1992

# A Multi-Population Genetic Algorithm and Its Application to Design of Manipulators

Jin-Oh Kim
*Carnegie Mellon University*

Pradeep Khosla
*Carnegie Mellon University*

Proc. IEEE/RSJ Int'l Workshop on Intelligent Robots and Systems
(IROS '92). Raleigh. NC. July 1992.

# A Multi-Population Genetic Algorithm and Its Application to Design of Manipulators

**Jin-Oh Kim and Pradeep K. Khosla**

**The Robotics Institute**
**Carnegie-Mellon University**
**Pittsburgh, PA, 15213**

**Abstract** - In this paper, we introduce a new algorithm called Multi-Population Genetic Algorithm (MPCA) as an efficknt optimization technique for highly nonlinear problems. Our MPCA is a parallel implementation of a CA and shows its robustness for our problem domain (Task Based Design; TBD) where we design a manipulator which is best suited for a given task. The MPGA has the same number of optimization functions as the number of task points and maintains almost constant complexity and does not depend on the number of task points. In addition, we develop a framework called Progressive Design, in which we design progressively b a d on coarse-fine approach.

## 1. Introduction

In this paper. we introduce a multi-population genetic algorithm (MPGA) to solve the highly nonlinear optimization problem of task based design (TBD). Genetic Algorithm (GA) is an adaptive search technique and has been used for a variety of problems [5]. GA is based on mechanics of natural genetics and natural selection. The artificial survival of the fittest is combined with genetic operators which are abstracted from nature. Our MPGA is a parallel implementation of GA and works very well for our problem domain (Task Based Design; TBD) where we design a manipulator which fits best for a given task. TBD puts forth the idea of designing manipulators based on the given task. The CMU RMMS (reconfigurable modular manipulator system [19]), which utilizes a stock of assemblable pint and link modules of different size and performance specifications, is m ideal application domain of TBD. We. however. do not limit the application of TBD to the RMMS and believe that it can be used for design of general manipulators.

If we use a traditional approach of optimization where one optimization function is defined as a sum of an objective function and several constraints. the complexity of TBD increases exponentially as a function of the number of task points (which approximates a given trajectory). The number of variables increases linearly as the number of task points. Thus. traditional optimization techniques with one optimization function are not suited for a large number of task points due to the huge search space and nonlinearity. To overcome this problem, we use MPGA, which has the same number of optimization functions as the number of task points. Each optimization function has an objective function and all of them are connected by connecting constraints. The major advantage of this MPGA is that each optimization function maintains almost constant complexity and does not depend on the number of task points.

A framework called Progressive Design is proposed wherein we design progressively based on coarse-fine approach. The design variables are *dimension* (link length). *pose* (pint angle) and *base position* of a manipulator. Progressive Design is composed of four steps (design, prototype. planning and control). As we go from design (the first step) to control (the last step). the number of variables decreases. the search spaces for the variables are reduced. and the size of task space increases. For example, we design all variables for finite number of task points in the first step. Task points are sampled fmm a given trajectory or a task *space* and include, typically, the closest and the furthest points fmm the base space. In the last step of control, the task space to be reached is continuous through the given trajectory, but only variable is a set of joint angles, and dimension m d base position an determined in previous steps.

This paper is *organized* as follows; In Section 2, we briefly describe our example problem and discuss a dexterity measure and constraints for TBD. In Section 3, we introduce simple genetic algorithm and In Section 4, we show the structure of OUT Multi-Population Genetic Algorithm (MPGA). In Section 5, optimization functions an formulated. Details of Progressive Design are presented in Section 6. An example of the Progressive Design is presented in Section 7 m d finally, we summarize this paper in Section 8.

## 2. Problem Description

Task Based Design (TBD) selects an optimal manipulator which satisfies a given task description. The optimality is based on the use of a dexterity measure. In general. tasks can be formulated as constraints. A dexterity measure and some task constraints are discussed in this section. As an example of TBD, we consider a design of a 3 DOF planar manipulator in Fig. 1 for *clank* turing task where the trajectory is a circle in a plane as shown in Fig. 2. Since we are concerned with only the position of the end-effector (for the example task), the 3 DOF planar manipulator is redundant with respect to this task. The base position is. for simplicity. assumed to be given at the origin of the global coordinate frame.

The design variables for this design include dimension (link length) and pose (joint angle). There an four design variables for each task point: $l_1$, $l_2$, $l_3$ and $\theta_1$ as shown in Fig. 1. Other joint variables of $\theta_2$ and $\theta_3$ can be determined by the inverse kinematics. An alternative between elbow up and down configurations from $\theta_2$ and $\theta_3$ is another variable to decide. In this paper, instead of using the above variables. we use $(x_1, y_1)$ for the second joint and $(x_2, y_2)$ for the third joint. By this formulation, we can avoid the elbow up and down problem. If $(x_2, y_2)$ is given, the link length $l_3$ is automatically

decided by the line connecting the last joint and the corresponding task point as shown in Fig. 1. Thus, we don't need to solve inverse kinematics for this design Even better, we are not bothered by the periodicity of pint variables. To summarize, a task point has four design variables $(x_1, y_1, x_2, y_2)$. The total number of design variables for traditional optimization techniques and for our parallel implementation of GA are compared in Section 4.

The crank is shown in Fig. 2(a) and its circular trajectory is approximated by four task points as in Fig. 2(b). These task points are used in the first step (design). In the remainder of this section, we discuss dexterity measure and constraints for design.

