A Global Optimal Approach for Robot Kinematics Design using the Grid Method

Joon-Young Park, Pyung-Hun Chang, and Jin-Oh Kim

Abstract: In a previous research, we presented the Grid Method and confirmed it as a systematic and efficient problem formulation method for the task-oriented design of robot kinematics. However, our previous research was limited in two ways. First, it gave only a local optimum due to its use of a local optimization technique. Second, it used constant weights for a cost function chosen by the manual weights tuning algorithm, thereby showing low efficiency in finding an optimal solution. To overcome these two limitations, therefore, this paper presents a global optimization technique and an adaptive weights tuning algorithm to solve a formulated problem using the Grid Method. The efficiencies of the proposed algorithms have been confirmed through the kinematic design examples of various robot manipulators.

Keywords: Adaptive weights tuning, global optimal kinematics, grid method, simulated annealing.

1. INTRODUCTION

In a previous research, we presented the Grid Method [1,2], a systematic, efficient problem formulation method for the task-oriented design of robot kinematics, and we confirmed its efficiency through various design examples. In this paper, we newly present a global optimization technique and an adaptive weights tuning algorithm that can be used to efficiently solve a formulated design problem using the Grid Method.

Previous researches on the optimal problem formulation of kinematic design [3-7] are problematic in that the number of design variables increases in proportion to d.o.f. and to the number of task points, which then increases the complexities of cost functions and constraints. In addition, these approaches become less efficient as d.o.f. and the number of task points increase, considering that search space generally increases exponentially as the number of design variables increases [8]. To solve such a problem, we proposed the Grid Method [1,2] on the basis of the widely used method in Finite Difference Method (FDM) or numerical analyses of heat transfer and fluid flow. In the Grid Method, our approach formulated an optimal design problem by using a constant number of design variables regardless of d.o.f. and the number of task points. In spite of its efficient problem formulation, however, our previous research was limited in two ways. First, it gave only a local optimum due to its use of a local optimization technique, Generalized Reduced Gradient Method (GRG). Second, it used constant weights for a cost function chosen by the manual weights tuning algorithm, thereby making a selection of the weights difficult and showing low efficiency in finding an optimum. To overcome these limitations, therefore, this paper presents a global optimization technique and an adaptive weights tuning algorithm for systematically and efficiently solving an optimal design problem formulated by using the Grid Method.

The literature reviews on the global optimization techniques used in previous approaches show that they can be mainly classified as either Genetic Algorithm (GA) [5-9] or Simulated Annealing [3,10]. Additionally, most previous approaches have used constant weights for the cost function [3,6,11]. Kim suggested a weights tuning algorithm that adaptively determined the weights through four selection phases [8]. However, the details of the weights tuning in each phase were not automatically determined but still

Manuscript received April 18, 2005; revised November 2, 2005 and April 2, 2006; accepted April 26, 2006. Recommended by Editorial Board member Sangdeok Park under the direction of Editor Jae-Bok Song. This work was supported by the Korea Science and Engineer-ing Foundation (KOSEF) and Human-friendly Welfare Robot System Engineering Research Center (HWRS-ERC).

Joon-Young Park is with Power Generation Laboratory, Korea Electric Power Research Institute, 103-16 Munji-dong, Yuseong-gu, Daejeon 305-380, Korea (e-mail: asura@kepco.co.kr).

Pyung-Hun Chang is with the Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea (e-mail: phchang@mecha.kaist.ac.kr).

Jin-Oh Kim is with the Department of Information and Control Engineering, Kwangwoon University, 447-1 Wolgye-dong, Nowon-gu, Seoul 139-701, Korea (e-mail: jokim@daisy.kw.ac.kr).

depended on manual tuning. Leger proposed the Requirement Prioritization to adaptively determine the weights in consideration of priority between the constraint functions included in the cost function [9]. In accordance with Leger's thesis, the constraint function in our paper is called a metric. However, this tuning scheme has no regard for dimensional nonhomogeneity between the constraint functions or for how far each constraint function deviates from its optimum. Its determination of the weights was based solely on whether or not a population of GA satisfied each constraint function.

In this paper, we chose the Very Fast Simulated Annealing (VFSA) [12,13], which has faster convergence than the Simulated Annealing, as a global optimization technique for solving a formulated problem using the Grid Method. Additionally, we used Recursive Evolution Technique (RET) [14] so as to further improve the convergence characteristics of VFSA. Through this application, we show that a global optimization technique as well as a local optimization technique can be applied to formulated problems using the Grid Method. Furthermore, in order to increase the efficiency of finding an optimum, we propose an adaptive weights tuning algorithm in consideration of dimensional nonhomogeneity in the constraint functions and deviation of each constraint function from its optimum, and confirm its effectiveness through actual applications.

The remainder of this paper is organized as follows. In the following section, we briefly introduce the Grid Method. Section 3 introduces VFSA and RET, which are used for global optimization, and finally presents the whole kinematic design algorithm obtained by combining the techniques above with a problem formulation using the Grid Method. In Section 4, we propose a simple and efficient adaptive weights tuning algorithm to increase the efficiency in finding a solution to the optimal design problem formulated by the Grid Method. Section 5 confirms the efficiencies of the proposed algorithms through specific applications. Finally, in Section 6, the results are summarized and conclusions are drawn.

2. PROBLEM FORMULATION USING GRID METHOD

In this section, we briefly introduce the Grid Method and give a simple example to help readers understand the Grid Method. Please refer to [1,2] for detailed information on the Grid Method.

2.1. Basic assumptions

(i) All the design variables can vary continuously.

(ii) We use Paul's notation [15] to represent DH parameters for robot kinematics. In the case of Fig. 1, DH parameters between joint i and joint i+1 can be represented by l_i, d_i, α_i, θ_i. Here, l_i denotes link length; d_i link offset; α_i twist angle; θ_i joint angle.

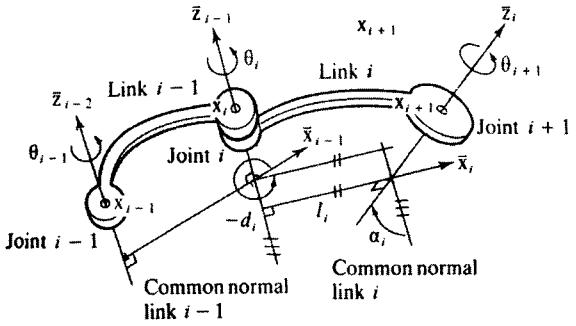


Fig. 1. DH parameters with Paul's notation.

(iii) As an object of our research, we deal with general spatial manipulators that have link lengths, link offsets, and twist angles as design variables.

(iv) The d.o.f. of the robot is given.

(v) The robot base position is fixed and known.

(vi) No self-collision is considered.

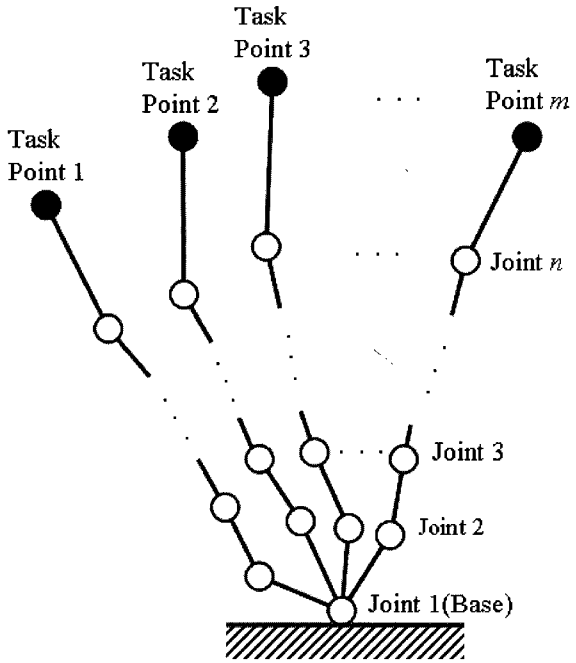
(vii) The robot can be represented by a set of straight line segments with a thickness of zero.

2.2. Problem formulation using the Grid Method

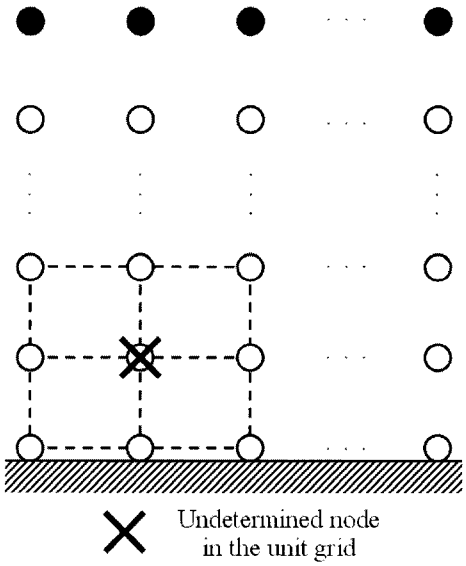
Let us consider the kinematic design of an n-d.o.f. robot under m task points given as kinematic requirements to be satisfied, as shown in Fig. 2(a). If we consider every joint and task point as grid nodes, the robots of Fig. 2(a) can be represented as a virtual grid space consisting of the grids shown in Fig. 2(b). The basic concept of the Grid Method regarding these grid nodes is that, just like a heat transfer problem, the base and task points are given as boundary conditions, and then the joint positions and orientations within are calculated through successive grid operations on the unit grids shown in Fig. 2(b).

Now, we present an actual problem formulation using the Grid Method. As stated above, the use of the Grid Method simplifies the design problem of Fig. 2(a) into that of the unit grid shown in Fig. 2(b). Let Fig. 3 represent the unit grid corresponding to the ith joint and jth task point. The corresponding joint position and twist angle are represented as $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})$ and $\alpha_{i-1,j} (1 \le i \le n+1, 1 \le j \le m)$, respectively. Here, $\alpha_{i-1,j}$ denotes the twist angle between the axes of joint $i-1$ and joint i, and $i = n+1$ indicates the task point that the end-effector reaches. Then the optimal design problem of robot kinematics can be formulated as that of the unit grid as follows:

Find $\mathbf{x}_{i,j}$, $\alpha_{i-1,j}$ of design variables to minimize a cost function



(a) n-d.o.f. manipulator for m task points.



