Journal of Universal Computer Science, vol. 3, no. 9 (1997), 1056-1073
submitted: 11/12/96, accepted: 2/7/97, appeared: 28/9/97

Genetic Trajectory Planner for a Manipulator with Acceleration Parameterization

Young Dae Lee
(Seoul National University, Korea
ydl@robot1.snu.ac.kr)

Beom Hee Lee
(Seoul National University, Korea
bhlee@robot1.snu.ac.kr)

Abstract: This paper presents a genetic trajectory planning method of a robot manipulator producing the optimal trajectory between two end points. Genetic algorithm based methods seldom require *a priori* knowledge of a problem. Furthermore, they do not tend to fall into local optima and proceed toward the global optimum. However, they have difficulty in handling equality constraints of trajectory boundary conditions because they use probabilistic transition rules to find a solution. In this paper, we investigate the proper genetic trajectory parameterization and develop an efficient scheme for the implementation of genetic trajectory planner. We demonstrate the effectiveness and validity of the proposed approach through some simulation studies.
Key Words: robot trajectory planning, genetic algorithm, constraint handling

1 Introduction

For industrial application of robot manipulators, automatic generation of optimal trajectories is essential to increase the productivity. This theme has been therefore one of the important issues in robotics. The trajectory planning can be classified into two categories. The one is the trajectory planning along a specified path and the other is without a given path.

The search space is greatly reduced in the problems related to the first category, thereby the following methods are available such as dynamic programming (DP) [Shin and Mckay 86, Singh and Leu 87] , *graph search* [Jacak 92], and *phase plane algorithms* [Dubowsky et al. 86, Bobrow et al. 85, Shin and Mckay 85]. DP and graph search method discretize the search space and can implement the general cost. Phase plane algorithm is especially efficient for the time optimal trajectory planning. It is difficult, however, to apply these methods to the high dimensional search problem.

The problem in the second category is more difficult than the first one because the path and trajectory planning must be considered together. It belongs to the two-point-boundary-value-problems in optimal control theory and Pontryagin's Maximum Principle (PMP) offers the basic analysis tool. *Shooting algorithm*, which is a kind of Newton Raphson search, is known as a typical numerical tool for solving this problem [Geering et al. 86]. Other methods for solving this problem are based on the trajectory parameterization and nonlinear programming (NLP) [Dissanayake et al. 90, Lee and et al. 95]. NLP and shooting method commonly need the auxiliary information such as Jacobian or Hessian to speed up the search and need considerable trials of initial feasible seeds for the global optimal solution. However, in spite of this point-to-point optimal trajectory planning, it is not so easy to solve the problem because of the highly coupled nonlinearity in manipulator dynamics. Thus, the problem in the second category remains a problem yet to be completely solved.

Recently, genetic algorithms (GAs) have been highlighted in a variety of fields and successfully applied in many engineering problems. The applications of GAs to robotics are gradually increasing now. Davidor used GA for the path planning of a redundant robot (see [Davidor 91]). Kim and Khosla suggested a new methodology of robot manipulator design using GA (see [Kim and Khosla 93]). Handley developed an intelligent genetic path planner for a mobile robot focusing on the automatic generation of genetic programs (see [Handley 93]). The use of GAs in the optimal trajectory planning has several advantages over the other conventional methods. They have the features that do not tend to fall into local minima, but proceed toward the global optimum using the combined information at many search points, which make GAs robust for highly nonlinear problems. Furthermore, they seldom require the auxiliary information and the specific knowledge of the problems *a priori*.

In this paper, we propose a GA based trajectory planner to solve the problem in the second category, especially for the optimal time trajectory planning. We present the formulation of GA based trajectory planning and its implementation focusing on genetic trajectory parameterization with acceleration. A typical characteristic of time optimal motion and improved performance results were observed in simulations in comparison with other method in [Dissanayake et al. 90], which shows the validity of the proposed approach .

The organization of this paper is as follows. In Section 2, a simple genetic algorithm is briefly reviewed. In Section 3, we formulate an optimal trajectory planning of a robot manipulator in the view of proper implementation of genetic algorithm. Section 4 contains the detailed implementation procedure of the proposed genetic trajectory planning scheme. In Section 5, the validity and effectiveness of the proposed method is verified through some simulation studies in comparison with the other method. In Section 6, we conclude our statements and present some future research plans.

2 Simple Genetic Algorithm

2.1 Brief Review of a Simple GA

Genetic Algorithms (GAs) are a sort of population-oriented search techniques which use probabilistic transition rules to evolve multiple potential solutions. GAs do not require the auxiliary information of objective function and constraints. They have robust characteristic for the problems with multiple local minima. These properties of GAs can give advantages over the other conventional optimal trajectory planners of a robot manipulator that have been mentioned previously.

A simple GA commonly uses binary coding for parameter representation. When the real parameter set with the parameter number N_p is given by $x = \{x_1, x_2, \cdots x_i, \cdots, x_{N_P}\}$ (hereafter, we define $x \triangleq \{\cup_{i=1}^{N_p} x_i\}$), it is encoded into a binary string $\hat{x}$ ($\triangleq \{\cup_{i=1}^{N_p} \hat{x}_i\}$) which is also called a chromosome. If each real parameter x_i, which has the maximum bound x_i^U and the minimum bound x_i^L, is encoded into the binary string $\hat{x}_i$ using the binary length $\hat{L}_i$, then the precision of the encoded parameter is given by

$$\delta \hat{x}_i = \frac{x_i^U - x_i^L}{2^{\hat{L}_i} - 1} \quad (i = 1, 2, 3, \cdots, N_p)$$

The above equation means that a simple GA, when it adopts a binary coding representation, has limited precision.

