'91 KACC 1991. 10. 22~24

PATH COORDINATOR BY THE MODIFIED GENETIC ALGORITHM

C. H. Chung and K. S. Lee
Dept. of Control and Instrumentation Eng.
Kwangwoon University

Abstract: Path planning is an important task for optimal motion of a robot in structured or unstructured environment. The goal of this paper is to plan the shortest collision-free path in 3D, when a robot is navigated to pick up some tools or to repair some parts from various locations. To accomplish the goal of this paper, the *Path Coordinator* is proposed to have the capabilities of an *obstacle avoidance strategy* [3] and a *traveling salesman problem strategy* (TSP) [23]. The *obstacle avoidance strategy* is to plan the shortest collision-free path between each pair of n locations in 2D or in 3D. The *TSP strategy* is to compute a minimal system cost of a tour that is defined as a closed path navigating each location exactly once. The *TSP strategy* can be implemented by the *Neural Network*. The *obstacle avoidance strategy* in 2D can be implemented by the *VGraph Algorithm*. However, the *VGraph Algorithm* is not useful in 3D, because it can't compute the global optimality in 3D. Thus, the *Path Coordinator* is proposed to solve this problem, having the capabilities of selecting the optimal edges by the *modified Genetic Algorithm* [21] and computing the optimal nodes along the optimal edges by the *Recursive Compensation Algorithm* [5].

1 Introduction

In many path planning algorithms, attempts are made to optimize the path between the start and the goal in terms of a Euclidean distance. The goal of this paper is to plan the shortest collision-free path in 3D, when a robot is navigated to pick up some tools or to repair some parts from various locations. In this paper, a *Planning Coordinator* is proposed to do this goal. To develop theories on the *Planning Coordinator* [4], the following assumptions are made:

- The obstacles are stationary polyhedral objects.
- A navigating robot is a polyhedral object.
- A system's cost function to be optimized is a Euclidean distance.

A navigating robot is shrunk to a configuration point in the *Configuration Space* [2], while the stationary obstacles are expanded to fill all space whenever the presence of a configuration point would imply a collision of the navigating robot with obstacles. Therefore, the path planning algorithm [17] can be formulated as a graph searching problem. The graph is formed by connecting all pairs of visible vertices. This is the well known *VGraph Algorithm* for calculating the shortest collision-free path.

To accomplish this goal, a *Planning Coordinator* must calculate the shortest collision-free path between each pair of the various locations. The *VGraph Algorithm* is useful since it can compute the global optimal path between each pair of the n locations in 2D.

Because the *VGraph Algorithm* cannot compute the global optimal path, the *VGraph Algorithm* cannot be directly applied for the previous subtasks. The *VGraph Algorithm* is not effective in view of the fact that the *VGraph Algorithm* cannot compute the global optimality in 3D. Instead, a robust algorithm is needed to solve the optimal path between each pair of the n locations in 3D. Since the optimal path in 3D may involve going through points on the optimal edges, a robust algorithm should have the capability of solving the following two problems:

- The first problem is how to select the optimal edges.
- The second problem is how to find the optimal nodes on the optimal edges.

The global optimality of the *Recursive Compensation Algorithm* is proved for 3D convex objects and the *Planning Coordinator* is implemented by using the *Recursive Compensation Algorithm*. While the modified Genetic Algorithm finds the optimal edges, the *Recursive Compensation Algorithm* calculates the optimal vertices along the optimal edges. The *Recursive Compensation Algorithm* can be applied to compute the cost function for the modified Genetic Algorithm.

The Genetic Algorithm (GA) may be used to find the optimal edges. However, the Genetic Algorithm does not always guarantee global optimality, since the Genetic Algorithm may converge to the local minima [9]. Therefore, a robust algorithm is needed to guarantee the global optimality. The modified Genetic Algorithm (MGA) used to find the optimal edges in the *Planning Coordinator* has been proved to converge in probability to the global minimum of the cost function [21].

Finally, the *Neural Network* is applied to a case of a mapping process to solve the TSP with the 3D obstacles. Most of the TSP problems have been solved in 2D. However, a robot could visit some locations to pick up some tools or repair some parts in deep sea, in space or on the earth. To solve the TSP with the 3D obstacles, a mapping mechanism is needed to compute the shortest path between two locations. If a location is visible to another location, then a straight line between the two locations is the shortest distance. If a location is not visible to another location, then obstacle-avoidance strategy is needed to calculate the shortest distance between the two locations.

