'92 KACC 1992. 10. 19~21

RECURSIVE COMPENSATION ALGORITHM APPLICATION TO THE OPTIMAL EDGE SELECTION.

C. H. Chung and K. S. Lee

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 optimal 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, the *Path Coordinator* is proposed to have the capabilities of an *obstacle avoidance strategy* and a *traveling salesman problem strategy* (TSP). 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 *Hopfield 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 used to solve this problem, having the capabilities of selecting the optimal edges by the *modified Genetic Algorithm* and computing the optimal nodes along the optimal edges by the *Recursive Compensation Algorithm*.

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* [1], 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 [3] 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 in 3D, 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 [4] 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* [5] 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 [6].

The Genetic Algorithm (GA) [7] 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 [7]. 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 [6].

Finally, the *Neural Network* is applied to a case of a mapping process to solve the TSP with the 3D obstacles [8]. 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 The Recursive Compensation Algorithm

One can get the compensated node, N_i^*. Because one can reconstruct the configuration space obstacles in 3D by the *Orthogonal Projection Theorem*, it is sufficient that one can prove the convergence in three orthogonal projection subspaces in order to prove convergence in three-dimensional configuration space.

The optimal distance d_k can be computed by the *Recursive Compensation Algorithm* in one of the orthogonal projection subspace. when the following vertices are known.

$$N_{i-1}(k) = P\{\omega_{i-1}(k), \varphi_{i-1}(k)\} \quad (1)$$
$$N_i(k-1) = P\{\omega_i(k-1), \varphi_i(k-1)\} \quad (2)$$
$$N_i(k) = P\{\omega_i(k-1), \varphi_i(k-1) - d_k\} \quad (3)$$
$$N_{i+1}(k-1) = P\{\omega_{i+1}(k-1), \varphi_{i+1}(k-1)\} \quad (4)$$

where $d_k > 0$.

The distance function through $N_i(k)$ between $N_{i-1}(k)$ and $N_{i+1}(k-1)$ can be given by

$$D(d_k) = \overline{N_{i-1}(k)N_i(k)} + \overline{N_i(k)N_{i+1}(k-1)} \quad (5)$$

Taking the derivative of $D(d_k)$ in order to get the minima, one can obtain

$$\frac{\partial D(d_k)}{\partial d_k} = 0. \quad (6)$$

Define

$$\Omega_1 = \{\omega_i(k-1) - \omega_{i-1}(k)\}^2 \quad (7)$$
$$\Omega_2 = \varphi_{i+1}(k-1) - \varphi_i(k-1) \quad (8)$$
$$\Omega_3 = \{\omega_{i+1}(k-1) - \omega_i(k-1)\}^2 \quad (9)$$
$$\Omega_4 = \varphi_i(k-1) - \varphi_{i-1}(k) \quad (10)$$
$$\sigma_1 = \Omega_1 - \Omega_3 \quad (11)$$
$$\sigma_2 = \Omega_1\Omega_2 + \Omega_3\Omega_4 \quad (12)$$
$$\sigma_3 = \Omega_1\Omega_2^2 - \Omega_3\Omega_4^2. \quad (13)$$

Since the k^{th} compensated node, $\varphi_i(k)$, can be computed by the subtraction of the previous compensated node and the k^{th} compensated distance, $\varphi_i(k)$ can be computed by

$$\begin{aligned}\varphi_i(k) &= \varphi_i(k-1) - d_k \quad (14)\\ &= \frac{1}{\sigma_1}\varphi_i(k-1)\{\sigma_1 - \Omega_1 + \Omega_3\} + \\ &\quad \frac{1}{\sigma_1}\varphi_{i+1}(k-1)[\Omega_1 - \{\omega_i(k-1) - \omega_{i-1}(k)\}\\ &\quad \{\omega_{i+1}(k-1) - \omega_i(k-1)\}] - \\ &\quad \frac{1}{\sigma_1}\varphi_{i-1}(k)[\Omega_3 - \{\omega_i(k-1) - \omega_{i-1}(k)\}\\ &\quad \{\omega_{i+1}(k-1) - \omega_i(k-1)\}]\\ &= \frac{\omega_i(k-1) - \omega_{i-1}(k)}{\omega_{i+1}(k-1) - \omega_{i-1}(k)}\varphi_{i+1}(k-1) + \\ &\quad \frac{\omega_{i+1}(k-1) - \omega_i(k-1)}{\omega_{i+1}(k-1) - \omega_{i-1}(k)}\varphi_{i-1}(k).\end{aligned}$$

Define γ_i,

$$\gamma_i = \frac{\omega_i(k-1) - \omega_{i-1}(k)}{\omega_{i+1}(k-1) - \omega_{i-1}(k)}, \quad (15)$$

where $0 < \gamma_i < 1$.

