International Journal of Innovative
Computing, Information and Control
Volume 8, Number 3(A), March 2012
 ISSN 1349-4198
pp. 1773–1786

HIERARCHICAL PATH PLANNING FOR MANIPULATORS BASED ON GENETIC ALGORITHM WITH NON-RANDOM INITIAL POPULATION

Chien-Chou Lin

Department of Computer Science and Information Engineering
National Yunlin University of Science and Technology
No. 123, Section 3, University Road, Douliou, Yunlin 64002, Taiwan
linchien@yuntech.edu.tw

Received October 2010; revised April 2011

Abstract. *A hierarchical genetic algorithm (GA) is proposed for the path planning of manipulators. The proposed algorithm consists of a global path planner (GPP) and a local motion planner (LMP). The GPP, which uses a MAKLINK-based approach, plans a trajectory for a robot end-effector from the starting free-space to the goal free-space. A GA with a non-random initial population is adopted to plan the manipulator configurations along the path given by the GPP. Once the optimal configuration has been obtained, the optimal chromosomes are reserved as the initial population for the next intermediate goal. Since the initial population is non-random, the evolution is more efficient and the planned path is smoother than those of the conventional GA. Simulation results show that the proposed algorithm works well, specifically in terms of path smoothing and computation efficiency. The proposed algorithm also significantly reduces the number of variations of joint angles of the manipulator.*
Keywords: Genetic algorithm, Path planning, Motion planning, Collision-avoidance, Hyper-redundant manipulator

1. **Introduction.** For robotic manipulators, the goal of path planning is to determine a collision-free trajectory from the original location and orientation (called the starting configuration) to the goal configuration [1], as shown in Figure 1. In recent years, many path planning algorithms have been proposed [1-32]. In general, the methods can be categorized into two basic types, namely the configuration space (c-space)-based approach [3-8] and geometric-based algorithms [9-23]. The c-space-based approach considers both the manipulator and the obstacles at the same time by identifying manipulator configurations that intersect the obstacles. A point of a c-space indicates that configuration of a manipulator. A configuration is usually encoded by a set of manipulator parameters, i.e., the angles of the links of manipulators. The forbidden regions in the c-space are points which imply manipulator configurations that intersect the obstacles. Thus, path planning is reduced to the problem of planning a path from the start point to the goal in free space [3]. The c-space algorithm is easy to implement and fast for manipulators with few degrees of freedom (DOF). The algorithm can also deal with manipulators with many DOF. Roadmap methods are the most popular type of c-space-based approach [4-6]. Many of the methods based on c-space can be applied to real robot systems. However, the overhead of preprocessing and updating the c-space significantly degrades the performance. In [5], the probabilistic roadmap planner (PRM), which does not need the entire c-space to be constructed, proceeds in two phases, namely the learning phase and the query phase. In the learning phase, the undirected graph is preprocessed to improve its connectivity. In the query phase, the method first attempts to find a path from the start

and goal configurations to two nodes of the roadmap. A graph search is then conducted to find a sequence of edges that connects these nodes in the roadmap. In practice, PRM has successfully solved many complex problems. The approach is also easy to implement and fast. The main drawback of PRM is that it is a probabilistically complete algorithm. In other words, the PRM algorithm will terminate under a time constraint even if the solution exists and still has not been found.

Unlike c-space-based approaches, geometric algorithms directly use spatial occupancy information of the workspace to solve path planning problems. Workspace-based algorithms usually extract relevant information about the free space and use it together with the robot geometry to find a path. In addition to collision avoidance, some approaches try to find paths with the minimum risk of collision. To minimize such a risk, repulsive potential fields between robots and obstacles were used in [17-20] to match their shapes in the path planning. In [17,18], heuristic search algorithms for robots in potential-based workspaces were proposed. The algorithms are more efficient than c-space approaches, but they are not complete. Furthermore, the potential algorithms may stop at local minima and fail to plan a complete path even if the path is available.

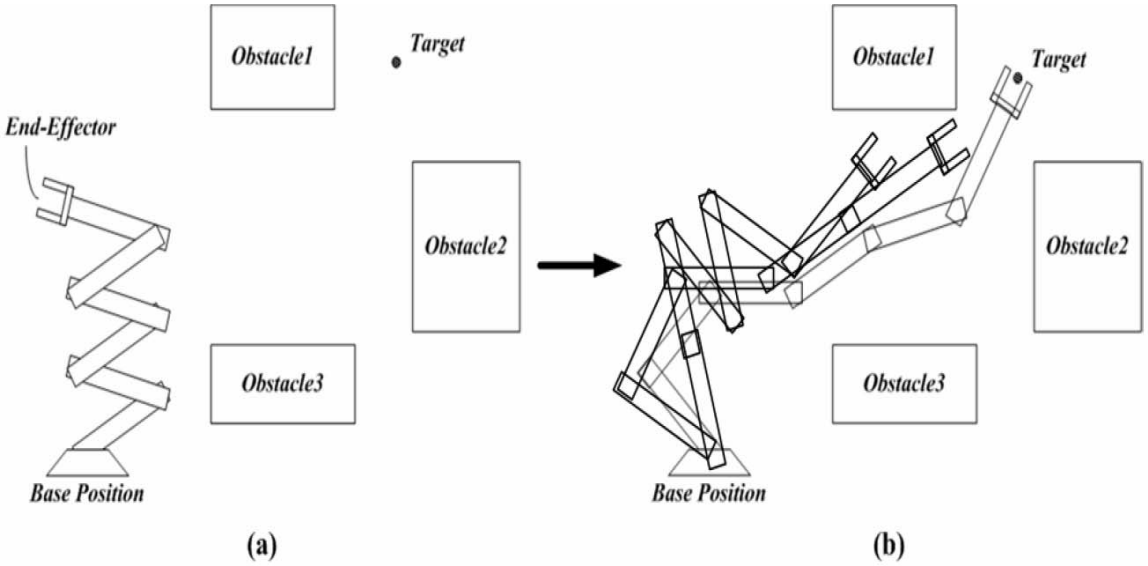

FIGURE 1. Path planning of a five-link manipulator: (a) the initial configuration and (b) a 3-configuration path to the goal

