2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

A Genetic Algorithm Approach to solve for Multiple Solutions of Inverse Kinematics using Adaptive Niching and Clustering

Saleh Tabandeh, *Member, IEEE*, Christopher Clark, *Member, IEEE*, and William Melek, *Member, IEEE*

***Abstract*— Inverse kinematics is a nonlinear problem that may have multiple solutions. A Genetic Algorithm(GA) for solving the inverse kinematics of a serial robotic manipulator is presented. The algorithm is capable of finding multiple solutions of the inverse kinematics through niching methods.**

Despite the fact that the number and position of solutions in the search space depends on the the position and orientation of the end-effector as well as the configuration of the robot, the number of GA parameters that must be set by a user are limited to a minimum through the use of an adaptive niching method. The only requirement of the algorithm is the forward kinematics equations which can be easily obtained from the link parameters and joint variables of the robot. For identifying and processing the outputs of this GA, a modified filtering and clustering phase is also added to the algorithm. The algorithm was tested to solve the inverse kinematic problem of a 3 degree-of-freedom(DOF) robotic manipulator.

I. Introduction

Path planning and control of robot manipulators require mapping from end effector cartesian space coordinates into corresponding joint positions. This mapping is referred to as the inverse-kinematics (IK) of the robot. Finding the position and orientation of the end-effector from the joint angles is called the forward-kinematics (FK) problem. Forward-kinematics of a robot manipulator can easily be solved by knowing the link parameters and joint variables of a robot, while the inverse kinematics is a nonlinear and configuration dependent problem that may have multiple solutions[1].

For some robot configurations the closed-form solution of the IK exist (e.g. PUMA, FANUC, etc.) [2], [1]. These solutions only exist for a few robot configurations and can not be obtained for all robots. Another approach to the IK problem is to use numerical methods [3], [4]. In numerical methods, the algorithm converges on the solution that is closest to the initial starting point of the algorithm. Since most of these methods are divergence based, they are vulnerable to local optimums. To solve IK for a redundant robot, a genetic algorithm was used in [5]. In that work, the focus is on finding the best solution among all the possible solutions that minimize the joint displacements. Most related is research from [6] and [7], where a fitness sharing niching method was used to find multiple solutions for a 2DOF robot. A prominent feature of these works is the use of real-coded GA in conjunction with tournament selection. A drawback is they suffer from the need to set numerous unknown parameters. These parameters depend greatly on the nature of the search space and are different from one robot configuration to another.

In this paper, an adaptive niching method to solve the IK problem is proposed. This algorithm is based on a minimizing GA to find the joint angles that produce the least positioning and orientation error of the end effector from those of the desired values. The contributions of this paper can be highlighted as:

- By using a niching method, the algorithm is able to find all the possible solutions of the IK problem.
- Unlike the other Niching Genetic Algorithms for solving IK, this algorithm requires few parameters to be set with the prior knowledge of the problem. This feature allows the algorithm to be used for solving IK of any robot configuration.
- A Real coded Simulated Binary Crossover [12] is used. This feature enables the algorithm to search in a continuous joint space, not a binary one.
- A formulation for incorporating the joint limits in the simulated binary crossover is presented.
- A modified Adaptive Niching method [10] was used to increase the algorithm speed without sacrificing the performance.
- A filtering and clustering method to find the solution regions is presented.
- Performance of the algorithm was tested in solving for 4 solutions of the IK problem of a 3DOF robot.

This work consist of six sections. Section II explains the IK problem and the objective function. In section III, the conventional niching methods and the adaptive niching method are explained. In section IV the proposed algorithm to solve the IK problem is explained. Section V describes the filtering and clustering method. Finally, the results of running the algorithm for positioning of a 6DOF robotic manipulator is illustrated in Section VI.

II. Kinematics and Objective Function

A. Forward and Inverse Kinematics

In Robotics, the problem of calculating the position and orientation of the end effector of a robot from the joint space coordinates is called the Forward Kinematics problem. The solution to this problem can be found by defining the position and orientation of each link frame with respect to the previous link frame as a function of the joint variable. This relative position and orientation of two consecutive links, (according to the Denavit-Hartenberg convention), is described by a Homogenous Transformation with the form:

0-7803-9487-9/06/$20.00/©2006 IEEE

$$\mathbf{T}_{i-1}^{i}(\theta_i) = \left(\begin{array}{c|c} \mathbf{R}_{i-1}^{i}(\theta_i) & \mathbf{P}_{i-1}^{i}(\theta_i) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right) \tag{1}$$

in this equation $\mathbf{R}_{i-1}^{i}(\theta)$ and $\mathbf{P}_{i-1}^{i}(\theta)$ describe the relative orientation and position of the frame respectively. The parameters of these matrices can be extracted from the physical shape and configuration of any robot. To calculate the position and orientation of the end-effector ($T_{oe}(\theta_1, \theta_2, \cdots \theta_n)$) with respect to the base of the robot for an arbitrary $[\theta_1,\ \theta_2,\ \cdots\ \theta_n]$ the transformation will be:

$$\begin{aligned} \mathbf{T}_{oe}(\theta_1,\ \theta_2,\ \cdots,\ \theta_n) &= \prod_{i=1}^{n} \mathbf{T}_{i-1}^{i}(\theta_i) \\ &= \left(\begin{array}{c|c} \mathbf{R}_{oe} & \mathbf{P}_{oe} \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right) \end{aligned} \tag{2}$$