A genetic algorithm is mainly composed of three basic operators : *reproduction, crossover,* and *mutation* [Goldberg 89]. Reproduction is a process in which the individual strings (chromosomes) in a population are replicated according to their fitness so that the individuals with a high suitability of solution contribute more offspring in the next generation.

After reproduction, crossover is proceeded with high probability (crossover probability : p_c) by exchanging the partial binary substrings of two mated parent strings. Otherwise, the parents are cloned into their children. This operator propagates the short, highly fit schemata through the population. The crossover site is picked at random and placed at the same bit position of two mated parent strings. For the following two parent strings $\hat{x}^1$ and $\hat{x}^2$ with a single parameter of the binary length 7, after one point crossover which is picked at the crossite ($|$), the offspring strings $\hat{x}_c^1$ and $\hat{x}_c^2$ are born as follows.

$$\hat{x}^1 = 1111|000, \ \hat{x}^2 = 0000|111 \longrightarrow \hat{x}_c^1 = 1111111, \ \hat{x}_c^2 = 0000000$$

Mutation operator changes some bits of individual strings occasionally with some low probability (mutation probability : $p_m << p_c$). By mutating the third bit of $\hat{x}_c^1$, $\hat{x}_c^1$ becomes $\hat{x}_m^1 = 1101111$. Otherwise, $\hat{x}_c^1$ is retained. Mutation has the effect of local search and increasing the diversity of a population. This operator is a secondary guarantee that the genetic search can access the whole search region. It helps to prevent loss of useful schemata and premature convergence to local points.

Elitism, which retains the best individual string over every generation, is important for an efficient application of GA. Through the implementation of elitism on the GA, the genetic solution can approach to the global optimum asymptotically as the generation proceeds Here, the convergence to optimum means the *ideal matching* case such that all parameters of the binary encoded optimal solution matches their corresponding parameter values of the real optimal solution within the desired precision (see [Yao and Sethares 94] and [Rudolph 94]).

Fig.1 shows a typical flowchart of a simple genetic algorithm.

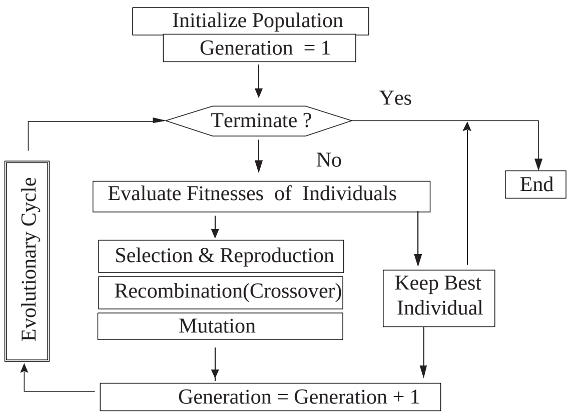


Figure 1: Flowchart of a simple genetic algorithm

2.2 Fitness Transformation with a Penalty Function

The optimal trajectory planning problem of a robot manipulator can be represented as a kind of constrained optimization problems such as many engineering problems. GA for a constrained optimization problem is fundamentally based upon a penalty function approach. The following constrained objective function with equality constraints and inequality constraints

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_k(x) \le 0 \quad (k = 1, 2, \cdots, M_1) \\ & && h_l(x) = 0 \quad (l = 1, 2, \cdots, M_2) \end{aligned} \tag{1}$$

can be converted into the following unconstrained auxiliary one with penalty functional description.

$$min \;\; P(x) = f(x) + \sum_{k=1}^{M_1} w_k \cdot \Phi(g_k(x)) + \sum_{l=1}^{M_2} w_l \cdot \Psi(h_l(x)) \tag{2}$$

where M_1 and M_2 are the numbers of inequality and equality constraints, respectively. $\Phi(\cdot)$ and $\Psi(\cdot)$ are the penalty functions for inequality and equality constraints, respectively, which are typically denoted as $\Phi(y)=|max(0,y)^m|$ and $\Psi(y)=|y|^m$. $max(x,y)$ returns the maximum value between two arguments. $|\cdot|$ denotes the absolute norm of the function and m is the positive number. In GAs, the fitness is defined as the maximization of the objective function and must be positive. The commonly used fitness transformations in GAs are the inverse of the auxiliary function in (2) or subtracting it from some large positive number C_{max} [Goldberg 89]. Thus the fitness for the above problem can be written as :

$$fit = max(0, 1/P(x)) \quad \text{or} \quad max(0, C_{max} - P(x)) \tag{3}$$

The fitness transformation with a penalty function above, however, seems a weak methodology for the heavily constrained optimization problems with many equality constraints and target variables. GAs use probabilistic transition rules to find a solution and does not utilize any auxiliary information of the function unlike conventional algorithms, such as NLPs. In addition, they have the limited precision in representing a parameter due to the binary representation, as described in [Section 2.1]. If we deal with the equality constraints using the fitness of GA directly in the form of a penalty function like, such as (2)-(3), it is highly possible that the genetic solution does not exactly match the equality constraints and stay on a local point in spite of the long generations of search. Even though a solution is found, the accuracy and repeatability of it may be very poor. Therefore, it is necessary to develop a genetic trajectory planner regarding the above mentioned properties of GA.