Recently, some genetic algorithm (GA)-based methods [24-33] have been proposed for robot path planning. In [24], the trajectories are derived using a pure GA. Therefore, some configurations of the trajectories are close to the obstacles. In [32], the immune GA was found to be more efficient than simple GAs for manipulator path planning. However, obstacles are not considered in the algorithm. In [33], an individual chromosome was used to describe the whole set of arm configurations of a trajectory. The evaluation function with multiple criteria can plan a series of smoothing configurations from the start point to the target point, but the search space is fairly large due to the large number of genes of a chromosome. In general, GA-based path planning algorithms are complete. In other words, they can find the path if it exists. However, GA-based approaches are slow because they use evolution and have no heuristic method. In order to obtain a safe path efficiently, the present study proposes a hierarchical GA for the path planning of manipulators. The proposed algorithm includes a global path planner (GPP) and a local motion planner (LMP). The rest of this paper is organized as follows. The GPP, which plans a trajectory

for a robot end-effector from the starting free-space to the goal free-space by adopting the MAKLINK approach, is introduced in Section 2. The GA with a non-random initial population adopted for planning the manipulator configurations along the path given by the GPP is described in Section 3. In Section 4, simulation results are presented for the path planning of manipulators in various 2-D workspaces. Section 5 summarizes this work.

2. **Global Path Planning.** Because global planning and local planning are coupled, many path planning algorithms require complex searching and suffer from the local minimum problem. The proposed hierarchical algorithm consists of the GPP and the LMP. The former determines the primary movement direction of manipulators and the latter is a motion planner that derives a manipulator configuration with the given position of its end-effector. Details of the two planning algorithms are given in the next section.

The GPP can be considered as a planner for point robots. In the present study, *MAKLINK* [2], which is based on the free link concept, is adopted since it is easy to implement and has been shown to work well in many cases. The key idea of *MAKLINK* is the division of the free workspace into several free spaces with free links. The free space is structured as a free convex area using the developed free link approach, as shown in Figure 2. In a convex area, any two points can be connected by a straight line. Thus, any point is reachable directly.

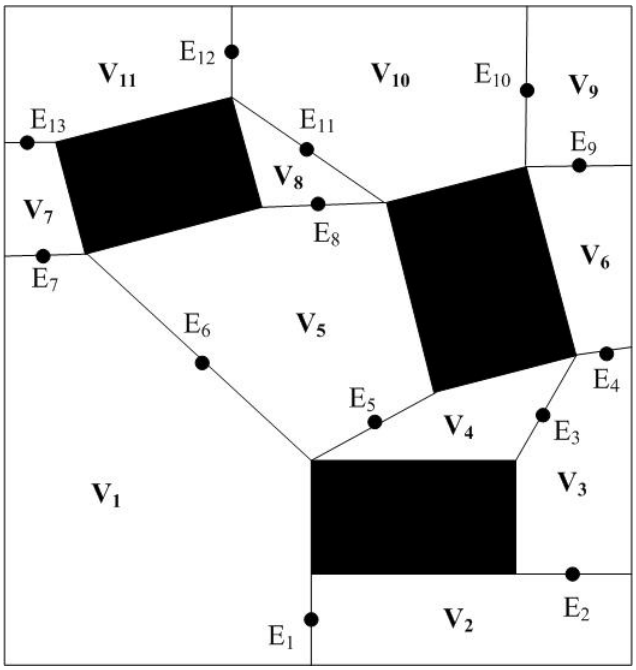

FIGURE 2. Free-link division of a free workspace into several free spaces

First, we connect all vertices of an obstacle to other obstacles and workspace boundaries, except if a connection intersects an edge of the obstacles. The free links of obstacle will be selected from the connections later. Second, the lines at a corner of an obstacle are sorted according to their length, from the shortest to the longest. Third, the outside angles of a corner are checked. The line is the best free link if the two angles are less than or equal to 180 degrees. Other links to the corner are ignored. Notably, if one of the two angles is more than 180 degrees, the link is a free link and is added to the list of free links. For such a corner, other links are needed to divide an outside angle which is more than 180

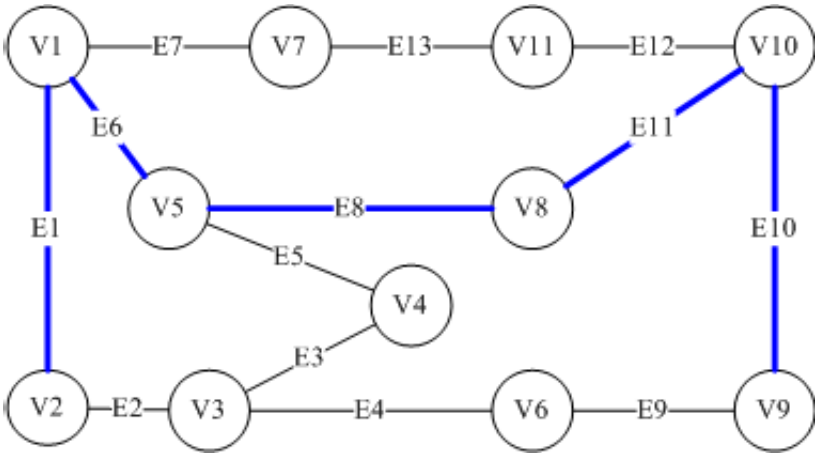

FIGURE 3. Graph representation of Figure 2, where free spaces are represented as vertices (V_i) of graph G and free links are represented as edges (E_i) of G

degrees. Therefore, the shortest links are selected sequentially from the free link list to divide the outside angles to be less than 180 degrees.

