstacles are preferred by this component of the evaluation function. In this case, path(2) is preferred over path(1).

The traveling distance is defined by the sum of all the segments of the path joining the start point and the goal point

$$D(p) = \sum_{i=1}^{N-1} \| (n_{i+1} - n_i) \|$$

where $\| \cdot \|$ represents the Euclidean norm.

The parameter $M_f$ in the repulsive potential term represents the number of steps required from node $n_i$ to node $n_{i+1}$ with increments of $\Delta d$, where $\Delta d$ is a discretization constant ($\Delta d = 0.2$ for all the simulations performed). The repulsive potential $U_{rpf}(n_{ij})$ is defined by

$$U_{rpf}(n_{ij}) = \begin{cases} C - \frac{1}{\rho_0} - \frac{1}{\rho_c} & \text{if } \rho(n_{ij}) \leq \rho_c \\ \frac{1}{\rho_0} & \text{if } \rho_c \leq \rho(n_{ij}) \leq \rho_0 \\ 0 & \text{if } \rho(n_{ij}) > \rho_0 \end{cases}$$

where $\rho(n_{ij})$ represents the shortest distance to the obstacles from node $n_{ij}$; $\rho_0$ and $\rho_c$ are high and low threshold constants, respectively, and $C$ is the proximity constant. In our experiments, which are described in Section VI, $\rho_0 = 1.0$, $\rho_c = 0.001$, and $C = 1000$. We utilize the I-COLLIDE [46] package for detecting collisions. Note that in the evaluation function, there is no explicit term that penalizes paths with many nodes. The repulsive potential along with the distance term acts like a complexity term that discourages paths with many nodes.

VI. EVOLUTIONARY PATH PLANNING ALGORITHM

The evolutionary path-planning algorithm uses a multiresolution path representation in the form of binary trees as detailed in Section III and incorporates a steady–state GA [47] with roulette wheel selection, a crossover operator, several mutation operators (see Section IV), and an evaluation procedure, as described in Section V. The steady–state GA uses overlapping populations based on a user-specified amount of overlap (50% of the population in our implementation). The steady–state GA is also referred to as $(\mu + \lambda)$—evolution strategy [48], where $\mu$ parents create $\lambda \geq 1$ offspring individuals by means of recombination and mutation. The $\mu$ best individuals out of a pool of parents and offspring are selected to form the next population. In our implementation, $\lambda = \mu / 2$.

Each individual in the population represents a (feasible or infeasible) path leading the robot to the goal location (see Section III for details on path representation). Each individual is evaluated using the evaluation criterion described in Section V. In the evolutionary loop, a set of individuals is selected for evolutionary crossover and mutation. A mutation operator is selected on the basis of a probability distribution. The crossover operator transforms two individuals (parents) into two new offspring by combining parts from each parent. The mutation operator operates on a single individual and creates an offspring by mutating that individual. The newly generated set of individuals are added to the existing population and evaluated based on the fitness function. The worst individuals of the extended population are then removed to reduce the population to its original size. In this scheme of evolution, the new individuals may only make it to the next generation if they have a higher fitness value than the worst individuals in the original population. The process terminates after a certain number of generations.

A high-level description of the evolutionary path-planning algorithm is as follows.

1. Form uniformly sampled initial population(s) having $\text{PopSize}$ individuals.
2. Evaluate candidate paths.
Fig. 7. (a) Paths found by the evolutionary path planner for a point robot in three different 2-D planning environments. (b) Current best performance of the algorithm versus generations count. All three simulations were run with a population size of 30 for 200 generations.

3. If the termination condition is not reached, go to Step 4. Otherwise, terminate execution. The algorithm terminates when the maximum number of generations is performed.
4. Select a certain percentage of the population for crossover and mutation.
5. Apply crossover operator to the selected individuals based on the crossover probability.
6. Select a mutation operator based on operator selection probabilities and probabilistically apply it to the selected individuals.
7. Add the newly generated individuals to the population.
8. Evaluate the newly generated candidates.
9. Remove the worst individuals from the extended population to return the population to its original size. The newly created individuals may only survive to the next
Fig. 8. Frequency histogram of genetic operations at various points of the evolutionary process for the top simulation in Fig. 7. Histograms show the number of times (cumulative) each operator is selected and applied to the individuals of the population during the evolutionary process. They are indicative of roles of different operators in leading to the chosen outcomes.