(b) Application of unit grid.

Fig. 2. Application of unit grids to n-d.o.f. manipulator for m task points.

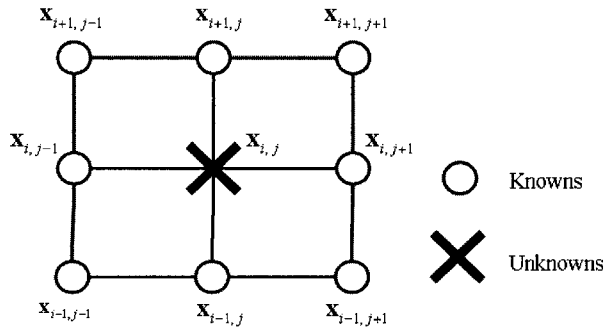


Fig. 3. Unit grid.

$$f\left(\mathbf{x}_{i,j}, \alpha_{i-1,j}\right) = f\left(x_{i,j}, y_{i,j}, z_{i,j}, \alpha_{i-1,j}\right), \quad i = 2,3,\cdots,n, \quad j = 1,2,\cdots,m, \tag{1}$$

subject to the p equality constraints:

$$h_s\left(\mathbf{x}_{i,j}, \alpha_{i-1,j}\right) = h_s\left(x_{i,j}, y_{i,j}, z_{i,j}, \alpha_{i-1,j}\right) = 0, \quad s = 1 \text{ to } p, \tag{2}$$

and the q inequality constraints:

$$g_t\left(\mathbf{x}_{i,j}, \alpha_{i-1,j}\right) = g_t\left(x_{i,j}, y_{i,j}, z_{i,j}, \alpha_{i-1,j}\right) \le 0, \quad t = 1 \text{ to } q. \tag{3}$$

Now we will describe the design variables, the cost function, and the constraints needed for the actual mathematical formulation of (1) to (3).

2.2.1 Design variables

The Grid Method takes the joint position $\mathbf{x}_{i,j}$ and twist angle $\alpha_{i-1,j}$ of the central node in the unit grid as a set of design variables. Here, notice that the twist angle was chosen as a design variable to represent joint orientation based on the fact that in Paul's notation [15], the Z axis of the first joint is always coincident with that of the world coordinate frame. Therefore, the Z axis of every joint can be found, provided we know every twist angle that represents the angle between two neighboring Z axes.

2.2.2 Cost function

By using the weighting function method, we define the cost function for the unit grid corresponding to the ith joint of the jth task point as follows, and it should be minimized for the sake of the optimal design.

$$\begin{aligned} f_{UG}\left(\mathbf{x}_{i,j}, \alpha_{i-1,j}\right) = {} & w_{EC} * f_{EC(i,j)} + w_{DOC} * f_{DOC(i,j)} \\ & + w_{LC} * f_{LC(i,j)} + w_{DC} * f_{DC(i,j)} \\ & + w_{OA} * f_{OA(i,j)} + w_{DM} * f_{DM(i,j)} \\ & + w_{JAC} * f_{JAC(i,j)} \end{aligned} \tag{4}$$

with 'UG' denoting the unit grid, and with $w_{(\cdot)}$ denoting a weight for each constraint function of $f_{UG}\left(\mathbf{x}_{i,j}, \alpha_{i-1,j}\right)$. In this section, we describe only the meaning of each function; their mathematical expressions are presented in Appendix A.

Equalization constraint (EC): *EC* makes the DH parameters of the jth task point close to those of the $j-1$th and $j+1$th task points in the unit grid. If *EC*s become zero for all the unit grids, we have the identical robot for all the task points.

Desired orientation constraint (DOC): Since the desired orientation is determined by other joint positions as well as the position of the end-effector,

we need this constraint to achieve the desired orientation.

Limit constraints (LC): *LC* is a constraint on the possible ranges of DH parameters.

Dimension constraint (DC) [8]: *DC* requires that an inner distance always be longer than an outer distance, where distance L between two adjacent joints is defined as $L=(l^2+d^2)^{1/2}$.

Obstacle avoidance measure (OA): We consider sphere-type obstacles basically because most oddly shaped obstacles can be approximated by a combination of several spheres. We determine their cost function using the concept of the potential function [16].

Dexterity measure (DM): Of various dexterity measures for the kinematic design of a robot manipulator, in this paper, we consider only Layout Conditioning [17]. We will explain the selection of *DM* in detail in Section 4.2.

Joint angle change constraint (JAC): *JAC* is used to prevent a large, sudden change of joint angles between two adjacent task points.

2.2.3 Constraints

In the case of a local optimization technique being used for solving the formulated problem, we used an inequality constraint to limit the search region to a finite region where each link is less than the maximum distance from the base to the task points. Since VFSA is chosen as a global optimization technique for our research, however, the role of the inequality constraint above can be replaced by 'stepsize adjustment'.

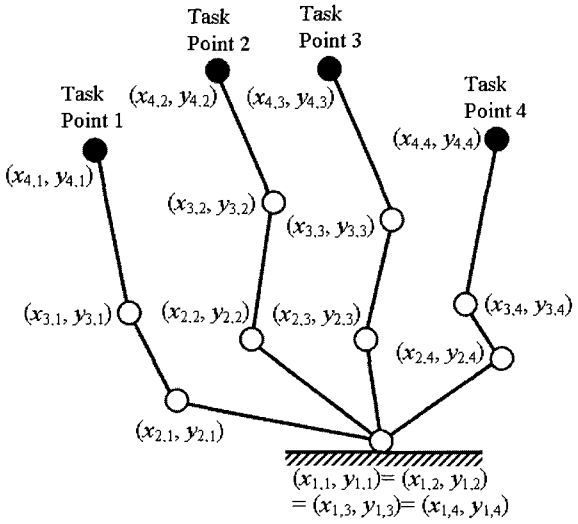
2.3. Simple example using the Grid Method

To help the reader gain a better understanding about the Grid Method, we present the formulations on the simple design example of a 3-d.o.f. planar manipulator under 4 task points shown in Fig. 4(a) using two different approaches - the widely used general formulation method and the Grid Method. Here, the general formulation method denotes the design method that mathematically formulates the optimum design problem using the general mathematical model, often called the standard design optimization model [18]. We used the *EC* only, the most basic constraints to be a solution in order to simplify the problem.

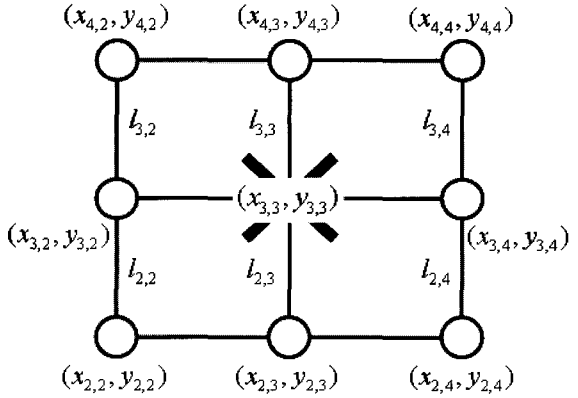
The general formulation method identifies all the parameters required to describe the design of a system as a set of design variables, thereby setting all joint positions as design variables. The problem, therefore, can be formulated as follows:

Find a vector of design variables

$$\mathbf{x}=\begin{pmatrix} x_{2,1}, y_{2,1} & x_{2,2}, y_{2,2} & x_{2,3}, y_{2,3} & x_{2,4}, y_{2,4} \\ x_{3,1}, y_{3,1} & x_{3,2}, y_{3,2} & x_{3,3}, y_{3,3} & x_{3,4}, y_{3,4} \end{pmatrix}^T.$$



(a) 3-d.o.f. manipulator for 4 task points.



(b) Unit grid corresponding to the 3rd joint and 3rd task point.

Fig. 4. Simple example using the Grid Method.

To minimize a cost function:

$$\begin{aligned} f(\mathbf{x}) = &\left(l_{1,1}-l_{1,2}\right)^2+\left(l_{1,2}-l_{1,3}\right)^2+\left(l_{1,3}-l_{1,4}\right)^2 \\ &+\left(l_{2,1}-l_{2,2}\right)^2+\left(l_{2,2}-l_{2,3}\right)^2+\left(l_{2,3}-l_{2,4}\right)^2 \\ &+\left(l_{3,1}-l_{3,2}\right)^2+\left(l_{3,2}-l_{3,3}\right)^2+\left(l_{3,3}-l_{3,4}\right)^2, \end{aligned}$$

where $l_{i,j}=\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}$

subject to the inequality constraints:

$$\begin{aligned} &0\le l_{1,1}\le L_{\max}, 0\le l_{2,1}\le L_{\max}, 0\le l_{3,1}\le L_{\max}, \\ &0\le l_{1,2}\le L_{\max}, 0\le l_{2,2}\le L_{\max}, 0\le l_{3,2}\le L_{\max}, \\ &0\le l_{1,3}\le L_{\max}, 0\le l_{2,3}\le L_{\max}, 0\le l_{3,3}\le L_{\max}, \\ &0\le l_{1,4}\le L_{\max}, 0\le l_{2,4}\le L_{\max}, 0\le l_{3,4}\le L_{\max}, \end{aligned}$$

where

$$L_{\max}=\max(\sqrt{(x_{4,j}-x_{1,j})^2+(y_{4,j}-y_{1,j})^2}).$$

You can see that in this approach, the number of design variables is '2(planar robot case) * (d.o.f-1) * (no. of task points)' = '2 * (3-1) * 4' = 16, which means that the number of design variables and complexity of the problem increase with the number of d.o.f. and task points. Therefore, this approach becomes more inefficient as the number of d.o.f and task points increases, considering that the search space in general increases exponentially as the number of design variables increases.

