Self-Learning Control of Cooperative Motion for Humanoid Robots

Yoon-Kwon Hwang, Kook-Jin Choi, and Dae-Sun Hong

Abstract: This paper deals with the problem of self-learning cooperative motion control for the pushing task of a humanoid robot in the sagittal plane. A model with 27 linked rigid bodies is developed to simulate the system dynamics. A simple genetic algorithm (SGA) is used to find the cooperative motion, which is to minimize the total energy consumption for the entire humanoid robot body. And the multi-layer neural network based on backpropagation (BP) is also constructed and applied to generalize parameters, which are obtained from the optimization procedure by SGA, in order to control the system.

Keywords: Cooperative motion, genetic algorithm, humanoid robot, neural network, optimization.

1. INTRODUCTION

Humans often use not only their arms but also their entire bodies to generate efficient manipulation force. Especially, the cooperative motion of the whole body is needed to accomplish heavy work. If the static force is insufficient to move a heavy object, a human accumulates momentum by moving the body and then generating a large impact to it. Accordingly, a human can accomplish a given task successfully by the large manipulation force which surpasses the maximum static frictional force of the object. The application of a humanoid robot rather than a human for a certain work is more useful than an ordinary one whose limbs are mounted on the ground, since its base has the capability of translational and rotational movement along and about all of the three Cartesian axes. Therefore, significant manipulation force can be generated and workspace can be enlarged through position and orientation of its mainbody adjusted by bending, stretching or moving its legs or arms. As a result, a humanoid robot is expected to perform more various and complicated works similar to those done by a human. Fig. 1 shows the typical examples for heavy works such as pushing a wall and twisting a valve. Most of these works are accomplished by proper control of the hand according to the surrounding circumstances. For multi-limbed robots, they experience the coordinative problem of the whole body while carrying out the given tasks.

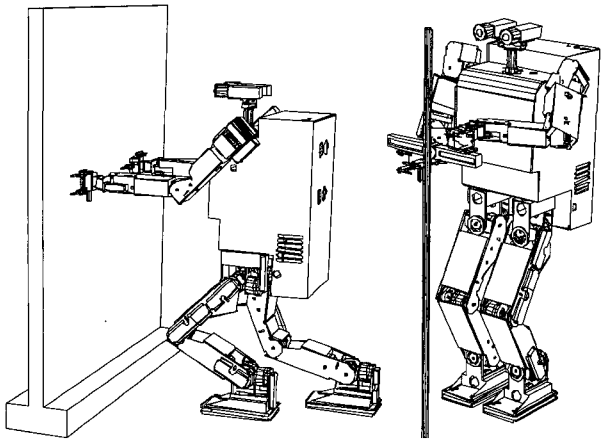


Fig. 1. Typical examples of heavy works which require the whole body cooperative motion; pushing a wall (left) and twisting a valve (right).

The related previous studies have been focused on the manipulability of the robotic mechanism [1,2]. Zheng and Yin have studied an optimal mechanism to coordinate the arm and mainbody configurations by the linear programming technique, in that the moving mainbody is much more expensive with respect to energy consumption [3,4]. Coordination of both the arm and leg was studied to avoid the degeneracy problem and to enlarge the workspace by Su and Zheng [5]. Recently, Yokokohji, Nomoto and Yoshikawa have performed studies to evaluate a given posture by the cost function using the nonlinear programming [6]. A weighted combination of energy consumption and load balancing was selected as a criterion of optimization. Yoshida and co-workers have proposed Jacobian matrices of the arm and leg to

Manuscript received October 19, 2005; revised May 15, 2006; accepted July 11, 2006. Recommended by Editorial Board member Eun Tai Kim under the direction of Editor Jae-Bok Song.

Yoon-Kwon Hwang is with the School of Mechatronics, Changwon National University, 9 Sarimdong, Changwon 641-773, Korea (e-mail: ykhwang@changwon.ac.kr).

Kook-Jin Choi and Dae-Sun Hong are with the Department of Mechanical Design and Manufacturing, Changwon National University, Korea (e-mails: {choi, dshong}@chang won.ac.kr).

give the robot adaptability to force change [7].

The present paper concentrates on how to coordinate the cooperative motion of the whole body for the humanoid robot during the pushing task. The approach taken here has been to search the position and orientation of the mainbody and the position of the hand in order to minimize summation of total joint torques according to the change of manipulation force using the simple genetic algorithm (SGA) [8,9].

Most of the optimization problems that we treat are not only complicated but also insufficient in regards to their reciprocal relationship between variables and *a priori* knowledge for a function. Those methods could be divided into derivative-based method and derivative-free method. The typical examples of the former are gradient method, Newton's method and conjugate gradient method, etc. However, they require much complexity to obtain the solution in calculation, although they could be calculated numerically. Methods belonging to the latter are simplex method, random search, genetic algorithm, genetic programming, evolutionary strategies, and simulated evolution, etc. Among them, the genetic algorithm is the probability based on the optimization method describing genetics and natural evolution numerically. It has been applied to many optimization problems because of the ability to search the global optimum and the convenience of the application. Slow convergence and many iterative computations, however, make it more difficult to apply and require high computational cost. The genetic algorithm has some characteristics distinguished from existing search algorithms. It can obtain a global solution in the multi-modality or complicated, large search environment, because it only requires the value of the objective function. Even if SGA is a primary algorithm proposed by Holland, it has still been applied to numerous optimization problems, due to the robustness in searching the solution. In particular, it is useful to the humanoid robot of nonlinear characteristics in that it is the hyper redundant multi-body system close to 30 degrees of freedom.

For the application of SGA to control the real robot, in the meanwhile, we should consider its disadvantages, which are slow conjunction and instability of solution. Accordingly, it is difficult to apply the results of SGA in the real time control of a robot. Therefore, we will introduce additionally the perceptron learning law of the neural network in order to overcome this limitation. Its performance learning is presented by generalization of the algorithm. This generalization, called backpropagation (BP), is used to train the multi-layer network [10]. This network is used to generalize parameters, which are the results obtained by SGA, as a training set in this paper.

2. NUMERICAL MODELING

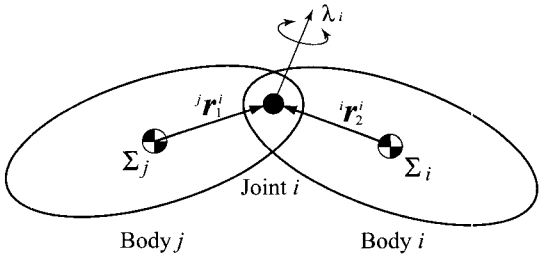


Fig. 2. Two contiguous bodies.