10. Go back to Step 3.

VII. EXPERIMENTAL RESULTS OF EVOLUTIONARY PATH PLANNING

The multiresolution path-planning algorithm has been implemented in a UNIX C++ programming environment on a Pentium PC running Linux and on a Sun Sparc 20 workstation running Solaris 2.5. The software developed also utilized portions of the GAlib library [47] and the I-COLLIDE [46] collision detection package. Experiments were conducted for planning paths in 2-D and three-dimensional (3-D) workspaces involving free-flying rigid and articulated robots with degrees of freedom varying from two to six. A few experimental trials demonstrated that a crossover probability of 0.65 and a mutation probability of 0.35 works well with the chosen genetic operators and for path-planning problems. Thus, the above probabilities are used in all the simulations. The selection of the mutation operator at each generation is done based on assigned probabilities to these operators. Table I (column 2) shows these assigned probabilities before a first feasible path is found. The assigned probabilities are adjusted after a first feasible path is found (shown in Table I, column 3). The same set of algorithm parameters were used throughout all the experiments with no arbitrary adjustment or selection of algorithm parameters. In some cases, the number of generations and population size have been adjusted to accommodate more difficult search spaces. These are reported for each simulation.

Section VII-A demonstrates experimental results for two degree-of-freedom experiments with either a point or a polygonal robot. In Section VII-B, robots having three degrees of freedom (two translational and one rotational in a 2-D worksite and three translational in a 3-D worksite) were used in planning paths in 2- and 3-D workspaces. Section VII-C presents results of planning paths for free-flying articulated robots having four degrees of freedom in 2-D work environments. In Section VII-D, results of six-dimensional path-planning experiments for polyhedral robots are described.

A. Two Degree-Of-Freedom Experiments

The two degree-of-freedom experiments were conducted with either a point robot or a planar robot on 2-D workspaces. Various experiments were performed with varying degrees of path-planning difficulty. Fig. 7 shows paths planned for a point robot in 2-D environments. The algorithm automatically adjusts the resolution of paths based on the complexity of the work environment and the difficulty of the path-planning problem with respect to the path representation. The best path found for the problem in the top left of Fig. 7 is represented by eight path vertices including the start and goal positions and is defined by the following sequence of path vertices: \{(1.0
Fig. 9. Frequency histogram of genetic operations at various points of the evolutionary process for the middle simulation in Fig. 7. Histograms show the number of times (cumulative) each operator is selected and applied to the individuals of the population during the evolutionary process. They are indicative of roles of different operators in leading to the chosen outcomes.

The plots on the right of each path-planning environment in Fig. 7 show the best current performance of the evolutionary process versus the generations count. Figs. 8–10 show the frequency histogram of the genetic operators at various points of the evolutionary process for the simulations in Fig. 7. These frequency histograms show the number of times (cumulative) each genetic operator is actually selected and applied to the individuals of the population during the evolutionary process. These histograms are indicative of the role of different operators in leading to the chosen outcomes. According to these frequency diagrams, before a feasible path is found, Swap-Subtree Crossover, Perturb_2 (large perturbations), Fix, and to some extent, Swap-Node play significant roles. After a feasible path is found, Swap-Subtree Crossover, Perturb_1 (small perturbations), and to some extent Insert-Node are used more frequently than other genetic operators.

In Fig. 11, increasingly more complex workspaces are presented for planning paths for a point robot from a specified starting position to a goal location. Previously found paths have been blocked by adding more obstacles to the workspace of the robot. In all cases, the evolutionary path planner is able to find intuitively good paths.

Fig. 12 shows the path planned for a point robot in a 2-D worksite. Fig. 12(a) represents the initial population of paths for this run. Fig. 12(b) provides a snapshot of the populations of individuals at generation 12. The population of paths at generation 100 is shown in Fig. 12(c). Fig. 12(d) shows the best path obtained at generation 100.