The selection of free links and the best free link is repeated for all obstacles. In Figure 2, the free workspace is divided into several free spaces, namely V_1, V_2, and V_{11}. The segmented workspace of Figure 2 can be represented as the graph, $G = \{V_i, E_k\}$, shown in Figure 3. Free spaces are represented as vertices (V_i) of G and free links are represented as edges (E_i) of G. The global path from the start (V_2) to the goal (V_{10}) is obtained using a simple search algorithm. The global path (E_1, E_6, E_8, E_{11}) is the blue path in Figure 3 and the dashed-line path in Figure 4.

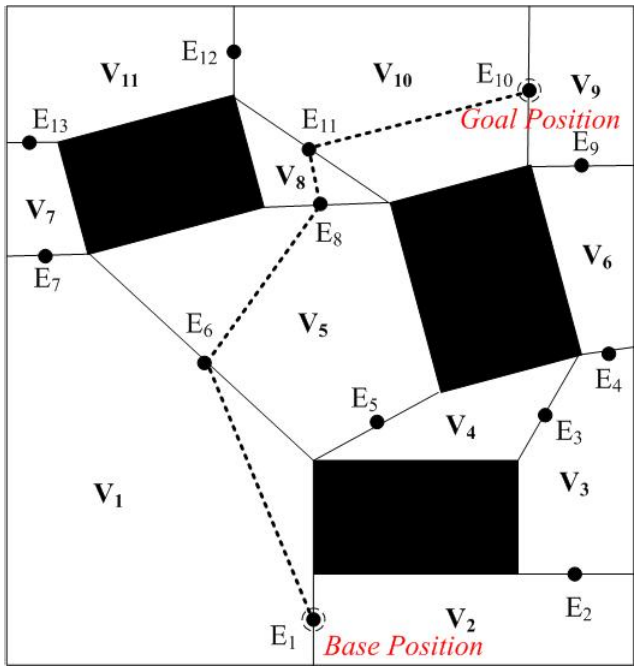

FIGURE 4. Planned global path of Figure 3

3. **Local Motion Planning.** The GPP generates a global path (E_1, E_6, E_8, E_{11}) which the end-effector of the manipulator should follow. In order to obtain fine trajectories,

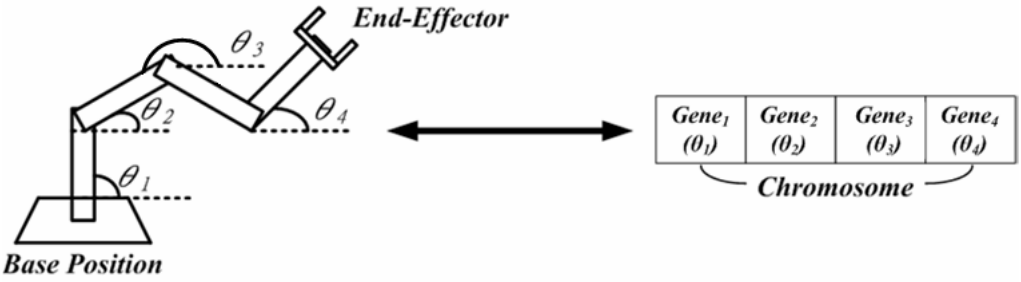

FIGURE 5. Configuration of a manipulator represented as a chromosome with the genes representing the joint angles

more middle points are interpolated within a free space. Then, a finer path is obtained as $(q_1, q_2, q_3, \ldots, q_n)$. For each position q_i, the GA-based LMP plans a configuration for the manipulator with the end-effector fixed at point q_i. Thus, the q_i can be represented as an intermediate goal, IG_i.

The GA is used for finding optimal solutions in the whole solution space; it thus avoids becoming trapped in local minima. To apply a GA search for the optimal configuration for manipulators, the joint angles of the manipulator configuration are encoded into one chromosome. A configuration of a k-link manipulator is denoted it terms of its joint angles $(\theta_1, \theta_2, \ldots, \theta_k)$. Each joint angle is represented as a gene of a chromosome. For each chromosome, the Cartesian coordinates of the end-effector can be determined. The chromosome is then constructed as shown in Figure 5. The proposed GA is shown below:

Algorithm Non-Random-Initial-GA

```
Begin
    i = 1; /* Initialize the first intermediate goal */
    t = 0; /* Initialize the genetic generations */
    Randomly Generate an initial population P_i(t);
    fitness(P_i(t));
    repeat until (reach final goal q_n) Do
        P_{i+1}(t) = P_i(t);
        repeat until (reach intermediate goal q_i) Do
            select P_i(t + l) from P_i(t);
            crossover(P_i(t + l));
            mutate (P_i(t + l));
            fitness(P_i(t + l));
            t = t + 1;
        end
        i = i + 1;
    end
End
```

The basic procedure of the proposed algorithm can be simplified as five steps:

i. Step 0: Initialization of the population for IG_i. If $i = 0$, the initial population is generated randomly; otherwise, the initial population is the offspring of IG_{i-1}.
ii. Step 1: Selection. The population is sorted by fitness and the top 50% of chromosomes are preserved for crossover and mutation.
iii. Step 2: Crossover. The crossover operation generates new offspring from two parent chromosomes.

iv. Step 3: Mutation. The mutation operation locally adjusts genes with different probabilities to generate new offspring.
v. Step 4: Evaluation of the population. If the optimal solution is found, set $i = i + 1$ and go to Step 0 to begin a new evolution for the next intermediate goal IG_{i+1}; otherwise, go to Step 1 to begin the next generation.

The implementation details of the five steps are given below.

3.1. **Step 0: Initialization of the population.** The population, $P_1(0)$, of the first intermediate goal is generated randomly, and the initial populations, $(P_i(0), i > 1)$, of other intermediate goals are obtained from the last generation of the preceding intermediate goal; that is, they are randomly generated. Since these initial populations are eugenic and inherit from the ancestor, the evolution time is reduced and the obtained path of the manipulator is smoother.