On the other hand, the inverse of the FK, the inverse kinematics, is the problem of finding $[\theta_1\ \theta_2\ \cdots\ \theta_n]$ from $\mathbf{T}_{oe}$. This problem is a mapping from the 3D task space to the joint angle space and usually has more than one solution. For instance, a PUMA560 robot may have 4 or 8 Inverse Kinematics solutions [1].

B. Objective Function

In this paper, the approach to solving the IK problem is to convert it to a minimization problem and then utilize a GA to find all the global minimums of the problem.

In GAs, a measure of the fitness of each individual is required to select the most potent individuals for crossover operation. This measure can be defined as the difference between the end-effector position and orientation of the individual and that of the desired location.

To measure the position error we use the Euclidean norm of the difference between the end-effector position of each individual and that of the desired point in the cartesian space.

$$\mathbf{E}_P = \|\mathbf{P}_{desired} - \mathbf{P}_{ind}\| \tag{3}$$

in this equation, $\mathbf{P}_{desired}$ and $\mathbf{P}_{ind}$ are the position vectors of the end effector in the desired point and the individual respectively. To find the orientation error the Euler angles of the hand, α , β and γ, are used which give the same orientation as $\alpha+\pi$, $-\beta$ and $\gamma-\pi$. The Euler angles of the desired points can be obtained directly from the homogenous transformation of the desired position/orientation of the end-effector. For the individuals, these values can be calculated from $\mathbf{R_{oe}}$ in (2). The orientation error formulation will be:

$$O_d = \begin{bmatrix} \alpha_d \\ \beta_d \\ \gamma_d \end{bmatrix}$$

$$\mathbf{E}_O = \min(\|O_d - \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}\|, \|O_d - \begin{bmatrix} \alpha_i + \pi \\ -\beta_i \\ \gamma_i - \pi \end{bmatrix}\|) \tag{4}$$

where O_d is the desired orientation defined in Euler angles, and $[\alpha_i\ \beta_i\ \gamma_i]$ are the Euler angles of the individual. With (3) and (4), the objective function for the minimization can be written as:

$$O.F. = w_p\, \mathbf{E}_P + w_o\, \mathbf{E}_O \tag{5}$$

w_p and w_o are the weighting factors and can be used to normalize their corresponding values. We recommend a value for w_p as:

$$w_p = \frac{1}{2P_{max}} \tag{6}$$

which P_{max} is the maximum reach of the robot. To choose w_o, the maximum and minimum allowable joint angles,q_{max} and q_{min} can be utilized to compute the following:

$$w_o = \frac{1}{q_{max} - q_{min}} \tag{7}$$

III. Background on the Niching Techniques

A Genetic Algorithm, through Selection, Cross-over and Mutation operations, finds the individuals that have the best fitness values and combines them to produce individuals that offer better fitness values than their parents. This process continues until the population converges around the single individual that have the best fitness value. However, in a large number of applications with multiple global (or local) optimums, identification of more than just one promising point per generation is required. For this purpose, niching methods modify the simple GA by changing the fitness value in a way to encourage convergence around multiple solutions in the search space[8]. In this section, we will briefly review the conventional niching techniques. Then the adaptive niching via coevolutionary sharing technique will be explained in greater detail.

A. Conventional Techniques

The sharing method[9], which is probably the most well-known niching technique, decreases the fitness value of the individuals in densely populated areas and as a result decreases their chance of being selected. The sharing method, with a complexity of $O(N^2)$, is computationally expensive. Also, in sharing methods a priori knowledge of the problem is required to tune the numerous parameters of the algorithm including niche radius parameter [8].

Moreover, the algorithm is more suitable for problems with equidistant niches. The limitation of this technique for our application is that prior to solving the problem no knowledge about the relative position of the solutions in the search space exists. In addition, number of niches changes for different configurations of robots. These solutions will also change with the position of the end-effector and are completely different from one robot configuration to another. Crowding methods, another approach to niching includes Standard Crowding, Deterministic Crowding and Restricted Crowding. These methods have a complexity of $O(N)$, however do not have the robustness of sharing methods [8].

B. Adaptive Niching via Coevolutionary Sharing

As noted, one of the disadvantages of Fitness sharing is the need to set the niche radius σ_s as accurately as possible. This requires a priori knowledge of the solutions of the problem, which is not available in IK problems.