In order to model a humanoid robot, twenty seven rigid bodies were used, linked by twenty-six revolute joints based on Saika-3 [11]. Each leg and arm are made of two rigid bodies, interconnected by a revolute joint as shown in Fig. 2. A frame is attached to each body and, to simplify matters, the origin is placed over the respective center of mass. And the unit vector λ_i is parallel to the rotational axis of joint i in the frame of body i. This model captures the essential characteristics of cooperative motion and is simple enough to permit rapid simulations, reducing also the search-space of the genetic algorithm, i.e. the dynamics of this system can be written in various ways. One of the fastest consists of describing the dynamics of each rigid body *per se*, in a common inertial frame, using an $O(n)$ algorithm [12].

This way, the velocity and angular velocity v_i^{i*}, ω_i for the mass center of body i can be expressed as follows:

$$\begin{bmatrix} v^{i*} \\ \omega^i \end{bmatrix} = \sum_{r=-5}^{i} \begin{bmatrix} v_r^{i*} \\ \omega_r^i \end{bmatrix} u_r, \tag{1}$$

where ω_r^i is called the r-th partial angular velocity for i in Σ_I, while v^{i*} is the r-th partial velocity for i^* in Σ_I. By the differentiation of (1), the acceleration and angular acceleration of the mass center for body i can be expressed as the summation of a term P_i including $\dot{u}_r\ (r<i)$, a term $Y_i\dot{u}_i$ including $\dot{u}$, and remainder terms a_t^i, α_t^i excluding them as follows:

$$\begin{aligned} \begin{bmatrix} v^{i*} \\ \omega^i \end{bmatrix} &= \sum_{r=-5}^{i-1} \begin{bmatrix} v_r^{i*} \\ \omega_r^i \end{bmatrix} \dot{u}_r + \begin{bmatrix} v_i^{i*} \\ \omega_i^i \end{bmatrix} \dot{u}_i + \sum_{r=-5}^{i} \begin{bmatrix} v_r^{i*} \\ \omega_r^i \end{bmatrix} u_r \\ &= P_i + Y_i\dot{u}_i + \begin{bmatrix} a_t^{i*} \\ \alpha_t^i \end{bmatrix}. \end{aligned} \tag{2}$$

The equations for each body are:

$$M_i\{P_i + Y_i\dot{u}_i\} + X_i = \begin{bmatrix} {}^iF^i - {}^iR_h{}^hF^h \\ {}^iN^i + {}^ir_2^i \times {}^iF^i - {}^iR_h{}^hN^h - {}^ir_1^h \times \left({}^iR_h{}^hF^h\right) \end{bmatrix}, \tag{3}$$

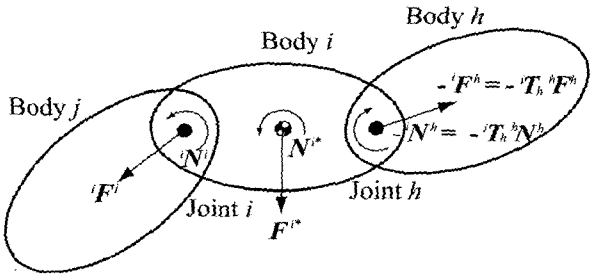


Fig. 3. Forces and moments applied to body i.

$$M_i = \begin{bmatrix} m_i E & 0 \\ 0 & I_i \end{bmatrix}, \tag{4}$$

$$X_i = \begin{bmatrix} m_i a_t^i - F^{i*} \\ I_i \alpha_t^i + \omega^i \times \left(I_i \omega^i \right) - N^{i*} \end{bmatrix}, \tag{5}$$

where m_i and I_i are the mass and moment of inertia for body i, E is the unit vector of 3×3, F^{i*} and N^{i*} are the external force and moment exerted on the mass center of body i, and ${}^{i}F^{i}$, ${}^{i}N^{i}$, ${}^{h}F^{h}$, ${}^{h}N^{h}$ are the internal forces and moments exerted on joint i and h, respectively (see Fig. 3). Also, ${}^{i}R_h$ is the transformation matrix from Σ_h toward Σ_i.

Interaction with the ground was also modeled using a three dimensional virtual spring-damper system in the normal direction and viscous damping in the tangential direction [13].

3. FORMULATION OF OPTIMIZATION

3.1. Optimization and control

A system with 26 joint actuators resulted from the previous modeling. The control objective is to specify a certain motion for the whole body and to control the execution of this motion. The choice of the control method is guided by:

- The desire to achieve a control adaptable to new requisites such as different motions, minimization of effort, etc.
- The complexity of the system's dynamics, strongly nonlinear.
- The use of minimal *a priori* information.

These reasons, mainly the last one, suggested the use of self-learning techniques from the so-called area of 'Intelligent Control'. To have a better idea of the dimension of the search-space associated with the present problem, assume that the objective is to specify one second of input to the 26 actuators, with an update frequency of 10Hz. Assume further, to simplify the analysis, that the dynamic range of each actuator is reduced to 5 discrete values. This way, it is necessary, for each second, to determine a set of 260 values, from a universe of $(5^{26})^{10} \approx 50^{181}$ possible combinations. Therefore, it seemed that the use of genetic algorithm for optimization is appropriate.

3.2. Performance criterion

In the case of the humanoid robot, which has capacity of locomotion, its mainbody has additional motion freedom introduced by its limbs. The limbs can thus afford to pose the adequate configuration corresponding to the external force. Generally, for example, when a human is doing work using an arm, he tends to choose spontaneously the posture that will minimize the potential energy expenditure in order to avoid joint fatigue. Robots should not also exceed the joint torque limit to avoid damage on the actuators during a task. The serial connection of the legs is more powerful than those of the arm. Weight of the arm, thus, is negligible in comparison with that of the trunk or leg. This implies that it is not possible to generate the large manipulation force by using the arm only, but a designed work can be carried out successfully by the distribution of surplus joint torque of legged limbs generated through adjusting the mainbody and legged limbs. Accordingly, we propose the optimized cooperative motion to minimize total joint torques with a constraint of which each joint torque is generating within range of its maximum limit. As a result, a robot is able to have a margin equivalent to total surplus torque with respect to initial configuration of the robot for external force, and to enlarge the range of work. In relation to this point, we adopt the objective function, which is to minimize summation of total joint torques for cooperative motion, as the performance criterion in this paper.