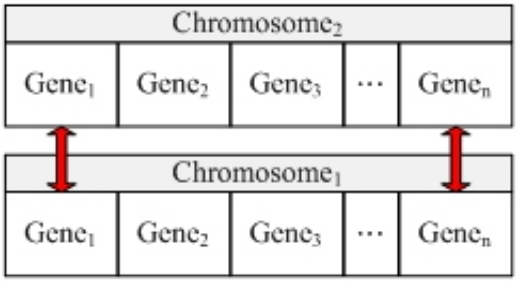

FIGURE 6. Crossover operator defined as the exchange of similarly positioned genes of a pair of chromosomes

3.2. **Step 1: Selection.** Selection in a genetic algorithm preserves the optimal chromosomes and abandons suboptimal chromosomes. Generally, selection is performed according to the fitness of every chromosome, where the fitness evaluation is an objective function for the chromosomes. Generally, there are several types of selection, including roulette, tournament, best, random, and top percent. Here, the top percent scheme is adopted. In this step, the top 50% of chromosomes in the population are preserved for Step 2 and Step 3 to generate new offspring.

3.3. **Step 2: Crossover.** The reproduction operators, crossover and mutation, generate new offsprings for the next generation. Crossover is performed between two selected chromosomes, called parents, by exchanging parts of their genomes to form two new chromosomes, called offspring. The most popular types of crossover operations are one-point, two-point, uniform, and blending. Here, since the i-th gene of a chromosome represents the i-th link of a manipulator, the crossover operator is defined as the exchange of similarly positioned genes of a pair of chromosomes, as shown in Figure 6.

3.4. **Step 3: Mutation.** For the mutation operator, an arbitrary bit in a genetic sequence is changed with a given probability. Mutation maintains genetic diversity from one generation to the next while attempting to avoid local minima. However, for manipulator motion planning, a small angle change of the base link of a manipulator will cause a large manipulator movement further away from the base. Here, the probability of mutation for every link is thus different. The probability of mutation increases from the base link to the distal link.

3.5. **Step 4: Evaluation criteria.** In this paper, an optimal chromosome (a configuration of a manipulator) should produce a path that is collision-free and that reaches the goal. Therefore, the fitness function can be defined as:

$$F = f_{collision} \cdot D_s \tag{1}$$

where F is the cost function and $f_{collision}$ is the collision coefficient, which is equal to 1 if the motion is collision-free and equal to $V_{\max}$ (punishment cost) otherwise. The cost of a collision-free configuration (D_s) is defined as (2), which is the distance between the end-effector and intermediate goals.

$$D_s = \sqrt{(x_g - x_{ep})^2 + (y_g - y_{ep})^2} \tag{2}$$

where (x_g, y_g) are the coordinates of the intermediate goal and (x_{ep}, y_{ep}) are the coordinates of the end-effector. (x_{ep}, y_{ep}) can be derived as:

$$\begin{aligned}\begin{bmatrix} x_{ep} \\ y_{ep} \\ 1 \end{bmatrix} = &\begin{bmatrix} \cos\theta_k & -\sin\theta_k & 0 \\ \sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_k \\ 0 \\ 1 \end{bmatrix} + \cdots + \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_2 \\ 0 \\ 1 \end{bmatrix} \\ &+ \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_1 \\ 0 \\ 1 \end{bmatrix}\end{aligned} \tag{3}$$

where l_i is the length of link i. If the lengths of all links are equal to l, (3) can be simplified as:

$$\begin{aligned}\begin{bmatrix} x_{ep} \\ y_{ep} \\ 1 \end{bmatrix} = &\left(\begin{bmatrix} \cos\theta_k & -\sin\theta_k & 0 \\ \sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} + \cdots + \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right. \\ &\left. + \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} l \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{k} \cos\theta_i & -\sum_{i=1}^{k} \sin\theta_i & 0 \\ \sum_{i=1}^{k} \sin\theta_i & \sum_{i=1}^{k} \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l \\ 0 \\ 1 \end{bmatrix}\end{aligned} \tag{4}$$

Thus:

$$\begin{aligned} x_{ep} &= \sum_{i=1}^{k} (\cos\theta_i \times l_i) = \sum_{i=1}^{k} \cos\theta_i \times l \\ y_{ep} &= \sum_{i=1}^{k} (\sin\theta_i \times l_i) = \sum_{i=1}^{k} \sin\theta_i \times l \end{aligned} \tag{5}$$

4. **Experimental Results.** In this section, simulation results are presented for path planning implemented using the open-source Java genetic algorithms and genetic programming package (JGAP) API (http://jgap.sourceforge.net). It provides basic genetic mechanisms that can be easily used to apply evolutionary principles to problem solutions.

The first example is a 6-link manipulator in a workspace with three obstacles. As shown in Figure 7, a smooth trajectory is obtained. Figures 7(a)-7(c) show partial trajectories of the 6-link manipulator. Figure 7(d) shows the whole planned trajectory. The population size was 100 and the maximum number of generations was 600. 10% of the initial population was non-random.

Figure 8 and Figure 9 show the average variations of each link between any two contiguous configurations of the trajectories of Figures 7(b) and 7(c), respectively. The dashed line was obtained by the conventional GA and the solid line was obtained by the proposed GA. The variation of the trajectory obtained by the proposed algorithm is smaller than