To address this drawback, Goldberg and Wang introduced an adaptive niching algorithm via coevolutionary sharing(CSN) [10]. This algorithm is loosely based on the economic model of *monopolistic competition*, in which businessmen try to position themselves, subject to a minimum distance, among geographically distributed customers to maximize their profit. In CSN two populations, businessmen and customers, work to maximize their separate interests.

These two populations interact with each other according to the economic model. Businessmen try to maximize their profit by finding locations with more customers, while customers try to shop from businessmen with better service, i.e. the closest businessman that is least crowded.

For the customer population, fitness function modification resembles that of the standard fitness sharing. If at any generation t, customer c is being served by the businessman b who is the closest businessman, and that b is serving a total $m_{b,t}$ customers, the shared fitness of c is calculated by:

$$f'(c) = \left.\frac{f(c)}{m_{b,t}}\right|_{c \in C_b} \tag{8}$$

where C_b denotes the customer set, whom businessman b serves. In other words, each customer shares its fitness value with the other customers of the same businessman. A Stochastic Universal Selection scheme and single point crossover has been used in the original paper [10].

The tendency of the businessmen is to place themselves in regions that are more densely populated by customers, subject to keeping a minimum distance of d_{min} from the other businessmen. The fitness value of the businessmen is simply the sum of the fitness values of its customers:

$$\phi(b) = \sum_{c \in C_{b,t}} f(c) \tag{9}$$

In their paper, Goldberg and Wang used only a mutation operation for the businessmen population. If a mutated individual: (1) is at least d_{min} far from other businessmen, and (2) is an improvement over the original businessman, it will replace the original one. If not, the mutation operation will be repeated up to a multiple of the businessman population. An imprint operation was also suggested which chooses new businessman randomly from the customer population instead of producing them by mutation. With imprint, the evolution of the businessman population will benefit from knowledge of the search space acquired by customers, and will not be completely random. If the chosen customer could satisfy the above two conditions it will replace the businessman. To find out if the selected customer is an improvement, the assignment of the customers to the businessmen must be repeated. To accomplish the assignment, the calculation of the customer distances from the members of the new set of businessmen is required.

Although this algorithm is not as sensitive to the values of d_{min} as the conventional fitness sharing technique is to σ_s, choosing an appropriate d_{min} is still of considerable importance.

CSN has been applied to a multi-objective softkill-scheduling problem with the imprint operation [11]. Rank based selection, elitist recombination, and non-dominated sorting are some of the prominent features of that work.

IV. Adaptive Sharing to solve IK problem

To solve the IK problem, the algorithm must be fast enough to evaluate the solution for a very large space (e.g. A 6 dimensional space for a PUMA or general purpose robots). It must be able to find multiple solutions for all the possible poses of the end-effector. This algorithm must also be able to solve the IK for any robot configuration by the knowledge of the FK equations. In this section, the proposed algorithm to solve the IK is explained.

A. The Algorithm

An overview of the proposed algorithm can be seen in Table. I. A detailed explanation of each of the steps is as follows:

1) Initialization: Two independent populations for Customers and Businessmen are randomly created. Each individual is consisted of n joint angles corresponding to the n joints of the robot:

$$Q = [q_1\ q_2\ \cdots\ q_n]^T \tag{10}$$

where $q_1,\ q_2,\ \cdots,\ q_n$ are all real numbers.

To allow more individuals to be associated to the IK solutions which are close to the reachable joint space borders, $[q_{min}$,$q_{max}]$, an extended range of permissible angles,$[q_{min} - \psi$,$q_{max} + \psi]$, is used.

After the Customer and Businessmen populations are randomly generated, the initial d_{min} is calculated using the following equation:

$$d_{min_{start}} = \frac{\kappa(\, q_{max} - q_{min})}{1 + \sqrt[n]{b}} \tag{11}$$

Step	Desciprtion
1	Randomly Initialize the Customer Population Randomly Initialize the Businessman Population Initialize d_{min} with $d_{min_{start}}$
2	Customers Raw Fitness Value Calculation Businessmen Raw Fitness Value Calculation Assignment of Customers to the closest Businessman Customers Shared Fitness Value Calculation
3	Forming the customers parent pool by Tournament Selection Adding the fittest businessmen to the pool
4	Customer Crossover
5	Businessmen Imprint
6	Updating d_{min}
7	If the termination criterion is not reached return step 2

TABLE I
THE IK ALGORITHM

where n, b, and κ correspond to the degree of freedom (i.e. number of joints), the number of businessmen, and the fitting index respectively. (11) uses $\kappa > 1$ multiplied by the distance between businessmen if they are spread equidistantly over the n dimensional joint space. That is, $d_{min_{start}}$ should be greater than the average distance between businessmen.