3 Formulation of Genetic trajectory Planning

3.1 Problem Statement

The optimal trajectory planning problem of a robot manipulator is described as follows.
For the given initial and final trajectory boundary conditions (BC) :

$$q(0) = q_0, \;\; q(t_F) = q_F \quad \{\, PBC \,\} : \text{position} \tag{4}$$

$$\dot{q}(0) = 0, \;\; \dot{q}(t_F) = 0 \quad \{\, VBC \,\} : \text{velocity} \tag{5}$$

and the robot manipulator dynamics :

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = T \quad \{\, R \,\} \tag{6}$$

find the optimal trajectories of the manipulator such that minimize some cost (In this paper, we consider the travel time as the cost) :

$$\int_0^{t_F} dt = t_F \quad \{\, COST \,\} \tag{7}$$

subject to the following joint limit constraints (LC) :

$$Q^L \leq q \leq Q^U \quad \{\ PLC\ \} : \text{position} \tag{8}$$
$$V^L \leq \dot{q} \leq V^U \quad \{\ VLC\ \} : \text{velocity} \tag{9}$$
$$A^L \leq \ddot{q} \leq A^U \quad \{\ ALC\ \} : \text{acceleration} \tag{10}$$
$$T^L \leq T \leq T^U \quad \{\ TLC\} : \text{torque} \tag{11}$$

where
n : degree of freedom (d.o.f.) of a robot manipulator,
t_F : travel time between initial position and terminal one,
$q, \dot{q}, \ddot{q} \in R^n$: generalized position, velocity and acceleration vectors ,
$T \in R^n$: generalized control torque (force) vector,
$M(q) \in R^{n \times n}$: inertia matrix,
$C(q, \dot{q}) \in R^n$: Corriolis and centrifugal force vector,
$G(q) \in R^n$: gravitational force vector,
U,L : superscript of upper and lower limit constraints respectively.

3.2 Genetic Trajectory Parameterization

The trajectory variables can be divided into two groups : kinematic trajectory variables and control torque. The kinematic trajectory variables are joint position, velocity and acceleration. The trajectory constraints are consisted of two parts : the equality constraints of trajectory boundary conditions (PBC, VBC) and inequality ones of trajectory limit constraints $(PLC, VLC, ALC$ and $TLC)$.

It is an important issue in robotics how to select the trajectory parameters among trajectory variables and how to select the optimization method for the trajectory planning . For illustration, the following works have been done for the trajectory planning with control torque and acceleration parameterization respectively. Dissanayake et al. adopted control torque parameterization and solved the optimal trajectory planning for two-link manipulator using sequential quadratic programming (SQP) (see [Dissanayake et al. 90]). Similarly, Lee performed the trajectory planning for dual cooperating manipulators with acceleration parameterization and SQP (see [Lee and et al. 95]).

In the view of SQP, which is a kind of typical NLP, it is not so difficult to match the trajectory boundary conditions because the search is done deterministically along the hyperplanes of equality constraints with the gradient information and the step size control of parameter variables.

However, if we adopt control torque for the genetic trajectory coding parameters, we have difficulty in the exact match of the trajectory boundary conditions. Control torque parameterization always involves the equality constraints of the trajectory boundary conditions explicitly and it is difficult to be adopted as the genetic trajectory coding parameters.

In this paper, we employed the joint acceleration parameters for the genetic trajectory parameterization. It was also employed in [Lee and et al. 95], however, our method and procedure to incorporate it with a genetic algorithm are much different from those that are based on the conventional nonlinear programming methods.

4 Procedure of Genetic Trajectory Planner

4.1 Acceleration Parametrizaton for GA

In this section, we describe the procedure of acceleration parameterization for the efficient implementation of genetic algorithm. For convenience, the notation afterwards will be denoted as follows :

Notations for acceleration parameterization
n : index of the joints of manipulator N : number of trajectory partition $\Delta t(=t_F/N)$: equally partitioned travel time $q_j, v_j(\triangleq \dot{q}_j), a_j(\triangleq \ddot{q}_j), T_j$: position, velocity, acceleration and torque of the j-th joint respectively $q_{j,i}, v_{j,i}(\triangleq \dot{q}_{j,i}), A_{j,i}(\triangleq \ddot{q}_{j,i})(i = 1, 2, \cdots, N)$: position, velocity, acceleration of the j-th joint in the i-th interval respectively $Q_{j,i}, V_{j,i}$ $(i{=}0, 1, \cdots, N)$: position, velocity of the j-th joint at the i-th knot position, here $i{=}0,N$ means initial and terminal knot respectively $Q \triangleq \{Q_{j,i} \mid j = 1, 2, \cdots, n, i = 0, 1, \cdots, N\}$: set of knot positions $V \triangleq \{V_{j,i} \mid j = 1, 2, \cdots, n, i = 0, 1, \cdots, N\}$: set of knot velocities $A \triangleq \{A_{j,i} \mid j = 1, 2, \cdots, n, i = 1, 2, \cdots, N\}$: set of accelerations

The trajectory discretization method used in this paper is as follows. At first, divide the total travel time interval $[0, t_F]$ into N equal subintervals. That is :

$$[0, t_F] = [t_0, t_1] \cup [t_1, t_2] \cup, \cdots, \cup [t_{N-1}, t_N] \tag{12}$$

where

$$\Delta t = t_i - t_{i-1} = \frac{t_F}{N} \ (i = 1, 2, \cdots, N) \tag{13}$$

Next, the joint accelerations are kept constant in each travel time subinterval, that is :

$$A_{j,i} = constant \ (i = 1, 2, \cdots, N) \tag{14}$$

To specify the travel time explicitly and to adopt it as a system parameter, rescale the travel time $t \in [t_{i-1}, t_i]$ by the normalized parameter τ as follows :

$$\tau = \frac{t - t_{i-1}}{\Delta t} \ (i = 1, 2, \cdots, N), \tau \in [0, 1] \tag{15}$$