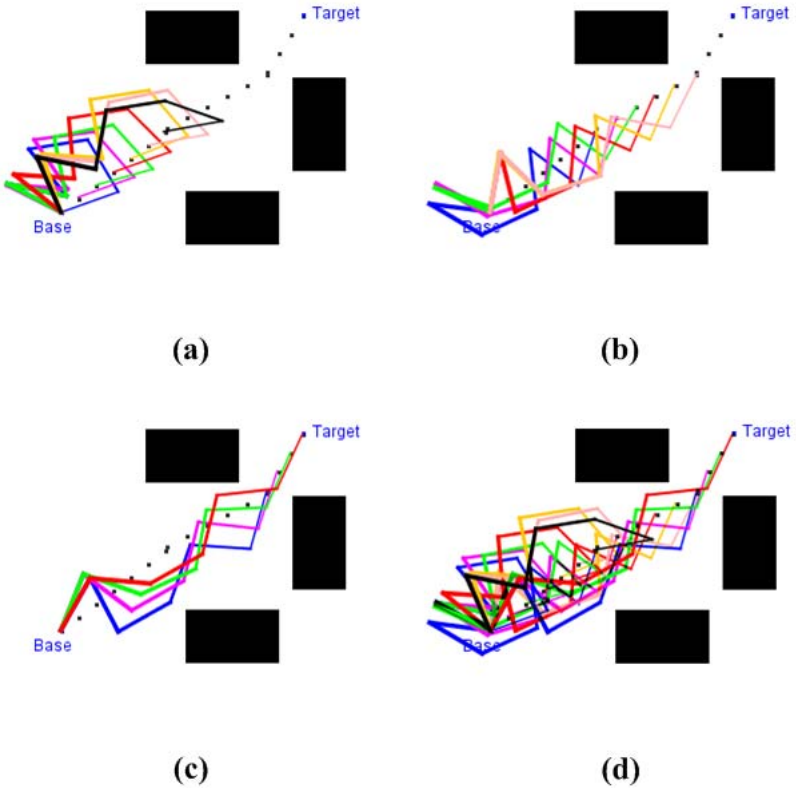

FIGURE 7. Path planning example of a 6-link manipulator

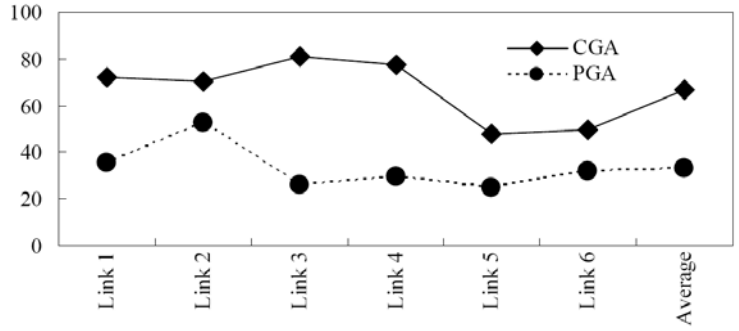

FIGURE 8. Average variations of each link between two contiguous configurations of the trajectories obtained by the conventional GA and the proposed GA of Figure 7(b), the average on the x-axis is the average variation of the six links

that of the trajectory obtained by the conventional GA. In other words, the trajectory obtained by the proposed GA is smoother and more efficient.

Another example is a 6-link manipulator in a workspace with two obstacles. The partial trajectories of this example are shown in Figures 10(a)-10(f). The whole trajectory obtained by the conventional GA is shown in Figure 11. Since all populations of the conventional GA are randomly generated, some unfeasible configurations are derived (e.g., the three configurations that go over the larger obstacle). The trajectory obtained by the proposed GA is smoother than that obtained by the conventional GA. Figure 12 shows the average variations of each link between two contiguous configurations of the trajectories of Figure 10(g) and Figure 11, respectively. The variation of the trajectory obtained by

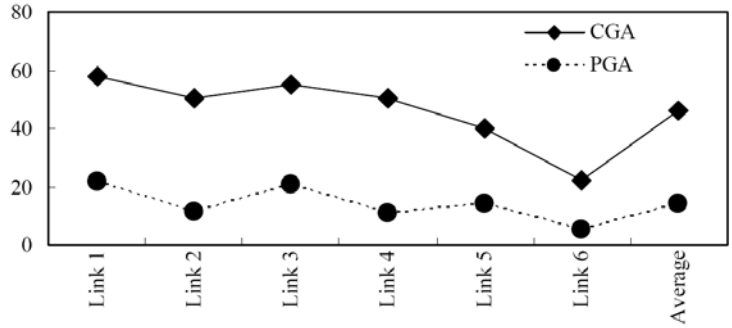

FIGURE 9. Average variations of each link between any two contiguous configurations of the trajectories obtained by the conventional GA and the proposed GA of Figure 7(c), the average on the x-axis is the average variation of the six links