Therefore, the k^{th} compensated node of i^{th} intermediate edge, $\varphi_i(k)$, can be represented by

$$\varphi_i(k) = (1-\gamma_i)\varphi_{i-1}(k) + \gamma_i\varphi_{i+1}(k-1) \quad (16)$$

where $\varphi_0(k) = \varphi_0 = \mu_1$ and $\varphi_{n+1}(k-1) = \varphi_{n+1} = \mu_2$.

$$\varphi_1(k) = (1-\gamma_1)\varphi_0(k) + \gamma_1\varphi_2(k-1) \quad (17)$$
$$\varphi_2(k) = (1-\gamma_2)\varphi_1(k) + \gamma_2\varphi_3(k-1) \quad (18)$$
$$\vdots$$
$$\varphi_i(k) = (1-\gamma_i)\varphi_{i-1}(k) + \gamma_i\varphi_{i+1}(k-1) \quad (19)$$
$$\vdots$$
$$\varphi_n(k) = (1-\gamma_n)\varphi_{n-1}(k) + \gamma_n\varphi_{n+1}(k-1). \quad (20)$$

With Equation 19, $\varphi_i(k)$ can be represented by

$$\varphi_i(k) = \{\prod_{j=1}^{i}(1-\gamma_j)\}\varphi_0 + \sum_{j=1}^{i}\{\gamma_j\prod_{m=j}^{i-1}(1-\gamma_{m+1})\}\varphi_{j+1}(k-1). \quad (21)$$

The state space equation of the compensated nodes can be represented by

$$\Psi(k+1) = \Gamma\Psi(k) + \Lambda U, \quad (22)$$

where

$$\Psi(k) = \begin{bmatrix}\varphi_1(k) & \varphi_2(k) & \cdots & \varphi_n(k)\end{bmatrix}^T \quad (23)$$

$$\Gamma = \begin{bmatrix} 0 & \gamma_1\prod_{j=1}^{0}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{1}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{2}(1-\gamma_{j+1}) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \gamma_1\prod_{j=1}^{n-1}(1-\gamma_{j+1}) & \cdots & \gamma_{n-1}\prod_{j=n-1}^{n-1}(1-\gamma_{j+1}) \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \prod_{j=1}^{1}(1-\gamma_j) & 0 \\ \prod_{j=1}^{2}(1-\gamma_j) & 0 \\ \prod_{j=1}^{3}(1-\gamma_j) & 0 \\ \vdots & \vdots \\ \prod_{j=1}^{n}(1-\gamma_j) & \gamma_n \end{bmatrix} \quad (24)$$

$$U = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix}^T. \tag{25}$$

The general solution to Equation 22 is represented by

$$\Psi(k) = \Phi(k)\Psi(0) + \sum_{j=0}^{k-1} \Phi(k-1-j)\Lambda U(j), \tag{26}$$

where $\Phi(k)$, the state transition matrix, is given by

$$\Phi(k) = \Gamma^k. \tag{27}$$

One technique for computing $\Phi(k)$ as a function of k is through the use of the $\mathcal{Z}$-transform. In Equation 22, let $U(k) = 0$. Then the $\mathcal{Z}$-transform of this equation yields

$$z\Psi(z) - z\Psi(0) = \Gamma\Psi(z). \tag{28}$$

Solving for $\Psi(z)$, one can obtain

$$\Psi(z) = z[zI - \Gamma]^{-1}\Psi(0) \tag{29}$$

Then

$$\begin{aligned} \Psi(k) &= \mathcal{Z}^{-1}[\Psi(z)]\Psi(0) \\ &= \mathcal{Z}^{-1}\{z[Zi - \Gamma]^{-1}\}\Psi(0) \end{aligned} \tag{30}$$