Then, the joint velocity in the i-th travel time subinterval is expressed by :

$$v_{j,i} = V_{j,i-1} + \int_{t_{i-1}}^{t} A_{j,i} dt \tag{16}$$

And, the joint displacement can be obtained as :

$$q_{j,i} = Q_{j,i-1} + \int_{t_{i-1}}^{t} v_{j,i} dt = Q_{j,i-1} + \frac{1}{2}\tau \Delta t (V_{j,i-1} + v_{j,i}) \tag{17}$$

The total summation of the products of joint acceleration and the corresponding travel time interval must be the discrepancy of final and initial velocity, which is zero in this case (VBC).

$$\int_{0}^{t_F} a_j dt \quad = \quad \sum_{i=1}^{N} A_{j,i} \Delta t = 0 \tag{18}$$

Then the joint velocity at the i-th knot position of travel time interval is recursively obtained as follows :

$$V_{j,i} = V_{j,i-1} + A_{j,i}\Delta t = \sum_{k=1}^{i} A_{j,k}\Delta t \tag{19}$$

The joint position at the i-th knot position of travel time interval can be written as :

$$Q_{j,i} = Q_{j,i-1} + \frac{1}{2}(V_{j,i-1} + V_{j,i})\Delta t \tag{20}$$

Now, substituting (19) for (20) recursively, the following equations are derived.

$$Q_{j,N} - Q_{j,0} = \sum_{i=1}^{N-1} V_{j,i}\Delta t = \sum_{i=1}^{N-1} (N-i) A_{j,i} \Delta t^2 \tag{21}$$

Substituting (18) for (21), the relationship between the joint position difference between two task points and joint accelerations is obtained as follows :

$$Q_{j,N} - Q_{j,0} = N(\sum_{i=1}^{N} A_{j,i} - A_{j,N})\Delta t^2 - \sum_{i=1}^{N-1} i A_{j,i} \Delta t^2 = -\sum_{i=1}^{N} i A_{j,i} \Delta t^2 \tag{22}$$

Hence, if we adopt the joint accelerations as the genetic coding parameters, then, two dependent joint acceleration parameters $A_{j,k}, A_{j,l}$ must satisfy the two simultaneous equations of (18) and (22) in order to match two trajectory boundary conditions (PBC, VBC). That is :

$$\begin{bmatrix} A_{j,k} \\ A_{j,l} \end{bmatrix} = \frac{1}{l-k} \begin{bmatrix} l & -1 \\ -k & 1 \end{bmatrix} \cdot \begin{bmatrix} -\sum_{i=1, i\neq k,l}^{N} A_{j,i} \\ \frac{Q_{j,0} - Q_{j,N}}{\Delta t^2} - \sum_{i=1, i\neq k,l}^{N} i A_{j,i} \end{bmatrix}$$

for $k, l = 1, 2, \cdots, N$ and $k \neq l$.

4.2 Trajectory Parameter Coding

When we adopt joint acceleration for the genetic trajectory coding parameter, without losing generality, we can select two dependent parameters as $A_{j,N-1}, A_{j,N}$ to match the two trajectory boundary conditions (PBC, VBC). The coding parameter set of each individual string is given as follows :

$$\hat{x} = \{\bigcup_{j=1}^{n} \bigcup_{i=1}^{N-2} A_{j,i} \text{ and } t_F\} \tag{23}$$

where $\{\bigcup_{j=1}^{n} \bigcup_{i=1}^{N-2} A_{j,i}\}$ are the acceleration coding parameters and t_F is the travel time coding parameter. The two dependent acceleration parameters selected to match the PBC and VBC are determined by :

$$\begin{bmatrix} A_{j,N-1} \\ A_{j,N} \end{bmatrix} = \begin{bmatrix} N & -1 \\ 1-N & 1 \end{bmatrix} \cdot \begin{bmatrix} -\sum_{i=1}^{N-2} A_{j,i} \\ \frac{Q_{j,0}-Q_{j,N}}{\Delta t^2} - \sum_{i=1}^{N-2} iA_{j,i} \end{bmatrix} \tag{24}$$

The coding parameter size for each individual string $\hat{x}$ is as follows :

$$N_p = n \cdot (N-2) + 1 \tag{25}$$

4.3 Handling the Limit Constraints

The trajectory limit constraints of a robot manipulator, which are expressed by the continuous inequality equations, are not so easy to handle, so we transform it into the canonical static constraints. For illustration, let us consider a dynamic system as follows :

$$\begin{array}{lll} \text{given} & : & \dot{x}(t) = f(x(t), u(t)) \\ \text{subject to} & : & x^L \le x(t) \le x^U, \forall t \in [0, t_F] \end{array} \tag{26}$$

where $x(t)$ is the state variable with the lower limit x^L, the upper limit x^U and u is the control input. We can transform the continuous dynamic inequality constraints in (26) into the static equality constraints as follows :

$$c(x) = \int_0^{t_F} w_x(min(x^U - x, 0))^2 + (min(x - x^L, 0))^2 dt \tag{27}$$

If $c(x)$ becomes zero, the inequality constraint in (26) is satisfied. We handle the trajectory limit constraints similarly. For convenience, let us denote x_j as a trajectory variable of the j-th joint such that $x_j \overset{\triangle}{=} q_j$(position), v_j(velocity), a_j(acceleration), T_j(torque). And let us denote the trajectory variable of (8)-(11) as $x_j(t) \in [x_j^L, x_j^U], \forall t \in [0, t_F]$, where x_j^L and x_j^U are the lower and upper bound of each trajectory variable of the j-th joint, respectively. Then, we can rewrite (8)-(11) as :