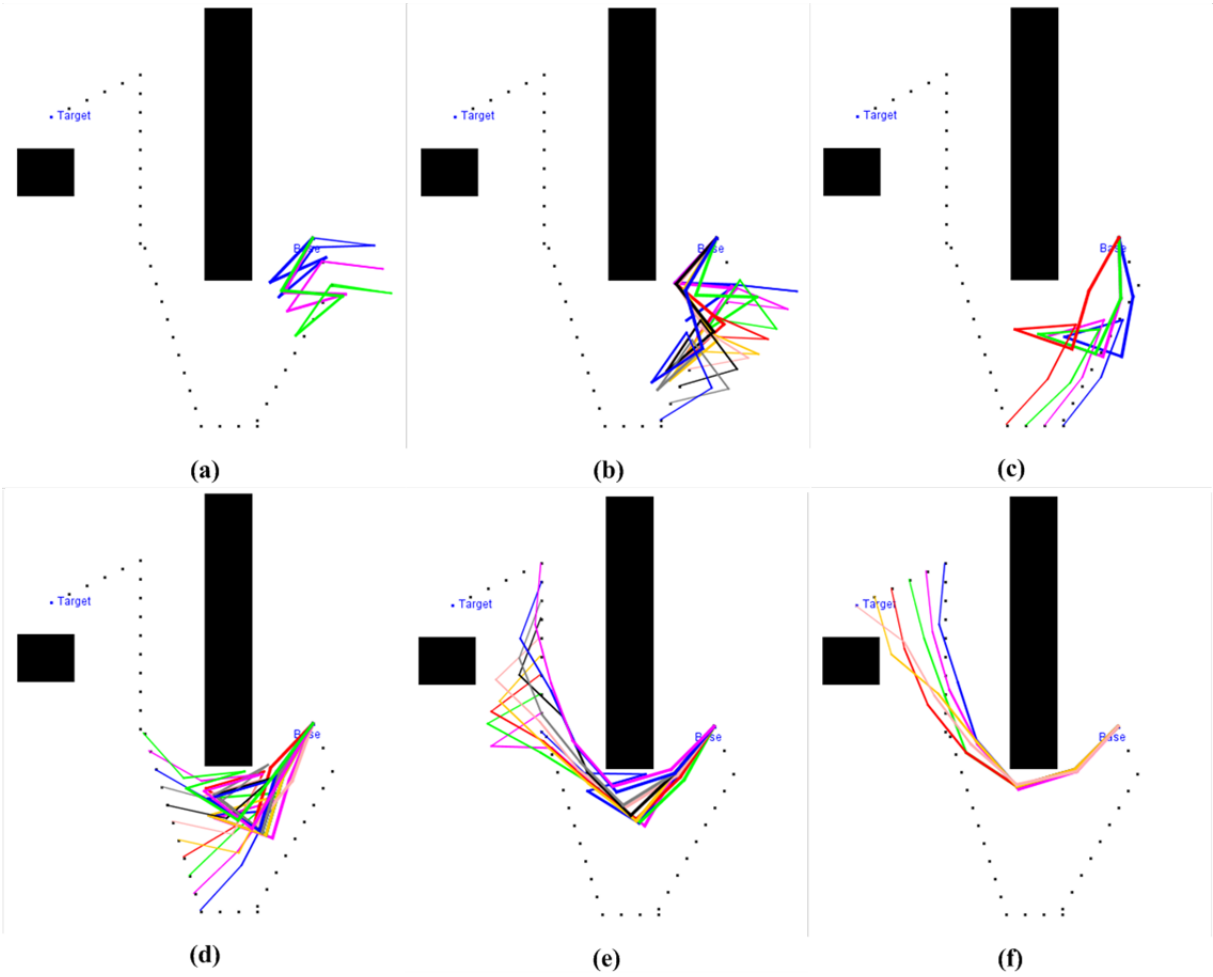

FIGURE 10. (a)-(f) Sequence of partial paths of a 6-link manipulator

the proposed algorithm is smaller than that of the trajectory obtained by the conventional GA.

The numbers of generations required to obtain optimal configurations for intermediate goals (IGs) are shown in Figure 13. In most simulations, the optimal configurations of some intermediate goals (e.g., IG24, IG25, and IG26) obtained by the conventional GA are far from the intermediate goals. These optimal configurations are also infeasible

FIGURE 11. Path obtained by the conventional GA for the workspace in Figure 10

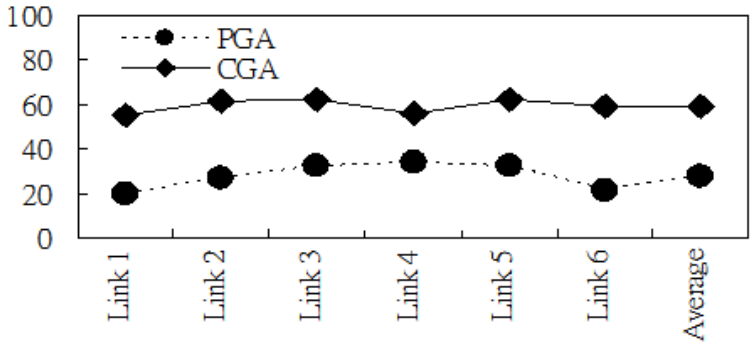

FIGURE 12. Average variations of each link between two contiguous configurations of Figure 10 and Figure 11, the average on the x-axis is the average variation of the six links

configurations which go over the obstacle, as shown in Figure 11. In Figure 13, the numbers of unsuccessful configurations of the conventional GA are not considered. As shown in Figure 13, the proposed algorithm requires fewer generations to reach the optimal configuration.

In the proposed algorithm, the initial generation is eugenically generated by the population of the preceding IG. The approach is different from the conventional GA, whose initial population is randomly generated. Figure 14 shows the convergence for populations of various sizes. The convergence greatly depends on the size of the population, but not on the mutation probabilities and crossover probabilities. An example of a crowded environment is shown in Figure 15. In this simulation, the manipulator passes through a

narrow passage to the goal. The proposed algorithm works well in terms of smoothness and efficiency.

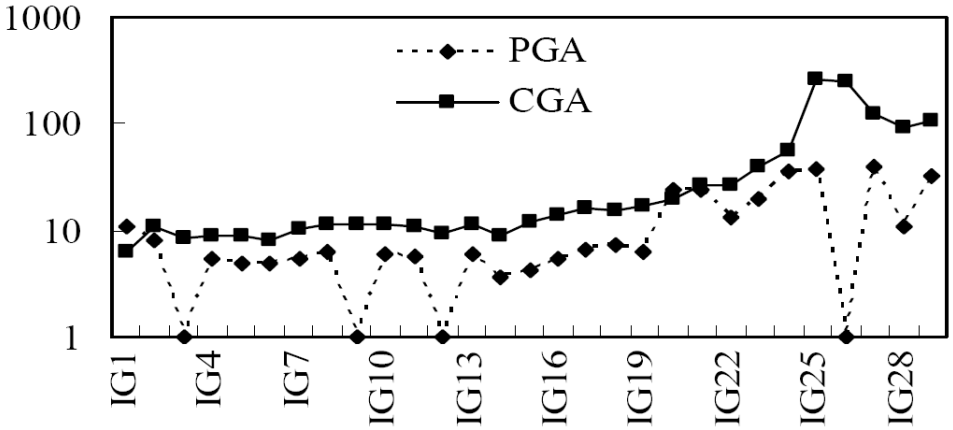

FIGURE 13. Numbers of generations required to obtain optimal configurations for intermediate goals (IGs)