It is well known that for an ordinary manipulator whose base is fixed on the ground, the equation of motion is $\tau = J(\theta)^T F$, where τ is the applied joint torque of the limb, F is the external force exerted on the origin of the end-limb, and $J(\theta)$ is the Jacobian matrix which relates the velocity of the end-effector to the angular velocities of joints. Apparently, Jacobian matrix of the serial connection is necessary to understand the relation between its behavior and external force. Fig. 4 illustrates configuration of numerical model during pushing work. Notice that the inertia frame is Σ_I, the positions of the center of gravity and hand are C_G, C_h and the orientation of mainbody about y axis is Θ_{Gy} designated in Σ_I, respectively. To evaluate how far the current joint torque is from the torque limit, the following index is introduced:

$$Z = \sum_{i=1}^{n} \left| \tau_i \tau_{i,max} \right|, \ (i = 1 \sim 26), \tag{6}$$

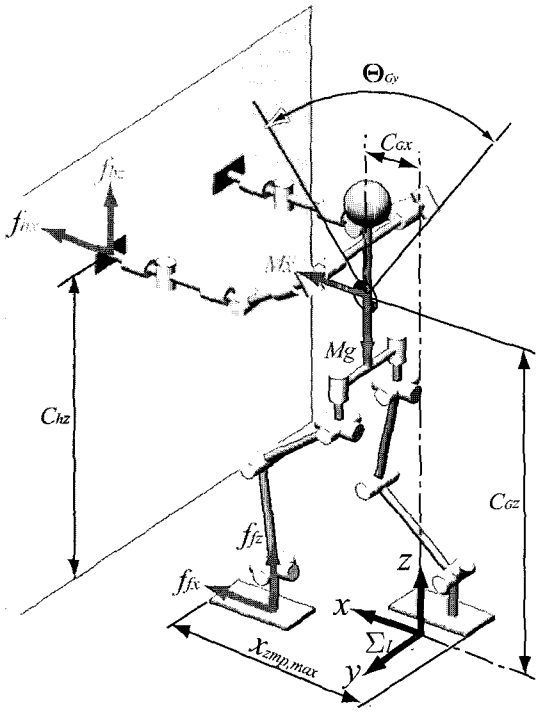


Fig. 4. Configuration of humanoid robot during the pushing task.

where τ_i and $\tau_{i,max}$ denote the current and maximum joint torques of i-th in limbs, respectively. The objective function denoted by $F(x)$ can be obtained by transformation of (6) in order to take the form of maximization with a positive value in the optimization process as follows:

$$F(x) = \sum_{i=1}^{n}\left(1 - \left|\tau(x)_i \tau(x)_{i,max}\right|^2\right). \tag{7}$$

It is unnecessary to change this object function according to the change of the body configuration for other tasks such as twisting a valve and lifting a box, and so on. The following system boundary constraints, however, should be changed according to the characteristics of work and/or environment appropriately.

3.3. Definition of system boundary

To obtain $\tau(x)_i$ of (7), however, constraints for system boundaries should be defined through analysis of motion expected in advance. They are expressed in the sagittal plane as follows:

$$\Omega = \{x \mid x^{(L)} \le x \le x^{(U)}\} \subseteq \Re^n, \tag{8}$$

where Ω is the solution space of the 26th-order vector which is composed of the inverse kinematics, and a subset of $\Re^n$. Ω is restricted by the values of lower limited vector $x^{(L)}$ and upper limited vector $x^{(U)}$ for $x = [\Theta_{Gy}\ C_{Gx}\ C_{Gz}\ C_h]$. $x \in \Omega$ satisfied all of the imposed constraints, which is known as the feasible solution. Its ranges are shown in Table 1.

To search the optimal solution of (7) in the imposed solution space, in the meanwhile, we must consider the following three terms:

(1) The degeneracy situation can be avoided. For an ordinary robot arm or leg with six or seven degrees of freedom, degeneracy happens when its Jacobian matrix becomes singular. A multi-limbed robot, however, can adjust the position and orientation of the mainbody to keep the arm or leg away from a degeneracy status. Therefore, the robot should not move over the limit of joint angle. Each joint of the robot has a certain motion range. If some of the joints reach the upper or lower limit of that range during the task, the robot might lose its balance and fall down. Therefore, it is preferable that each joint has the certain margin from the upper and lower bounds of motion range (see Fig. 5) by the geometric condition of the robot as follows:

$$\theta_{i,min} \le \theta_i \le \theta_{i,max}, \tag{9}$$

where θ_i denotes the angle of i-th joint, and $\theta_{i,max}$ and $\theta_{i,min}$ are its upper and lower limits, respectively.

(2) A robot should not exceed the maximum limited joint torque to avoid damage from occurring on the actuators. Therefore, another constraint is necessary as follows:

$$\tau_{i,min} \le \tau_i \le \tau_{i,max}, \tag{10}$$

where $\tau_{i,max}$ is set as shown in Table 2 based

Table 1. Ranges of feasible regions for constraints.

Constraint	x_{init}	$x^{(L)}$	$x^{(U)}$
Orientation of the center of gravity about y-axis Θ_{Gy} [°]	0	-30	30
Position of the center of gravity along x-axis C_{Gx} [m]	0.15	0.30	0.0
Position of the center of gravity along z-axis C_{Gz} [m]	0.90	0.75	1.05
Position of the hand from the center of gravity along z-axis C_{hz} [m]	0.83	0.68	0.98

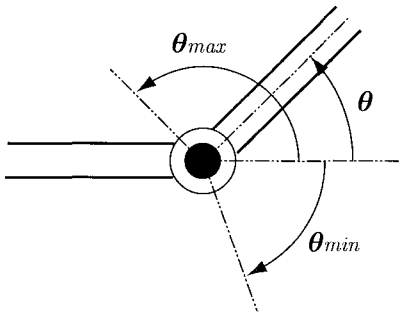


Fig. 5. Joint angle limits.

Table 2. The values of maximum limited joint torque.

Joint		Max. torque limit [Nm]
Arm	Shoulder, Elbow	36.1
	Wrist	10.4
Leg	Hip, Ankle	189.8
	Knee	227.8

on Saika-3.

(3) The stability of the system should be sustained. The typical way to evaluate the stability of the locomotion system is ZMP (Zero Moment Point), which is defined as a point on the ground that the summation of all moments generated at each body by inertial and gravitational forces becomes zero. If there is no consideration given to stability, a robot easily loses its balance and topples over. This scheme is considered under the condition of which no slip occurs between the foot and the ground. The equilibrium of force and moment is expressed in the sagittal plane as follows:

$$N_{COG} = f_{hx}(C_{hz} - C_{Gz}) - f_{hz}C_{hx} - f_{fx}C_{Gz} + f_{fz}x_{zmp}, \quad (11)$$

$$Mg = f_{fz} + f_{hz}, \quad (12)$$

$$M\ddot{x} = f_{fx} + f_{hx}, \quad (13)$$

where N_{COG} is the moment occurring around the center of gravity, M is the total mass of the robot, f_{hx}, f_{hz}, f_{fx}, and f_{fz} are forces generated on hands and feet along x and z directions, respectively (see Fig. 4). If there is no slip between floor and foot, reaction force and moment are given as follows:

$$f_{fx} = -f_{hx}, \quad (14)$$

$$N_{COG} = f_{hx}C_{hz} - f_{hz}C_{hx} + f_{fz}x_{ZMP}. \quad (15)$$

From (15), x_{zmp} is obtained as follows:

$$x_{zmp} = -f_{hx}C_{hz} + f_{hz}C_{hx}f_{fz}. \quad (16)$$

In order not to generate N_{COG}, the range of ZMP in the feasible region should be satisfied in the interval as follows:

$$x_{zmp,min} \le x_{zmp} \le x_{zmp,max}. \quad (17)$$

4. OPTIMIZATION USING GENETIC ALGORITHM

Although the various algorithms deforming and developing SGA have been developed so far, those mechanisms are basically equal to SGA, in terms of including factors such as encoding of parameter, generation of initial individual population, use of genetic operator, and evaluation of fitness for an individual in the population, etc. [8,14]. The reason is that effects of these factors are dominant in the performance of genetic algorithms. In this Section, we will describe the procedure applying SGA to the numerical model of the humanoid robot in more detail.

4.1. Encoding of parameter

The genetic operators and fitness are performed in two spaces of coding based on the binary string and the solution based on parameters. To represent points in the search space, each range of feasible region for constraints as shown in Table 1 is transformed to natural parameter called string or chromosome in the coding space. Fig. 6 shows the procedure that transforms constraint x from the vector of binary string s into the vector of chromosome length l. The constraint x_i is obtained from range of feasible region, the binary string s_i is calculated by binary encoding of x_i, and then, the chromosome length l_i is obtained by the following equation:

$$l_i \ge \log_2\left[(|x_i^{(U)}| - |x_i^{(L)}|)10^{d_i} + 1\right] \ (i = 1 \sim 4), \quad (18)$$

where the resolution of digit for each parameter is $d_i = (0,0,0,0)$. From (18) and Table 1, total length of chromosomes l becomes 21-bit.

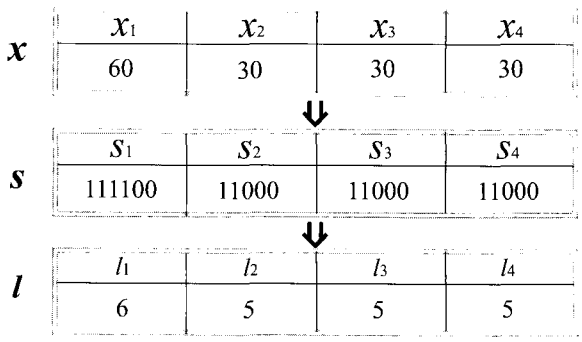


Fig. 6. Binary encoding s and chromosome length l for real vector x.

4.2. Generation of initial population

The initial population must perform the simulated evolution through the genetic operators. The population takes over the role of integral memory while evolving individuals. The population, denoted by $P(k)$, in k-th generation is defined as a set of individuals for the number of N:

$$P(k)=[s_1(k),s_2(k),\cdots,s_N(k)], \tag{19}$$

where $s_i(k)$ is a point in search-space as i-th chromosome and $N(>1)$ is the population size. The initial population $P(0)$ is generated by the random initialization method. This method is used to initialize the chromosome as a binary constant of which the number of Nl is generated by the random number generator. Accordingly, total bit number becomes $50\times 21=1050$ by setting the population size $N=50$ in this paper.

4.3. Genetic operators

4.3.1 Reproduction

In the evolutionary procedure from a population of current generation $P(k)$ to a population of next generation $P(k+1)$ through the genetic operators, two temporary populations $\bar{P}(k+1)$, $\tilde{P}(k+1)$ are created, respectively. Especially, the former is called the mating pool. The reproduction selects individuals in $P(k)$ and creates $\bar{P}(k+1)$. This selection has the characteristic to increase the selection probability in the total population. Among reproduction methods, the roulette wheel selection is well known in SGA. However, it has the disadvantage to not select the fittest certainly in the procedure of selection because of the probabilistic attribute. As a result, the optimized individual of current generation might be eliminated in the subsequent generation. The elimination of the optimized individual causes depression of search by loss of valuable characteristics of the chromosome. Accordingly, the elitist strategy is also introduced in order to preserve the best individual. Its procedure is to evaluate $P(k)$, to check the survival of the best individual, and then, to swap the saved one with the weakest in $P(k)$, if the best one is destroyed.

4.3.2 Crossover

Crossover is to select the parent chromosome arbitrarily from $\bar{P}(k+1)$, and to generate offspring by changing and binding of each bit from the crossover point in order to search the new point in a search-space. Among crossover methods, the one-point crossover is the most simple and well known as the genetic operator in SGA. Its procedure is divided into three steps during one cycle; (1) select two chromosomes randomly from $\bar{P}(k+1)$, (2) determine whether they will crossover at a randomly generated crossover point based on the crossover rate P_c, (3) copy the offspring into $\tilde{P}(k+1)$. When P_c is set highly, crossover will occur frequently. The number of chromosomes generated by crossover is $P_cN=0.8\times 50=40$.

4.3.3 Mutation

Mutation is a mechanism by which the chromosome is to escape from a suboptimal solution or dead point during the simulated evolution by the operations of selection and crossover. Simple mutation is a typical method in SGA. Its procedure is divided into three steps; (1) get one bit in the population, (2) decide whether it will mutate or not for a selected bit based on the mutation rate P_m, (3) copy the bit into the new population. When P_m is set high, a mutation will occur easily. The number of bits generated by the mutation at each generation is $P_mNl=0.01\times 50\times 35=17.5$.

4.4. Evaluation of fitness

When the new population is created by reproduction, crossover and mutation, fitness of individuals has been evaluated by the calculation of the objective function. Accordingly, an individual that has higher value of fitness should receive a lot of compensation. The fitness function $f(x)$ is described by a form of maximization with a positive value as follows:

$$f(x)=F(x)+\varepsilon P(x)-\gamma. \tag{20}$$

In (20), the penalty function $P(x)$ is:

$$P(x)=\sum_{j=1}^{2} w_j g_j^+(x), \tag{21}$$

where w_j is the value of weight, and $g_j^+(x)$ is the function to satisfy the following condition:

$$g_j^+(x)=\begin{cases} g_j(x), & x \text{ is infeasible solution,} \\ 0, & x \text{ is feasible solution.} \end{cases}$$

By imposing the penalty function, it is possible to transform the optimization problem without constraints when the algorithm searches the feasible region in the calculation of the fitness function. If Jacobian matrix becomes singular or ZMP is not located within the stable region generated by the random generator, the value of penalty function is imposed to the fitness function. Here, the constants

are set $\varepsilon = -1$ and $\gamma = -12$ in order to satisfy the relation of $f(x) \geq 0$ as small as possible.