$$G(x_j) = W_x^t g(x_j) \tag{28}$$

where

$$g(x_j) \triangleq \begin{bmatrix} \int_0^{t_F} |min(1 - x_j/x_j^L, 0)|^m dt \\ \int_0^{t_F} |min(1 - x_j/x_j^U, 0)|^m dt \end{bmatrix} \in R^{2\times 1}$$

is the constraint violation vector, $W_x^t \triangleq [w_x^L \ w_x^U] \in R^{1\times 2}$ is a weighting vector related to the lower and upper bound of x and m is a positive exponent number. If $G(x_j)$ goes to zero, it means that trajectory variable x_j satisfy its continuous limit constraint in (8)-(11).

By reflecting the trajectory limit constraint in (28) on the fitness function, the trajectory fitness for the genetic trajectory planning is denoted as follows :

$$fit = max(0, \frac{1}{t_F + \sum_x \sum_{j=1}^n G(x_j)}) \tag{29}$$

where

t_F : travel time ($COST$ to minimize),

$\max(x, y)$: maximum value between two arguments x and y,

x_j : type of trajectory variable of the j-th joint such that $x \triangleq q$(position), v(velocity) , a(acceleration) and T(torque),

$G(x)$: modified limit constraints related to the type of x (PLC, VLC, ALC, TLC),

$\sum_x$: notation reflecting each trajectory limit-constraint on the trajectory fitness.

4.4 Algorithm of Overall Procedure

The procedure of newly developed technique for the genetic trajectory planning for a robot manipulator is described as follows.

Notations	$\hat{x}=\{\hat{x}_i \mid i = 1, 2, \cdots, N_p\}$: trajectory chromosome in (23) $k = 1, 2, \cdots, N_{pop}$: index for the k-th individual of a population $\hat{X} = \{\hat{x}^k \mid k = 1, 2, \cdots, N_{pop}\}$: chromosomes of a population $Fit = \{fit^k \mid k = 1, 2, \cdots, N_{pop}\}$: fitness vector of a population $\Omega_s : \hat{X} \times R^{N_{pop}} \mapsto \hat{X}$; selection operator $\Omega_c : \hat{X} \mapsto \hat{X}$; crossover operator $\Omega_m : \hat{X} \mapsto \hat{X}$; mutation operator	
Initial settings	robot	link parameters $Q^L, \cdots, T^U$: lower and upper bounds of LC in (8)-(11)
	GA	$p_c, p_m \in [0, 1]$: crossover, mutation probability $[\hat{x}^L, \hat{x}^U]$: encoding bounds for a chromosome $\hat{x}$ $\hat{L} = \{\hat{L}_i \mid i = 1, 2, \cdots, N_p\}$: encoding lengths for $\hat{x}$ N_{pop} : population size, N_{gen} : max. generation number
	etc.	$d\tau$: sampling time, N : travel time partition number
Input	q_0, q_F : initial and terminal position of a robot (PBC)	
Halt	$\Lambda : \hat{X} \mapsto \{\text{Yes, No}\}$; halt when generation number reaches N_{gen}	
Output	trajectory of elite string fulfilling halting criteria	

Pseudo code for the algorithm
1. Initialization
1.1 Perform *initial settings* for the parameters of a robot, GA and etc
1.2 Input task points (PBC)
1.3 $gen \leftarrow 1$, $\hat{X}(gen) \leftarrow Init\_Chromosomes(\hat{x}^L, \hat{x}^U, \hat{L}, N_{pop})$
while $(\Lambda(\hat{X}(gen) \neq \text{Yes})$ *do*
2.   Evaluate $Fit(gen)$ with $\hat{X}(gen)$
for k=1 to N_{pop} *do*
2.1 $\hat{x} \leftarrow \hat{x}^k$
2.2 Compute Δt using (13)
2.3 Compute acceleration set A using (24)
2.4 Compute knot velocity set V, position set Q using (19)-(20) respectively
2.5 Compute acceleration $\ddot{q}$ from A, velocity $\ddot{q}$ and position q using (16)-(17) and torque T using (6) respectively
2.6 Compute fitness fit using (28)-(29)
2.7 $fit^k \leftarrow fit$
end
3.   *GA operation*
3.1 $\hat{X}_s(gen) \leftarrow \Omega_s(\hat{X}(gen), Fit(gen))$
3.2 $\hat{X}_c(gen+1) \leftarrow \Omega_c(\hat{X}_s(gen))$
3.3 $\hat{X}_m(gen+1) \leftarrow \Omega_m(\hat{X}_c(gen))$
4. $gen \leftarrow gen + 1$, $\hat{X}(gen) \leftarrow \hat{X}_m(gen)$
end
5. return *Output*

5 Simulation

The robot used for the test has the same link parameters and TLC like as in [Dissanayake et al. 90]. The main differences between this method and our method are compared in [Tab.1]. The specification of the robot is shown in [Tab.2] and [Fig.2]. The dynamics of the robot are as follows :

$$T_1 = M_{11}\ddot{q}_1 + M_{12}\ddot{q}_2 - 2h\dot{q}_1\dot{q}_2 - h\dot{q}_2^2, \quad T_2 = M_{22}\ddot{q}_2 + M_{12}\ddot{q}_1 + h\dot{q}_1^2$$

where $M_{11} = I_1 + I_2 + m_1 l_1^2 + m_2(L_1^2 + l_2^2 + 2L_1 l_2 cos(q_2))$, $M_{12} = I_2 + m_2 l_2^2 + m_2 L_1 l_2 cos(q_2)$, $M_{22} = I_2 + m_2 l_2^2$, $h = m_2 L_1 l_2 sin(q_2)$.