2 Modified Genetic Algorithm

[Problem Statement: where no information on the optimal edges is known.]
Solve the TSP of 10 locations with the polyhedral obstacles, assuming that $A = B = 50$, $C = 20$, $D = 50$, $u_0 = 0.02$, $n = 15$, and $\tau = 1$ [23]. Calculate the shortest path from the L_i node to the L_j node, assuming that the obstacles are polyhedrons and no optimal edges are known. Suppose that the polyhedral, L_i node and L_j node are represented by the following vertices; $P_1(8,1,-3)$, $P_2(2,1,-3)$, $P_3(2,3,-3)$, $P_4(8,3,-3)$, $P_5(8,1,5)$, $P_6(2,1,5)$, $P_7(2,3,5)$, $P_8(8,3,5)$, $L_i(3,0.5,0.5)$, $L_j(3.5,4,3)$.
(Solution) Most of the TSP problems have been solved in 2D. However, a robot could visit some locations to pick up some tools or repair some parts in deep sea, in space or on the earth. To solve the TSP with the 3D obstacles, a mapping mechanism is needed to compute the shortest path between two locations. If a location is visible to another location, then the straight line between two locations is the shortest distance. If a location is not visible to another location, then obstacle-avoidance strategy is needed to calculate the shortest distance between two locations. Suppose that the polyhedral and Locations are represented by the following vertices; $P_1(8,1,-3)$, $P_2(2,1,-3)$, $P_3(2,3,-3)$, $P_4(8,3,-3)$, $P_5(8,1,5)$, $P_6(2,1,5)$, $P_7(2,3,5)$, $P_8(8,3,5)$, $L_i(3,0.5,0.5)$, $L_j(3.5,4,3)$. The *modified Genetic Algorithm* [21] updates the information on the optimal edges at each generation, according to the fitness of the cost function. Since the *modified Genetic Algorithm* reproduces its population for each generation, the *modified Genetic Algorithm* needs a fast function in order to evaluate the fitness. So the Recursive Compensation Algorithm is applied to compute the fitness function. I set the following parameters; size of population = 20, length of chromosome = 12, maximum # of generation = 10, probability of crossover = 0.7, probability of mutation = 0.02, fitness function = $D_{fitness}/D_{RCA} \times 100\%$, where $D_{fitness} = 0.5519 \times 10$ and D_{RCA} = distance computed by the Recursive Compensation Algorithm according to information on the chromosome. To construct the chromosome, I set e_i, $i = 1, 2, \cdots, 12$ as the edges of a polyhedral; $e_1 = \overline{P_1P_2}$, $e_2 = \overline{P_2P_3}$, $e_3 = \overline{P_3P_4}$, $e_4 = \overline{P_1P_4}$, $e_5 = \overline{P_1P_5}$, $e_6 = \overline{P_2P_6}$, $e_7 = \overline{P_3P_7}$, $e_8 = \overline{P_4P_8}$, $e_9 = \overline{P_5P_6}$, $e_{10} = \overline{P_6P_7}$, $e_{11} = \overline{P_2P_7}$, $e_{12} = \overline{P_5P_8}$.
The Tables show that the optimal edges are e_6 and e_7. The optimal vertices are N_1 along e_6 and N_2 along e_7 and the shortest path between L_i and L_j is 0.5519453549×10;

$$N_1 = (2.0, 1.0, 1.0682)$$
$$N_2 = (2.0, 3.0, 2.0842).$$

Table 1: Generation 0 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(0, 0)	0	111100000000	1.599E+01	35 %
2	(0, 0)	0	000001100010	1.078E+01	51 %
3	(0, 0)	0	100010011000	1.939E+01	28 %
4	(0, 0)	0	100010010000	1.337E+01	41 %
5	(0, 0)	0	000010010000	1.190E+01	46 %
6	(0, 0)	0	010100000000	8.778E+00	63 %
7	(0, 0)	0	000010010000	1.190E+01	46 %
8	(0, 0)	0	000010011000	1.505E+01	37 %
9	(0, 0)	0	000000001101	1.503E+01	37 %
10	(0, 0)	0	000000000101	1.163E+01	47 %
11	(0, 0)	0	001001100010	1.461E+01	38 %
12	(0, 0)	0	010100000000	8.778E+00	63 %
13	(0, 0)	0	000000001111	1.757E+01	31 %
14	(0, 0)	0	001001100010	1.461E+01	38 %
15	(0, 0)	0	110100000000	1.325E+01	42 %
16	(0, 0)	0	000001100010	1.078E+01	51 %
17	(0, 0)	0	000010010000	1.190E+01	46 %
18	(0, 0)	0	010100000000	8.778E+00	63 %
19	(0, 0)	0	100010011000	1.939E+01	28 %
20	(0, 0)	0	000010011000	1.505E+01	37 %