Now we apply the Grid Method to this design problem. As stated in Section 2.2, considering every joint and task point as grid nodes, we can divide the design problem of Fig. 4(a) into those of the unit grids like Fig. 4(b). Fig. 4(b) shows an example of the unit grid that corresponds to the 3rd joint and 3rd task point. The unit grid of Fig. 4(b) can be formulated as follows:

Find a vector of design variables

$$\mathbf{x} = \left(x_{3,3}, y_{3,3}\right).$$

To minimize a cost function:

$$f(\mathbf{x}) = \left(l_{2,2} - l_{2,3}\right)^2 + \left(l_{2,3} - l_{2,4}\right)^2 + \left(l_{3,2} - l_{3,3}\right)^2 + \left(l_{3,3} - l_{3,4}\right)^2$$

subject to the inequality constraints:

$$0 \le l_{2,3} \le L_{\max},\ 0 \le l_{3,3} \le L_{\max}.$$

Here, notice that the number of design variables remains constant (i.e., '2(planar case)') irrespective of the number of d.o.f. and of task points. This implies that the Grid Method becomes more efficient as the number of d.o.f. and of task points both increase. Due to this, the cost function f and the inequality constraints can be represented very simply compared with those of general formulation method.

The joint positions of all the nodes except the central one of the unit grid in Fig. 4(b) are given as boundary conditions. Through the application of an optimization technique to the equations above, the Grid Method determines a new joint position $(x_{3,3}^{new},\ y_{3,3}^{new})$ of the central node, with the cost function f optimized and with the given constraints satisfied under such boundary conditions. We define this Grid Method operation as Grid Operation. To determine all the joint positions, one should perform consecutive Grid Operations for all the unit grids in the following order:

```
for j = 1 to 4(no. of task points)
  for i = 2 to 3(d.o.f.)
        Grid Operation(i, j)
  end
end
```

3. VERY FAST SIMULATED ANNEALING FOR GRID METHOD

In this paper, we chose VFSA, a widely used global optimization technique [12,13], to find a global optimal kinematics and, moreover, we used RET (Recursive Evolution Technique) [14] to accelerate the convergence characteristics of the Metropolis algorithm in VFSA. In the following subsections, we briefly introduce each of the techniques above, and finally propose the whole design procedure obtained by combining these techniques with a problem formulation using the Grid Method.

3.1. Very fast simulated annealing and recursive evolution technique

Simulated Annealing (SA), together with Genetic Algorithm (GA), has been widely used as a global optimization technique due to its effectiveness in finding a global optimum from local optima and its simple, compact algorithm. The idea for SA was based on the statistical mechanics algorithm of Metropolis [19-21]. Theoretical studies on the algorithm of SA have shown that a global optimum of the optimization problem can be reached with a probability one [22]. The main disadvantage of Classical SA, as is well known, is its slow rate of convergence. There are, of course, many algorithms to compensate for this, such as Fast Simulated Annealing [23], Very Fast Simulated Re-annealing [12,13], and so on. Of these, we chose VFSA as a global optimization technique for our research. Since the re-annealing part of Ingber's Very Fast Simulated Re-annealing is used to increase its convergence characteristics and is not an indispensable part for SA, we do not use it in this paper.

The basic algorithm of VFSA can be summarized as follows [13]:

(i: Initialization) Start with a high initial temperature T and a random starting point $\mathbf{x}$ where $\mathbf{x} = \left\{x_i; x_i \in \left[A_i, B_i\right], i = 1 \ldots D\right\}$.

$$T \leftarrow T_0,\ \mathbf{x} \leftarrow \mathbf{x}_0 \tag{5}$$

(ii: Initial Evaluation) Find the function value of the starting point.

$$E \leftarrow f(\mathbf{x}) \tag{6}$$

(iii: Random Generation) Generate a new point $\mathbf{x}'$ using the random variable $y_i \in [-1, 1]$. In other words, y_i's are repeatedly generated until a valid set $\mathbf{x}'$ is found.

$$x_i' \leftarrow x_i + y_i\left(B_i - A_i\right),\ x' \in \left[A_i, B_i\right] \tag{7}$$

The generating distribution of VFSA is defined as

$$g_T(y) = \prod_{i=1}^{D} \frac{1}{2(|y_i| + T)\ln(1 + 1/T)}. \quad (8)$$

To generate new points according to this distribution, new values of y_i are generated from the uniform distribution $u_i \in U[0, 1]$ as follows.

$$y_i = \mathrm{sgn}\left(u_i - \frac{1}{2}\right) T\left[(1 + 1/T)^{|2u_i - 1|} - 1\right]. \quad (9)$$

(iv: Function Evaluation) Calculate the function value of $\mathbf{x}'$.

$$E' \leftarrow f(\mathbf{x}') \quad (10)$$

(v: Metropolis Criterion) Accept or reject the new point $\mathbf{x}'$ using the Metropolis criterion.

If $\Delta E < 0$ where $\Delta E = E' - E$, accept the new point.

If $\Delta E > 0$, accept the new point with the probability density function defined as

$$p(\mathbf{x}') = \frac{1}{1 + \exp(\Delta E / T)}. \quad (11)$$

(vi: Temperature Annealing) Reduce the temperature T by the annealing schedule.

$$T' \leftarrow T(0)\exp\left(-c\,k^{1/D}\right) \quad (12)$$

(vii: Convergence Criterion) Return $\mathbf{x}$ and E as the optimal point and the optimal function value, respectively. If the convergence criterion is not satisfied, go to (iii). Otherwise, stop executing the algorithm.

In many cases, the above procedure for the SA algorithm includes the step of 'stepsize adjustment.' As the number of iterations increases, the stepsize adjustment algorithm gradually reduces $[A_i, B_i]$ of (7), the region that should be searched for an optimum, and is used to improve the convergence characteristics of SA. There exist various forms of stepsize adjustment [14,24], but we used the following algorithm:

$$S_i' \leftarrow \alpha * S_i, \quad \text{where} \quad 0 < \alpha < 1, \; S_i = [A_i, B_i] \quad (13)$$

and gave the following as its initial values for an optimal search starting from a sufficiently large region:

$$S_i(0) = 0.5 * (L_{(i-1,j)\max} + L_{(i,j)\max}) \quad \text{for } \mathbf{x}_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j}), \quad (14)$$
$$S_i(0) = \pi/4 \quad \text{for } \alpha_{i-1,j}.$$

Jung and Cho proposed RET to accelerate the convergence characteristics of the Metropolis algorithm [14]. The basic concept behind RET is that if $f^{\min}$ is not improved during a given number of trials, the recursive evolution of solutions is actuated from $\mathbf{x}^{\min}$.

$$E \leftarrow f^{\min}$$
$$\mathbf{x} \leftarrow \mathbf{x}^{\min} \quad (15)$$

That is to say, the basic idea of RET is that if the minimum is not updated in spite of our efforts to find an optimum smaller than the current minimum during a given number of trials, a new search attempt starts again from the minimum.

3.2. Whole design procedure by combining VFSA & the Grid Method

In this subsection, we propose a whole design procedure for the global optimization of robot kinematics. This procedure is obtained by combining the optimization techniques introduced in the previous subsection with a problem formulation using the Grid Method. The detailed explanation of the whole design procedure is as follows:

(i) Initialization.
(ii) Grid Operation Loop: the detailed procedure for Grid Operation Loop is presented in Fig. 5.
(iii) Is the number of step adjustments larger than $n_{s\_\max}$? If no, go to (v).
(iv) Adjust the step size. Reset the number of step adjustments to 0.
(v) Is the number of temperature annealing larger than $n_{t\_\max}$? If no, go to (vii).
(vi) Reduce temperature. Reset the number of temperature annealing to 0.
(vii) Increase the number of step adjustments. Increase the number of temperature annealing.
(viii) Set a current point as an optimum.
(ix) Recursive Evolution Technique.
(x) Is the convergence criterion satisfied? If no, return to (ii). Otherwise, stop executing the algorithm.

3.3. Some remarks on the actual application

In actual application, the global optimal solutions of most kinematic design problems can not be solved explicitly. This is the reason why we use the global optimization techniques for such problems. By using SA, for example, a global optimum of the optimization problem can be theoretically reached with a probability one [22]. However, enormous efforts (iterations) are usually required to reach a global optimum on account of the random generation

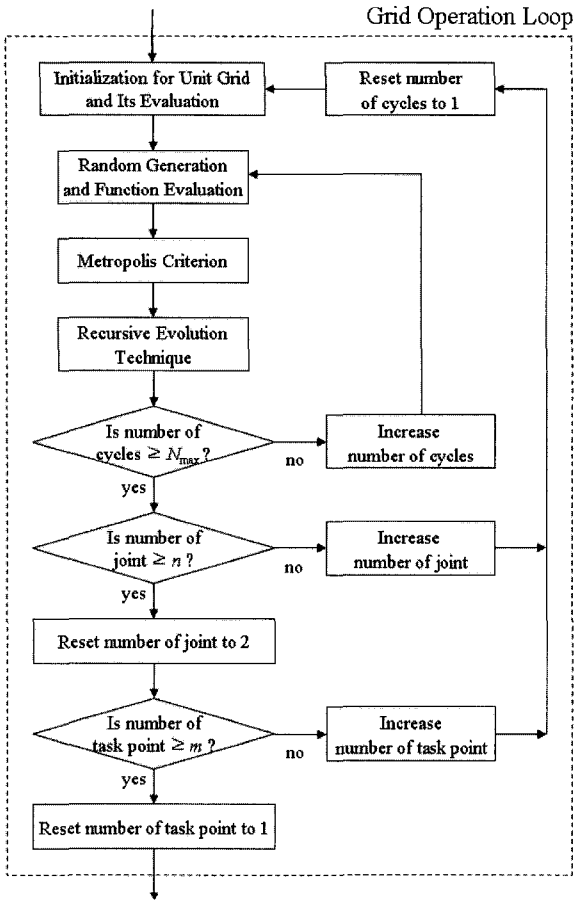


Fig. 5. Detailed algorithm for grid operation loop.

characteristics of the global optimization techniques. For this reason, a convergence constant for a global optimization technique is set to a sufficiently small value (see Section 4.3), which is dependent upon a designer's decision, and when its cost function reaches this constant, we usually accept it as a global optimal solution. Using a global optimization technique to obtain a global optimal solution, you can find that it takes reasonable efforts for its cost function to reach a somewhat small value close to the convergence constant. In order to reach the convergence constant beyond the small value, however, tremendous efforts should be made due to its random generation characteristics. Considering such aspects, it is more reasonable to use a global optimization technique for searching only a region where a global optimal solution may exist, and to use a local optimization technique in order to find a global optimal solution within the region.