B. Three Degree-Of-Freedom Experiments

Figs. 13 and 14 demonstrate the motion of a rigid rectangular robot having three degrees of freedom (two translational and one rotational) in 2-D work environments. The motion of a L-shaped robot (nonconvex) is shown in Fig. 15. The L-shaped robot has three degrees of freedom (two translational and one rotational) in 2-D planning environments. The L-shaped robot has to travel with rotations in order to reach the goal position without any collisions. The motion of a cross-shaped robot (nonconvex) is shown in Fig. 16. The cross-shaped robot has three degrees of freedom (two translational and one rotational) in a 2-D planning environment. In the evaluation function, no angular constraints are specified so any feasible orientation of the robot is as good as any other feasible one. Even though that is the case, the evolutionary path planner is able to generate practical motions. However, the approach is very flexible in allowing specification of such constraints in the evaluation function. The motions of polyhedral robots with three translational degrees of freedom...
in 3-D workspaces are shown in Fig. 17. In Fig. 17, the robot is confined in a closed box with an opening on top. There is a rectangular block in the center of the outer box. The robot has to go around the inside block from the starting position (bottom corner) to a goal position outside the opening on the top corner of the outer block. The diagrams on the right of Figs. 14–18 show the mean performance curves for 30 trials of each studied problem setting. These curves give an indication of the probability of discovering useful solutions to the problems studied based on multiple samples of the same problem. They also offer an assessment of the reliability of the path-planning approach.

C. Four Degree-Of-Freedom Free-Flying Articulated Robot Experiments

Fig. 18 shows the motion of a free-flying articulated robot with four degrees of freedom (two translational and two rotational) in a 2-D environment. The robot has to deform into a shape using its rotational degree of freedom at its joint in order to travel through the narrow regions of the workspace. The path planning of articulated robots present an interesting class of problems. The robustness and generality of the evolutionary path-planning algorithm to a wide variety of path-planning domains are further revealed by these articulated robot path-planning experiments.

D. Six Degree-Of-Freedom Experiments

The motion of a six degree-of-freedom (three translational and three rotational) polyhedral rigid robot in a 3-D work environment is demonstrated in Fig. 19. The diagram on the right of Fig. 19 shows the best current score of the population versus the generation count during the evolutionary process. The motion of a L-shaped (nonconvex) six degree-of-freedom (three translational and three rotational) rigid robot in a 3-D work environment is demonstrated in Fig. 20. The robot has to travel with rotations through a narrow opening in the rectangular object. The diagram on the right of Fig. 20 shows the best current score of the population versus the generation count during the evolutionary process.

VIII. SCALING AND COMPUTATIONAL ISSUES

In this section, we perform experiments to assess scaling and computational issues of the evolutionary path planner. Two types of experiments were performed.

Category 1) Experiments with increasing problem dimensionality. The complexity of the environment (number of obstacles) and the complexity of the required solution path change as well.

Category 2) Experiments with increasing number of obstacles. The complexity of the required solution path does not change or changes incrementally.
Table II shows the execution times for the experiments in Category 1. As shown in Table II, the computation times are most strongly related to the complexity of the required path, and are not dominated by the dimensionality of the problem. Thus, Timed Experiment 1 (Fig. 12), a 2-D problem, actually required the largest computation time due to the number and spacing of the obstacles, while Timed Experiment 5 (Fig. 20) required less computation time for a full six degree of freedom plan. The number of obstacles (or faces) in the environment certainly impacts the computation time for collision checking (see Category 2 experiments), but the arrangement and spacing determine the local path resolution required and, therefore, the number of path vertices. The results of these experiments indicate that the execution times do not increase exponentially with the dimensionality of the problem unlike most of the traditional path-planning approaches which are exponential in the degree-of-freedom of the robot.

Table III shows the execution times for the experiments in Category 2. In these experiments, we have used a large number of obstacles and systematically increased the number of obstacles (number of faces) in the experiments without greatly affecting the complexity of the solution path. The purpose of these experiments were to report execution times on environments with large number of obstacles and also to assess the scalability of the algorithm implementation to even larger environments. Figs. 21–23 show the 3-D workspaces and the results of planning a path for a free-flying polyhedral robot having six degrees of freedom (three translational and three rotational). In Table III, the execution times for finding the first feasible path increased from 42–57 s to 69 s for 3-D workspaces having 208, 310, and 406 faces, respectively. The differences in times are due mostly to the increase in the number of distance computations for environments with larger number of obstacles.