Table 2: Generation 2 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(7,18)	9	000001100010	1.078E+01	51 %
2	(7,18)	9	000010011000	1.505E+01	37 %
3	(4,11)	12	010100000000	8.778E+00	63 %
4	(4,11)	12	001001100010	1.461E+01	38 %
5	(17, 6)	8	001001100010	1.461E+01	38 %
6	(17, 6)	8	000001100010	1.078E+01	51 %
7	(10,17)	1	000001100010	1.078E+01	51 %
8	(1, 1)	12	010100000000	8.778E+00	63 %
9	(14,14)	12	010100000000	8.778E+00	63 %
10	(14,14)	12	010100000000	8.778E+00	63 %
11	(17, 5)	1	010100000000	8.778E+00	63 %
12	(17, 5)	1	000001100010	1.078E+01	51 %
13	(11, 8)	10	001001100010	1.461E+01	38 %
14	(11, 8)	10	000010010000	1.190E+01	46 %
15	(17,13)	9	000001100000	5.519E+00	100 %
16	(17,13)	9	000010010000	1.190E+01	46 %
17	(16,12)	12	000010010000	1.190E+01	46 %
18	(16,12)	12	010100000000	8.778E+00	63 %
19	(7, 4)	12	000001100010	1.078E+01	51 %
20	(7, 4)	12	010100000000	8.778E+00	63 %

The *optimal population ratio* (opr) is defined by

$$\frac{\textit{number of optimal population in a generation}}{\textit{population size in a generation}} \times 100\%,$$

Figure 1 shows that the modified Genetic Algorithm converges to the global optimality, while the Genetic Algorithm converges to a local minima. Figure 2 shows the initial state in the Neural TSP and Figure 3 shows the final state in the Neural TSP.

Table 3: Generation 4 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(11,20)	12	010100000000	8.778E+00	63 %
2	(11,20)	12	000001100010	1.078E+01	51 %
3	(4, 8)	12	010100000000	8.778E+00	63 %
4	(4, 8)	12	000001100000	5.519E+00	100 %
5	(8, 1)	12	000001100000	5.519E+00	100 %
6	(8, 1)	12	010100000000	8.778E+00	63 %
7	(5, 3)	10	010100000000	8.778E+00	63 %
8	(5, 3)	10	010100000000	8.778E+00	63 %
9	(8,18)	5	010100000000	8.778E+00	63 %
10	(8,18)	5	000001100000	5.519E+00	100 %
11	(8, 8)	11	000001100000	5.519E+00	100 %
12	(20,20)	12	000001100010	1.078E+01	51 %
13	(18,11)	9	010100000000	8.778E+00	63 %
14	(18,11)	9	010100000000	8.778E+00	63 %
15	(8,17)	6	001001100000	8.167E+00	68 %
16	(8,17)	6	000001100010	1.078E+01	51 %
17	(4, 8)	2	000001100000	5.519E+00	100 %
18	(4, 8)	2	010100000000	8.778E+00	63 %
19	(12, 3)	11	000001100000	5.519E+00	100 %
20	(12, 3)	11	010100000000	8.778E+00	63 %