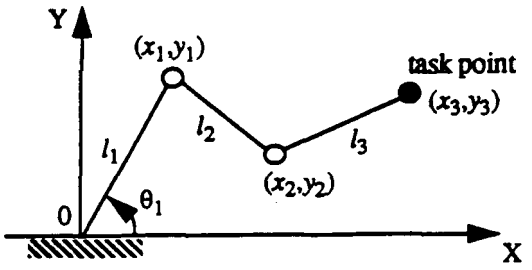


Fig. 1.3 DOF Planar Manipulator

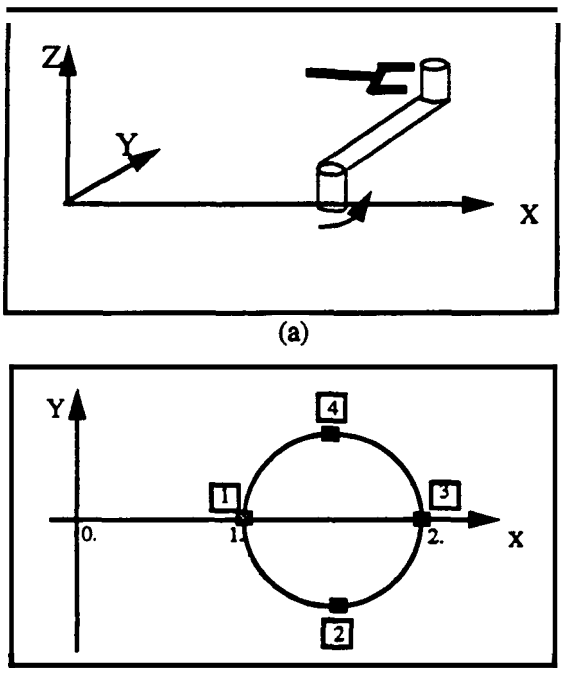


Fig. 2 Crank Turning Task

## 21 Dexterity Measure (DM)

The main role of a dexterity measure in design is to choose one optimal design from the candidates which satisfy given constraints. For a dexterity measure to be used for design, it must be independent of the scale of a manipulator. In this paper, we use the relative manipulability as a dexterity measure. The relative manipulability is a dimensionless scale independent measure of manipulability and is defined as [9]

$$M_r = \frac{M}{f_M} \quad , \tag{1}$$

where $M$ is the order independent manipulability and $f_M$ is a function of dimension ([length]$^2$). The measure $M$ is obtained as

$$M = \sqrt[m]{\det(JJ^T)} \tag{2}$$

whae $J$ is the Jacobian matrix of instantaneous kinematics and $m$ is the order of task space. For a 3 DOF planar manipulator with 2 dimensional task space. m = 2. For this case, this measure $M$ is equal to the measure of manipulability in [20]. For m > 2, however, this M has an advantage since its order is always equal to 2 without depending on the order of task space. We we $l^2$ for $f_M$, where $l$ is a total length of a manipulator. The total length of a manipulator is defined as a sum of link lengths. In general, link length is defined as $(l_i \equiv \sqrt{a_i^2 + d_i^2})$, where $a_i$ and $d_i$ are D-H parameters [7].

## 23 Constraints

The design constraints considered in this paper are:

1. Reachability constraint (RC)
2. Heuristic constraint (HC)
3. Joint limit constraint (JLC)
4. Joint angle change between two adjacent task points (JAC)
5. Task specific constraint (TC).

The RC is the most important constraint that must be satisfied by a designed manipulator. Every task point must be reached by the designed manipulator. The important role of the RC is to solve inverse kinematics implicitly. For our design problem of a planar manipulator, we don't need this RC because of the formulation where we use the position of the first two pints as design variables and the third link is expressed by a line connecting position of the second joint and the corresponding task point. When a manipulator is given, however, we need to use RC to avoid the inverse kinematics.

The HC, as the name implies, can take many different forms. The HC that we use is that the inner link is always longer than the outer link. This constraint is based on the observation rhat the outer link is more appropriate for fine motion and the inner link for gross motion. This HC also helps eliminate an awkward design such as zero link length for redundant manipulator [9]. However, for tasks that include obstacle avoidance. we may choose to turn off this HC. The JLC is included to take into account the fact that the total range of each actuator is limited. The JAC is based on the fact that the joint angle change between two adjacent task points must be kept small. We Qn't want a drastic change m the pose. like from elbow up to down, between two adjacent task points. The TC depends on a given task and may include bracing, obstacle avoidance, singularity avoidance, dexterity, and so on. Among them, we consider singularity avoidance and dexterity [2][3][15].

## 3. Simple Genetic Algorithm (SGA)

Genetic Algorithms (GAs) are adaptive search procedures based on mechanics of natural genetics and natural selection and have been used for a variety of sea-ch problems. GAs have proven to be

more efficient than other search algorithms for highly nonlinear and complex problems [4]. GAs have no requirement for continuity in the derivatives, so virtually any fitness function can be selected for optimizing. In contrast to conventional methods, GAs use population in which many individuals are selected based on the fitness value. In this section, we briefly review one GA called Simple Genetic Algorithm (SGA) [5]. SGA is used as a basis for building the Multi-Population GA (MPGA) structure in tlk following section.

SGA starts by generating a random population and continues to generate subsequent populations until some halting condition is mer. Each individual of a population represents a possible solution with different fitness and is coded by a binary string. One string (called "genotype") is matched to a set of variables (called "phenotype"). Each variable corresponds to a gene. When evaluating each gene, we decode the binary String (called "chromosome") mto a real value (called "dele") by using a linea mapping. After evaluation of all individuals. the next population is obtained by using genetic operators. The genetic operators used in SGA are:

1. Reproduction,
2. Crossover. and
3. Mutation.

Reproduction realizes the survival of the fittest within the SGA. There are many ways to achieve effective reproduction. One simple proportionate scheme selects individual s h g s for reproduction according to their fitness. where fitness is defined as the nonnegative figure of goodness being maximized. Individuals with higher fitness values have a higher probability of being selected for mating (crossover) and subsequent genetic action (mutation).