Comparing Equation 30 with Equation 26, we see that

$$\Phi(k) = \mathcal{Z}^{-1}\{z[Zi - \Gamma]^{-1}\} \tag{31}$$

Definition 2.1 *Define the* RCA matrix, Γ, *as the discrete state transition matrix of Equation 22 and* λ_i *as its eigenvalue for any* i *such that* $1 \le i \le n$.

$$\Gamma = \begin{bmatrix} 0 & \gamma_1\prod_{j=1}^{0}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{1}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{2}(1-\gamma_{j+1}) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \gamma_1\prod_{j=1}^{n-1}(1-\gamma_{j+1}) & \cdots & \gamma_{n-1}\prod_{j=n-1}^{n-1}(1-\gamma_{j+1}) \end{bmatrix}$$

The convergence of the *Recursive Compensation Algorithm* can be proved by showing that all the eigenvalues of the *RCA matrix* are within a unit circle.

Theorem 2.1 *All the eigenvalues of the RCA matrix are within a unit circle.*

Proof. D_1 is the *Gerŝgorin disk* at 0 of radius $|\gamma_1|$. Therefore λ_1 is within a unit circle. D_2 is the *Gerŝgorin disk* at $\gamma_1(1-\gamma_2)$ of radius $|\gamma_2|$. Therefore λ_2 is within a unit circle, since $|\gamma_1(1-\gamma_2)| + |\gamma_2| < 1$. For any i such that $1 < i < n-1$, D_i is the *Gerŝgorin disk* at $\gamma_{i-1}(1-\gamma_i)$ of radius

$$\sum_{j=1}^{i-2} |\gamma_j \prod_{k=j+1}^{i} (1-\gamma_k)| + |\gamma_i|. \tag{32}$$

Assume λ_i is within a unit circle, then D_{i+1} is the *Gerŝgorin disk* at $\gamma_i(1-\gamma_{i+1})$ of radius

$$\sum_{j=1}^{i-1} |\gamma_j \prod_{k=j+1}^{i+1} (1-\gamma_k)| + |\gamma_{i+1}|. \tag{33}$$

The extreme boundary of λ_{i+1} can be represented by

$$\begin{aligned} &|1-\gamma_{i+1}|\{\sum_{j=1}^{i-1} |\gamma_j \prod_{k=j+1}^{i} (1-\gamma_k)| + |\gamma_i|\} + |\gamma_{i+1}| \\ &= |1-\gamma_{i+1}||\lambda_i| + |\gamma_{i+1}| < 1. \end{aligned} \tag{34}$$

Therefore λ_{i+1} is also within a unit circle. Finally, D_n is the *Gerŝgorin disk* at $\gamma_{n-1}(1-\gamma_n)$ of radius

$$\sum_{j=1}^{n-2} |\gamma_j \prod_{k=j+1}^{n} (1-\gamma_k)|. \tag{35}$$

The extreme boundary of λ_n can be represented by

$$\begin{aligned} &|1-\gamma_n|\{\sum_{j=1}^{n-2} |\gamma_j \prod_{k=j+1}^{n-1} (1-\gamma_k)| + |\gamma_{n-1}|\} \\ &= |1-\gamma_n||\lambda_{n-1}| < 1. \end{aligned} \tag{36}$$

Hence, λ_n is also within a unit circle. Therefore, all the eigenvalues of the *RCA matrix* are within a unit circle.

Q.E.D.

Definition 2.2 *Define the RCA sequences as* $\varphi_i(k) - \varphi_i(k+1)$ *of the node* i, *where* $\varphi_i(k)$ *is the* k^{th} *compensated node by the Recursive Compensation Algorithm.*