Table 5: Generation 8 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(18, 1)	12	000001100000	5.519E+00	100 %
2	(18, 1)	12	000001100000	5.519E+00	100 %
3	(7, 5)	2	000001100000	5.519E+00	100 %
4	(7, 5)	2	000001100000	5.519E+00	100 %
5	(2,10)	9	000001100000	5.519E+00	100 %
6	(2,10)	9	000001100000	5.519E+00	100 %
7	(1, 1)	1	000001100000	5.519E+00	100 %
8	(10, 7)	12	000001100000	5.519E+00	100 %
9	(8,15)	3	010100000000	8.778E+00	63 %
10	(8,15)	3	010100000000	8.778E+00	63 %
11	(10, 6)	12	000001100000	5.519E+00	100 %
12	(10, 6)	12	000001100000	5.519E+00	100 %
13	(12,11)	7	000001100000	5.519E+00	100 %
14	(12,11)	7	000001100000	5.519E+00	100 %
15	(20,11)	12	000001100000	5.519E+00	100 %
16	(20,11)	12	000001100000	5.519E+00	100 %
17	(2, 6)	11	000001100000	5.519E+00	100 %
18	(2, 6)	11	000001100000	5.519E+00	100 %
19	(11, 5)	1	000001100000	5.519E+00	100 %
20	(11, 5)	1	000001100000	5.519E+00	100 %

Table 4: Generation 6 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(12,17)	12	000001100000	5.519E+00	100 %
2	(12,17)	12	010100000000	8.778E+00	63 %
3	(16, 1)	2	000001100000	5.519E+00	100 %
4	(16, 1)	2	010100000000	8.778E+00	63 %
5	(8, 6)	4	010100000000	8.778E+00	63 %
6	(8, 6)	4	010100000000	8.778E+00	63 %
7	(7,18)	12	000001100000	5.519E+00	100 %
8	(7,18)	12	010100000000	8.778E+00	63 %
9	(19, 3)	12	000001100000	5.519E+00	100 %
10	(19, 3)	12	000001100000	5.519E+00	100 %
11	(3, 9)	12	000001100000	5.519E+00	100 %
12	(3, 9)	12	000001100000	5.519E+00	100 %
13	(1, 1)	12	000001100000	5.519E+00	100 %
14	(13,11)	1	000001100010	1.078E+01	51 %
15	(1, 7)	10	000001100000	5.519E+00	100 %
16	(1, 7)	10	000001100000	5.519E+00	100 %
17	(12,19)	5	000001100000	5.519E+00	100 %
18	(12,19)	5	000001100000	5.519E+00	100 %
19	(7, 2)	12	000001100000	5.519E+00	100 %
20	(7, 2)	12	000001100000	5.519E+00	100 %

Table 6: Generation 10 by the Modified Genetic Algorithm

#	parents	loc.	edges	distance	o.p.r.
1	(10, 9)	1	000001100000	5.519E+00	100 %
2	(10, 9)	1	000001100000	5.519E+00	100 %
3	(7,15)	12	000001100000	5.519E+00	100 %
4	(7,15)	12	000001100000	5.519E+00	100 %
5	(14,11)	12	000001100000	5.519E+00	100 %
6	(14,11)	12	000001100000	5.519E+00	100 %
7	(7,16)	7	000001100000	5.519E+00	100 %
8	(7,16)	7	000001100000	5.519E+00	100 %
9	(14,15)	12	000001100000	5.519E+00	100 %
10	(14,15)	12	000001100000	5.519E+00	100 %
11	(11, 1)	10	000001100000	5.519E+00	100 %
12	(11, 1)	10	000001100000	5.519E+00	100 %
13	(14,19)	12	000001100000	5.519E+00	100 %
14	(14,19)	12	000001100000	5.519E+00	100 %
15	(14, 8)	12	000001100000	5.519E+00	100 %
16	(14, 8)	12	000001100000	5.519E+00	100 %
17	(11, 8)	12	000001100000	5.519E+00	100 %
18	(1, 1)	12	000001100000	5.519E+00	100 %
19	(13, 4)	12	000001100000	5.519E+00	100 %
20	(13, 4)	12	000001100000	5.519E+00	100 %