We used a two point crossover, roulette wheel selection, bit-flipping mutation and elitism for GA operators (see [Goldberg 89]). For details of the GA specification, see [Tab.3]. ALC is needed for the encoding bounds of acceleration parameters of an individual chromosome in our method , however, since ALC is not specified in [Dissanayake et al. 90], we heuristically set it to [-100 100] (rad/sec^2) for all joints by dividing the torque bound of each joint with its link inertia. The encoding bound of travel time parameter is set to [0.5 1.0](sec) and the penalty weight, W_T^t, for TLC is set to [10 10]. The trajectory partition number, N, is set to 10 as in [Dissanayake et al. 90] and sampling time, $d\tau$, is set to 0.5. Each parameter in a chromosome $\hat{x}$ is coded with 8 bits.

The optimal travel times obtained by three trajectory planners are compared in [Tab.4], where the conventional trajectory planner is the one that manipulates joint actuators according to a triangular velocity profile. The solutions obtained by the proposed method were slightly improved more than those of the method in [Dissnayake 90] for these cases (about 9% enhancement). The simulation was carried out using PC 586 with a 100 Mhz CPU clock. All programs were written in m files of Matlab which is a widely used package in control engineering. The fitness computation time took 3.6-4.0 *sec* for a population in one generation for the test cases.

[Fig.3] (a) and (b) show the fitness change of the elite individual during generation for case 1 and 3, respectively. [Fig.4] (a) and (b) show the objective (travel time) and TLC violations of the elite individual for case 1 and case 3, respectively, where it is seen that TLC violations are diminished to zero after some generations, that is, they are satisfied within 200 generations.

Thus, TLC is satisfied for all test cases when N is 10, population size N_{pop} is 30 and generation number N_{gen} is 200 (the total number of fitness evaluation is $N_{pop} \cdot N_{gen} = 6000$). However, as N is increased to 15 , TLC is not satisfied for all test cases within 200 generations.

To get a hint of this property, observe a constrained problem with the limit constraint of $|x_i| \leq x^U$ $(i = 1, 2, \cdots, N)$ and the equality constraint of $\sum_{i=1}^{N} x_i = 0$. Selecting a dependent parameter of a chromosome $\hat{x}$ to match the equality constraint as $\hat{x}_N = -\sum_{i=1}^{N-1} x_i$, we can estimate the extreme value of $|\hat{x}_N|$ as $(N-1)x^U$. If the parameters $\hat{x}_i$ $(i = 1, 2, \cdots, N-1)$ of a chromosome $\hat{x}$ are encoded with the bound of x^U, the dependent parameter $\hat{x}_N$ can violate the limit constraint with $O(N)$ in the worst case. This property can degrade the performance of our GA planner since the search efforts might be more concentrated on finding feasible trajectories rather than minimizing the cost during the GA search. From the above illustration, it seems that a large trajectory partition is not desirable for our GA planner, with a small population and a small generation cycle. Therefore, a trade-off is to be considered between solution performance and computational efficiency in selecting the parameters of the proposed GA planner.

[Fig.5] (a) and (b) show the time optimal motion for case 1 and 3, respectively, where the inward motion of links is observed during the traveling, which is a typical characteristic of the time optimal motion. This means that the robot manipulator takes a minimum inertia configuration to reduce the mechanical kinetic energy of links during the optimal motion. The joint position, velocity, acceleration and torque trajectories of genetic optimal solution are shown for case 1 from [Fig.6] to [Fig.9]. It is seen that the robot manipulator utilizes its torque range almost fully in order to achieve the minimum time motion under TLC.

Table 1: Comparison between the method

Description	[Dissanayake et al. 90]	Our Method
$COST$	Travel time	
Optimization tool	SQP (NLP)	GA
Parameterization	Torque	Acceleration
Handing BC	Penalty function	Matched
Handling TLC	Penalty function	
Trajectory partition number N	10	

Table 2: The specification of link parameters for a robot

Description	Symbol & Unit	Value
Length of the 1st link	L_1 (m)	0.4
Length of the 2nd link	L_2 (m)	0.4
Distance from the 1st joint to its C.G	l_1 (m)	0.2
Distance from the 2nd joint to its C.G	l_2 (m)	0.2
Mass of the 1st link	m_1 (kg)	0.5
Mass of the 2nd link	m_2 (kg)	0.5
Inertia of the 1st link about its C.G	I_1 $(kg \cdot m)$	0.1
Inertia of the 2nd link about its C.G	I_2 $(kg \cdot m^2)$	0.1
TLC at joint 1	$[T_1^L \;\; T_1^U]$ $(N \cdot m/rad)$	[-10 10]
TLC at joint 2	$[T_2^L \;\; T_2^U]$ $(N \cdot m/rad)$	[-10 10]

Table 3: Specification of genetic trajectory planner

Description	Symbol	Specification
Population size	N_{pop}	30
Selection scheme	Ω_s	Roulette wheel selection
Crossover type	Ω_c	Two point crossover
Crossover probability	p_c	0.677
Mutation probability	p_m	0.033
Max. generation number	N_{gen}	200
Elitism	$\bullet$	Used
Parameter set (chromosome)	$\hat{x}$	$\{\cup_{j=1}^{n} \cup_{i=1}^{N-2} A_{j,i}, t_F\}$