4.5. Termination condition

When individuals are going to converge toward the solution, most individuals make it to the next generation without change of characteristics. Accordingly, it is very important to determine the accurate termination condition in order to save the computation time. Therefore, we adopt the limitation of the maximum generation number and comparison of maximum fitness value and average fitness value. The iteration is terminated if the average fitness value is over 70% of the maximum fitness value when the iteration number reaches the maximum generation number. In this research, the number of maximum generation, M is set as 100 experimentally.

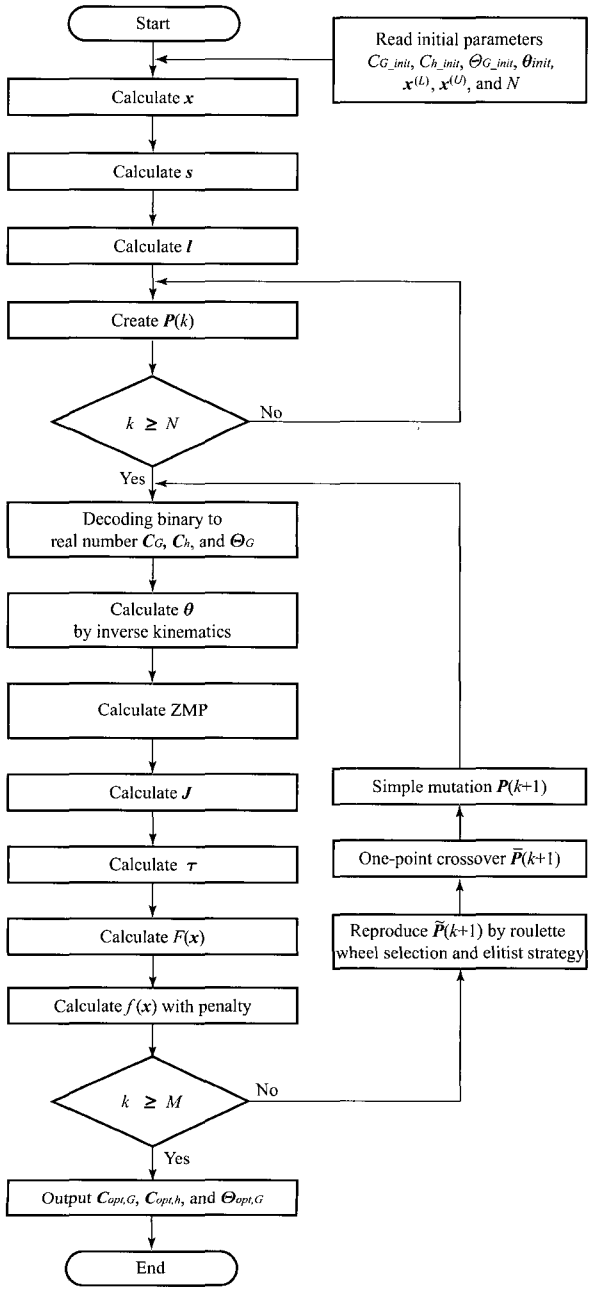


Fig. 7. Flow chart for procedure of simulated evolution.

5. RESULTS OF SIMULATED EVOLUTION

Fig. 7 shows the flow chart for the evolutionary procedure to obtain the optimized parameters (Θ_{Gy}, C_{Gx}, C_{Gz}, and C_{hz}) from the initial posture of the humanoid robot by SGA. Each simulated evolution is carried out at the manipulation forces $p = 5$, $10, \cdots, 90$N, respectively. To verify the convergence of a solution satisfying the termination condition in the simulated evolution, Fig. 8 shows comparison of maximum fitness value and average fitness value for each manipulation force.

Next, we present torques of main joints such as hip, knee, shoulder, and elbow according to the change of manipulation force in the initial and optimized postures as shown in Fig. 10. Even though the maximum joint torques of hip, knee and elbow do not surpass those limited values during the task (see Figs.

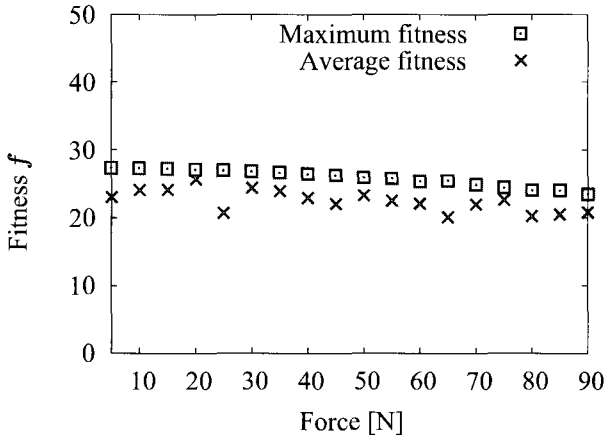


Fig. 8. Comparison of maximum fitness value and average fitness value.