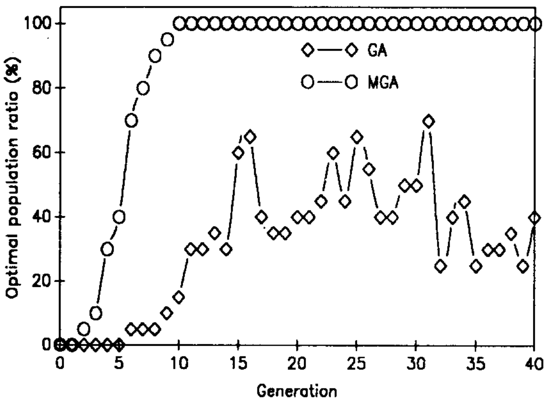

Figure 1: A Comparison between MGA and GA

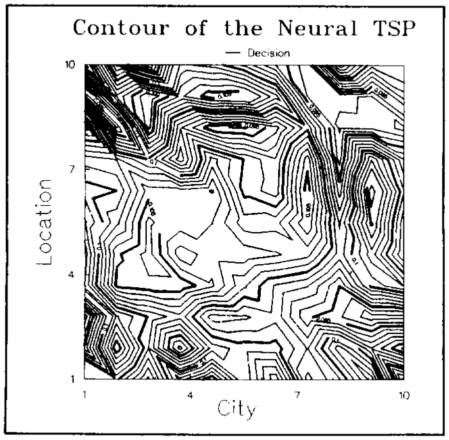

Figure 2: Initial State in the Neural TSP

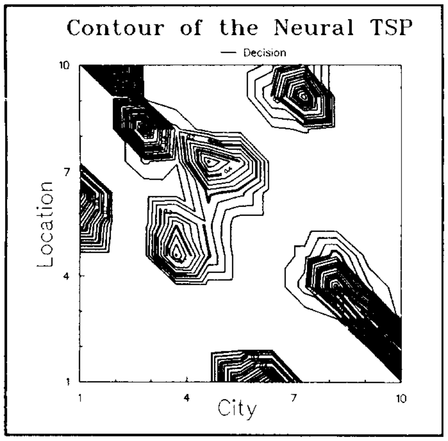

Figure 3: Final State in the Neural TSP

3 Conclusions

First, the *Recursive Compensation Algorithm* was applied to a case where information on the optimal edges is available and no global search is necessary. Comparing the efficiency of the algorithm when the accuracy, ϵ, is set to 10^{-5}, the *Branch and Bound Method* needs 0.230×10^{7} milliseconds, while the *Recursive Compensation Algorithm* needs only 0.416×10^{3} milliseconds on a VAX-11/750. A *Euclidean* distance obtained by the *Recursive Compensation Algorithm* is 48% less than that obtained by the *VGraph Algorithm*. Since the accuracy is defined by ϵ whose value is set to be very small and n is the number of polyhedral obstacles, Lozano-Pérez's Modified VGraph Algorithm needs a lot of memory space to store $(2+8\times n\times\epsilon^{-1})$ vertices for the *VGraph*, while the *Recursive Compensation Algorithm* needs a small memory space to store $(2 + 8 \times n)$ vertices for the *VGraph*. Simplifying the *VGraph*, the *Recursive Compensation Algorithm* can save not only the memory space to represent the *VGraph* but also the graph-searching time.

Second, the *Neural Network* is applied to a case of a mapping process to solve the TSP with the 3D obstacles.

Finally, the *Recursive Compensation Algorithm* was also applied to a case where no information on the optimal edges is available. However, the *Recursive Compensation Algorithm* requires information on the optimal edges to get the optimal solution. The exhaustive search algorithm guarantees the optimal edges for the shortest path. It may be applied to find the globally optimal edges for the shortest path in the workspace that has less than 4 obstacles. However, it is not an adequate algorithm to find the optimal edges for the shortest path in the workspace that has many obstacles, since it needs the exponential search time. For $(4 \sim 10)$ obstacles, the Modified VGraph algorithm is suggested. This algorithm generates some additional vertices along the edges of the grown obstacles so that no edge is longer than a prespecified maximum length. However, it is not easy to decide how many vertices should be added along the edges of the grown obstacles and the additional vertices need much more computation time. If there are a lot of obstacles in a workspace, this algorithm is not adequate, because it needs a lot of memory space to represent the additional vertices which will result in more computation time. The modified Genetic Algorithm is useful to find the optimal edges for the shortest path in the workspace that has many obstacles. Figure 1 shows that the modified Genetic Algorithm converges to the global optimality, while the Genetic Algorithm converges to a local minima. While the modified Genetic Algorithm finds the optimal edges, the *Recursive Compensation Algorithm* calculates the optimal vertices along the optimal edges. The *Recursive Compensation Algorithm* works to compute the cost function for the modified Genetic Algorithm.

4 Future Work

There are many interesting problems deserving further research for CIM environment. Below some of them are listed in no particular order.