In crossover, two individual strings, selected using the reproduction operator. create two new individuals by exchanging partial strings with each other. A crossover site along the string length as shown in Fig. 3 is selected uniformly at random, and position values are swapped between the two strings following the crossover site. This crossover allows combination of advantageous substructures mto individuals more fit than either parents. Fig. 3 shows how crossover works for two individuals selected by reproduction.

In a binary coded GA, mutation is the occasional (low probability) alteration of a bit position (the changing of a 1 to a 0, and a 0 to a 1). By itself, mutation is a simple random walk through the string space. When used sparingly in combination with reproduction and crossover, mutation is an insurance policy against the loss of important genetic material at a particular position. This mutation helps avoid local minima.

For our design problem, there are four design variables $(x_1, y_1, x_2, y_2)$. Each variable is called a "gene". These are represented by string of binary digits. For example, if each gene is of 5-bit string (chromosome) as m Fig. 3, then each individual is represented by a 20-bit String (genotype). A genotype is mapped onto 4 unsigned integers which in turn an mapped onto real values of 4 alleles by a linear function based on the respective maximum and minimum values of $(x_1, y_1, x_2, y_2)$. A set of rul values (phenotype) for an individual is used to obtain the fitness value of each individual.

Based on the fitness value, reproduction selects two mating genes for crossover. Assume that the i-th and $j$-th individuals are selected as shown in Fig. 3. The fitness values are chosen arbitrarily for explanation. Between genotype and phenotype, there are implicit mapping functions different for all genes $(x_1, y_1, x_2, y_2)$. For this example, the crossover site is chosen randomly after the 6-th binary digit. New individuals ($I_1$ and $I_2$) of the next generation are obtained by crossover. When there an $m$ individuals (genes). we need $m/2$ times of reproduction and crossover for obtaining the next generation. Mutation is applied bit by bit at a low rate (1 or 2 times per every 100 binary bits) after crossover.

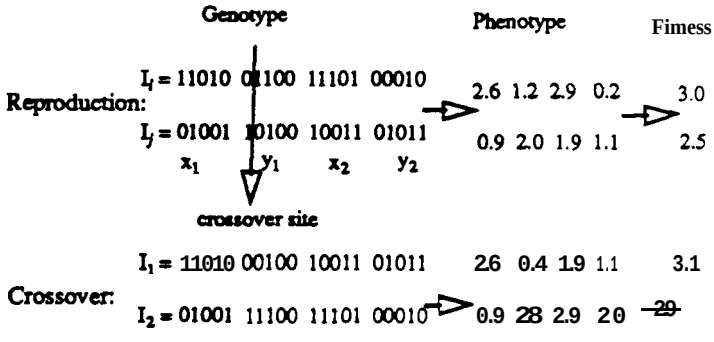


Fig. 3. Genetic operators

## 4. Multi-Population Genetic Algorithm

Different from simple GA (SGA) in previous section and other GAs, MPGA uses multiple populations. Each population works on one optimization function corresponding to one task point. The advantage of MPGA is that the complexity does not increase much as the number of task points increases. MPGA is different from GAs for multi-optimization function problems [18], since our MPGA solves problems with single complex optimization function and divides the complexity into multi-optimization functions.

For TBD, the number of variables increases linearly as the number of task points (TP). If we use one optimization function, the total number of variables is equal to

$$(n-2)*t + n + 2 \quad \text{for a planar manipulator.} \tag{3}$$

where $n$ is equal to DOF and $t$ is the total number of task points. This is computed assuming that $l_i$ (link length) and $\theta_i$ (joint angles) are used as design variables because these provide the minimum number of variables. For the first task points. there are 2 variables for a base position, $n$ variables for $n$ link lengths and $(n-2)$ joint variables assuming the use of inverse kinematics. That is. there are totally $2n$ variables for the first task point. The remaining $(t-1)$ task points have $(n-2)$ pint variables, respectively. Therefore, the total number of variables is $2n + (t-1)*(n-2) = (n-2)*t + n + 2$.

The search space increases exponentially as the number of variables increases. This implies that the quality of the solution decreases rapidly as the search space increases. To avoid this. we use $t$ optimization functions corresponding to $t$ task points. They are optimized simultaneously and modified depending on the optimal manipulators for other task pints. This is as if we are designing $t$ manipulators for $t$ task points. The advantage of this parallel implementation is that the number of variables for each optimization function is fixed [$t$=1 in (3)]. This advantage is obtained at the cost of inclusion of two additional constraints: link constraint (LC: the $t$ manipulators must have the same link lengths) and base position constraint (BC: the $t$ manipulators must have the sume base position). These are necessary because we want a single manipulator, which satisfies all constraints.

The dexterity measure (DM) in Section 2.1. all constraints in Section 2.2 and two additional constraints (LC and BPC) form @-&on functions and determine the fitness value of all individuals in all populations. *Among* these, LC, BPC and JAC serve as constraints connecting all populations and modify fitness values of all individuals as shown in Fig. 4. Objective functions computes fimess values of individuals for the corresponding task points, whereas connecting constraints controls optimization function based on the designs for other task points. For each task point, there are *m* individuals. A MPGA for a task with *t* task points has *t* populations.