IX. MULTIPATH PLANNING USING EVOLUTIONARY SPECIATION

In path planning, it may be difficult to combine all the important factors into a single evaluation function. There are evolutionary algorithms especially devised for solving multiobjective optimization problems [49]–[53]. Another way to approach this problem is to use a few important criteria in constructing the evaluation function and generating alternative solutions to the problem using evolutionary computation. From these selected paths, one can then choose the best one based on further analysis of the problem. For example, in dynamic environments one or more of these plans may become infeasible, so one of the feasible alternatives must then be chosen.

In this section, we are interested in finding multiple, alternative paths simultaneously. Fig. 24 shows an example where three solution paths exist from start to the goal location. To our knowledge, no other path-planning method has been developed for generating multiple alternative solutions simultaneously. We have developed and implemented a unique technique that is based on the concept of evolutionary speciation using minimal representation size criterion and cluster analysis. We make use of MRSC_GA [44], [45] with fitness sharing, which utilizes multiple populations to track species for generating multiple alternative paths simultaneously.

The minimal representation criterion was introduced as a criterion for model inference by Segen and Sanderson [54]. The ideas of Solomonoff [55], Kolmogorov [56], and Chaitin [57] pioneered the development of algorithmic information theory to which the method of minimal representation size is related. The approach was demonstrated on problems in attributed image matching [58], polynomial fitting and clustering [44], [45], [54].

According to the minimal representation size criterion, the best description of the data is given by the smallest size program. This corresponds to the model with the minimal representation size.
In [44] and [45], we formulated speciation in GAs using minimal representation size clustering. The clustering problem was posed as a statistical model identification problem where within-class probability distributions and the distribution of each class were known in a general form. A unique GA using minimal representation size cluster analysis was designed and implemented for solving multimodal function optimization problems [44], [45]. MRSC_GA allows one to run multiple populations of chromosomes that evolve separately with occasional cross-species interactions among populations. Section IX-A describes the multipath-planning algorithm with evolutionary speciation that generates multiple alternative paths simultaneously.

A. Multipath-Planning Algorithm

The concepts of speciation in evolutionary processes as described in [44] and [45] and the evolutionary path-planning algorithm presented in Section VI were used to develop an evolu-
Fig. 14. (a) Path of a rectangular robot having three degrees of freedom (two translational and one rotational) in a 2-D environment (Generation = 300, PopSize = 50). Diagram shows the best path found by the evolutionary path planner based on the evaluation function. (b) Mean performance curves for 30 trials. Curve is an indication of the probability of discovering useful solutions to the problem studied based on multiple samples of the same problem.

Fig. 15. (a) Path of an L-shaped rigid robot having three degrees of freedom (two translational and one rotational) in a 2-D environment (Generation = 300, PopSize = 30). Robot has to use its rotational degree of freedom in order to avoid hitting the obstacles when traveling from start to the goal location. Diagram shows the best path found by the evolutionary path planner based on its evaluation function. Robot posed itself in a V-shape orientation before getting into the constrained region of the workspace, then it moved through the narrow region in this orientation almost in a straight line, and at the end of the constrained region, the robot took its final position and orientation at the goal location. (b) Mean performance curves for 30 trials. Curve is an indication of the probability of discovering useful solutions to the problem studied based on multiple samples of the same problem.