- Solve the collision-free path planning for the dynamic obstacles.
- Exchange the knowledge on the path planning with other coordinators for an intelligent robot.
- Consider the general polyhedral objects for the collision free shortest path.
- Consider not only a Euclidean distance but also time or energy as a system cost function.

References

[1] Aarts, E. and Korst, J., "Simulated Annealing and Boltzmann Machines," *John Wiley* & Sons, New York, 1989.

[2] Brooks, Rodney A., "Planning Collision-Free Motions for Pick-and-Place Operations," *The International Journal of Robotics Research*, Vol. 2, No. 4, Winter 1983, pp. 19-44.

[3] Chung, C. H. and Saridis, G. N., "An Obstacle Avoidance Motion Organizer for an Intelligent Robot," *RAL-TR*-88-117, Robotics and Automation Laboratory, Rensselaer Polytechnic Institute, Troy, New York, 12180-3590.

[4] Chung, C. H. and Saridis, G. N , "Obstacle Avoidance Path Planning by the Extended VGraph Algorithm," *CIRSSE-TR*-89-12, Center for Intelligent Robotic Systems for Space Exploration, NASA, Jan. 1989.

[5] Chung, C. H. and Saridis, G. N., "The Recursive Compensation Algorithm for Obstacle Avoidance Path Planning," *IEEE International Workshops on Intelligent Robots and Systems,* Tsukuba, Japan, Sep. 1989.

[6] Chung, C. H. and Saridis, G. N., "Path Planning for an Intelligent Robot by the Extended Vgraph Algorithm," *IEEE International Symposium on Intelligent Control,* Albany, NY, Sep. 1989.

[7] Chung, C. H. and Saridis, G. N., "Recursive Compensation Algorithm for Collision Free Path Planning," *IEEE Transactions on Systems, Man and Cybernetics* (submitted).

[8] Cuykendall R. and Reese, R., "Scaling the Neural TSP Algorithm," *Biological Cybernetics*, Vol. 60, pp. 365-371, 1989.

[9] Goldberg, David E., "Genetic Algorithms in Search, Optimization, and Machine Learning," *Addison-Wesley Publishing Company, INC.*, Reading, Massachusetts, 1989.

[10] Hart, P. and Nilsson, N. J. and Raphael, B. A., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybernetics*, Vol. SSC-4, No. 2, July 1980, pp. 100-107.

[11] Hinton, G. E. and Sejnowski, T. J., "Learning and Relearning in Boltzmann Machines," pp. 282-317, in D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1:Foundations*, MIT Press, 1986.

[12] Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, Vol. 79, pp. 2554-2558, April 1982.

[13] Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. USA*, Vol. 81, pp. 3088-3092, May 1984.

[14] Hopfield, J. J., "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52, pp. 141-152, 1985.

[15] Hopfield J. J. and Tank, D. W., "Computing with Neural Circuits: A Model," *Science*, Vol. 233, pp. 625-633, August 1986.

[16] Lawler, E.L and Lenstra, J. K. and Rinnooy Kan, A. H. G. and Shmoys, D. B., "The Traveling Salesman Problem," *John Wiley & Sons*, New York, 1986.

[17] Lozano-Pérez Tomás, "Automatic Planning of Manipulator Transfer Movements," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-11, No. 10, October 1981, pp. 681-698.

[18] Luenberger, David G., "Linear and Nonlinear Programming," *Addison Wesley Publishing Company*, Reading, MA, 1984.

[19] Maudlin, M. L., "Maintaining Diversity In Genetic Algorithm," *Proc. National Conference on Artificial Intelligence,* 1984, pp. 247-250.

[20] Moed, M. C. and Saridis, G. N., "A Boltzmann Machine for the Organization of Intelligent Machines", *Proceedings of 2 Telerobotics Conference*, Pasadena, CA, Jan. 1989.

[21] Moed, M. C., "The Organizer: Planning Tasks With An Emergent Connectionist/ Symbolic Systems," *CIRSSE-TR*-89-42, Center for Intelligent Robotic Systems for Space Exploration, NASA, Sep. 1989.

[22] Saridis, G. N., "Expanding Subinterval Random Search For System Identification And Control," *IEEE Transactions on Automatic Control,* 1979, pp. 405-412.

[23] Wilson, G. V. and Pawley, G. S., "On the stability of the travelling salesman problem algorithm of Hopfield and Tank," *Biological Cybernetics*, Vol. 58, pp. 63-70, 1988.