2) Fitness value calculation: The fitness values $f(c)$ and $f(b)$ of customers and businessmen are calculated using (5). Then, customers are assigned to the closest businessman, where closeness is measured using the Euclidean distance between customers and businessman. Based on these assignments, the shared fitness values $f'(c)$ of customers are calculated with the following equation. This equation is

$$f'(c) = f(c) \cdot m_{b\,,t} \tag{12}$$

3) Selection: Since Tournament Selection does not require a priori knowledge of the problem, it was used to create a parent pool.

From the customers, n_t individuals are selected at random. Of this subset, the customer with the least fitness value (error) is transferred to the parent pool. Choosing $n_t \geq 2$ individuals encourages faster algorithm convergence.

In this step, b_t businessmen with the best fitness values ($f(b)$) are also added to the parent pool. By letting some businessmen in the pool, there will be an increase customers moving towards businessmen with high fitness values but few customers. These businessmen might not otherwise attract customers.

4) Customer crossover: In this step, the new generation of customers are produced from the parent pool that was created in the previous step. Simulated Binary Crossover[12] is used for the crossover operation. Details can be found in section IV-B.

5) Businessmen imprint: Each businessman is compared with an individual randomly selected from the newly formed parent pool. If this individual is an improvement over the businessman and was d_{min} away from all the other businessmen, it will replace the corresponding businessman.

To check for improvement, we define the following value:

$$F(x) = \left. \frac{f(x)}{N_{d_{min}}} \right|_{x \in P \vee B} \tag{13}$$

where P and B are the parent pool and the businessman populations respectively. $N_{d_{min}}$ denotes the number of customers in the distance d_{min} from individual x and $f(x)$ is the raw fitness value of individual x. $F(x)$ is a measure of the potential of the neighborhood of x for being a solution region. Regions with higher customer densities and better fitness values (lower errors) at the center have lower $F(x)$. That is, individuals with lower $F(x)$ are recognized as better businessmen.

Use of $F(x)$ enables us to evade the computationally costly process of reassigning the customers to businessmen in the imprint step of the original algorithm [10].

For each businessman, this process is repeated n_{limit} times, or until it is replaced by a better candidate. Here n_{limit} is a multiple of the population number of businessmen.

As the algorithm progresses, d_{min} will decrease. Since the businessmen in close vicinity of the solution regions have a high concentration of customers around them, even with the decrease of d_{min}, they will still have customers in their local regions. Meanwhile businessmen further from solution regions will be left without customers and are forced to find better regions.

6) Updating d_{min}: d_{min} is closely related to the accuracy of the end solutions. Lower values of d_{min} bring flexibility to the businessmen to locate regions with better fitness values and more concentrations of solutions.

In step 1), d_{min} was set at its maximum value to prevent the GA from converging immaturely on only one niche. As the iterations continue, the niches begin to establish themselves around the solution points and the difference between their fitness values decreases.

In this step, d_{min} is decreased in small step sizes towards its lowest value in the last iterations. In the GA proposed here, the following function was used for updating d_{min} :

$$d_{min} = d_{min_{start}}(1 - \lambda \frac{t}{t_{max}}) \tag{14}$$

where t and t_{max} correspond to the current iteration and maximum iteration number. λ is the coefficient that defines how small d_{min} can become.

7) Check for the termination criterion: If the termination criterion is not satisfied the algorithm will return to step (2).

B. The continuous crossover operation

Crossover operation randomly selects two parents, P_1 and P_2, from the parent pool and produces two children, C_1 and C_2, from them. It has been shown that for continuous search spaces, real coded GAs are more suitable than binary coded algorithms [12]. In this paper, we use a Simulated Binary Crossover (SBX) [12] to apply the variable-by-variable crossover. The idea behind SBX is to create a random distribution of offsprings in the domain of real numbers. This distribution matches the distribution of the common binary crossovers. In other words, SBX uses a randomly generated number, $u_\beta(i)$ to produce a random expansion ratio $\beta(i)$ that defines how similar the offsprings are to their parents:

$$\beta(i) = \left| \frac{C_2(i) - C_1(i)}{P_2(i) - P_1(i)} \right|. \tag{15}$$

Crossover is carried out with the following steps:

1) Of the n joints, l joints ($l \leq n$) are randomly selected for the crossover operation. The rest of the joint angles will be transferred from the parents to the children, unchanged. In our implementation $l = 0.5n$ [13].
2) For each of the l joints angle selected in the last step, a random number, $u_\beta(i)$, is generated. The expansion ratio, β, is then calculated using:

$$\beta(i) = \begin{cases} (2u_\beta(i))^{\frac{1}{\eta+1}} & \text{if } u_\beta(i) \leq 0.5 \\ \left(\frac{1}{2(1-u_\beta(i))}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{16}$$

where η denotes the distribution index and can be any nonnegative real number. For small values of η, points far away from the parents have higher probability of being chosen, while with large values of η points closer to the parents are more likely to be chosen. A value of 2–5 produces a good estimate of the binary coded crossover [13]. In our algorithm, η initially has a small value (2) and with the progress of the algorithm it will increase(to around 5) to let the solutions fine tune into the centers of the solution regions.
When joint angles have physical limits, (as commonly found), (16) must be modified to produce offsprings that are located inside the joint limits. To accomplish this, the following method has been proposed. First, β_L and β_H are calculated from the following equation for each of the joint variables:

$$\begin{aligned} \beta_L(i) &= \frac{0.5(P_1(i) + P_2(i)) - q_{min}}{|P_1(i) - P_2(i)|} \\ \beta_H(i) &= \frac{-0.5(P_1(i) + P_2(i)) + q_{max}}{|P_1(i) - P_2(i)|} \end{aligned} \tag{17}$$

where $P_1(i)$ and $P_2(i)$ denote the ith variable of the two parents. Value of u_β is then updated as follows:

$$\beta_{limit}(i) = \begin{cases} \beta_L(i) & \text{if } \beta_L \leq \beta_H \\ \beta_H(i) & \text{otherwise} \end{cases} \tag{18}$$

$$\begin{aligned} k(i) &= \frac{1}{1 - \frac{0.5}{\beta_{limit}(i)^{\eta+1}}} \\ u_\beta(i) &= \frac{u_\beta(i)}{k(i)} \end{aligned} \tag{19}$$

This modification, by changing the probability distribution of $\beta(i)$, will guaranty that the produced children are inside the variable range. Since the expansion ratio of the children to parents is limited by the joint variable limits, (17) calculates the maximum allowable value of this parameter corresponding to each limit. (18) and (19) modify u_β in a way to set the probability of choosing a β less than $\beta_{limit}(i)$, equal to one. In other words, for any arbitrary u_β, the produced children will be in range $[q_{min}, q_{max}]$.
Finally, $\beta(i)$ is calculated from (16) with the updated $u_\beta(i)$.
3) In the last step children are produced from the following equation:

$$\begin{aligned} C_1(i) &= 0.5[\,(1 + \beta(i))\, P_1(i) \\ &\quad + (1 - \beta(i))\, P_2(i)\,] \end{aligned} \tag{20}$$

$$\begin{aligned} C_2(i) &= 0.5[\,(1 - \beta(i))\, P_1(i) \\ &\quad + (1 + \beta(i))\, P_2(i)\,] \end{aligned} \tag{21}$$

and will be placed in the new generation population.

V. Processing the output

The output of the GA is a set of points with high population density around the solution regions, and with lower concentration in the rest of the search space. In order to distinguish solution regions, a mechanism to detect the regions with high concentration of individuals and low error is required.

If the robot has 2 DOF, identifying these results in the 2D space can be accomplished by observation, which is not convenient if the GA is a building block for other software peripherals. Moreover, for robots with more DOFs (for example a 6 DOF PUMA), identifying these solution regions must be done in a 6 dimensional space, which is not possible by visual inspection. Hence, a robust algorithm for clustering the results is required.

In this section, an overview of the filtering and clustering method is presented.

A. Filtering

The fitness function of each individual is the orientation/position error from the desired value. It is convenient to

use the fitness function as a measure of filtering the results before the clustering.

In the filtering phase, individuals with high fitness value (Error) are rejected and individuals with lower fitness value are transferred to the clustering step.

B. Clustering

Since no priori knowledge about the number of solutions of IK exist, the number of solution niche or clusters is also unknown. To resolve this issue, Subtractive Clustering [14] is used to find niches. Subtractive Clustering is a one-pass algorithm for estimating the number of clusters and their centers for a set of data when the number of the clusters is unknown.

Subtractive clustering assumes that every point in the data set is a potential cluster center. This algorithm measures the potential of each of the points based on the density of the data set around it and then assigns the point with the highest potential a cluster center. It then removes all other points in the $R_{cluster}$ from the cluster center, and repeats the process until all of the points in the data set are within the radius of a cluster.

Choosing $R_{cluster}$ has a great effect on the number of clusters that are detected by the algorithm. The larger $R_{cluster}$, the less clusters detected. Because the GA has filtering and runs until a relatively good convergence is achieved, $R_{cluster}$ can be set to small values to detect all the solution regions with good precision.