The evolutionary multipath-planning algorithm incorporates a modified version of the MRSC_GA, which has been described in our previous work [44], [45], and the evolutionary path planner with multiresolution path representation presented in this paper as a basis for generating multiple alternative paths. The modified MRSC_GA uses a steady-state GA with fitness sharing instead of a simple GA. Fig. 25 illustrates the key components of the evolutionary process of the multipath-planning algorithm. The evolutionary process may begin with only a single population of individuals that may evolve into multiple species based on the minimal representation size clustering. The sharing EPP runs sequentially, processing each species until the next cluster interval is reached. One generation of the sharing EPP involves selection based on scaled scores, crossover, mutation, and evaluation procedures. The same crossover and mutation operators as well as the same evaluation procedure are used as those detailed for the evolutionary path planner. Cross-species interaction is an operator similar to the crossover operator in a single population. In cross-species interaction, the individuals are selected from different species and then crossover operator is applied to produce two new offspring. The cross-species interaction is controlled by Cross-Species-Rate. The cross-species interaction is performed for all pairs of populations. Cross-Species-Rate is usually very low in order not to destroy the resulting speciation. A Cross-Species-Rate of zero means that no cross-species interactions ever happen. Minimal representation size cluster analysis is performed at each Cluster-Interval (see Fig. 25). Cluster analysis is a basis for forming species and drawing boundaries among them. In our implementation, the KMEANS clustering algorithm with Euclidean distance metric is used to classify encoded individuals (the space of path representations)
into various species. The number of species is determined by the minimal representation size criterion. The individuals are then distributed into their respective locations according to the minimal representation size classification algorithm.

Fitness sharing is used to encourage speciation within a single population and to keep the existing species discovered by the minimal representation size clustering stable so that they are not destroyed during the process of single population evolution between clustering intervals. In the fitness sharing scheme, an individual’s objective score is derated by an amount specified by the sharing function. The amount of scaling depends on the similarity of that individual to other population members. The sharing function used in the multipath-planning algorithm is as follows:

\[ Sh(d_i) = \begin{cases} 1 - \left( \frac{d_i}{\sigma_{share}} \right)^\alpha, & \text{if } d_i < \sigma_{share}; \\ 0, & \text{otherwise.} \end{cases} \] (1)

The parameter \( \alpha \) determines the degree of convexity of the sharing function. It controls the degree of sharing between neighboring individuals. We use an \( \alpha \) value of 1.0, which results in a triangular sharing function. The cutoff value \( \sigma \) the limiting distance between the individuals to be shared and is shown in Table IV for each experiment performed. The
parameter $d_i$ is the distance between current individual and the individual $i$. The distance function must return a value of zero or higher, where zero means that the two values are identical (no diversity). The distance function used in this work is based on the Euclidean distance between the contents of the root nodes of two individuals in encoded (representation) space. For an individual, the scaled score is given by

$$
\hat{f} = f \cdot \sum_{i=1}^{N} S_h(d_i)
$$

where $N$ is the number of individuals in the population.

The high level multipath-planning algorithm is as follows.

1. Form uniformly sampled initial population(s) having $PopSize$ individuals.
2. Set current number of populations $CurNumPop = Initial_No_Pops$.
3. Run the evolutionary path-planning algorithm (see Section VI) with fitness sharing until generation counter has value Cluster-Interval. At this point, perform a Cross-Species Interaction among individuals from different species (populations). See Fig. 25.
4. If maximum number of cluster analysis $Max-Cluster-Count$ is performed, go to step 10.
Fig. 20. (a) Path of an L-shaped robot having six degrees of freedom (three translational and three rotational) in a 3-D environment (Generations = 100, PopSize = 30). Robot has to go through a narrow opening in the rectangular object. Object shown above is confined in a rectangular box which is not shown in the diagram for easier visualization of results. (b) Plot of the best current performance versus the generation count during the evolution process.

TABLE II
FULL EXECUTION TIMES OF THE EXPERIMENTS PERFORMED FOR A GIVEN POPULATION SIZE (PopSize) AND MAXIMUM NUMBER OF GENERATIONS (MaxGens)

<table>
<thead>
<tr>
<th>Timed Experiments</th>
<th>d.o.f</th>
<th>Total No. Faces</th>
<th>PopSize</th>
<th>MaxGens</th>
<th>Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Fig. 12)</td>
<td>2</td>
<td>184</td>
<td>20</td>
<td>100</td>
<td>141 (197)</td>
</tr>
<tr>
<td>2 (Fig. 13)</td>
<td>3</td>
<td>66</td>
<td>20</td>
<td>150</td>
<td>68 (90)</td>
</tr>
<tr>
<td>3 (Fig. 16)</td>
<td>3</td>
<td>38</td>
<td>50</td>
<td>50</td>
<td>68 (95)</td>
</tr>
<tr>
<td>4 (Fig. 17)</td>
<td>3</td>
<td>54</td>
<td>20</td>
<td>50</td>
<td>88 (116)</td>
</tr>
<tr>
<td>5 (Fig. 20)</td>
<td>6</td>
<td>36</td>
<td>30</td>
<td>100</td>
<td>130 (146)</td>
</tr>
<tr>
<td>6 (Fig. 15)</td>
<td>4</td>
<td>48</td>
<td>30</td>
<td>100</td>
<td>32 (90)</td>
</tr>
</tbody>
</table>