4. ADAPTIVE WEIGHTS TUNING ALGORITHM FOR GRID METHOD

In order to actually obtain an optimal solution by applying the Grid Method and the optimization technique, we should determine the weights included in each constraint function for the cost function of (4). To this end, we had proposed a manual weights tuning algorithm using the constant weights for the Grid Method [2]. This manual weights tuning algorithm, however, requires great time and effort in determining especially, the weight for *DM*. In addition, it is defective in that its constant weights cause inefficiency in finding an optimum. To remedy these problems, we newly propose the normalization of the cost function and an adaptive weights tuning algorithm. To begin with, we describe the normalization of the cost function.

4.1. Normalization of cost function

As seen from Table 1, the constraint functions for the cost function of (4) can be classified into the following: functions of length (distance) variables, functions of angle variables and functions of both variables. Here, the angle variables in the unit of radian vary within $-\pi \le (\theta \text{ or } \alpha) \le \pi$, while the length variables in cm usually vary within a relatively large magnitude of the order of 10^2. Therefore, considering that each constraint function of the cost function has the function type of squaring these variables, we can easily confirm that non-homogeneity in the physical units of the variables causes the large differences in magnitude of variation between the functions of each variable. Such imbalance makes it difficult for each constraint function to participate equally in the optimization of the cost function. To cope with this problem, we normalize all the variables, thereby keeping them all within a range of 0 to 1. In this way, we present a normalization procedure for *LC*. The normalized variables and functions for each constraint function are given in Appendix A.

LC before normalization is as follows: to limit DH parameter φ $(=l,d,\alpha \text{ or } \theta)$ to $\varphi_{\min} \le \varphi \le \varphi_{\max}$, we use the following constraint [2].

$$w_{LC} * f_{LC} = \begin{cases} w_{LC} * (\varphi - \varphi_{\max})^2 & \text{if } \varphi > \varphi_{\max} \\ w_{LC} * (\varphi - \varphi_{\min})^2 & \text{if } \varphi < \varphi_{\min} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

Table 1. Classification of each constraint function included in cost function.

variables	constraint functions
length (distance)	Terms related to l, d among *EC*; *DOC*; *LC* on l, d; *DC*; *OA*
angle	Terms related to α, θ among *EC*; *LC* on α, θ; *JAC*
length & angle	*DM*

Here, if we normalize the variable φ ($\varphi_{\min} \le \varphi \le \varphi_{\max}$) as follows,

$$\varphi^* = \frac{\varphi - \varphi_{\min}}{\varphi_{\max} - \varphi_{\min}} \quad \text{where} \quad 0 \le \varphi^* \le 1 \tag{17}$$

then the constraint function in the case of $\varphi > \varphi_{\max} (= \varphi^* > 1)$ in (16) can be transformed as follows:

$$\begin{aligned} & w_{LC}(\varphi - \varphi_{\max})^2 \\ & = w_{LC}\left((\varphi_{\max} - \varphi_{\min}) \cdot \frac{\varphi - \varphi_{\min} - (\varphi_{\max} - \varphi_{\min})}{\varphi_{\max} - \varphi_{\min}}\right)^2 \\ & = w^*_{LC}(\varphi^* - 1) \quad \text{where} \quad w^*_{LC} = w_{LC}(\varphi_{\max} - \varphi_{\min})^2. \end{aligned} \tag{18}$$

Developing the case of $\varphi < \varphi_{\min} (= \varphi^* < 0)$ in the same way, we can obtain the normalized function of (16) as follows:

$$w^*_{LC} * f^*_{LC} = \begin{cases} w^*_{LC} * (\varphi^* - 1)^2 & \text{if } \varphi^* > 1 \\ w^*_{LC} * (\varphi^*)^2 & \text{if } \varphi^* < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

All the constraint functions, except *DM*, can be normalized in the same manner.

As seen from Table 1, because *DM* has a complicated function form that includes both kinds of variables, it is difficult to normalize *DM* in the same way as *LC*. Observing carefully the normalized constraint functions, except *DM*, we can divide them into the following two classes:

A kind of equality constraint function that in itself varies between 0 and 1, and that has zero as its optimum; *EC* and *DOC* belong to this class.

A kind of inequality constraint function that, when its normalized variable varies between 0 and 1, has zero as its optimum and that, when its normalized variable deviates from the range of 0 to 1, limits the search space by increasing its magnitude in proportion to the square of the deviation; *LC*, *DC*, *OA*, and *JAC* belong to this class.

Of these two classes, *DM* belongs to the former, because *DM* is a measure with a specific value representing the kinematic performance of a robot, and is not the kind of inequality constraint that limits the search space to a certain range. Therefore, for our research, we intended to choose a normalized *DM*, which has zero as its optimum and varies between 0 and 1, as in the former constraint functions. It is the Layout Conditioning $\kappa_L(\bar{\mathbf{J}})$ [17] using the normalized Jacobian $\bar{\mathbf{J}}$ that totally satisfies such requirements among various dexterity measures. However, because $\kappa_L(\bar{\mathbf{J}})$ has one as its optimum and $1 \le \kappa_L(\bar{\mathbf{J}}) \le \infty$, we transform it into a minimization measure that has zero as its optimum and that varies between 0 and 1 as follows:

$$\begin{aligned} & (\text{isotropy}) 1 \le \kappa_L(\bar{\mathbf{J}}) \le \infty (\text{singularity}) \\ & \Rightarrow (\text{singularity}) 0 \le \frac{1}{\kappa_L(\bar{\mathbf{J}})} \le 1 (\text{isotropy}) \\ & \Rightarrow (\text{isotropy}) 0 \le 1 - \frac{1}{\kappa_L(\bar{\mathbf{J}})} \le 1 (\text{singularity}) \\ & \therefore f^*_{DM} = \left(1 - \frac{1}{\kappa_L(\bar{\mathbf{J}})}\right)^2. \end{aligned} \tag{20}$$

4.2. Development of adaptive weights tuning algorithm

Through the above normalization of the variables included in the constraint functions, we were able to transform their variation ranges into a range of 0 to 1. However, even so, the actual variation ranges of the normalized constraint functions can differ, unlike those of normalized variables. As stated in the previous section, the constraint functions belonging to a kind of equality constraint function always vary between 0 and 1, while those of an inequality constraint function are proportional to the square of the deviation of their normalized variables from a range between 0 and 1. In addition, even if all the constraint functions have a variation range between 0 and 1, there can still be a relatively considerable difference, as much as between 0.00001 and 0.9, between each constraint function. The manual weights tuning algorithm in our previous research could not help but have a limited effect on balancing such differences between each term varying at every iteration, because all the weights were initially determined as fixed values according to initial conditions in order to achieve a rough balance between each term. Therefore, for equal participation of each term in the optimization of the cost function, we need an adaptive weights tuning algorithm to adaptively maintain the balance between each term according to current conditions at every iteration.

The basic idea for the adaptive weights tuning algorithm that we propose in this paper is as follows:

All of the normalized constraint functions are minimization measures that have zero as their optimum. Therefore, if a constraint function is much larger than others due to its insufficient minimization, the adaptive weights tuning algorithm, in order to keep the balance between each term, gives priority to the minimization of this constraint function, thereby making this constraint function similar in magnitude to others. To this end, the weight corresponding to this

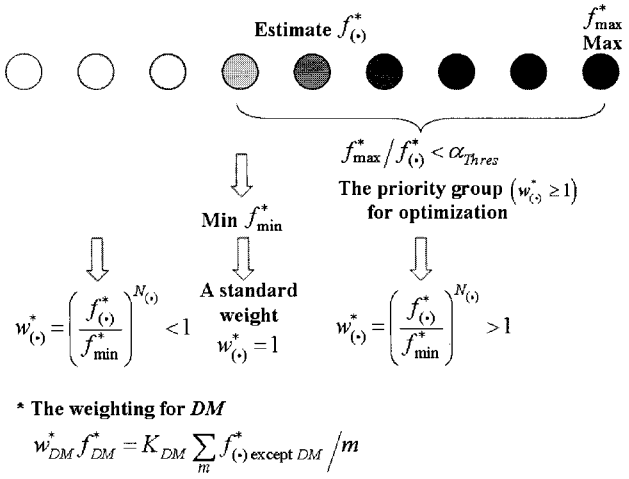


Fig. 6. Conceptual description for adaptive weights tuning algorithm.

constraint function should be determined as a relatively larger value than other weights. Then the optimization technique will minimize this constraint function preferentially over other functions. To conclude, we estimate each constraint function at every iteration, choosing a priority group on the basis of such estimates. Then, by determining the weights for the priority group as relatively large values, we can eventually achieve a balance between each term of the cost function. Fig. 6 is a conceptual description for the adaptive weights tuning algorithm we propose in this paper. Here, the row of circles represents the normalized constraint functions, with the exception of *DM*. The colors of the circles represent their magnitude; the deeper the color, the larger the estimated magnitude. Notice that, for clarity, we sorted the circles by the magnitude of the normalized constraint functions.

Now, we will give a detailed explanation of each step in the adaptive weights tuning algorithm:

(i) All the weights are initially set to 1.

(ii) Grid Operations are consecutively executed on all the unit grids by using the determined weights.