Suppose that some individual for some task point has high fitness value of objective function and its link length is much different from the average of (*t*-1)**m* individuals of the other populations. Then, the effect of LC (connecting constraint) to that specific individual is negative. This negative value of LC neutralizes the high fitness of the objective function. The fitness value for that optimization function is adjusted this way and used for reproduction of the next generation. That is, the total fitness of an individual is a summation of the value from the corresponding objective function and the adjusting value from connecting constraints which is obtained by comparison with individuals in the other populations. The CC is to connect all task points and 10 result m a single manipulator as a solution. The BPC works in the same way. The JAC adjusts the fitness value based on the average poses of populations for the two adjacent task pints.

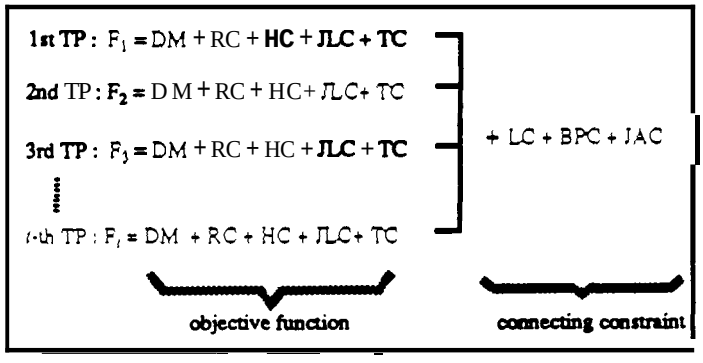


Fig. 4. Optimization functions

The MPGA is *similar* to having *t* SGA's operating in parallel as shown m Fig. 5. For each task point (population), there are *m* individuals and each of these has variables *corresponding* to dimension (D), pose (P) and base position (B). As an optimal solution, we E-quest that all manipulators *on* the right *side* of Fig. 5 have the same dimension (link length) and base position. and smooth change of poses dong the given trajectory. The connecting constraints (CC) enable to satisfy these requirements, which are satisfied progressively as the number of generations increases.

Fig. 6 shows the data structure for MPGA where we see *t* populations and *m* individuals for each population. D, P and B represent dimension, pose and base position, respectively. The advantage of MPGA is not only to maintain complexity of design constant, but also to be *applied* for subproblems of design. One of the subproblems of design is called *path placement* where we seek an optimal use of a given manipulator for a given task [6][14][15]. This kind of problem is called an analysis problem in contrast to our design problem. Our TBD with MPGA can be used to solve this path placement just by fixing variables corresponding to dimension (D) and turning off LC and HC, since a manipulator is given. Another

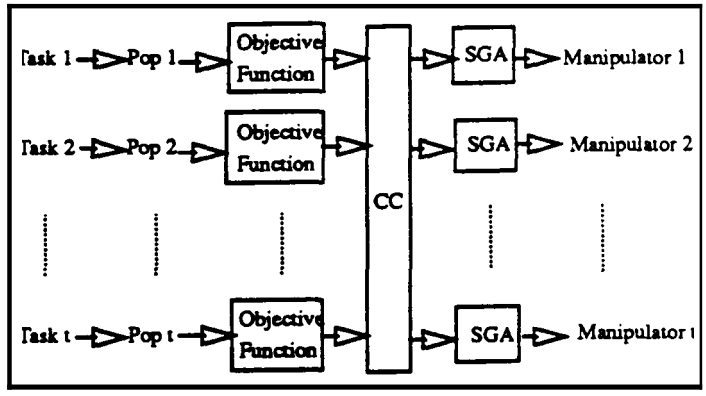


Fig. 5. MPGA structure

advantage of the MPGA is its ability to be extended to solve design problem of a manipulator with a mobile base. If the mobile base is omnidirectional, we need just turn off the BPC in Fig. 6. Otherwise. the BPC can be used to task into account its mobility constraint. A prismatic joint can be included in design by modifying the link length constraint (LC).

In summary, parallel implementation of optimization provides a general technique for design of manipulators with/without a prismatic pint or mobile base, and for design problem as well as analysis problem such as path placement with a given manipulator. Furthermore, it is not necessary to solve inverse kinematics because we use the RC. The complexity of each optimization function remains almost constant without depending *on* the number of task points, which is a main advantage of *our* MFGA formulation.

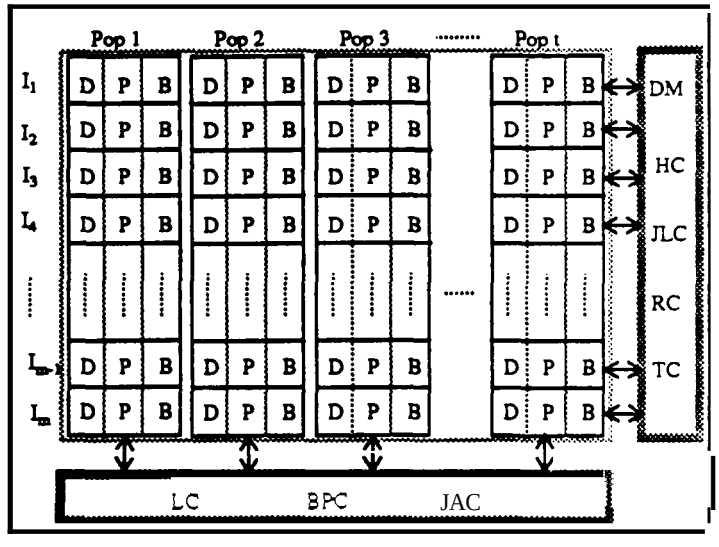


Fig. 6. MPGA data structure

## 5. Optimization Function

The optimization (fitness) function $F_{ij}$ for *the j-th* individual of the *i-th* population is

$$F_{ij} = DM_{ij} + TC_{ij} + RC_{ij} + HC_{ij} + JLC_{ij} + LC_{ij} + BPC_{ij} + JAC_{ij} \quad (4)$$

We assume the above functions are added using some weights implicitly. SGA creates a new generation by using genetic operators based on fitness values of all individuals and the fitness is defined as a nonnegative value. *An* individual with higher fitness value has a higher probability of being selected for mating (crossover) and mutation. The first five functions *on* the right side of (4) *rre* calculated from each individual, whereas the last three functions (CC) are *can*-straints connecting all populations.