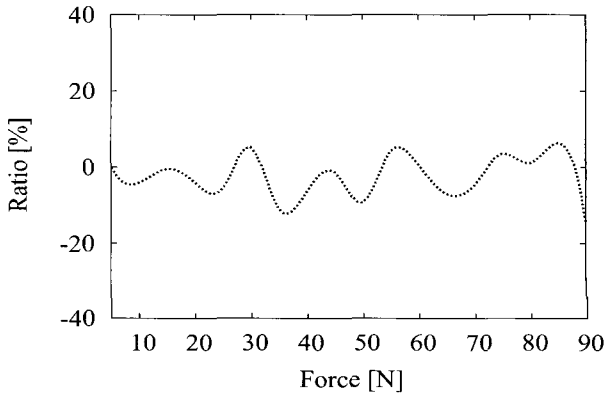
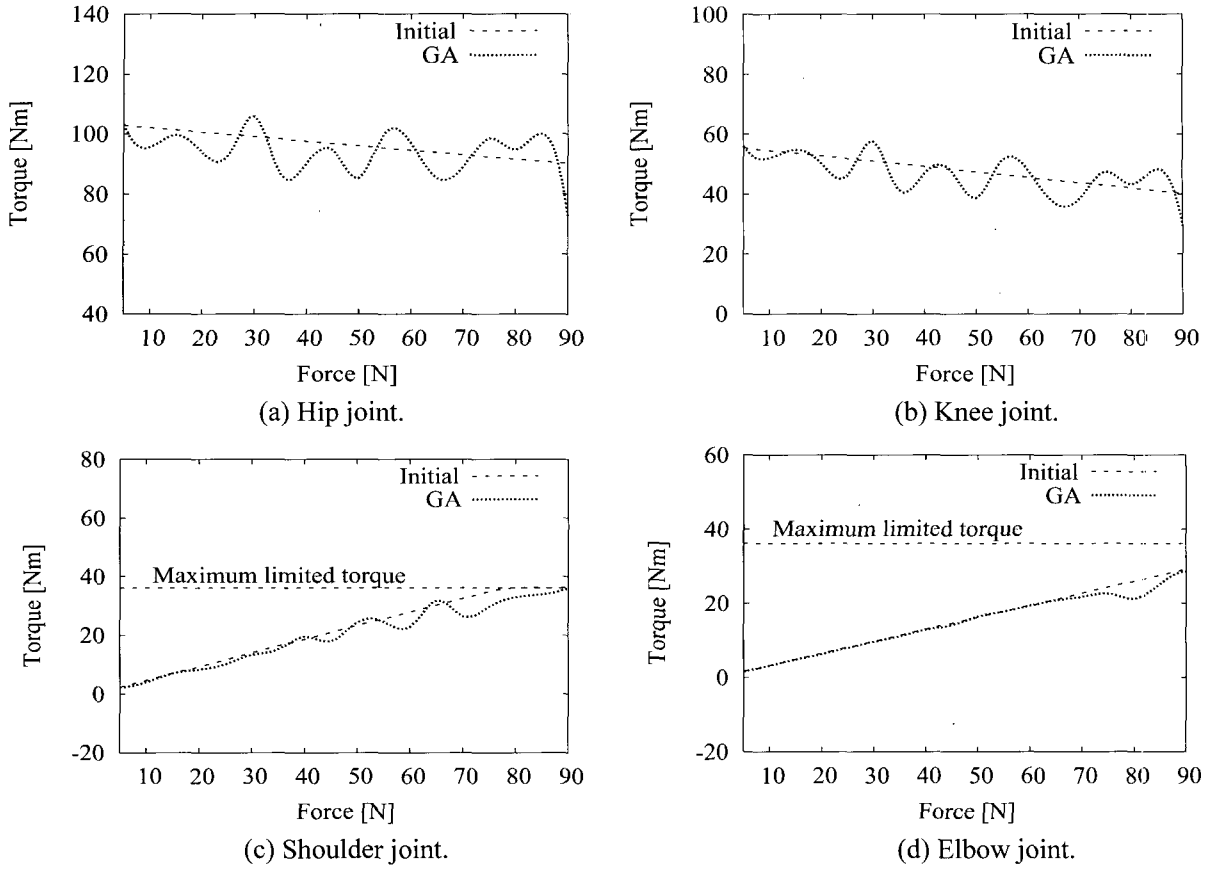


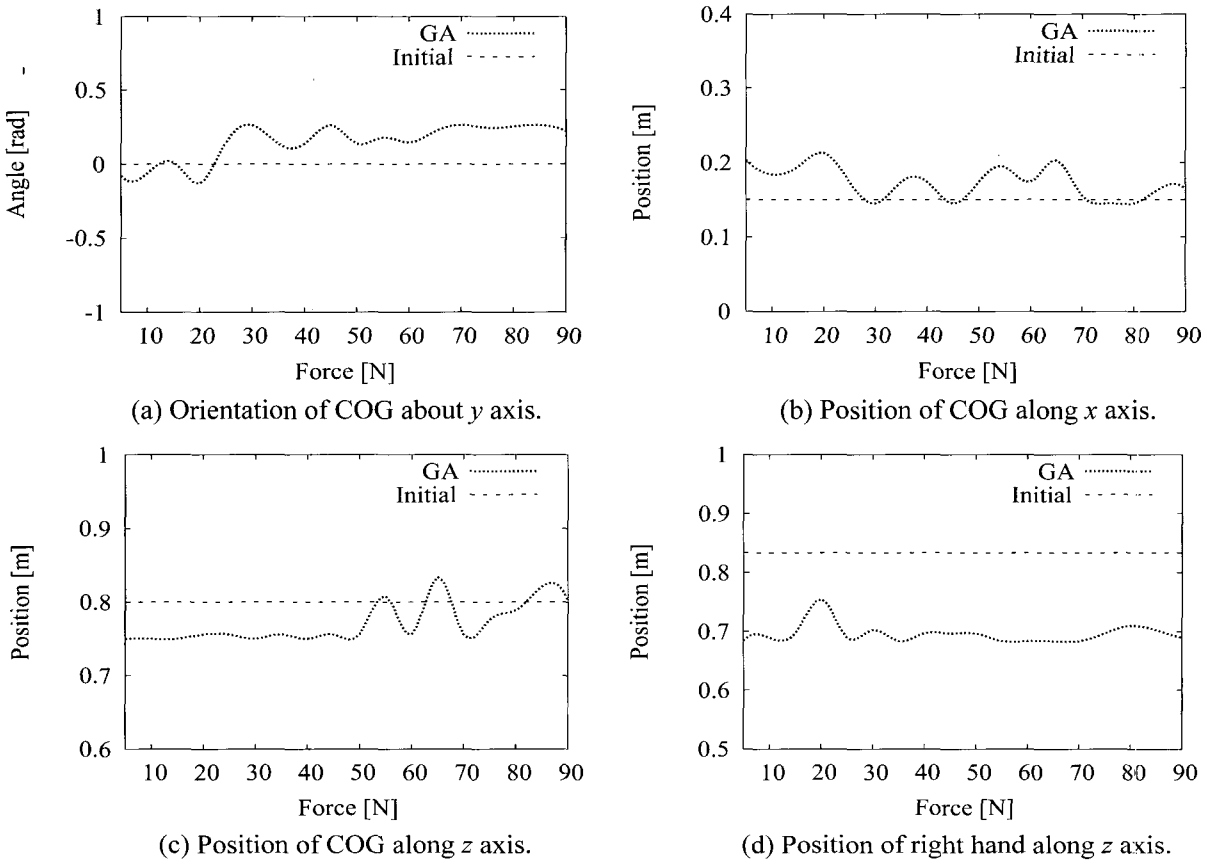
Fig. 9. Ratio of total joint torques.



(a) Hip joint. (b) Knee joint.

(c) Shoulder joint. (d) Elbow joint.

Fig. 10. Results of joint torques in the right leg and arm.



(a) Orientation of COG about y axis. (b) Position of COG along x axis.

(c) Position of COG along z axis. (d) Position of right hand along z axis.

Fig. 11. Results of main parameters.