Note that in the GA, in cases where $q_{min} = -\pi$ and $q_{max} = \pi$, if an IK solution is close to π or $-\pi$, a concentration of individuals close to the other end of the joint space(around $-\pi$ or π respectively) might form. For example, in a 2D search space, if one solution of the IK is $[-\pi + \varepsilon, \cdots]$, a concentration of individuals around $[\pi - \varepsilon, \cdots]$ might form in the population. In the clustering phase, these concentrations will be detected as two different regions. To counter this problem, the following equation is proposed for calculating the distance of the joint variables of two individuals:

$$d = \min(\ q_1 - q_2 \quad q_1 - (q_2 - 2\pi) \quad q_1 - (q_2 + 2\pi)\ ) \quad (22)$$

VI. Results: PUMA560 IK solution

The algorithm was used to solve the IK problem of the first three joints of a PUMA560. The reason that only the first three joints were used is that these joints are responsible for most of the positioning of the robot. The responsibility of the 3 distal joints, the spherical wrist joints, is only setting the orientation of the hand. In other words, the problem can be decoupled to only a positioning by the first three degrees of freedom and then an orientation for the wrist joints. The second part of this problem, orienting with the wrist, can be solved analytically, so here we try to solve the first part of the problem.

Parameter	Value
Customer Population	600
Businessman Population	30
κ	1.2
λ	0.9
q_{min}	$-\pi$
q_{max}	π
ψ	π
η	2-5
$R_{cluster}$	0.795
Filtering threshold	0.1
n_{limit}	90
n_t	10
b_t	15

TABLE II

Parameters used in the Genetic Algorithm

Case Number	x	y	z
Case 1	-0.3071	-0.5193	-0.0249
Case 2	0.4151	-0.6273	0.285
Case 3	0.5525	-0.3913	-0.5522

TABLE III

Test Hand Positions For Puma560

The parameters used to run the algorithm are shown in Table II. The algorithm was used to solve the IK problem for three different hand positions. The coordinate of these points in the cartesian space are given in Table. III.

Tables. IV, V, and VI show the actual solution of each case in comparison to the results of the algorithm. All these values are expressed in the joint space and are in radians. In Fig. 1, Fig. 2 and Fig. 3 the results of the algorithm and the actual solutions of the IK are shown. In these figures, the smaller circles denote the output of the filtering algorithm, while the larger circles denote the output of the clustering algorithm. The large filled circles are the actual solutions and the crosses represent the position of the businessmen.

As can be seen from Fig. 1 and Fig. 2, some of the individuals had converged on the points close to the limits of the joint angle on a position mirror to the actual solution point. With the modifications to the subtractive clustering that was explained in section V-B, these points were assigned to the same cluster they tried to converged on. In the two

Actual Solution	q_1	q_2	q_3
1	0.7854	2.3562	0.1309
2	0.7854	-2.2711	3.1046
3	-1.8534	0.7854	3.1046
4	-1.8534	-0.8705	0.1309
Algorithm Solutions	q_1	q_2	q_3
1	0.8015	2.3975	0.0554
2	0.7256	-2.3149	3.0705
3	-1.8453	0.8953	3.0716
4	-1.8776	-0.8388	0.0722

TABLE IV

Actual solutions and algorithm output for Case 1

Actual Solution	q_1	q_2	q_3
1	1.9546	2.3562	-0.6914
2	1.9546	-3.0941	-2.3562
3	-0.7854	0.7854	-2.3562
4	-0.7854	-0.0475	-0.6914
Algorithm Solutions	q_1	q_2	q_3
1	1.9638	2.3363	-0.7530
2	1.9357	-3.0838	-2.4796
3	-0.7775	0.7547	-2.2749
4	-0.7791	-0.0356	-0.7024

TABLE V

ACTUAL SOLUTIONS AND ALGORITHM OUTPUT FOR CASE 2

Actual Solution	q_1	q_2	q_3
1	2.3019	-2.5337	-1.3464
2	2.3019	-2.3562	-1.7012
3	-0.3927	-0.6079	-1.7012
4	-0.3927	-0.7854	-1.3464
Algorithm Solutions	q_1	q_2	q_3
1	2.2651	-2.4617	-1.5835
2	2.3038	-2.3154	-1.8715
3	-0.4047	-0.6220	-1.6732
4	——-	——-	——-

TABLE VI

ACTUAL SOLUTIONS AND ALGORITHM OUTPUT FOR CASE 3

figures convergence regions that are actually the same have been distinguished with arrows.

In case 3, the solution regions are very close to each other. As a result, the algorithm may confuse the two very close regions as one cluster and converge on them as a single solution. In Fig. 3, which is the output of case 3, the algorithm has detected 3 of the 4 possible solutions. As can be observed from Table. VI, Two of these found solutions are in the close vicinity of each other. The other solution that was located by the algorithm was actually two very close solutions of the IK.

Table. VII the errors in detecting the different niches are shown. These results were obtained after 15 iterations. For each of the points, the algorithm reached the results on an average of 60 seconds, with MatLab on a P4 processor with 512MB RAM. If higher precisions are required, the algorithm can be run for longer iterations to converge more on the niches. However, increasing the iterations brings the risk of loosing some of the niches.