Dexterity Measure $DM_{ij}$ allows a choice of one optimal manipulator among various candidates that satisfy all constraints. One of dexterity measures that can be used for design is the relative manipulability [9]. This is a scale-independent mas= and is defined as

$$DM_{ij} = (n\sqrt{J_{ij}J_{ij}^T}) / \left(\sum_{k=1}^{n} l_{ijk}\right)^2 \tag{5}$$

*where* $J_{ij}$ is the Jacobian matrix the j-th individual of the i-th population and $l_{ijk}$ is the k-th link length of the j-th individual of the i-th population.

Task Constraint $TC_{ij}$ is a constraint to take m account specific task requirements that cannot be satisfied by the *otha* constraints in this paper. Any task which *can* be expressed by dimension, pose and bare position is a candidate for *TC*. For example, all features of velocity/force ellipsoids can be a *powerful* tool to guarantee dexterity of a manipulator [2][15]. In *this* paper, we do not use these features for our example *(crank* turning). *However,* for a complicated task DM alone is not enough to define dexterity over the whole trajectory and more detailed task specification using features of velocity/force ellipsoid are desirable. *A* simple obstacle avoidance constraint can be a *TC*.

Reachability Constr.int $RC_{ij}$ is a constraint used as a substitute of the inverse kinematics. When the i-th task point is at $X_i$ and the end-effector of the *j-th* individual at $E_{ij}$, the penalty for this individual is

$$RC_{ij} = |X_i - E_{ij}| \ . \tag{6}$$

Heuristic Constraint $HC_{ij}$ may be turned off for special tasks such as obstacle avoidance in which an outer link can be longer than an *inner* link *to* avoid obstacles close *to* the base. Since for spatial manipulators. some link length may become zero, HC does not compare this link, but compares the *other* nonzero links. *HC* can be *expressed* as

$$HC_{ij} = \sum_{k=1}^{n-1} \max(0,\ l_{ijk} - l_{ij(k+1)}) \tag{7}$$

Jdnt Limit Constraint $JLC_{ij}$ takes into account joint range availability. When the maximum $\theta_{k,max}$ and the minimum $\theta_{k,min}$ joint limits are given, a simple formulation of JLC can be expressed as

$$JLC_{ij} = \sum_{k=1}^{n} [\max(0,\ \theta_{ijk} - \theta_{k,max}) + \max(0,\ \theta_{k,min} - \theta_{ijk})] \tag{8}$$

when $\theta_{ijk}$ is the *k-th* pint angle of the *j-th* individual of the *i-th* population.

Link Length Constraint $LC_{ij}$ compares the k-th link length of the j-th individual of the i-th population with the average *k-th* link length of the other populations and can be written as

$$LC_{ij} = \sum_{k=1}^{n} \left| l_{ijk} - \left(\sum_{h=1}^{t} l_{hk} - l_{ik}\right) / (t-1) \right| \tag{9}$$

when $l_{ik} = \left(\sum_{j=1}^{m} l_{ijk}\right)/m$ is the average of the *k*-th link length for the i-th population.

Base Position Constraint $BPC_{ij}$ compares base position of the j-th individual of the i-th population with average base position of the other populations and can be written as

$$BPC_{ij} = \left| X_{0,ij} - \left(\sum_{h=1}^{t} \bar{X}_{0,h} - \bar{X}_{0,i}\right) / (t-1) \right| \tag{10}$$

where $\bar{X}_{0,i} = \left(\sum_{j=1}^{m} X_{0,ij}\right)/m$ is the average base position of the *i*-th population.

Joint Angle Change Constraint $JAC_{ij}$ is to prevent *an* abrupt change of joint angles between tow adjacent task points. The first order continuity condition, where we consider two neighboring (before and next) joint angles, *can* be expressed as

$$JAC_{ij} = \sum_{k=1}^{n} |\theta_{ijk} - (\bar{\theta}_{(i-1)k} + \bar{\theta}_{(i+1)k})/2| \tag{11}$$

where $\bar{\theta}_{ik} = \left(\sum_{j=1}^{m} \theta_{ijk}\right)/m$ is the average of the *k-th* joint angle for the i-th population.

## 6. Progressive Design

In this section, we introduce a framework called Progressive Design *where* we find an optimal solution based on coarse-fine approach. *Progressive* Design reduces the number of variables and the *search space* progressively. while its task space increases from finite number of task points in the beginning of design to the whole task space in the last step of control. Progressive Design consists of the following four steps:

(1) Kinematic Design

(2) Prototype

(3) Planning and

(4) Kinematic Control.

In *the* first step. we derive an optimal value of design variables corresponding *to* dimension, pose and base position. During this first step, we don't need to include the *RC* for design of planar manipulaton since *the* reachability is satisfied automatically by connecting the last joint (design variable) and the corresponding task point.

The second step is composed of two stages. The first stage is to choose link modules from a stock of the RMMS based on the result of the first step of design. When building a real manipulator, we may have to choose a slightly different dimension due to the limited number of available dimensions. In the second stage, therefore, we need to optimize pose and base position as variables at the same task points with a manipulator chosen in the first stage. The search space for pose and base position can be reduced considerably without losing possible optimal values by using the result of the first step as center points for these variables. We need RC but don't need LC and HC since the dimension of a manipulator is decided in the first stage of the second step. The result of the second step is used for fine planning (third step) with more task points (sub–task points) as shown in Fig. 7.