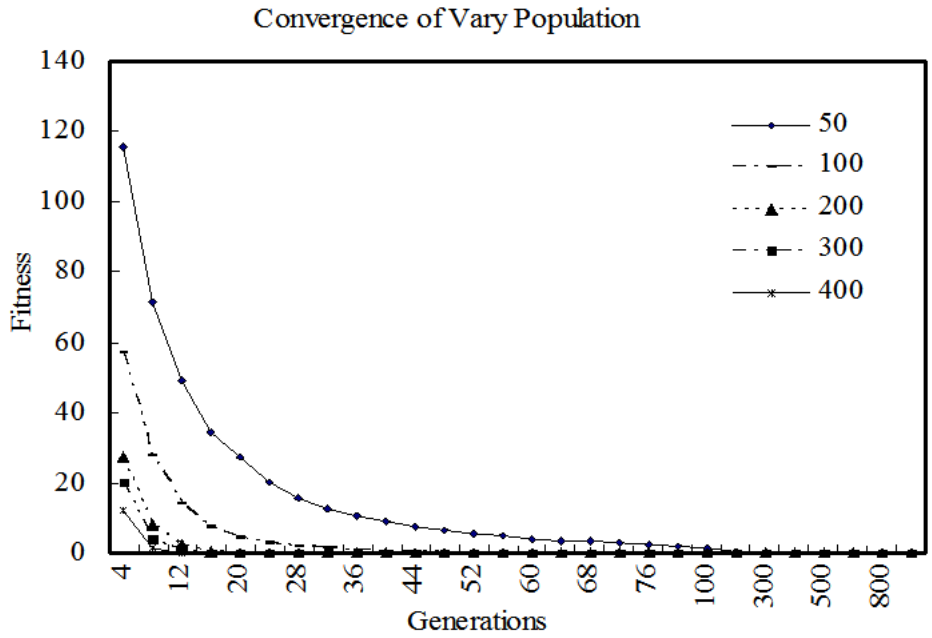

FIGURE 14. Convergence populations of various sizes

Figure 16 shows the path planning results for an example similar to that shown in [33]. In our simulation, a chromosome consists of 3 genes, whereas a long encoded chromosome with 120 genes is used in [33] for a 40-configuration trajectory from the given starting point to the goal. The search space, which is proportional to the exponential function of the number of genes in a chromosome, of the approach in [33] is thus fairly large. Furthermore, if a finer trajectory with more configurations is needed, the algorithm proposed in [33] must extend its chromosomes, further extending the time required to obtain results. In contrast, the proposed algorithm is scalable because it can plan additional configurations without changing the chromosome coding.

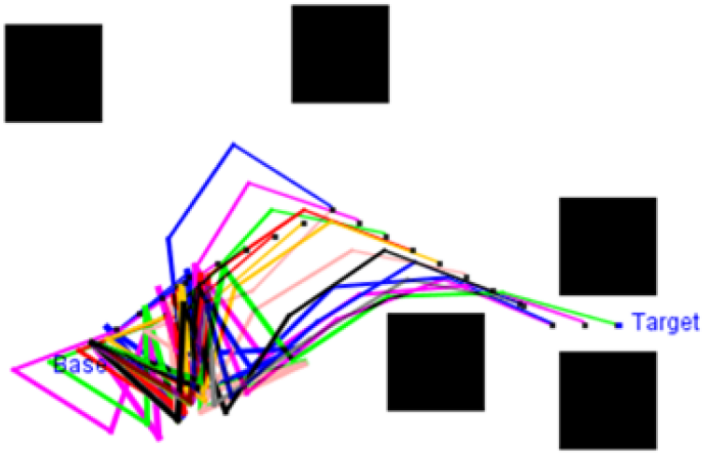

FIGURE 15. Path planning example of a 6-link manipulator within a narrow passage

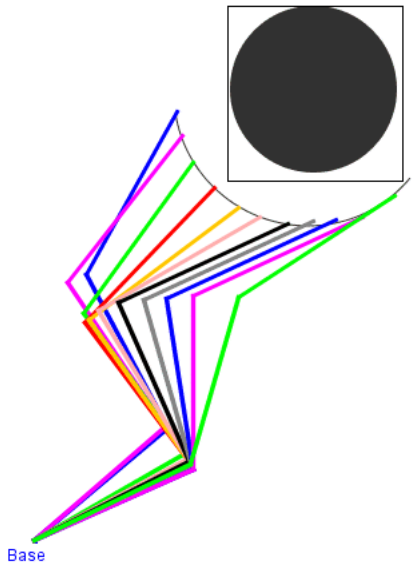

FIGURE 16. Path planning results for example similar to that shown in [33]

5. **Conclusion.** A hierarchical GA path planner for hyper-redundant manipulators was proposed. Unlike most planning algorithms for manipulators that consider path planning and motion planning together, the proposed algorithm consists of a global path planner (GPP) and a local motion planner (LMP). The GPP plans a trajectory for the end-effector and the LMP plans manipulator configurations along the path. Since the proposed algorithm processes path planning and motion planning individually, the complexity of the proposed GA is reduced. Furthermore, since the initial population of the GA is non-random, the continuity of the manipulator's configurations is maintained and the obtained trajectories are smooth.

Simulation results show that the proposed algorithm is able to plan a smooth trajectory to the goal while reducing evolution time. The smooth path is very efficient since it reduces the variations of the joint angles of the manipulator by 50%. In addition, the infeasible configurations which may be obtained by the conventional GA are avoided. The proposed algorithm has better convergence than the conventional GA, making it

more computationally efficient. In most cases, the conventional GA requires more time to plan an optimal configuration.

Since the major difference between 2D manipulator and 3D manipulator is the type of joints (which are encoded as genes in our algorithm), the proposed algorithm can be extended to 3D workspaces without significant changes. In future works, the fitness function of robot configurations will be modified for 3D coordinate systems.

Acknowledgment. This work was supported by the National Science Council of Taiwan under grants NSC 95-2221-E-366-010 and NSC 97-2221-E-366-004.

REFERENCES

[1] J. H. Reif, Complexity of the mover's problem and generalizations, *Proc. of Foundations of Computer Science*, pp.421-427, 1979.