Theorem 2.2 *The RCA sequences are Cauchy sequences.*

Proof. With Equation 22, the state space equation for the sequences generated by the Recursive Compensation Algorithm can be represented by

$$\Psi(k+1) = \Gamma\Psi(k) + \Lambda U, \tag{37}$$

where

$$\Psi(k) = \begin{bmatrix} \varphi_1(k) & \varphi_2(k) & \cdots & \varphi_n(k) \end{bmatrix}^T \tag{38}$$

$$\Gamma = \begin{bmatrix} 0 & \gamma_1\prod_{j=1}^{0}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{1}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1\prod_{j=1}^{2}(1-\gamma_{j+1}) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \gamma_1\prod_{j=1}^{n-1}(1-\gamma_{j+1}) & \cdots & \gamma_{n-1}\prod_{j=n-1}^{n-1}(1-\gamma_{j+1}) \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \prod_{j=1}^{1}(1-\gamma_j) & 0 \\ \prod_{j=1}^{2}(1-\gamma_j) & 0 \\ \prod_{j=1}^{3}(1-\gamma_j) & 0 \\ \vdots & \vdots \\ \prod_{j=1}^{n}(1-\gamma_j) & \gamma_n \end{bmatrix} \tag{39}$$

$$U = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix}^T. \tag{40}$$

Since Theorem 2.1 shows that all the eigenvalues of the RCA matrix are within a unit circle, the system represented by Equation 22 is stable and all states by Equation 22 converge to some real numbers. Therefore, the sequences generated by the Recursive Compensation Algorithm are Cauchy sequences, because a Cauchy sequence converges to some real number.

Q.E.D.

One can conclude that the number of compensations

for the *Recursive Compensation Algorithm* could be calculated if the accuracy, ϵ, is known. Or the accuracy, ϵ, could be computed if the number of compensations of the *Recursive Compensation Algorithm* is given, because the *RCA sequences* are Cauchy sequences.

Theorem 2.3 *The global optimal solution can be calculated by the Recursive Compensation Algorithm, where the optimal edges are selected for the Recursive Compensation Algorithm.*

Proof. Define φ_i^* as $\lim_{k\to\infty}\varphi_i(k)$ of the node i. φ_i^* is the bounded real number by the Theorem 2.2. Since the Recursive Compensation Algorithms shows

$$\tan\theta_i = \frac{\varphi_i(k)-\varphi_{i-1}(k)}{w_i(k)-w_{i-1}(k)}, \tag{41}$$

Equation 42 should be proved for the global optimality of the Recursive Compensation Algorithm.

$$\tan\theta_i = \tan\theta_{i+1}. \tag{42}$$

In Equation 42, one can get

$$\frac{w_i-w_{i-1}}{w_{i+1}-w_{i-1}} = \frac{\varphi_i(k)-\varphi_{i-1}(k)}{\varphi_{i+1}(k)-\varphi_{i-1}(k)}. \tag{43}$$

Since the RCA sequences are Cauchy sequences by the Theorem 2.2, one can obtain

$$\frac{\varphi_i^*-\varphi_{i-1}^*}{\varphi_{i+1}^*-\varphi_{i-1}^*} = \frac{w_i-w_{i-1}}{w_{i+1}-w_{i-1}}. \tag{44}$$

By Equation 15, the right term of Equation 44 is γ_i. Let's evaluate the left term of Equation 44. By Equation 16 and Theorem 2.2. one can get

$$\begin{aligned}\frac{\varphi_i^*-\varphi_{i-1}^*}{\varphi_{i+1}^*-\varphi_{i-1}^*} &= \frac{\{(1-\gamma_i)\varphi_{i+1}^*+\gamma_i\varphi_{i+1}^*\}-\varphi_{i-1}^*}{\varphi_{i+1}^*-\varphi_{i-1}^*}\\ &= \gamma_i\end{aligned} \tag{45}$$

Therefore, the Recursive Compensation Algorithm can compute the global optimal solution.

Q.E.D.

The global optimal path by the *Recursive Compensation Algorithm* is solved in the case of two nodes for simplicity and clarity without loosing the generality. We are interested in the convergence of the compensated nodes by the *Recursive Compensation Algorithm*, i.e., $\{\varphi_1(k)\}$, $\{\varphi_2(k)\}$.

$$\begin{aligned}\varphi_1(k) &= \varphi_0+\frac{\varphi_2(k-1)-\varphi_0}{\omega_2-\omega_0}(\omega_1-\omega_0)\\ &= \gamma_1\varphi_2(k-1)+(1-\gamma_1)\varphi_0\end{aligned} \tag{46}$$

$$\begin{aligned}\varphi_2(k) &= \varphi_1(k)+\frac{\varphi_3-\varphi_1(k)}{\omega_3-\omega_1}(\omega_2-\omega_1)\\ &= (1-\gamma_2)\varphi_1(k)+\gamma_2\varphi_3.\end{aligned} \tag{47}$$