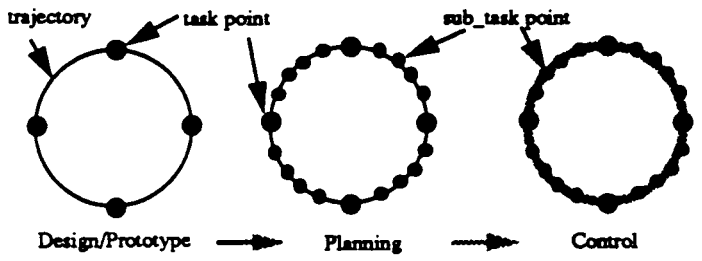


Fig. 7. Progressive Design

The third step is to interpolate two adjacent task pomts while satisfying the given trajectory at finite number of sub_task points. BPC is not necessary any more since we use the optimal base position from the second step. The only remaining variable is the pose (joint angles), whose search space can be reduced drastically. As a center point of the search space for each joint angle at all sub_task points, we use the linear interpolation of two boundary poses determined in the second step. We check any possible violations of constraints at sub–task pomts. If a violation is detected at some sub_task points. we return to the first step. This time, a set of task points for design is modified to reflect the regions w h m violations are detected. This step corresponds to the fine planning, whm as the second step to the coarse planning. Fig. 8 shows when a COND.int is applied at each step of the Progressive Design.

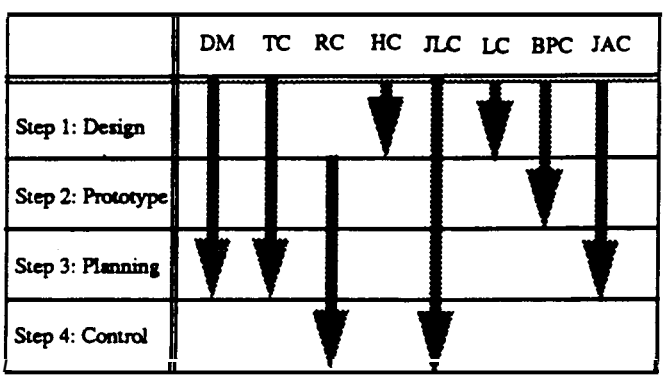


Fig. 8. Constraints

The last step (kinematic control) is to interpolate continuously between two adjacent sub–task points and to justify design and planning of the first three steps. Here again, by connecting two adjacent sub–task points. we check any possible violation of constraints such as JLC along the given trajectory. If violation is detected at some points, then we go back to the first step with a new set of task points reflecting regions where violations are detected We do not vv the MPGA in this step. This step is pure trajectory following problem where initial and final poses are determined in the previous step. Assume that $\theta_i$ and $\theta_{i+1}$ are joint angles at two adjacent task points (including sub_task points) and we want to move through this interval by $n_1$ steps. For non-redundant manipulators, this step is straight forward because we can use the RMRC (resolved motion rate control). However. for redundant manipulators, we need to control the null motion (self motion) to &@ boundary conditions. One of the simplest control algorithms for redundant manipulators is to use the pseudoinverse approach. Our pseudoinverse approach is different from others [12][13], since our problem is a two point boundary value problem. For small displacement dx at the end-effector, the desired point displacement is obtained as

$$d\theta = J^{+}\, dx + (I - J^{+}J)\, k \qquad (12)$$

where $J^{+}$ $(=J^{T}(JJ^{T})^{-1})$ is the Moore-Penrose pseudoinverse [20], $I$ the identity matrix and $k$ an arbitrary vector. The magnitude of net motion is determined by the first term of (12) and the magnitude of null motion is determined by the vector $k$. We have to choose this vector such that the final joint angle is satisfied at the $n_1$-th step. In addition, a desirable behavior of the null motion is that the magnitude is continuous and almost constant as the end-effector approached the end boundary. The vector $k$ is determined by

$$k = k_1 \cdot j/n_1 \qquad (j=1, \ldots n_1) \qquad (13)$$

where $k_1 = (-V\,|\theta - \theta_{i+1}|^2)/2$. The vector $k_1$ approaches to 0 and $j/n_1$ increases to 1 as $\theta$ approaches $\theta_{i+1}$. Thus, the magnitude of $k$ is almost constant through the trajectory between two boundary joint angles. In the following section. we present results of our MFGA and Progressive Design for the crank turning problem.

## 7. Results of Progressive Design

Fig. 9 shows the variance of $F_{1,max}$ (the best fitness for the first task point) versus generations. Since a GA works only with positive values, the negative fitness in this figure was internally shifted to positive values during computation

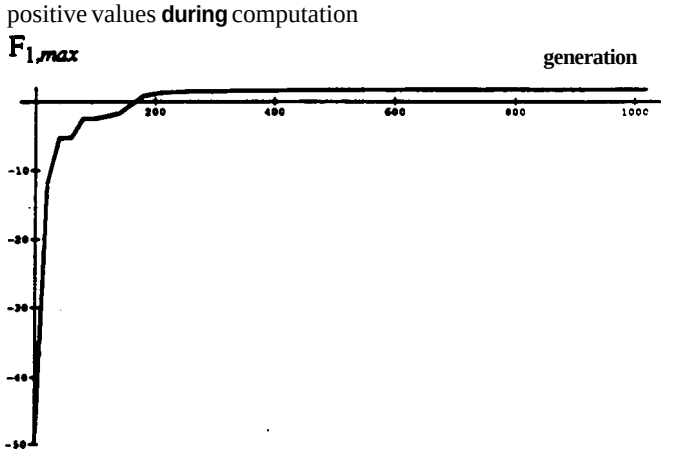


Fig. 9. $F_{1,max}$ vs. generation

Fig. 10 shows the variance of total link length for four task points which are constrained by LC ($l_i$ = total link length of the individual which has the best fitness. $F_{i,max}$ among the i-th popula-