[2] M. K. Habib and H. Asama, Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach, *Proc. of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, vol.2, pp.563-567, 1991.

[3] T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Communications of the ACM*, vol.22, pp.560-570, 1979.

[4] T. Lozano-Perez, Spatial planning: A configuration space approach, *IEEE Transactions on Computers*, vol.C-32, pp.108-120, 1983.

[5] L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, Probabilistic roadmap for path planning in high-dimensional configuration spaces, *IEEE Transactions on Robotics & Automation*, vol.12, no.4, pp.566-580, 1996.

[6] T. Lozano-Perez, A simple motion-planning algorithm for general robot manipulator, *IEEE Journal of Robotics and Automation*, vol.3, no.3, pp.224-238, 1987.

[7] Y. D. Dai, M. Konishi and J. Imai, RNN-based cooperative motion control of 2-DOF robot arms, *International Journal of Innovative Computing, Information and Control*, vol.3, no.4, pp.937-952, 2007.

[8] J. J. Kuffner and S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, *Proc. of IEEE International Conference on Robotics and Automation*, pp.995-1001, 2000.

[9] B. Li, X. J. Yang, J. G. Zhao and P. Yan, Minimum time trajectory generation for a novel robotic manipulator, *International Journal of Innovative Computing, Information and Control*, vol.5, no.2, pp.369-378, 2009.

[10] E. Ralli and G. Hirzinger, Fast path planning for robot manipulators using numerical potential fields in the configuration space, *Proc. of the IEEE/RSJ/GI Intl. Conf. on Intelligent Robots and Systems, Advanced Robotic System and the Real World*, vol.3, pp.1922-1929, 1994.

[11] J. Barraquand and J. C. Lotombe, Robot motion planning: A distributed representation approach, *International Journal of Robotics Research*, vol.10, no.6, pp.628-649, 1991.

[12] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *International Journal of Robotics Research*, vol.5, no.1, pp.90-98, 1986.

[13] J. C. Latombe, Motion planning: A journey of robots, molecules, digital actors, and other artifacts, *International Journal of Robotics Research*, vol.18, no.11, pp.1119-1128, 1999.

[14] J. Barraquand, B. Langois and J. C. Latombe, Numerical potential field techniques for robot path planning, *IEEE Transactions on Robotics and Automation*, vol.22, pp.224-241, 1992.

[15] L. H. Jiang, M. C. Deng and A. Inoue, Obstacle avoidance and motion control of a two wheeled mobile robot using SVR technique, *International Journal of Innovative Computing, Information and Control*, vol.5, no.2, pp.253-262, 2009.

[16] T. Henmi, T. Ohta, M. Deng and A. Inoue, Tracking control of a two-link planar manipulator using nonlinear model predictive control, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.2977-2984, 2010.

[17] C.-C. Lin, C.-C. Pan and J.-H. Chuang, A novel potential-based path planning of 3-D articulated robots with moving bases, *Robotica*, vol.22, no.4, pp.359-367, 2004.

[18] J.-H. Chuang, C.-C. Lin and L.-W. Kuo, Potential-based path planning for robot manipulators, *Journal of Robotic Systems*, vol.22, no.6, pp.313-322, 2005.

[19] C.-C. Lin and J.-H. Chuang, A potential-based path planning algorithm for hyper-redundant manipulators, *Journal of the Chinese Institute of Engineers*, vol.33, no.3, pp.415-427, 2010.

[20] J. H. Chuang, Potential-based modeling of three-dimensional workspace for obstacle avoidance, *IEEE Transactions on Robotics and Automation*, vol.14, no.5, pp.778-785, 1998.
[21] V. Kroumov, J. L. Yu and K. Shibayama, 3D path planning for mobile robots using simulated annealing neural network, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.2885-2899, 2010.
[22] R. Brooks and T. Lozano-Perez, A subdivision algorithm in configuration space for findpath with rotation, *IEEE Transcations on System, Man and Cybernetics*, vol.15, no.2, pp.224-233, 1985.
[23] O. Aron and T. Lozano-Perez, Visible decomposition: Real-time path planning in large planar environments, *AI Memo 1638*, 1996.
[24] E. J. Solterio Pires and J. A. Tenrerio Machado, A trajectory planner for manipulators using genetic algorithm, *Proc. of the IEEE International Symposium on Assembly and Task Planning*, pp.163-168, 1999.
[25] J. K. Parker, A. R. Khoogar and D. E. Goldberg, Inverse kinematics of redundant robots using genetic algorithms, *Proc. of IEEE International Conference on Robotics and Automation*, pp.271-276, 1989.
[26] A. R. Khoogar and J. K. Parker, Obstacle avoidance of redundant manipulators using genetic algorithms, *Proc. of IEEE International Conference on Robotic and Automation*, pp.317-320, 1991.
[27] H. Chen, X. Du and W. Gu, Global path planning based on neural network and genetic algorithm in a static environment, *Artificial Neural Networks*, pp.34-42, 2004.
[28] M. Gill and A. Zomaya, A parallel collision-avoidance algorithm for robot manipulators, *IEEE Concurrency*, vol.6, no.1, pp.68-78, 1998.
[29] M. Srinivas and L. M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, vol.24, no.4, pp.656-667, 1994.
[30] A. C. Nearchou, Path planning of a mobile robot using genetic heuristics, *Robotica*, vol.16, no.5, pp.575-588, 1998.
[31] S. Mittal and K. Deb, Three-dimensional offline path planning for UAVs using multi-objective evolutionary algorithms, *IEEE Congress on Evolutionary Computation*, pp.3195-3202, 2007.
[32] X. Luo and W. Wei, A new immune genetic algorithm and its application in redundant manipulator path planning, *Journal of Robotic Systems*, vol.21, no.3, pp.141-151, 2004.
[33] L. Tian and C. Collins, Motion planning for redundant manipulators using a floating point genetic algorithm, *Journal of Intelligent and Robotic Systems*, vol.38, no.3-4, pp.297-312, 2003.