10(a), (b), and (d)), the joint torque of shoulder surpasses its limit value at 75N in the initial posture (see Fig. 10(c)). This means that the generated maximum manipulation force might be restricted by the shoulder joint, if the optimal configuration of the robot is not considered while doing work. By the proposed method, however, it can be enlarged until 90N. We also consider the efficiency of the proposed method from another point of view. It is the energy consumption for total joints of the robot. Fig. 9 presents the ratio of summation of total joint torques calculated in the initial and optimized configuration of the robot. It is calculated by $\left(\left(\sum_{GA}|\tau_i|-\sum_{Init}|\tau_i|\right)/\sum_{Init}|\tau_i|\right)\times 100\,\%$. Although the total energy consumption of the robot increased occasionally in certain cases according to the change of manipulation force, it decreased about 10% in total.

Finally, Fig. 11 shows the change of Θ_{Gy}, C_{Gx}, C_{Gz}, and C_{hz} of optimal configuration calculated from the initial configuration of the robot, according to the change of the manipulation force.

6. MULTI-LAYER PERCEPTRON

In the previous sections, we laid the foundation for optimal configuration of the entire body for the pushing task of a humanoid robot based on the genetic algorithm. The application of the genetic algorithm in the control of the real robot, however, has a problem. It requires too many arithmetic operations to search the optimal solution. Accordingly, we will adopt the perceptron learning law of neural network based on the backpropagation algorithm, and then, generalize the parameters (Θ_{Gy}, C_{Gx}, C_{Gz}, and C_{hz}), which were obtained from the optimization procedure by SGA, as a training set in order to overcome this limitation.

6.1. Multi-layer network architecture

The multi-layered network is more powerful than the single-layered one. The single-layer network suffers from the disadvantage that it is only able to solve linearly separable classification problems. However, it can overcome this limitation. The multi-layer perceptron trained by the backpropagation algorithm has been used widely in the neural network. Accordingly, it can be used to approximate almost any function, if we have enough neurons in the hidden layers. However, we cannot say, in general, how many layers or how many neurons are necessary for adequate performance [10].

Fig. 12 shows a five-layer network constructed in this research. Each individual input of manipulation forces ($f_{frx}, f_{flx}, \cdots, f_{fhlx}$) is weighted by corresponding elements of the *weights matrix* w. Each layer has its own weight matrix w and an output vector a, which includes a bias vector and a net input vector. There are six inputs and the outputs of layers one to four are the inputs for layers two to five. A layer whose output is the network output is called an *output layer* (layer 5). The other layers are called *hidden layers* (layers 1 to 4).

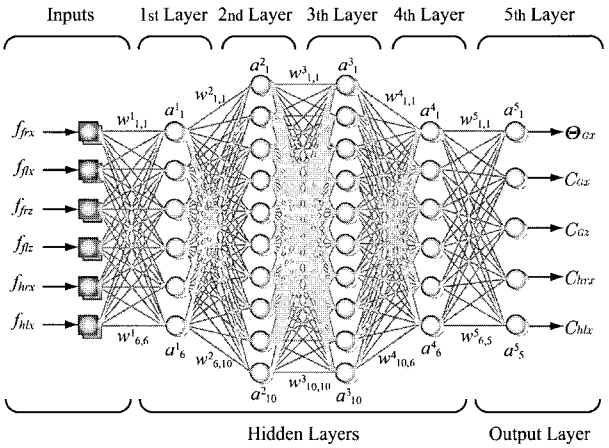


Fig. 12. Neural network.

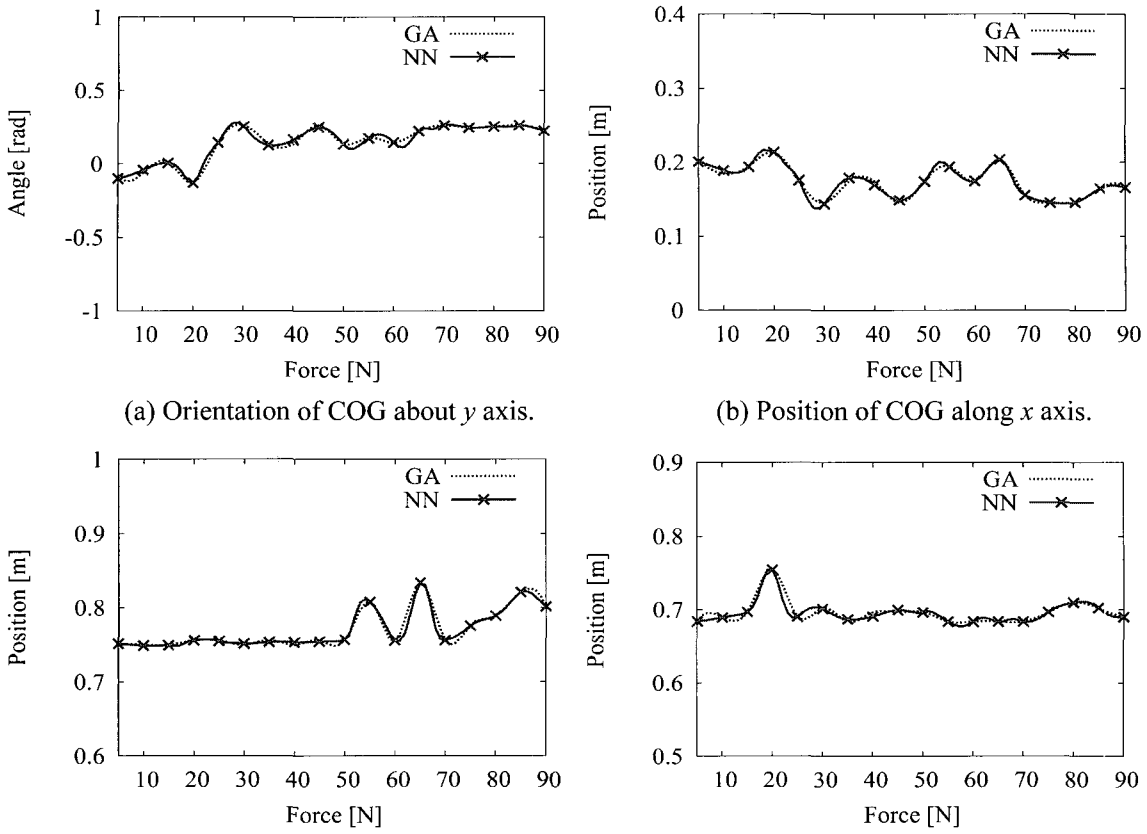
6.2. Performance optimization

When the basic backpropagation algorithm is applied to a practical problem, the training may take hours or days of computation time. To accelerate the convergence, an algorithm includes techniques such as the learning rate, momentum and rescaling variables. The learning law is described by the fact that during training the network parameters (weights and biases) are adjusted in an effort to optimize the performance of the network.