tion). The total link lengths at four task points are not exactly the same even at generation=1000. This can be overcome by trying more generations. Instead of more generations in the first step, we choose the average link lengths in the beginning of the second step (prototype). Therefore, from the first step (design) we decide only dimension (link length) and results for other variables (pose and base position) are used to reduce search space in the second step. The result of prototype is shown in Fig. 11. In general, pose and base position obtained in the second step is better than those of the first step, since the second *step* has less constraints to be satisfied. We don't need HC and LC, whereas we must include RC in the second step because we have a manipulator determined in the first step. The optimal base position is determined in this step. But, for our example. the base position is assumed to be *given* at the origin for Simplicity.

In the Third step. we do fine planning based on the result of the second step. constraints used for the third *step* are almost the same as those for the second step except BPC. as shown in Fig. 8. Four sub_task points are equally spaced between two adjacent task points as in Fig. 7. The result of the third step (planning) is shown in Fig. 12. The poses at 20 task points (4 task pomt and 16 sub_task points) are shown in configuration space m Fig. 13. Finally the result of the last step (kinematic control) are obtained by using the pseudoinverse approach and shown in Fig. 14.

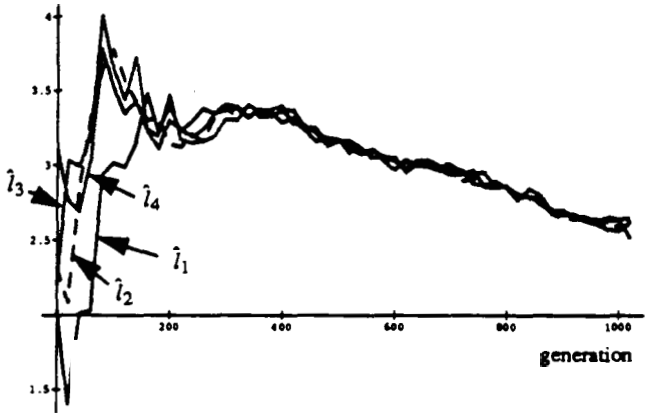


Fig. 10. Link length vs. generation

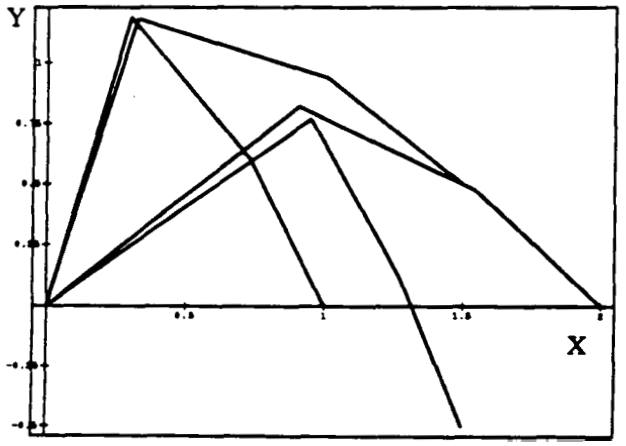


Fig. 11. Optimal dimension and poses at task points

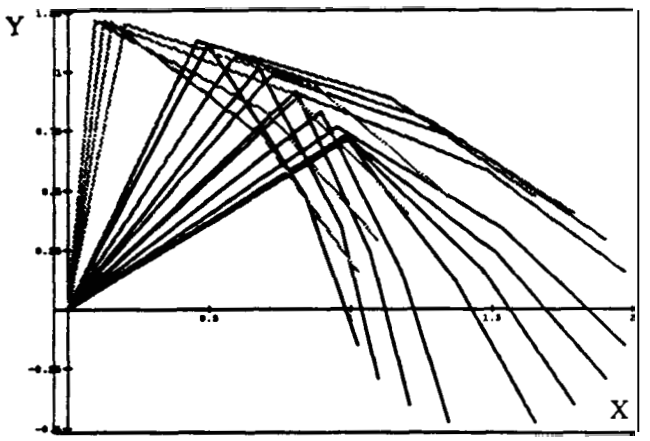


Fig. 12. Results of planning at sub–task points

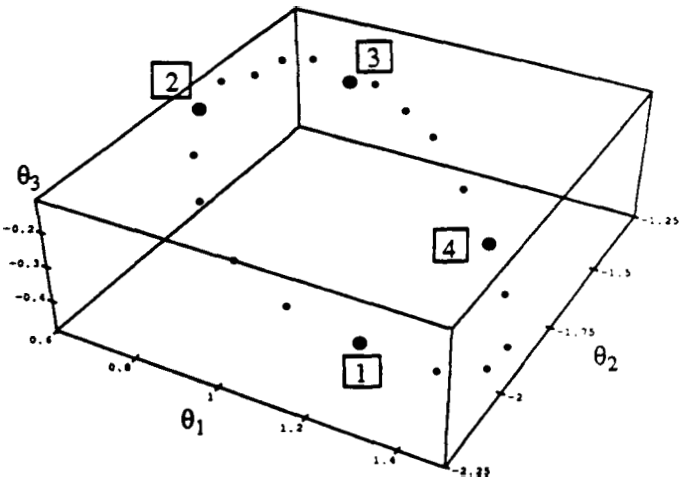


Fig. 13. Results of prototype and planning *in* configuration space

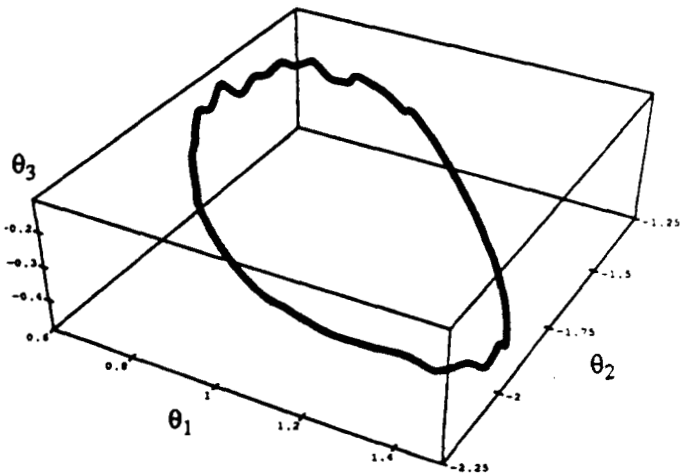


Fig. 14. Results of control in configurationspace

## 8. Summary

In this paper, we introduced the Multi-Population Genetic Algorithm (MPGA) which is a parallel implementation of optimization problem. GAs including our MPGA have no requirement for conti-

nuity in the derivatives, so virtually any fitness function can be selected for optimizing. Different from conventional methods. GAs use population in which individuals are selected based on their fitness value. Our MPGA has the same number of optimization functions as the number of task points. Each optimization has its own objective function and they are modified by CC (connecting constraints). Due to this implementation, the complexity of TBD (task based design) was maintained almost constant, independently of the number of task points. In addition, by selecting proper constraints from a set of CC, the same MPGA could be applied for manipulators with mobile bases, or with links with prismatic joint.