Times show the number of seconds it took to find the first feasible solution. Numbers in parenthesis show the full duration times of the experiments from initialization to termination (Sun Sparc 20 workstation with 64-MB RAM).

TABLE III
SCALING AND COMPUTATIONAL ISSUES IN RELATION TO ENVIRONMENT COMPLEXITY (NUMBER OF OBSTACLES)

<table>
<thead>
<tr>
<th>Timed Exp.</th>
<th>d.o.f</th>
<th>No. of Verts.</th>
<th>No. of Faces</th>
<th>PopSize</th>
<th>MaxGens</th>
<th>Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>6</td>
<td>278</td>
<td>208</td>
<td>30</td>
<td>100</td>
<td>42 (539)</td>
</tr>
<tr>
<td>S2</td>
<td>6</td>
<td>414</td>
<td>310</td>
<td>30</td>
<td>100</td>
<td>57 (769)</td>
</tr>
<tr>
<td>S3</td>
<td>6</td>
<td>542</td>
<td>406</td>
<td>30</td>
<td>100</td>
<td>69 (1157)</td>
</tr>
</tbody>
</table>

In these three experiments, we have systematically increased the number of obstacles in the environment to assess scaling and computational issues in environments involving a large number of obstacles. Additional obstacles were added in such a way so that the path complexity required for a feasible path would not change. The times show the number of seconds it took to find the first feasible solution. Numbers in parenthesis show the full duration times of the experiments from initialization to termination (Sun Sparc 20 workstation with 64-MB RAM).

8. Update cluster interval: \( \text{Cluster-Interval} = K \times \text{Cluster-Interval} \). We use \( K = 2 \).

9. Distribute chromosomes into \( \text{CurNumPop} \) populations (see Fig. 25). For populations having more than PopSize individuals, discard the worst chromosomes from the population. For populations having less than PopSize individuals, generate individuals that are perturbed around the cluster means. The maximum perturbation along each dimension is equal to the standard deviation of the corresponding variable.

10. If the termination condition is not reached, go back to step 3. The algorithm terminates when maximum number of generations is performed.

B. Multipath-Planning Simulation Results

Experiments were conducted for planning multiple alternative paths in 2-D workspaces involving both point and rigid robots with two degrees of freedom. A crossover probability of 0.65 and a mutation probability of 0.35 were used in all the experiments. The selection of the mutation operator at each generation is done based on assigned probabilities to these operators. Table I (column 2) of Section VII shows these assigned probabilities before a first feasible path is found. The assigned probabilities are adjusted after a first feasible path is found (shown in Table I, column 3 of Section VII). The same set of algorithm parameters were used throughout all the experiments with no arbitrary adjustment or selection of algorithm parameters. In some cases, the number of generations and the sharing sigma have been adjusted to accommodate different landscapes of the search space.

Various experiments were conducted using different planning environments with either a point or a polygonal robot. Table IV presents the set of parameters used to run two of these experiments. In Fig. 26, results for the PLE_2 planning environment
are shown. In this workspace, three different solution paths are expected. One of the paths would clearly have a shorter path length, which makes this example a good test case for validating the correctness of the multipath-planning algorithm. As shown in Fig. 26, the algorithm was able to successfully discover three species (left diagram) and obtain three solution paths (right diagram) for the path-planning problem. The bottom diagram of Fig. 26 shows species formation and convergence of each species. Three different species have evolved during the evolutionary process. An intermediate species has also evolved at generation 3 and went into extinction at generation 6. This species is not shown, since the performance values were relatively much higher than those shown in the diagram. Fig. 27 demonstrates the results for a more complex planning environment where two solution paths were discovered.