Generally, algorithms are distinguished by the choice of the search direction. In the backpropagation algorithm, it is composed of the method of steepest descent for the approximate mean square error. For steepest descent, a large learning rate will be taking large steps and is expected to converge faster. If it is set too large, however, the algorithm will become unstable. We set the learning rate as $\alpha = 0.6$ with an iteration number of $\eta = 10^5$. In addition, the momentum coefficient is set to $\gamma = 0.1$ $(0 \le \gamma < 1)$. The use of momentum is that convergence might be improved to smooth out the oscillation in the trajectory. As a result, we are able to accelerate convergence when the trajectory is moving in a consistent direction.

6.3. Generalization

In most cases the multi-layer network is trained with a finite number of examples of proper network behavior:



(a) Orientation of COG about y axis.

(b) Position of COG along x axis.

(c) Position of COG along z axis.

(d) Position of right hand along z axis.

Fig. 13. Comparison of responses.

$$\{p_1,t_1\},\{p_2,t_2\},\cdots,\{p_Q,t_Q\},\ (Q=1\sim 18). \qquad (22)$$

This training set is normally representative of a much larger class of possible input/output pairs. It is important that the network successfully generalize what it has learned to the total population. The training set, $g(p)$ is composed of the results of genetic algorithm obtained at each of the manipulation forces $p=5,10,15,\cdots,90$. There are 18 input/target pairs. Fig. 13 shows the comparison of responses. The dotted line represents $g(p)$, the solid line represents the network response, and the '+' symbols indicate the training set. The network response almost accords with the response of SGA. This network generalizes well. By the application of the network response, the system can be controlled in real time.

7. CONCLUSIONS

This paper has presented two main issues. One is the determination of optimal configuration for the pushing task of the humanoid robot by the simple genetic algorithm. The other is the generalization of the parameters of SGA by the multi-layer neural network based on the backpropagation algorithm.

The former focuses on determination of optimal posture to minimize the energy consumption according to the change of manipulation force. As a result, the range of workspace can be enlarged under the same physical and environmental conditions of the robot. Since the genetic algorithm has characteristics that can handle quantitative constraints, it is more useful in complexity of structure and in accomplishment of high level tasks like a humanoid robot. The prerequisites, however, are the accurate and adequate selection of the objective function and the system boundary constraints for a robot. They are identified in sophisticated works with varying environments and remain of severe interest for researchers in this related field. The latter is the perceptron by the practical neural network architecture, learning algorithm and training technique. As a result, parameters to pose optimal configuration according to the change of manipulation force were generalized almost similar to those of the genetic algorithm. Even though, we cannot say exactly how many layers or how many neurons are necessary for adequate performance in the humanoid robot because of change of possible input/output pairs according to the works and/or environments. The proposed neural network has been generalized well.

With this work the authors reached three important conclusions: (1) The genetic algorithm is a simple and

powerful search method that proved to be effective even with a search-space so vast as that of this problem. Probably, there are not a great number of optimization methods capable of performing so well. (2) Most optimization algorithms including the genetic algorithm always imply that the more information the algorithm gets, the less it learns. It means that the algorithm should have a great knowledge base obtained via simulation and much care should be taken before on-line implementations of a learning algorithm in real and costly systems. (3) The system boundary constraints seem to be identical to the other tasks such as twisting a value and lifting a box, etc. In relation to this point, it is necessary how the proposed approach could be extended to more general behaviors. Future topics for study will be to analyze various motions from the viewpoint of identification and to investigate applicability in more detail.

REFERENCES

[1] D. E. Orin and S. Y. Oh, "Control of force distribution in robotic manipulator mechanisms containing closed kinematic chains," *Journal of Dynamics Systems, Measurement, and Control*, vol. 102, pp. 134-141, 1981.

[2] T. Yoshikawa, "Dynamic manipulability of robot manipulators," *Journal of Robotic Systems*, vol. 2, no. 1, pp. 113-124, 1985.

[3] Y. F. Zheng and J. Y. S. Luh, "Joint torque for control of two coordinating moving robots," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1375-1380, 1986.

[4] Y. F. Zheng and Q. Yin, "Coordinating multilimbed robots for generating large cartesian force," *Proc. of Int. Conf. on Robotics and Automation*, pp. 1653-1658, 1990.

[5] C. Su and Y. F. Zheng, "Task decomposition for multilimbed robots to work in the reachable-but-unorientable space," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1659-1664, 1990.

[6] Y. Yokokohji, S. Nomoto, and T. Yoshikawa, "Static evaluation of humanoid robot postures constrained to the surrounding environment through their limbs," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 1856-1863, 2002.

[7] H. Yoshida, K. Inoue, T. Arai, and Y. Mae, "Mobile manipulation of humanoid robots - Optimal posture for generating large force based on statics-," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2271-2276, 2002.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Michigan, 1975.

[9] M. Mitchell, *An Introduction to Genetic Algorithm*, Massachusetts Institute of Technology, 1996.

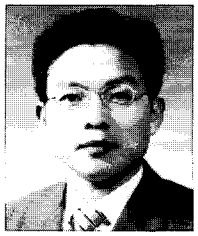
[10] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing Co., 1995.

[11] A. Konno, N. Kato, S. Shirata, T. Furuta, and M. Uchiyama, "Development of a light-weight biped humanoid robot," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1565-1570, 2000.

[12] D. E. Rosenthal, "An order *n* formulation for robotic system," *The Journal of the Astronautical Sciences*, vol. 38, no. 4, pp. 511-529, 1990.

[13] Y. K. Hwang, E. Inohira, A. Konno, and M. Uchiyama, "An order *n* dynamic simulator for a humanoid robot with a virtual spring-damper contact model," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 31-36, 2003.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co. Inc., NY, 1989.



Yoon-Kwon Hwang is a Research Professor in the Department of Mechanical Design and Manufacturing Engineering at Changwon National University, Korea. He received the M.S. and Ph.D. degrees from Tohoku University, Japan in 2001 and 2004, respectively. He received the B.S. from Chosun University, Korea in 1993. He worked as an Assistant Manager at Hyundai Motor Co., Korea in 1993-1999. His research interests include dynamic control of humanoid robot and optimal design of parallel manipulator.

Kook-Jin Choi is a Ph.D. candidate in the Department of Mechanical Design and Manufacturing Engineering at Changwon National University, Korea. He received the M.S. degree from Changwon National University in 2005. His current research areas are automation system design, neural network, robotics, optimization and genetic algorithm.

Dae-Sun Hong is a Professor in the Department of Mechanical Design and Manufacturing Engineering at Changwon National University, Korea. He received the M.S. and Ph.D. degrees from KAIST, Korea in 1986 and 1995, respectively. He received the B.S. from Seoul National University, Korea in 1982. He worked as a Principal Engineer at Samsung Aerospace Ind. Ltd., Korea in 1982-1997. His current research interests include assembly automation, artificial intelligence and automatic control.