In addition, we proposed a framework for TBD called Progressive Design. in which we design progressively by using coarse-fine approach. The Progressive Design is composed of four steps: design, prototype, planning and control. With a design of an optimal 3 DOF redundant planar manipulator for crank turning, we showed how our MPGA works well and presented results from each step of the Progressive Design. The framework for Progressive Design can also be used for analysis problems such as path placement and workspace design by using the last three steps (prototype, planning and control) since a manipulator is g i v a

## ACKNOWLEDGMENT

This research was funded in part by NASA under grant NAG-1-1075, DOE under grant DE-F902-89ER14042, the Department of Electrical and Computer Engineering. and The Robotics Institute. Carnegie-Mellon University.

## REFERENCES

[1] J.W. Burdick IV *Kinematic Analysis and Design of Redundant Manipulators.* Ph.D. thesis, Department of Computer Science, Stanford University, March 1988.

[2] S. Chiu, "Kinematic Characterization of Manipulators: An Approach to Defining Optimality". IEEE int. conf. on Robotics and Automation. pp. 828-833.1988.

[3] S. Chiu, 'Task Compatibility of Manipulators Postures". The International Journal of Robotics Research, vol.7, No. 5, pp. 13-21. October, 1988.

[4] Y. Davidor, *Genetic Algorithms and* Robotics, World Scientific: an international publisher, New Jersey, 1990.

[5] D.E. Goldberg, Genetic *Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley publishing company, 1989.

[6] J.S. Hemmerle, Optimal Path *Placement for Kinematically Redundant Manipulators,* Ph.D. Thesis, Department of Mechanical Eng. Carnegie-Mellon University, May. *1989.*

[7] R.S. Hartenberg and J. Denavit, *Kinematic Synthesis of Linkages,* McGraw-Hill Book Company, 1964.

[8] Jim-Oh Kim and P. Khosla, "Dexterity Measures for Design and Control of Manipulators", IEEE/RSJ int. workshop on Intelligent Robots and Systems (IROS'91), Osaka, Japan, Nov. 2-5, 1991.

[9] A. Liégeois, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms", IEEE transaction on Systems. Man, and Cybernetics. Vol. SMC-7(12), pp. 868-871.1977.

[10] Y. Nakamura, *Kinematic Studies on the Trajectory Control of Robot Manipulators, Ph.D. dissertation,* Kyoto University. June, 1985.

[11] B. Nelson. K. Pedersen and M. Donath, "Locating Assembly Tasks in a Manipulator's Workspace", IEEE int. conf. on Robotics and Automation, pp. 1367-1372, 1987.

[12] J. Alfonso Pamanes G. and Said Zeghloul, "Optimal placement of Robotic Manipulators Using Multiple Kinematic Criterion", IEEE int. conf. on Robotics and Automation, pp. 933-938, Sacramento, CA, 1991.

[13] C.J. Paredis, and P.K. Khosla, "An Approach for Mapping Kinematic Task Specifications into a Manipulator Design", Fifth International Conference on Advanced Robotics, Pisa, Italy. June, 1991.

[14] J.K. Parker, A.R. Khoogar and D.E. Goldberg, "Inverse Kinematics of Redundant Robots using Genetic Algorithms". IEEE int. conf. on Robotics and Automation, pp.271-276,1989.

[15] J.D. Schaffer, "Multiple Objective Optimization With Vector Evaluated Genetic Algorithms", Proceeding of International Conference on Genetic Algorithms and Their Applications",pp. 93-100.1985.

[16] D.E. Schmitz, P. Khosla and T. Kanade, 'The CMU Reconfigurable Modular Manipulator System". Proceedings of the 18-th ISIR, Australia, 1988.

[17] T. Yoshikawa, "Manipulability of Robotic Mechanisms", Robotics Research: The second int. symp.. pp. 439-446, 1985.