X. DISCUSSION AND CONCLUSION

This paper has described a new evolutionary algorithm for solving multidimensional path-planning problems for robotic applications. The evolutionary planning algorithm utilizes an iterative multiresolution path representation for encoding candidate paths. The hierarchical representation of a path reduces the expected search length for the path-planning problem. If a path is found at a low level of resolution, further expansion of that portion of the path search is not necessary. The encoded search space takes advantage of this by automatically adjusting the path length. The algorithm is flexible in that it can handle multidimensional path-planning problems, utilizes domain-specific knowledge for making decisions, and can accommodate different optimization criteria. During the planning process, the computation of the C-space is not required as the candidate paths are evaluated with respect to the workspace.

The implementation and experiments described in this paper have demonstrated that the evolutionary path-planning approach yields consistent results with computation times that are acceptable for practical applications. As shown in Table II, the computation times are most strongly related to the complexity of the required path and are not dominated by the dimensionality of the problem. The number of obstacles (or faces) in the environment certainly impacts the computation time for collision checking, but the arrangement and spacing determine the local path resolution required and, therefore, the number of path vertices.

These properties of the representation and search approach may be highly effective in many realistic applications where difficulty in planning is dominated by a small number of bottlenecks. For example, mobile robots often navigate in regions of relatively free space with a small number of local bottlenecks at passages (doors) or worksites. Also, in assembly path planning, tight constraints often occur in local regions, while simple
straight-line paths are very desirable in other regions. The techniques described in this paper could also be applied to other applications where a multiresolution spatial representation is appropriate, for example, in the optimization of structural designs or in graphical tessellation problems.

This paper has discussed experiments for multidimensional path planning of free-flying rigid and articulated robots and assembly trajectory problems. However, the evolutionary path-planning algorithm is flexible and general enough for path planning of fixed-based industrial manipulators such as Puma 560 as well. The only additional constraints posed by such a robot is due to joint limit problems and singularity concerns. These issues can easily be handled by the evolutionary path planner as follows.

1) Singularity Concerns: Although the evolutionary path planner performs no explicit C-space mappings, the path planner is C-space based. Thus, singularities are not a concern for this approach when planning paths for single robots since singularities are a task space phenomenon.

2) Joint Limit Problems: The evolutionary planner can easily and implicitly handle joint limit problems by considering configurations that violate joint limit constraints as infeasible and penalizing paths which have such infeasible configurations. The work envelope violations and obstacle collisions are already handled in this fashion. As the candidate paths evolve such poor candidates are eliminated and replaced by more fit candidates.

The evolutionary path planner, due to its path representation, may not be able to efficiently solve problems in a maze type of environment, which very rarely occur in practical applications. Due to scarcity of these problems in practical applications we have decided not to extend the evolutionary planner to solve such problems more efficiently. However, it is possible to generate subgoals (feasible configurations) for these problems and apply the evolutionary path planner to these subgoals. This type of approach will result in a more efficient handling of maze type problems.

Table IV

<table>
<thead>
<tr>
<th>Parameters Used for Evolutionary MultiPath Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLE.2</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Initial No Pops</td>
</tr>
<tr>
<td>Max No Pops</td>
</tr>
<tr>
<td>Crossover Rate</td>
</tr>
<tr>
<td>Mutation Rate</td>
</tr>
<tr>
<td>Population Size</td>
</tr>
<tr>
<td>Max Generations</td>
</tr>
<tr>
<td>Sharing Sigma</td>
</tr>
<tr>
<td>Sharing Alpha</td>
</tr>
</tbody>
</table>
Fig. 23. Complexity of the environment in Fig. 22 has been systematically increased to include more obstacles having a total number of 542 vertices and 406 faces. Obstacles were added in such a way so that the complexity of the required path does not change drastically. Experiment was performed using a free-flying polyhedral robot having six degrees of freedom (three translational and three rotational). Table III (Experiment S3) reports execution time of this experiment on a SUN Sparc 20 with 64-MB RAM.

Fig. 24. Simultaneous generation of multiple alternative solutions to the path-planning problem.

The results of these experiments are very encouraging and they indicate important contributions to the areas of evolutionary computation and path planning in robotics. In a related effort, the extension of these algorithms to include evolutionary speciation [59], where multiple alternative paths are generated simultaneously is discussed in this paper as well.