(iii) In order to determine the weights for achieving balance between the terms, we should first know the current magnitude of each constraint function. Notice here that the weights are determined as identical values for all the unit grids, not for only a single unit grid. For this reason, the magnitude of each constraint function should be obtained as the total sum of cost functions of all the unit grids as follows:

$$f^*_{(\cdot)} = \sum_j \sum_i f^*_{(\cdot)(i,j)}. \tag{21}$$

(iv) Unlike the case of the manual weights tuning algorithm, *DM* in the adaptive weights tuning algorithm is also a minimization measure that has zero as its optimum and that varies between 0 and 1. However, *DM* is a measure that should be minimized as far as possible, though not necessarily to 0, while other constraint functions must be minimized to 0. Moreover, in actual applications, it is very difficult for *DM*, as well as for all other constraint functions, to simultaneously converge to 0. Usually the obtained optimum has a DM that is significantly larger than 0. Therefore, as in the manual weights tuning algorithm, we should determine the weight for *DM* to balance *DM* terms with other terms, after determining the weights for all constraint functions other than *DM*. This will later be explained in detail. As stated above, the weight for *DM* is determined differently than that of other constraint functions. In this step, therefore, we obtain only a maximum $f^*_{\max}$ among constraint functions, excepting f^*_{DM}. This value will be used in $f^*_{\max}/f^*_{(\cdot)} < \alpha_{Thres}$ of the step (v) to select a priority group, which is a set of constraint functions that have priority over other ones in terms of participation in the optimization of the cost function.

(v) When selecting a priority group for optimization, we exclude the constraint functions, which already have sufficiently small values owing to ample optimization, by using the inequality of $f^*_{\max}/f^*_{(\cdot)} < \alpha_{Thres}$, and then obtain a minimum $f^*_{\min}$ among the priority group. Here, the larger the value of α_{Thres}, the more constraint functions can be included in the priority group. In accordance with our experience, α_{Thres} is set to a value between 100 and 1000.

(vi) The priority group obtained from the steps above, that is, the constraint functions that have values between $f^*_{\min}$ and $f^*_{\max}$ should take precedence over all other constraint functions in the minimization of a total cost function. Our basic weighting formula for this purpose is as follows:

$$w^*_{(\bullet)} = \frac{f^*_{(\bullet)}}{f^*_{\min}}. \tag{22}$$

Let us describe in detail the formula above. Using this weighting formula, we set the weight of the minimum $f^*_{\min}$ among the priority group to 1. On the basis of this value, the weights of more than 1 are assigned to the priority group, and those of less than 1 are assigned to the constraint functions excluded from the priority

group. Moreover, as the magnitude of the constraint function grows larger compared with its criterion $f^*_{\min}$, this weighting formula assigns a larger value to its corresponding weight, thereby making the minimization of the constraint function precede those of other functions.

Furthermore, we additionally introduced the concept of priority between each constraint function into the basic weighting formula. That is to say, to increase the effectiveness of the weighting scheme according to the importance of each constraint function, we modified the weighting formula of (22) by introducing the concept of priority as follows:

$$w^*_{(\bullet)} = \left(\frac{f^*_{(\bullet)}}{f^*_{\min}} \right)^{N_{(\bullet)}} . \tag{23}$$

Here, $N_{(\bullet)}$ is a priority index given according to the priority of each constraint function. As seen from (23), a constraint function that has higher priority therefore has a larger priority index than other functions, which increases the effectiveness of the weight for the constraint function. However, even so, the criterion weight of $f^*_{\min}$ is still 1, and it is independent of the priority index $N_{(\bullet)}$.

Table 2 shows the priority levels, the priority indices $N_{(\bullet)}$, and their corresponding constraint functions, as obtained from our experience. However, the classification of each constraint function by priority can differ according to the characteristics of a given design problem. Therefore, a designer should determine it so as to be suitable for the characteristics of each individual design problem.

(vii) Finally, we state the weighting scheme for *DM*. As previously mentioned, unlike other constraint functions, it is not easy for *DM* to reach the optimal value of 0, and it usually has a relatively large value compared with others. In consideration of such characteristics, the weighting method for the *DM* proposed in our research determines the weight for *DM* at every iteration in order to appropriately balance the magnitude of *DM* term with those of other terms chosen as the priority group for optimization. To this end, through the following weighting formula, we determined the weight for *DM* so that *DM* term can be the average level of all other terms.

$$\begin{aligned} w^*_{DM} f^*_{DM} &= K_{DM} \sum_m f^*_{(\bullet) \text{ except } DM} \Big/ m \\ \therefore w^*_{DM} &= K_{DM} \sum_m f^*_{(\bullet) \text{ except } DM} \Big/ \left(m * f^*_{DM} \right) \end{aligned} \tag{24}$$

Here, K_{DM} is a constant for determining the weight for *DM* on the basis of the average of all other constraint functions. Also, K_{DM} should be chosen as a value that optimizes *DM* the most and that simultaneously causes all other constraint functions to converge to zero. K_{DM} is tuned from the starting value of '1' and, in our experience, is usually set to a value between 1 and 20.

(viii) If all the weights are determined through the above procedure, return to (ii) and repeatedly execute the procedure above.

Table 2. Priority level, priority index and corresponding constraints.

Priority Level	$N_{(\bullet)}$	Constraints
1	3	*LC*
2	2	*EC* for twist angle; *DOC*
3	1	*EC* for link length and offset; *DC*; *JAC*
4	0.5	*OA*

4.3. How to determine the convergence criterion

In order to judge whether an optimal solution has been obtained or not, we have adopted a total cost function, which is the total sum of cost functions of all the unit grids, as follows:

$$\begin{aligned} f_{total} &= \sum_j \sum_i f_{UG}(\mathbf{x}_{i,j}), \text{ where} \\ f_{UG}(\mathbf{x}_{i,j}) &= w_{EC} * f_{EC(i,j)} + w_{DOC} * f_{DOC(i,j)} \\ &\quad + w_{LC} * f_{LC(i,j)} + w_{DC} * f_{DC(i,j)} \\ &\quad + w_{OA} * f_{OA(i,j)} + w_{DM} * f_{DM(i,j)} \\ &\quad + w_{JAC} * f_{JAC(i,j)}. \end{aligned} \tag{25}$$

Close inspection of $f_{UG}(\mathbf{x}_{i,j})$ reveals that every constituent normalized constraint function is a minimization measure having zero as its global minimum. Hence, when f_{total} converges to a value close to zero (usually less than 10^{-5}, in our experience), we can conclude that an optimal solution has been obtained.

5. APPLICATIONS

In this section, we apply our proposed design approach to a 2-d.o.f. planar manipulator design and a PUMA-type spatial robot design.

5.1. Design of a 2-d.o.f. planar manipulator

A 2-d.o.f. planar manipulator was designed, the tip positions of which are given for seven task points as follows: (-10,40), (5, 35), (10, 40), (12, 42), (17, 40), (22, 35), (30, 30). The robot is to avoid two sphere-type obstacles, which have a radius of 5 and a safety factor of 3. Under such conditions, we applied the local optimization technique of GRG and the global optimization technique of VFSA, respectively. Through these applications, we intend to verify if the global optimization technique can always find the global optimum regardless of initial postures, unlike local optimization techniques that are sensitive to initial postures. Kinematic requirements for this design are as follows: *EC*, *LC* of $0 \le l_i \le 30$ $(i = 1, 2)$, *DC*, *OA*, *DM*, *JAC* of $\Delta\theta_{\max,i} = 60°$ $(i = 1, 2)$. The global optimum of this design can be easily obtained as $l_1 = l_2 = 30$.

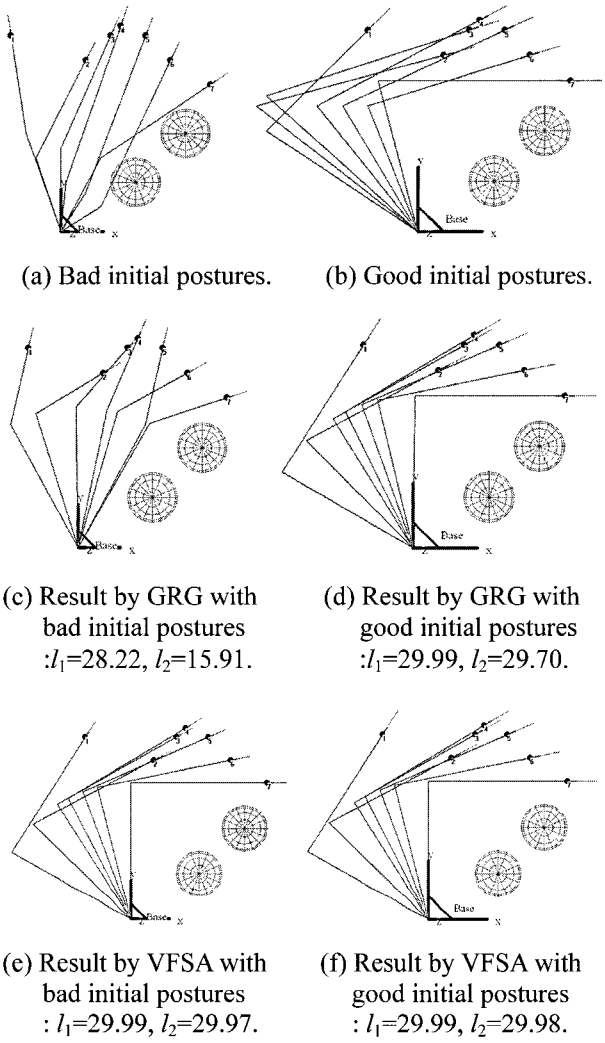
The parameter values for the whole design procedure in Section 3.2 are as follows:

For the whole design procedure in Section 3.2,
- number of stepsize adjustments : 50
- number of temperature annealing : 100
- number of trials for Recursive Evolution Technique : 100
- reduction rate α of stepsize adjustments : 0.99.

For Grid Operation Loop in Fig. 5,
- number of cycles : 100
- number of trials for Recursive Evolution Technique : 25.