In Fig. 4 the decrease of error of each case is shown as the generations evolve.

Fig. 5 shows that the average of error of the individuals is also decreasing.

Case number	Niche.1	Niche.2	Niche.3	Niche.4
1	0.049	0.02	0.049	0.035
2	0.066	0.101	0.019	0.037
3	0.005	0.031	0.041	—-

TABLE VII

THE ERRORS OF CASES 1–3 (M)

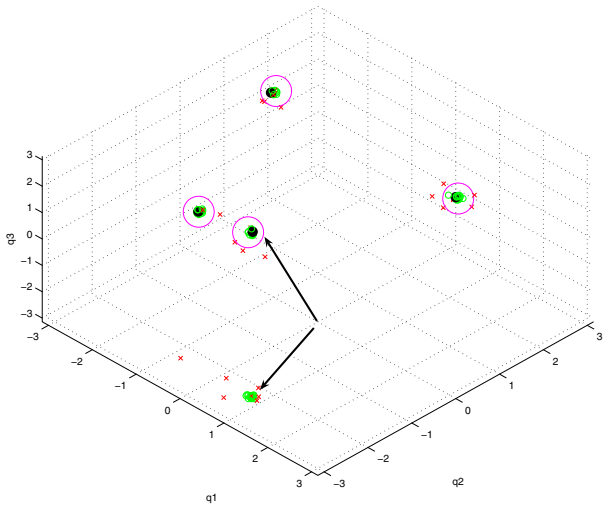

Fig. 1. Output of the Algorithm for case 1

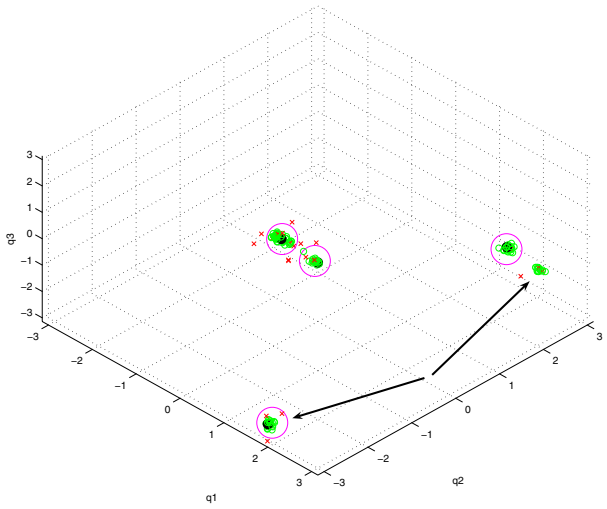

Fig. 2. Output of the Algorithm for case 2

VII. CONCLUSION AND DISCUSSION

An adaptive niching strategy was presented and used to solve for multiple solutions of the IK problem. Since this algorithm uses the minimum preset parameters, it can be generalized to solve IK of a robot with unknown degrees of freedom and configuration. The algorithm is benefiting from real coding, adaptive minimum distance setting of businessmen and adaptive real coding distribution index. To process the results, a subtractive clustering algorithm was also modified for the application. It was shown that the algorithm works with good precision/speed performance in the whole search space and for all the possible hand position/orientations in space.

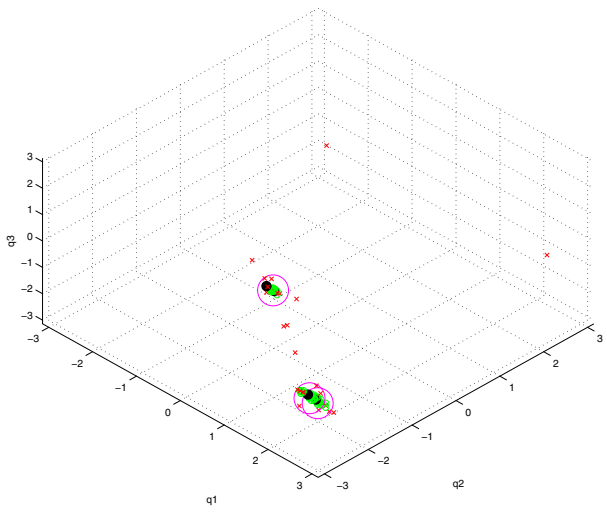

Fig. 3. Output of the Algorithm for case 3

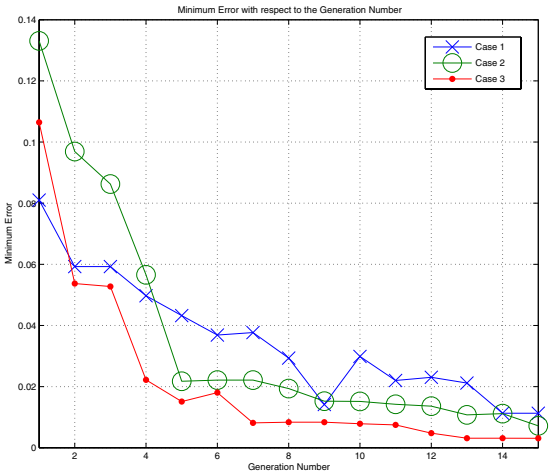

Fig. 4. minimum error of the cases 1–3 with respect to iteration number

This algorithm can also be used in conjunction with a numerical method in order to increase the resolution and precision of the results. The regions that were detected in this algorithm will be used in the numerical method as the initial search points and the numerical method can continue the convergence of the result to any required precision.