This paper has also presented a novel algorithm for solving multipath-planning problems for robotic applications. The multipath-planning algorithm utilizes minimal representation size cluster GA (MRSC_GA) with fitness sharing and the evolutionary path planner for generating multiple alternative solutions to the path-planning problem.
The simulation results demonstrated that this multipath-planning algorithm performs well on 2-D path-planning problems. The extension of the algorithm to higher dimensional path-planning problems is not difficult conceptually, but will involve significant extensions to the existing implementation of clustering algorithms and interpath distance metrics. These extensions are
a topic for continuing research. We have already demonstrated that the evolutionary path planner itself performs well for higher dimensional path-planning problems and in future research, we will address the extension of the multipath-planning algorithm to higher dimensional problems including the interpath distance metric for high dimensional paths and an extended clustering algorithm for higher dimensional feature vectors.

ACKNOWLEDGMENT

The authors would like to thank D. Fogel and the anonymous reviewers who made many useful suggestions. Computing resources of the Electronics Agile Manufacturing Research Institute (EAMRI) at Rensselaer and the State University of New York, Stony Brook, have also been used during this research in addition to software implementation utilized portions of the GAlib package by M. Wall.

REFERENCES


Cem Hocaöglu received the B.S., M.S., and Ph.D. degrees in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY.

He is currently a Consultant. Previously, he was an Assistant Professor of Electrical and Computer Engineering at the State University of New York, Stony Brook. From December 1994 to December 1998, he was a Research Scientist at the Electronics Agile Manufacturing Research Institute at Rensselaer Polytechnic Institute. His research interests include distributed object computing and evolutionary computation, and their applications to distributed manufacturing, path and assembly sequence planning, and autonomous air vehicles.

Dr. Hocaöglu was an Invited Member of the ECSE Honors and Research Program for Outstanding Undergraduates at Rensselaer Polytechnic Institute and the program committee for the 1999 Genetic and Evolutionary Computation Conference. He is a Member of Tau Beta Pi, Eta Kappa Nu, the New York Academy of Sciences, and the IEEE Robotics and Automation Society. He received the Henry J. Nolte Memorial Prize for outstanding academic achievement and promise for a successful career.

Arthur C. Sanderson (S’66–M’68–SM’86–F’91) received the B.S. degree from Brown University, Providence, RI, in 1968 and the M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, PA, in 1970 and 1972, respectively.

He has held faculty positions at Carnegie Mellon University, Pittsburgh, PA, from 1973 to 1987, where he was Co-Director of the Robotics Institute, and visiting positions at Delft University of Technology, Universidad Iberoamericana, Mexico City, Mexico, and Philips Laboratories, Briarcliff Manor, NY.

In 1987, he joined Rensselaer Polytechnic Institute as Professor and served as Department Head of the Electrical, Computer, and Systems Engineering Department from 1987 to 1994. He is currently a Professor of Electrical, Computer, and Systems Engineering and the Vice President for Research at Rensselaer Polytechnic Institute Troy, NY. During 1996, he was on sabbatical leave as a Visiting Professor in the Intelligent Robotics Laboratory, Institute for Information Sciences and Electronics, University of Tsukuba, Japan. From 1998 and 1999, he was on leave from Rensselaer Polytechnic Institute serving as the Director of the Division of Electrical and Communications Systems at the National Science Foundation. He has authored or coauthored over 250 publications and proceedings in the areas of biomedical signal processing, robotics and automation systems, sensor-based control, computer vision, and applications of knowledge-based systems. He recently coauthored the following books: Intelligent Task Planning using Fuzzy Petri Nets (Singapore: World Scientific, 1996), A Modular Approach to Reconfigurable Parallel Robotics, (Norwell, MA: Kluwer, 1998), and Multisensor Fusion: A Minimal Representation Framework (Singapore: World Scientific, 1999).

Dr. Sanderson is a Fellow of the AAAS. He was President of the IEEE Robotics and Automation Society from 1989 to 1990 and has been active in editorial activities and conference organization. He has recently been the corecipent of several awards for innovation in undergraduate engineering education.