For objective comparison, we applied the adaptive weights tuning algorithm to both cases. The priority level of each constraint function for the adaptive weights tuning algorithm and its corresponding index $N_{(\bullet)}$ are given in Table 2, and the constant K_{DM} for weight selection of *DM* was determined as 15.

In Fig. 7(a), we gave bad initial postures that were far from the global optimum, while Fig. 7(b) shows good initial postures that were relatively close to the global optimum. Under such initial postures, Fig. 7(c) and 7(d) show the design results obtained by using GRG in each case, and Fig. 7(e) and 7f show those obtained by using VFSA. The results show that GRG finds only a local optimum in the case of bad initial postures, although it can find an optimum very close to the global optimum in the case of good initial postures. As seen from Fig. 7(e) and 7(f), however, the global optimization technique of VFSA always finds an optimum very close to the global optimum regardless of initial postures. The above results verify that our proposed design approach can be applied to actual design examples well and, moreover, that a global optimization technique should be used in order to get the global optimum regardless of initial postures.



(a) Bad initial postures. (b) Good initial postures.

(c) Result by GRG with bad initial postures :l_1=28.22, l_2=15.91.

(d) Result by GRG with good initial postures :l_1=29.99, l_2=29.70.

(e) Result by VFSA with bad initial postures : l_1=29.99, l_2=29.97.

(f) Result by VFSA with good initial postures : l_1=29.99, l_2=29.98.

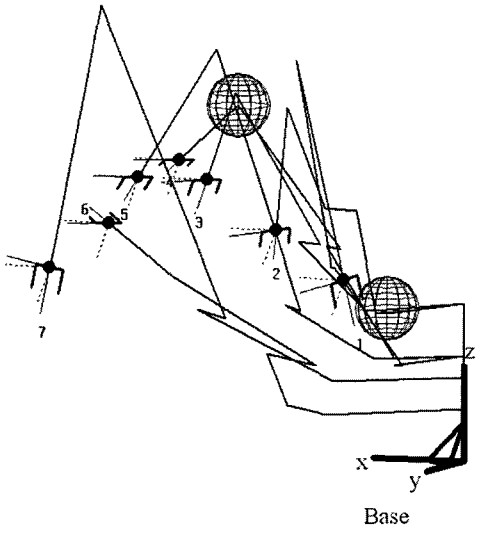
Fig 7. Design results for 2-d.o.f. planar manipulator.

5.2. Design of a 6-d.o.f. spatial manipulator with a spherical wrist

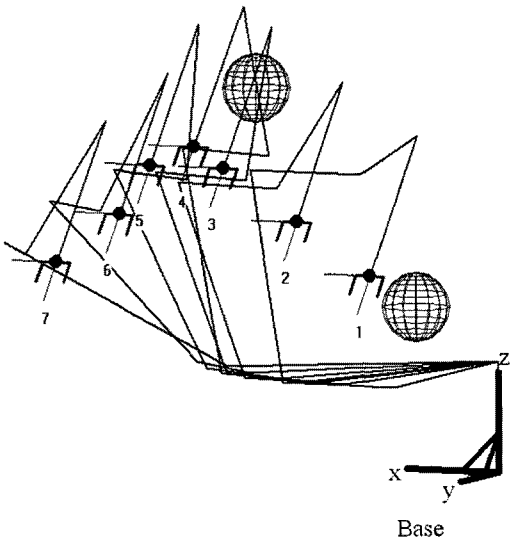
A 6-d.o.f. spatial manipulator with a spherical wrist was designed, the tip positions and orientations of which are given for seven task points in Table 3. Here, desired orientations are represented by Z-Y-X Euler angles. The twist angles of the robot are given as $\boldsymbol{\alpha} = [-90°, 0°, -90°, 90°, -90°, 0°]$. The robot is to

Table 3. Positions and orientations of given task points.

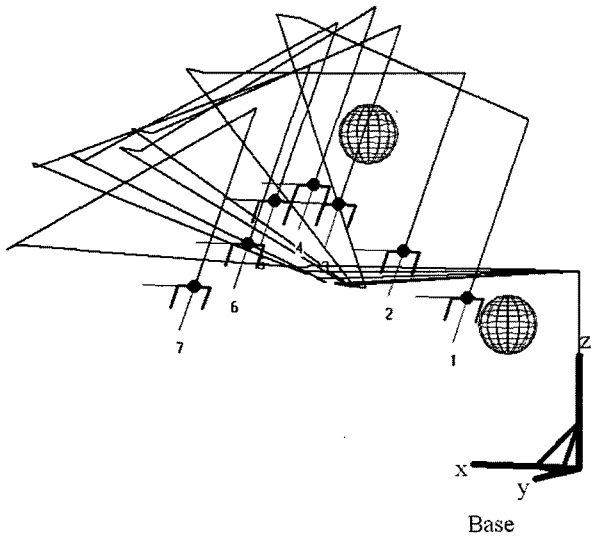
Task point	Position	Desired Orientation
1	(0, 30, 20)	(0°, 0°, −135°)
2	(5, 35, 25)	(0°, 0°, −135°)
3	(10, 40, 30)	(0°, 0°, −135°)
4	(12, 42, 32)	(0°, 0°, −135°)
5	(17, 40, 30)	(0°, 0°, −135°)
6	(22, 35, 25)	(0°, 0°, −135°)
7	(30, 30, 20)	(0°, 0°, −135°)



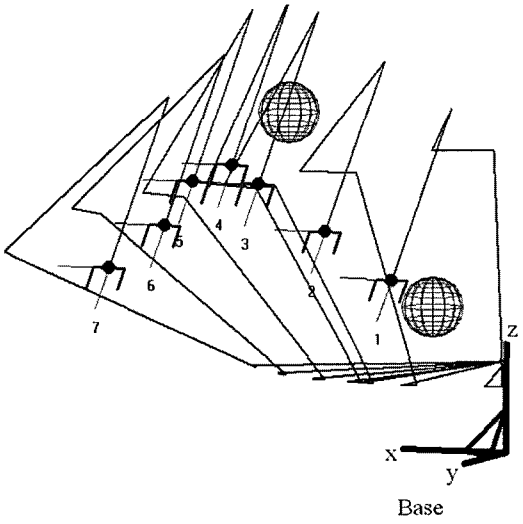
(a) Initial postures.



(b) Local optimal design by GRG using manual tuning.



(c) Local optimal design by GRG using adaptive weights tuning.



(d) Global optimal design by VFSA using adaptive weights tuning.

Fig. 8. Results of optimal design for 6-d.o.f. spatial manipulator with spherical wrist.

avoid two sphere-type obstacles, which have a radius of 3 and a safety factor of 2. Kinematic requirements for this design are as follows: *EC*, *DOC*, *LC* of $0 \le l_i \le 30, -30 \le d_i \le 30$ $(i = 1,2,\cdots,6)$, *DC*, *OA*, *DM*, *JAC* of $\Delta\theta_{\max,i} = 60°$ $(i = 1,2,\cdots,6)$.

5.2.1 Verification of the efficiency of the adaptive weights tuning algorithm

In this subsection, we apply the manual weights tuning algorithm, which was proposed in our previous research, and the newly proposed adaptive weights tuning algorithm to the problem above, respectively, and through comparison of these results, confirm the efficiency of the adaptive weights tuning algorithm. For an objective comparison of the two tuning algorithms, we chose GRG as an optimization technique due to the random generation characteristic of VFSA. In the case of the manual weights tuning algorithm, we followed the procedure proposed in our previous research [2], and determined the weight w_{DM} for *DM* as 0.02 through its best tuning. A priority level of each constraint function for the adaptive weights tuning algorithm and its corresponding index $N_{(\bullet)}$ are given in Table 2, and the constant K_{DM} for weight selection of DM is determined as 15.

Fig. 8(b) and 8(c) are, respectively, the design results obtained by applying the manual weights tuning algorithm and the adaptive weights tuning algorithm to the initial postures of Fig. 8(a). In the

case of the manual weights tuning algorithm, it took ten trials to choose the weight for *DM* through its best tuning, and then 2763 iterations were required to get an optimum using the chosen weight; in the case of the adaptive weights tuning algorithm, only four trials and 1531 iterations were necessary. In the manual weights tuning algorithm the starting value is not given, so its value should be gradually increased, starting from a very small value, resulting in the necessity for so many trials and iterations. The adaptive weights tuning algorithm, on the other hand, requires fewer trials than the manual weights tuning algorithm in choosing the weight for *DM* because the starting value for the weight tuning of *DM* is given as 1. Therefore, these results show that the adaptive weights tuning algorithm is more efficient than the manual weights tuning algorithm.

5.2.2 Design results for global optimal kinematics

In this subsection, we design the global optimal kinematics of the 6-d.o.f. spatial manipulator with a spherical wrist by using the whole design procedure in Section 3.2 and the adaptive weights tuning algorithm, and compare it with the design result obtained using the local optimization technique of GRG in the previous subsection.

The parameter values for the whole design procedure in Section 3.2 are as follows:

For the whole design procedure in Section 3.2,
- number of stepsize adjustments : 50
- number of temperature annealing : 100
- number of trials for Recursive Evolution Technique : 100
- reduction rate α of stepsize adjustments : 0.99.

For Grid Operation Loop in Fig. 5,
- number of cycles : 100
- number of trials for Recursive Evolution Technique : 25.

The priority level of each constraint function for the adaptive weights tuning algorithm and its corresponding index $N_{(\bullet)}$ are given in Table 2, and the constant K_{DM} for weight selection of *DM* was determined as 15.

In Table 4, we have listed various performance measures for the initial postures and the design results by VFSA/GRG. Here, the result of Table 4(c) is a numerical representation of Fig. 8(c) obtained by using GRG and the adaptive weights tuning algorithm in the previous subsection and, as stated before, is given for comparison with the result using VFSA. All the performance measures in Table 4 are given as normalized values except *EC*; this is done for easy decision of *EC*'s convergence, because the normalized value of *EC*, even before the optimization procedure is performed, may be too small to judge its convergence.