The sequences generated by the *Recursive Compensation Algorithm* are convergent by the Theorem 2.1. Let's compute $\varphi_1(k)$ and $\varphi_2(k)$, assuming that there are two polyhedral obstacles in 3D. From Equation 22, the state equation is given by

$$\Psi(k+1) = \Gamma\Psi(k)+\Lambda U \tag{48}$$

where

$$\Psi(k) = \begin{bmatrix}\varphi_1(k) & \varphi_2(k)\end{bmatrix}^T \tag{49}$$

$$\Gamma = \begin{bmatrix}0 & \gamma_1\\ 0 & \gamma_1(1-\gamma_2)\end{bmatrix} \tag{50}$$

$$\Lambda = \begin{bmatrix}(1-\gamma_1) & 0\\ (1-\gamma_1)(1-\gamma_2) & \gamma_2\end{bmatrix} \tag{51}$$

$$U = \begin{bmatrix}\mu_1 & \mu_2\end{bmatrix}^T \tag{52}$$

The solution of the state equation is given by

$$\Psi(k) = \Phi(k)\Psi(0)+\sum_{j=0}^{k-1}\Phi(k-1-j)\Lambda U(j) \tag{53}$$

where $\Phi(k)$, the state transition matrix, is given by $\Phi(k) = \Gamma^k$. One technique for evaluating $\Phi(k)$ as a function of k is through the use of the Z-transform.

$$\begin{aligned}\Phi(k) &= Z^{-1}\{z(zI-\Gamma)^{-1}\} &(54)\\ &= Z^{-1}\left\{\frac{1}{z-\gamma_1(1-\gamma_2)}\begin{bmatrix}z-\gamma_1(1-\gamma_2) & \gamma_1\\ 0 & z\end{bmatrix}\right\}\\ &= \begin{bmatrix}\delta(k) & \frac{-1}{1-\gamma_2}\{\delta(k)-[\gamma_1(1-\gamma_2)]^k\}\\ 0 & [\gamma_1(1-\gamma_2)]^k\end{bmatrix} &(55)\\ &= \begin{bmatrix}\delta(k) & \frac{-1}{1-\gamma_2}\{\delta(k)-\alpha^k\}\\ 0 & \alpha^k\end{bmatrix} &(56)\end{aligned}$$

where $\alpha \equiv \gamma_1(1-\gamma_2)$. Therefore, the solution of the state equation is obtained by

$$\Psi(k) = \Phi(k)\Psi(0)+\sum_{j=0}^{k-1}\Phi(k-1-j)\Lambda U(j) \tag{57}$$

$$\begin{aligned}\begin{bmatrix}\varphi_1(k)\\ \varphi_2(k)\end{bmatrix} &= \begin{bmatrix}\delta(k) & \frac{-1}{1-\gamma_2}\{\delta(k)-\alpha^k\}\\ 0 & \alpha^k\end{bmatrix}\begin{bmatrix}\varphi_1(0)\\ \varphi_2(0)\end{bmatrix}+\\ &\sum_{j=0}^{k-1}\begin{bmatrix}\delta(k-1-j) & \frac{-1}{1-\gamma_2}\{\delta(k-1-j)-\alpha^{k-1-j}\}\\ 0 & \alpha^{k-1-j}\end{bmatrix}\\ &\begin{bmatrix}(1-\gamma_1) & 0\\ (1-\gamma_1)(1-\gamma_2) & \gamma_2\end{bmatrix}\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}\end{aligned} \tag{58}$$

3 The Optimal Edge Selection

[Problem Statement] 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 $\tau = 1$ [9]. 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 dis-

tance 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* [6] 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 [5] is applied to compute the fitness function. We set the parameters as follows; 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, We set e_i, $i = 1,2,\cdots,12$ as the edges of a polyhedral; $e_1 = \overline{P_1P_2}$, $e_2 = \overline{P_2P_3}$, $e_3 = \overline{P_3P_4}$, $e_4 = \overline{P_1P_4}$, $e_5 = \overline{P_1P_5}$, $e_6 = \overline{P_2P_6}$, $e_7 = \overline{P_3P_7}$, $e_8 = \overline{P_4P_8}$, $e_9 = \overline{P_5P_6}$, $e_{10} = \overline{P_6P_7}$, $e_{11} = \overline{P_2P_7}$, $e_{12} = \overline{P_5P_8}$.