Table 4: Comparison of the travel time for three trajectory planners
(a) conventional planner, (b) the planner in [Dissannayake et al. 90], (c) the proposed GA planner

PBC (rad)		Travel time solution (sec)		
Case	$q_0 \Rightarrow q_F$	(a)	(b)	(c)
1	(0.00,-2.00)$\Rightarrow$ (1.00,-1.00)	1.092	0.6711	0.6255
2	(1.00,-1.00)$\Rightarrow$ (0.00,-2.00)	1.079	0.6732	0.6686
3	(1.32,-2.64)$\Rightarrow$ (2.80,-2.37)	0.727	0.6404	0.5267

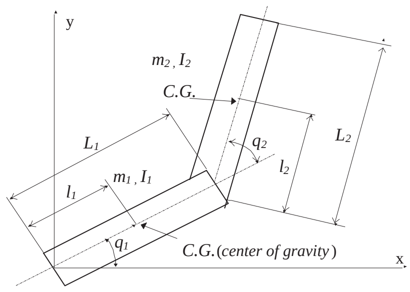
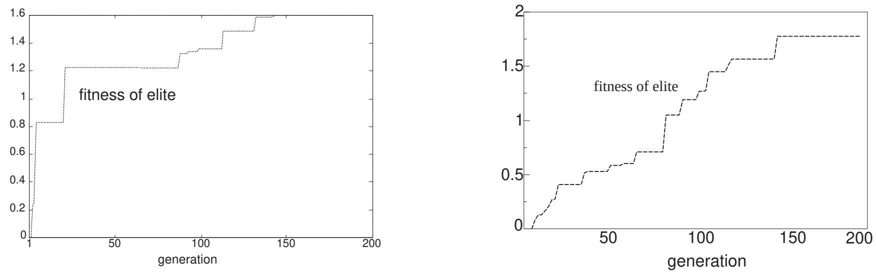
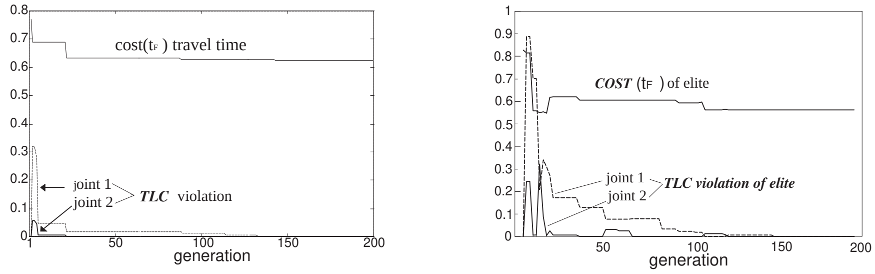


Figure 2: Two link robot for simulation



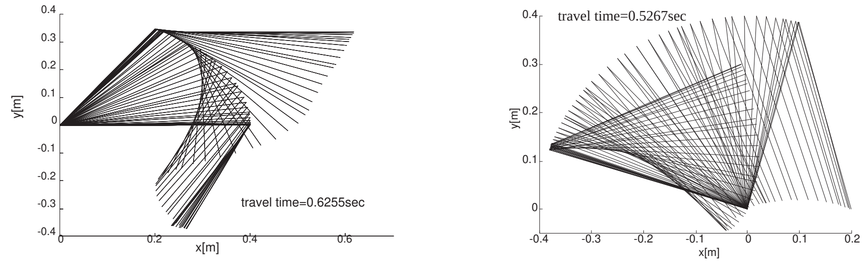
(a) (b)

Figure 3: Fitness change of elite during generation (a) case 1 (b) case 3



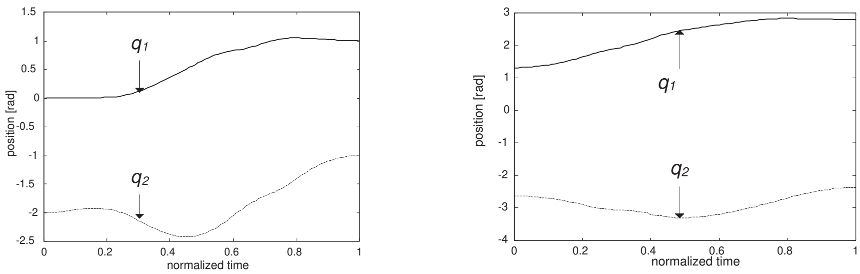
(a) (b)

Figure 4: Constraint (TLC) violation and $COST$ change of elite during generation (a) case 1 (b) case 3



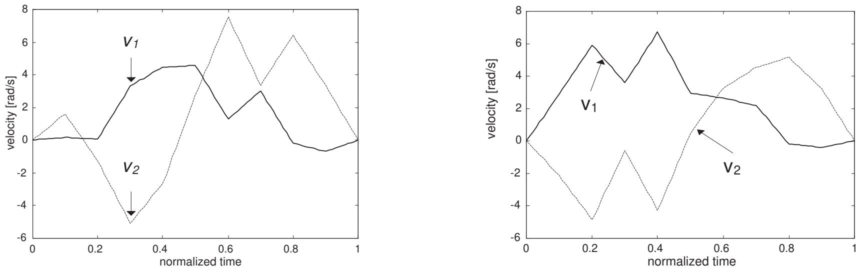
(a) (b)