Fig. 8(a) and 8(d) is a graphical representation of Tables 4(a) and 4(b), and shows initial postures and their corresponding design result using the whole design procedure in Section 3.2 and the adaptive weights tuning algorithm. As seen from Table 4, initial link lengths and offsets are not identical, because the user arbitrarily gives initial joint positions: $f_{EC\_total} = 1414 \neq 0$. Through the Grid Method, however, we can get identical link lengths and offsets for every task point; i.e., $f_{EC\_total} = 0.007263 \approx 0$. Careful observation of Fig. 8 leads us to find two types of lines attached to each end-effector. Here, the dotted lines represent the X and Z axes of the desired orientation of each task point, the black circle in the intersection of these lines represents the position of each task point, and the solid lines represent the X and Z axes of the actual coordinate frame of the end-effector at each task point. From this figure, we can see that in the design result each end-effector exactly indicates the desired orientation, while in the initial postures none of the end-effectors do. This fact can be confirmed from the following: $f^*_{DOC\_total} = 0.8654 \Rightarrow 4.1e-7$. Moreover, we achieve obstacle avoidance from the initial postures that collide with two obstacles. In addition, this result is optimized from the viewpoint of *DM*, as seen from $f^*_{DM\_total} = 6.6662 \Rightarrow 5.0500$, which is more optimized than the result obtained using GRG of $f^*_{DM\_total} = 6.6662 \Rightarrow 5.2968$, and the total cost function is also minimized as follows: $f^*_{total} = 74970031 \Rightarrow 9.7e-6$. Fig. 9 shows the transient variation of DH parameters in the case of Fig. 8(d) as the iteration goes on. The left side of Fig. 9 denotes the transient variation of the average of each DH parameter, and the right side denotes that of its standard deviation. From this figure, you can see that

Table 4. Results optimal design for 6-d.o.f. spatial manipulator with spherical wrist.

function	value		
	(a) Initials	(b) VFSA	(c) GRG
f_{EC_total}	1414	0.007263	0.001981
$f^*_{DOC\_total}$	0.8654	4.1e−7	7.3e−8
$f^*_{LC\_total}$	0	2.5e-8	0
$f^*_{DC\_total}$	6.8004	0	0
$f^*_{OA\_total}$	74970014	1.3e-6	0
$f^*_{DM\_total}$	6.6662	5.0500	5.2968
$f^*_{JAC\_total}$	0.8996	0	0
f^*_{total}	74970031	9.7e−6	9.0e−6

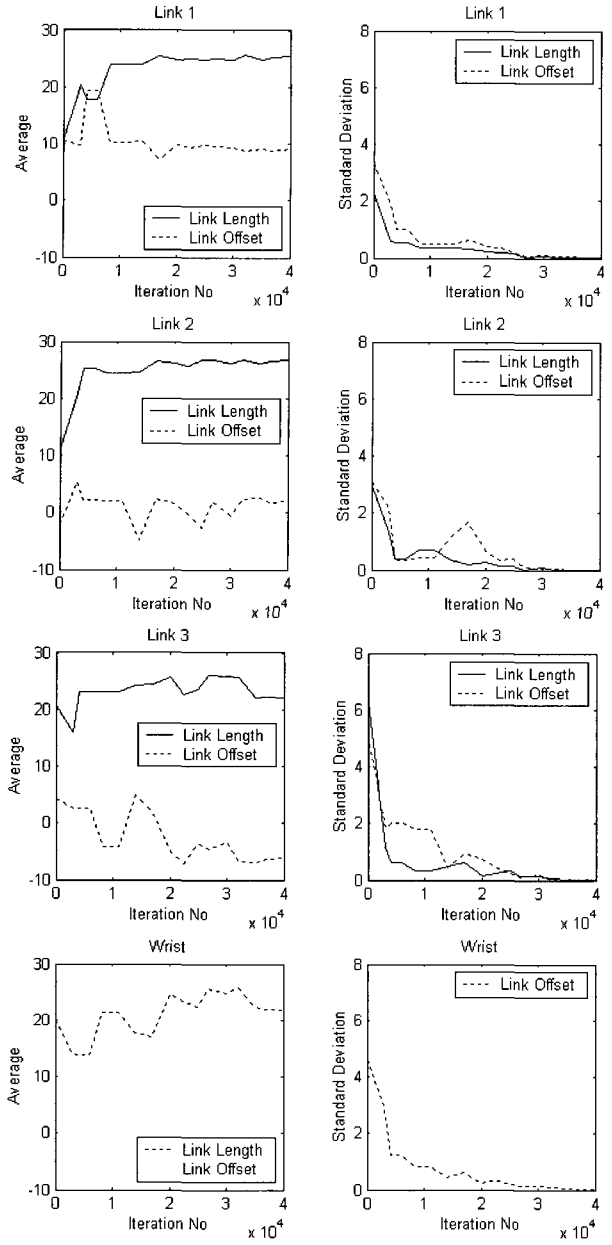


Fig. 9. Transient variation of DH parameters in the case of Fig. 8(d).

as the iteration goes on, the average of DH each parameter converges on a value and its standard deviation goes to zero, which means that VFSA and the adaptive weights tuning algorithm proposed in this paper work well with the Grid Method.

6. CONCLUSIONS

In this paper, we presented a simple, systematic, effective design approach for robot optimal kinematics, one that is composed of a problem formulation using the Grid Method, VFSA as a global optimization technique, and the adaptive weights tuning algorithm for the Grid Method. Therefore, if we input initial postures for every task point to the computerized program for these algorithms, the program automatically produces a global optimum of robot kinematics. By virtue of this, even a novice designer can easily design robot kinematics and, furthermore, we expect that the necessary development period for the robot will be shortened.

APPENDIX A: COST FUNCTION

A.1. *EC*

EC before normalization [2] is as follows:

$$f_{EC(i,j)} = f_{L(i-1,j)} + f_{L(i,j)} + w_{ang} * f_{\alpha(i-1,j)}, \quad (26)$$

where

$$f_{L(k,j)} = (l_{k,j} - l_{k,j-1})^2 + (l_{k,j} - l_{k,j+1})^2 + \begin{cases} (d_{k,j} - d_{k,j-1})^2 + (d_{k,j} - d_{k,j+1})^2 \\ \qquad \text{for revolute joint } k \\ w_{ang} * \left((\theta_{k,j} - \theta_{k,j-1})^2 + (\theta_{k,j} - \theta_{k,j+1})^2 \right) \\ \qquad \text{for prismatic joint } k, \end{cases}$$

$$f_{\alpha(i-1,j)} = (\alpha_{i-1,j} - \alpha_{i-1,j-1})^2 + (\alpha_{i-1,j} - \alpha_{i-1,j+1})^2.$$

Here, w_{ang} denotes a weight for equalizing the magnitude order of joint angles with that of link lengths and offsets, $f_{L(i-1,j)}$ is a cost function concerning a lower link of the unit grid, $f_{L(i,j)}$ is concerning an upper link of the unit grid and $f_{\alpha(i-1,j)}$ is concerning twist angles of the unit grid.

If $\varphi \left(\varphi_{\min} \le \varphi \le \varphi_{\max} \right)$ is normalized as follows,

$$\varphi^* = \frac{\varphi - \varphi_{\min}}{\varphi_{\max} - \varphi_{\min}}, \quad \text{where } 0 \le \varphi^* \le 1. \quad (27)$$

(26) is normalized as follows.

$$f^*_{EC(i,j)} = f^*_{L(i-1,j)} + f^*_{L(i,j)} + w^*_{ang} * f^*_{\alpha(i-1,j)}, \quad (28)$$

where

$$f^*_{L(k,j)} = (l^*_{k,j} - l^*_{k,j-1})^2 + (l^*_{k,j} - l^*_{k,j+1})^2 + \begin{cases} (d^*_{k,j} - d^*_{k,j-1})^2 + (d^*_{k,j} - d^*_{k,j+1})^2 \\ \qquad \text{for revolute joint } k \\ w^*_{ang} * \left((\theta^*_{k,j} - \theta^*_{k,j-1})^2 + (\theta^*_{k,j} - \theta^*_{k,j+1})^2 \right) \\ \qquad \text{for prismatic joint } k, \end{cases}$$

$$f^*_{\alpha(i-1,j)} = (\alpha^*_{i-1,j} - \alpha^*_{i-1,j-1})^2 + (\alpha^*_{i-1,j} - \alpha^*_{i-1,j+1})^2.$$

A.2. *DOC*

DOC before normalization [2] is as follows:

$$f_{DOC(i,j)} = \left\|\bar{\mathbf{x}}_d - \bar{\mathbf{x}}_{ee}\right\|^2 + \left\|\bar{\mathbf{z}}_d - \bar{\mathbf{z}}_{ee}\right\|^2. \quad (29)$$

Here, $\bar{\mathbf{x}}_d$ and $\bar{\mathbf{z}}_d$ denote the unit vectors representing the X and Z axes of the desired orientation, respectively, while $\bar{\mathbf{x}}_{ee}$, $\bar{\mathbf{z}}_{ee}$ denote those of the end-effector's coordinate frame. This constraint should be used for Grid Operation in the unit grid related to the robot's d.o.f. for determining the desired orientation.