Table 1 and Table 2 show that the optimal edges are e_6 and e_7. The Recursive Compensation Algorithm shows that the optimal vertices are $N_1(2.0,1.0,1.0682)$ along e_6 and $N_2(2.0,3.0,2.0842)$ along e_7 and the shortest path between L_i and L_j is 0.5519453549×10.

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

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

4 Conclusions

The state-space equations for the *Recursive Compensation Algorithm* are derived in Equation 22. The general solution for the *Recursive Compensation Algorithm* can be obtained by Equation 26. Defining the *RCA matrix*, Γ, as the discrete state transition matrix of Equation 22 and λ_i as its eigenvalue for any i such that $1 \le i \le n$, Theorem 2.1 is proved to show that all the eigenvalues of the *RCA matrix* are within a unit circle with the *Gerŝgorin Theorem*. Therefore, the state-space equation for the *Recursive Compensation Algorithm* has the bounded solution.

$$\Gamma = \begin{bmatrix} 0 & \gamma_1 \prod_{j=1}^{0}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1 \prod_{j=1}^{1}(1-\gamma_{j+1}) & \cdots & 0 \\ 0 & \gamma_1 \prod_{j=1}^{2}(1-\gamma_{j+1}) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \gamma_1 \prod_{j=1}^{n-1}(1-\gamma_{j+1}) & \cdots & \gamma_{n-1} \prod_{j=n-1}^{n-1}(1-\gamma_{j+1}) \end{bmatrix}$$

Defining the *RCA sequences* as $\varphi_i(k) - \varphi_i(k+1)$ of the node i, where $\varphi_i(k)$ is the k^{th} compensated node by the *Recursive Compensation Algorithm*, Theorem 2.2 is proved to show that the *RCA Sequences* are *Cauchy sequences*. Therefore, Theorem 2.2 shows the convergent property of the *Recursive Compensation Algorithm*. One can conclude that the number of compensations for the *Recursive Compensation Algorithm* could be calculated if the accuracy, ϵ, is known. Or the accuracy, ϵ, could be computed if the number of compensations of the *Recursive Compensation Algorithm* is given, because the *RCA sequences* are Cauchy sequences.

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 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%

Theorem 2.3 is proved to show that the global optimal solution can be calculated by the *Recursive Compensation Algorithm*, where the optimal edges are selected for the *Recursive Compensation Algorithm*. Therefore, the *RCA sequences* converge to some bounded numbers by Theorem 2.2 and they are global optimal numbers by Theorem 2.3.

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

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. 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.

References

[1] C. H. Chung and G. N. Saridis, "Obstacle avoidance path planning by the extended vgraph algorithm," Tech. Rep. CIRSSE-TR-89-12, Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute, January 1989.

[2] R. A. Brooks, "Planning collision-free motions for pick-and-place operations," *The International Journal of Robotics Research*, vol. 2, no. 4, pp. 19–44, 1983.

[3] T. Lozano-Pérez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 11, pp. 681–698, October 1981.

[4] C. H. Chung and G. N. Saridis, "Path planning for an intelligent robot by the extended vgraph algorithm," in *IEEE International Symposium on Intelligent Control*, (Albany, NY), September 1989.

[5] C. H. Chung and G. N. Saridis, "The recursive compensation algorithm for obstacle avoidance path planning," in *IEEE International Workshops on Intelligent Robots and Systems*, (Tsukuba, Japan), September 1989.

[6] M. C. Moed and G. N. Saridis, "A boltzmann machine for the organization of intelligent machines," in *Proceedings of 2 Telerobotics Conference*, (Pasadena, CA), January 1989.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley Publishing Company, INC., 1989.

[8] C. H. Chung and K. S. Lee, "Neural network application to the obstacle avoidance path planning for cim (computer integrated manufacturing)," in *IEEE International Workshops on Intelligent Robots and Systems*, (Osaka, Japan), November 1991.

[9] G. V. Wilson and G. S. Pawley, "On the stability of the travelling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, pp. 63–70, 1988.