Figure 5: Robot motion (a) case 1 (b) case 3



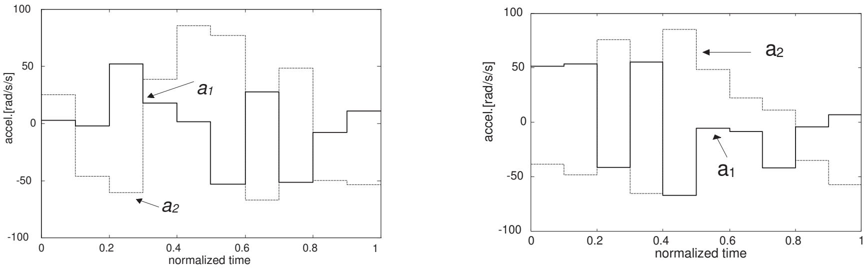
(a) (b)

Figure 6: Position trajectories (a) case 1 (b) case 3



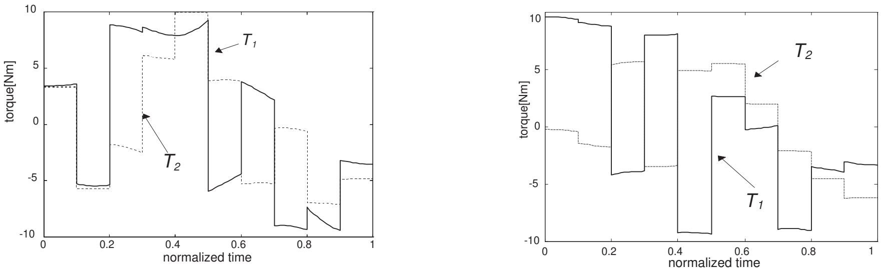
(a) (b)

Figure 7: Velocity trajectories (a) case 1 (b) case 3



(a) (b)

Figure 8: Acceleration trajectories (a) case 1 (b) case 3



(a) (b)

Figure 9: Torque trajectories (a) case 1 (b) case 3

6 Conclusion

A new genetic trajectory planner is proposed for the trajectory optimization of a robot manipulator. The trajectory parameterization is investigated for the proper implementation of GA. The control torque is shown to be difficult to be adopted as genetic trajectory coding parameters unlike as in typical NLP trajectory planner. We modified the acceleration parameterization for the proper implementation of GA. By incorporating our developed method in GA, we eliminated the mismatch of trajectory boundary conditions. The validity and effectiveness of the proposed method is demonstrated by the comparisons with the others. The presented genetic trajectory planner is not applicable in real time, but can be available for off-line application. Although, only the travel time considered as the cost in this paper, it can be easily extended to the other costs such as energy and torque consumption. In the future, we will consider more constraints and another tasks such as obstacle avoidance and multiple robot cooperation. In addition, we have a plan to study more efficient evolutionary trajectory planner for robot manipulators.

References

[Shin and Mckay 86] K. G. Shin and N. D. Mckay.: "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators", IEEE Trans. Automatic Control, AC-31, 6 (1986), 491-500.

[Singh and Leu 87] Singh and M. C. Leu.: "Optimal Trajectory Generation for Robotic Manipulators Using Dynamic Programming", Trans. of the ASME J. of Dynamic System and Measurement Control, 109 (1987), 88-96.

[Jacak 92] Witold Jacak.: "A Graph-Searching Approach to Trajectory Planing of Robots", Robotica, (1992), 10, 531-537.

[Dubowsky et al. 86] S. M. Dubowsky, M. A. Norris and Z. Schille.: "Time-Optimal Trajectory Planning for Robot Manipulator", Proc. IEEE International Conference on Robotics and Automation, (1986), 1906-1912.

[Bobrow et al. 85] J. E. Bobrow, S. Dubowsky and J. S. Gibson.: "Time-Optimal Control of Robotic Manipulators along Specified Paths", Int. J. Robotics Research, 4, 3 (1985), 3-17.

[Shin and Mckay 85] K. G. Shin and N. D. Mckay.: "Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints", IEEE Transactions on Automatic Control, AC-30, 6 (1985), 531-541.

[Geering et al. 86] H. P. Geering, L. Guzzella, S. A. R. Hepner and C. H. Onder.: "Time-Optimal Motions of Robots in Assembly Tasks", IEEE transactions on Automatic Control, AC-31, (1986), 512-518.

[Dissanayake et al. 90] M. W. M. G. Dissanayake, C. J. Goh and N. Phan-Thien.: "Time-Optimal Trajectories for Robot Manipulators", Robotica, 9, (1990), 131-138.

[Lee et al. 95] K. Y. Lee and M.W.M.G. Dissanayake.: "Near Minimum-time Trajectory for Two Coordinated Manipulator", Robotica, 13 (1995), 177-184.

[Kim and Khosla 93] J. O. Kim and P. K. Khosla.: "A Formulation for Task Based Design of Robot Manipulators", Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems (1993), 2310-2317.

[Rudolph 94] Rudolph.: "Convergence Analysis of Canonical Genetic Algorithms", IEEE Trans. on Neural Networks, 5, 1, (1994).

[Yao and Sethares 94] Leehter Yao, William A. Sethares.: "Nonlinear Parameter Estimation via the Genetic Algorithm", IEEE Trans. on Signal Processing, 42, 4 (1994).

[Goldberg 89] D. E. Goldberg : "Genetic Algorithms in Search, Optimization, and Machine learning", Addison Wesly (1989).

[Davidor 91] Yuval Davidor.: "Genetic Algorithms and Robotics - A Heuristic Strategy for Optimization", World Scientific Series in Robotics and Automated Systems, 1, (1991).

[Handley 93] S. Handley.: "The Genetic Planner : The Automatic Generation of Plans for a Mobile Robot via Genetic Programming", Proc. of IEEE International Symposium on Intelligent Control, (1993), 190-195