Here, $0 \le \left\|\bar{\mathbf{x}}_d - \bar{\mathbf{x}}_{ee}\right\| \le 2$ and $0 \le \left\|\bar{\mathbf{z}}_d - \bar{\mathbf{z}}_{ee}\right\|^2 \le 2$, because $\bar{\mathbf{x}}_d$, $\bar{\mathbf{z}}_d$, $\bar{\mathbf{x}}_{ee}$ and $\bar{\mathbf{z}}_{ee}$ are all unit vectors. Hence, if we determine the normalized variables as follows,

$$d_X^* = \frac{\left\|\bar{\mathbf{x}}_d - \bar{\mathbf{x}}_{ee}\right\|}{2}, \quad d_Z^* = \frac{\left\|\bar{\mathbf{z}}_d - \bar{\mathbf{z}}_{ee}\right\|}{2} \quad (30)$$
$$(0 \le d_X^* \le 1,\ 0 \le d_Z^* \le 1).$$

(29) can be normalized thus:

$$f_{DOC(i,j)}^* = (d_X^*)^2 + (d_Z^*)^2. \quad (31)$$

A.3. *LC*

The normalized variables and functions for *LC* are as follows:

$$f_{LC(i,j)}^* = \begin{cases} (\varphi^* - 1)^2 & \text{if } \varphi^* > 1 \\ (\varphi^*)^2 & \text{if } \varphi^* < 0, \\ 0 & \text{otherwise,} \end{cases} \quad (32)$$

$$\text{where} \quad \varphi^* = \frac{\varphi - \varphi_{\min}}{\varphi_{\max} - \varphi_{\min}} (0 \le \varphi^* \le 1).$$

A.4. *DC* [8]

The normalized variables and functions for *DC* are as follows:

$$f_{DC(i,j)}^* = \begin{cases} (L_{i,j}^* - 1)^2 & \text{if } L_{i,j}^* > 1 \\ 0 & \text{otherwise,} \end{cases} \quad (33)$$

$$\text{where} \quad L_{i,j}^* = \frac{L_{i,j}}{L_{i-1,j}} (0 \le L_{i,j}^* \le 1).$$

A.5. *OA*

The normalized variables and functions for *OA* are as follows:

$$f_{OA(i,j)}^* = \begin{cases} (d_c^* - 1)^2 & \text{if } d_c^* > 1 \ \&\ \text{collision} \\ 0 & \text{otherwise,} \end{cases} \quad (34)$$

$$\text{where} \quad d_c^* = \frac{R_s}{d_c} (0 \le d_c^* \le 1).$$

Here, R_s s the radius of a sphere-type obstacle and d_c is the distance from the sphere center to a link.

A.6. *DM*

As stated in Section 4.1, we use the following function as a normalized *DM*.

$$f_{DM(i,j)}^* = \left(1 - \frac{1}{\kappa_L(\bar{\mathbf{J}})}\right)^2, \quad (35)$$

where $\kappa_L(\bar{\mathbf{J}})$ denotes the Layout Conditioning [17].

A.7. *JAC*

$\left|\theta_{i,j} - \theta_{i,j-1}\right| \le \Delta\theta_{\max,i}$, $\left|\theta_{i,j} - \theta_{i,j+1}\right| \le \Delta\theta_{\max,i}$ of *JAC* can be transformed into a form of *LC*. For example, $\left|\theta_{i,j} - \theta_{i,j-1}\right| \le \Delta\theta_{\max,i}$ can be transformed into the following:

$$\theta_{i,j-1} - \Delta\theta_{\max,i} \le \theta_{i,j} \le \theta_{i,j-1} + \Delta\theta_{\max,i}. \quad (36)$$

As with *LC*, therefore, if we determine the normalized variables as follows,

$$\theta_{i,k}^* = \frac{\theta_{i,j} - \theta_{C\min,(i,k)}}{\varphi_{C\max,(i,k)} - \varphi_{C\min,(i,k)}} \quad (37)$$
$$(0 \le \theta_{i,k}^* \le 1;\ k = j-1 \text{ or } j+1),$$

where $\theta_{C\max,(i,k)} = \theta_{i,k} + \Delta\theta_{\max,i}$, $\theta_{C\min,(i,k)} = \theta_{i,k} - \Delta\theta_{\max,i}$, then *JAC* can be normalized as follows:

$$f_{JAC(i,j)}^* = f'^*_{JAC(i,j-1)} + f'^*_{JAC(i,j+1)}, \quad (38)$$

$$\text{where } f'^*_{JAC(i,k)} = \begin{cases} (\theta_{i,k}^* - 1)^2 & \text{if } \theta_{i,k}^* > 1 \\ (\theta_{i,k}^*)^2 & \text{if } \theta_{i,k}^* < 0 \\ 0 & \text{otherwise.} \end{cases}$$

REFERENCES

[1] P.-H. Chang, J.-Y. Park, and J.-Y. Yang, "Task oriented design of robot kinematics using grid method and its applications to nuclear power plant," *Proc. of Int. Symp. on Artificial Intelligence, Robotics and Human Centered Technology for Nuclear Applications*, pp. 114-123, 2002.

[2] J.-Y. Park, P.-H. Chang, and J.-Y. Yang, "Task oriented design of robot kinematics using the Grid Method," *Advanced Robotics*, vol. 17, no. 9, pp. 879-907, 2003.

[3] C. J. J. Paredis and P. K. Khosla, "Kinematic design of serial link manipulators from task specifications," *International Journal of Robotics Research*, vol. 12, no. 3, pp. 274-287, 1993.

[4] I.-M. Chen and J. W. Burdick, "Determining task

optimal modular robot assembly configurations," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 132-137, 1995.

[5] G. Yang and I.-M. Chen, "Task-based optimization of modular robot configurations: Minimized degree-of-freedom approach," *Mechanism and Machine Theory*, vol. 35, no. 4, pp. 517-540, 2000.

[6] O. Chocron and P. Bidaud, "Evolutionary algorithms in kinematic design of robotic systems," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1111-1117, 1997.

[7] J.-O. Kim and P. K. Khosla, "A formulation for task based design of robot manipulators," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, pp. 2310-2317, 1993.

[8] J.-O. Kim, *Task Based Kinematic Design of Robot Manipulators*, Ph.D. thesis, Carnegie-Mellon University, 1992.

[9] C. Leger, *Automated Synthesis and Optimization of Robot Configurations: An Evolutionary Approach*, Ph.D. thesis, Carnegie-Mellon University, 1999.

[10] C. J. J. Paredis and P. K. Khosla, "An approach for mapping kinematic task specifications into a manipulator design," *Proc. of the Fifth Int. Conf. on Advanced Robotics 'Robots in Unstructured Environments*, vol. 1, pp. 556-561, 1991.

[11] J. Wunderlich and C. Boncelet, "Local optimization of redundant manipulator kinematics within constrained workspaces," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 127-132, 1996.

[12] L. Ingber, "Very fast simulated re-annealing," *Mathematical and Computer Modelling*, vol. 12, no. 8, pp. 967-973, 1989.

[13] B. Rosen, "Function optimization based on advanced simulated annealing," *Proc. of Workshop on Physics and Computation*, pp. 289-293, 1992.

[14] W. S. Jung and N. Z. Cho, "Determination of design alternatives and performance criteria for safety systems in a nuclear power plant via simulated annealing," *Reliability Engineering and System Safety*, vol. 41, no. 1, pp. 71-94, 1993.

[15] R. P. Paul, B. Shimano, and G. E. Mayer, "Kinematic control equations for simple manipulators," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-11, no. 6, pp. 449-455, 1981.

[16] A. Mohri, X. D. Yang, and M. Yamamoto, "Collision free trajectory planning for manipulator using potential function," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 3069-3074, 1995.

[17] F. Ranjbaran, J. Angeles, and A. Kecskemethy, "On the kinematic conditioning of robotic manipulators," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol. 4, pp. 3167-3172, 1996.

[18] J. S. Arora, *Introduction to Optimum Design*, McGraw-Hill, New York, 1989.

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087-1092, 1953.

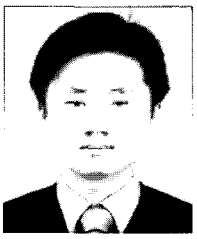
[20] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[21] D. Vanderbilt and S. G. Louie, "A Monte Carlo simulated annealing approach to optimization over continuous variables," *Journal of Computational Physics*, vol. 56, pp. 259-271, 1984.

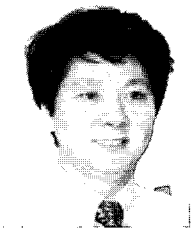
[22] M. Lundy and A. Mees, "Convergence of annealing algorithm," *Mathematical Programming*, vol. 34, pp. 111-124, 1986.

[23] H. H. Szu and R. L. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, pp. 157-162, 1987.

[24] A. D. Belegundu and T. R. Chandrupatla, *Optimization Concepts and Applications in Engineering*, Prentice-Hall, Inc., 1999.



Joon-Young Park received the B.S. degree in Electrical Engineering, in 1995, and the M.S. and Ph.D. degrees in Mechanical Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1997 and 2004, respectively. He is now a Senior Researcher at the Power Generation Laboratory in Korea Electric Power Research Institute (KEPRI). His research interests include robot systems for the electric power industry, robust control of nonlinear systems as well as optimum kinematic design of a robot manipulator.



Pyung-Hun Chang received the B.S. and M.S. degrees in Mechanical Engineering from Seoul National University (SNU) in 1974 and 1977, respectively. He received the Ph.D. degree in Mechanical Engineering from the Massachusetts Institute of Technology (MIT) in 1987. From 1984-1987, he was involved in a research project in the field of robotics as a Research Assistant at the Artificial Intelligence Laboratory of MIT. Since 1987, he has been on the faculty of and is now a Professor at the Department of Mechanical Engineering in KAIST. His research interests include the high accuracy/ speed control with application to mechanical systems, robust control of nonlinear plants such as a robot manipulator and task-oriented design of a robot manipulator.

Jin-Oh Kim received the B.S. and M.S. degrees in Mechanical Engineering from Seoul National University in 1983 and 1985, respectively. He also received the Ph.D. in Robotics from the Robotics Ph.D. Program in School of Computer Science, Carnegie-Mellon University in 1992. He had studied surveillance robot systems at the SECOM Intelligent System Lab in Tokyo, Japan before his joining Samsung Electronics in 1994. He had developed more than 50 industrial robots and robotic systems in Samsung Electronics as a principal researcher and had led a new robot business team as the general manager until 1998. He has joined the Department of Information and Control Engineering in Kwangwoon University since March, 1999. He has also served as the Director of the Korea Intelligent Robot National Initiative Steering Committee since January, 2004. He has joined AI Lab, Stanford University since 2005 as a Visiting Associate Professor. His research interest includes design methodology of robots and robot systems, agile and intelligent robotic manufacturing system as well as robot technology innovation for business.