To incorporate a spherical wrist in the problem, the positioning part of IK can be solved with the proposed algorithm. Then the joint variables of the spherical wrist can be calculated analytically in order to set the orientation of the end-effector to the desired values.

The proposed algorithm can also be used in conjunction of another optimization measure, e.g. minimum joint angle change, obstacle avoidance, and joint singularity avoidance to solve the IK problem of redundant robotic manipulators.

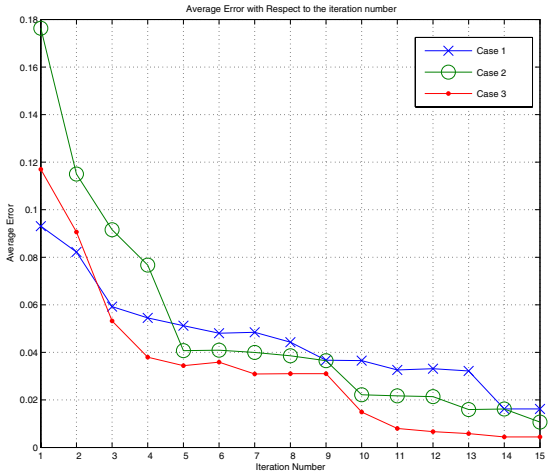

Fig. 5. Average error of the cases 1–3 with respect to iteration number

REFERENCES

[1] R.P. Paul, *Robot Manipulator: Mathematics, Programming and Control* ,MIT Press , Cambridge, Mass., 1981.

[2] L. Tsai and A. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods", *Journal of Mechanisms, Transmissions and Automation in Design*, Vol.107, pp.189–200, 1985.

[3] A.A. Goldenberg, B. Benhabib and G. Fenton, "A Complete Generalized Solution to the Inverse Kinematics of Robots", *IEEE J. of Robotics and Automation*, Vol.RA-1, No.1, March 1985.

[4] A.A. Goldenberg and D.L. Lawrence, "A generalized solution to the inverse kinematics of robotics manipulators", *ASME J. Dynamic Syst., Meas., Contr.*, Vol. 107, pp.103–106, Mar. 1985.

[5] J.K. Parker, A.R. Khoogar, D.E. Goldberg, "Inverse kinematics of redundant robots using genetic algorithm", *IEEE Int. Conf. on Robotics and Cybernetics*, Vol.1, pp.271–276, 1989.

[6] P. Karla, P.B. Mahapatra and D.K. Aggarwal, "On the solution of Multimodal Robot Inverse Kinematic Function using Real-coded Genetic Algorithms", *IEEE Int. Conf. on Systems, Man and Cybernetics*, Vol.2, pp. 1840–1845, 2003.

[7] P. Karla, P.B. Mahapatra and D.K. Aggarwal, "On the Comparison of Niching Strategies for Finding the Solution of Multimodal Robot Inverse Kinematics", *IEEE Int. Conf. on Systems, Man and Cybernetics*, Vol.6, pp. 5356–5361, 2004.

[8] B. Sareni, L. Krähenbühn, "Fitness sharing and niching methods revisited", *IEEE Trans. on Evolutionary Computation*, Vol.2, No.3, SEP. 1998.

[9] D.E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization", *Proc. 2nd. Int. Conf. Genetic Algorithms*, 1987, pp.41–49.

[10] D.E. Goldberg and L. Wang, "Adaptive niching via coevolutionary sharing", *Illigal Report No.97007, Urbana, IL: University of Illinois at Urbana-Champaign*, 1997.

[11] M. Neef, D. Thierens, H. Arciszewski, "A case study of a multiobjective recombinative genetic algortihm with coevolutionary sharing", *Proc. of 1999 Congress on Evolutionary Computation*, Vol.1, 1999.

[12] K. Deb and R.B. Agrawal, "Simulated binary crossover for continuous search space", *Complex systems*, Vol.9, No.2, pp.115–148, 1995.

[13] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design", *Computer Science and Informatics*, Vol.26, No.4, pp.30–45, 1995.

[14] S. Chiu, "Fuzzy Model Identification Based on Cluster Estimation," *Journal of Intelligent and Fuzzy Systems*, Vol.2, No. 3, Sept. 1994.