Robotica (2010) volume 28, pp. 493–507. © Cambridge University Press 2009
doi:10.1017/S0263574709005803

An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots

Saleh Tabandeh†*, William W. Melek† and Christopher M. Clark‡

†*Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1*
‡*Department of Computer Science, California Polytechnic State University, 1 Grand Avenue, San Luis Obispo, CA 94307-0354, USA*

(Received in Final Form: April 27, 2009. First published online: May 26, 2009)

SUMMARY
Inverse kinematics (IK) is a nonlinear problem that may have multiple solutions. A modified genetic algorithm (GA) for solving the IK of a serial robotic manipulator is presented. The algorithm is capable of finding multiple solutions of the IK through niching methods. Despite the fact that the number and position of solutions in the search space depends on the position and orientation of the end-effector as well as the kinematic configuration (KC) 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 (FK) equations which can be easily obtained from the Denavit–Hartenberg link parameters and joint variables of the robot. For identifying and processing the outputs of the proposed GA, a modified filtering and clustering phase is also added to the algorithm. For the postprocessing stage, a numerical IK solver is used to achieve convergence to the desired accuracy. The algorithm is validated on three KCs of a modular and reconfigurable robot (MRR).

KEYWORDS: Pose estimation and registration; Serial manipulator design and kinematics; Motion planning; Modular robots; Space robotics.

1. 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. FK of a robot manipulator can easily be formulated if the link parameters and joint variables of a robot are known, while the IK is a nonlinear configuration-dependent problem that may have multiple solutions.[1]

For a handful of robot configurations, closed-form solutions of the IK exist (e.g. PUMA, FANUC, etc.) .[1–4] For many other serial manipulators, the IK analytical solution does not exist. Another approach to the IK problem is to use numerical methods.[5–7] In numerical methods, the algorithm converges on the solution dependent on the initial starting point of the algorithm. In ref. [8], an algorithm to numerically solve the IK for modular and reconfigurable robot (MRR) was presented. In ref. [9] a method for analytically solving the IK for a number of MRR kinematic configurations (KCs) was presented. The algorithm in ref. [9] was based on the product-of-exponentials formula and was developed with the intention of finding single solutions of IK. The solver could cope with IK of all robots with 4 degree-of-freedom (DOF) or less, 90 % of the 5-DOF robots, and 50 % of the 6-DOF robots.

To solve IK for a redundant robot, a genetic algorithm (GA) was used in ref. [10], where the focus was on finding the solution that minimizes the joint displacements among all the possible solutions. In ref. [11], the IK problem for a 12-DOF redundant robot was tackled. According to the article, even after addition of heuristics to the algorithm, the results were still not completely satisfactory. In ref. [12], GA in conjunction with fuzzy systems and hybrid immune algorithms were used to solve the FK of a Stewart platform. A GA and a neural network were used in ref. [13] to solve the IK for the optimal joint motions. In ref. [14], a genetic programming algorithm was used to produce an estimate analytical formula for the IK offline. The extracted formula is then used to solve for an approximate solution to IK with a high speed. In ref. [18] a GA was used to find the optimized kinematic structure and the pose of a manipulator capable of performing a certain task. In ref. [18] the IK was solved in the same GA that was resolving the kinematic optimization problem. Hence, finding the optimized structure and pose of a manipulator for the prescribed task simultaneously.

Most related is research from refs. [15, 16] , where a fitness sharing niching method was used to find multiple solutions of the IK for positioning of a 2-DOF planar robot. A prominent feature of these works is the use of real-coded GA in conjunction with tournament selection. A drawback is that 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. More importantly, in these works

* Corresponding author. E-mail: tabandeh@uwaterloo.ca

no considerations has been made for robots with more than 3-DOF, where identification of the solutions among the output of the GA is not possible with observation anymore. This is viewed as a major drawback since most articulated industrial robots use at least a 3-DOF arm for end point positioning and at least one more DOF for orienting the end-effector.

In ref. [17] an IK solver based on GA with postprocessing stages to extract distinct solution regions was presented. In that paper, the algorithm was validated in solving the IK for positioning of the end-effector of a 3-DOF PUMA560 robot. Although the mentioned algorithm converged on IK solutions well, the accuracy and resolutions of the solutions were not satisfactory for precise control of the end-effector.

In this paper, an adaptive niching method to solve the IK problem is proposed. The proposed algorithm solves the IK problem while considering both positioning and orienting of the end-effector. 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 work can be highlighted as:

- By using a niching method, the algorithm is able to calculate multiple solutions of the IK problem for both positioning and orienting the end-effector. Moreover, unlike the other computational algorithms for solving IK, this method requires few parameters to be set with the prior knowledge of the problem. This feature allows for solving IK of a wide range of robots with distinct kinematics. For instance, the proposed algorithm can be used to calculate the IK solutions for MRRs in which the robot can assume a large range of distinct KCs.
- A real coded simulated binary crossover (SBX)[26] is used. This feature enables the algorithm to search in a continuous joint space, not a discrete binary one.
- A new formulation for incorporating the joint mechanical limits in the simulated binary crossover is presented.
- A modified adaptive niching method via coevolutionary sharing[24] was adopted to increase the algorithm speed without sacrificing the performance.
- A postprocessing stage consisting of filtering, clustering, and numerical IK is proposed. The filtering and clustering stages allow for using the algorithm to solve IK for robots with more than 3-DOF by identifying the solution regions automatically. Then the numerical IK solver achieves convergence to any desired joint angle accuracy. The numerical stage also enables the algorithm to distinguish the global optimums from the local optimums in the output of the niching GA.

Performance of the algorithm is validated by solving for multiple solutions of the IK problem for three distinct KCs. Because of the capability of MRRs to generate different kinematic structures, the tested KCs are obtained from an MRR joint and link modules. The algorithm is applied to a 4-DOF spatial, a 6-DOF spatial, and a 7-DOF KC with several GA runs for two different task points.

This paper is divided into six sections. Section 3 explains the IK problem and the objective function. In Section 4, the conventional niching methods and the adaptive niching method are explained. In Section 5 the proposed algorithm to solve the IK problem is explained. Section 6 describes how to process the results and details the filtering, clustering and numerical IK stages of the proposed algorithm. Finally, the results of running the algorithm for three distinct manipulators are presented in Section 7.

2. Motivation

Industrial robotic manipulators are designed to perform a certain task thousands of times during their operational age. Hence, any change to these manipulators that could cause an increase in their performance could have huge rewards in terms of decreasing operations, maintenance and repair costs, and manufacturing time.

Different approaches and solutions for optimizing the performance of robotic manipulators exists. The least costly, in terms of material and time, is probably modifying the robot trajectory such that a set of operational performance parameters of the robot will improve. For instance, the trajectory of the manipulator could be created in order to minimize the required torque, power consumption, operation's time, etc.

Having multiple solutions of the IK can lead to more optimal solutions to the trajectory generation problems. To demonstrate the effect of considering multiple IK solutions in trajectory optimization, the results of comparing the trajectories of a PUMA560 6-DOF spatial manipulator based on two of distinct IK solutions are presented.

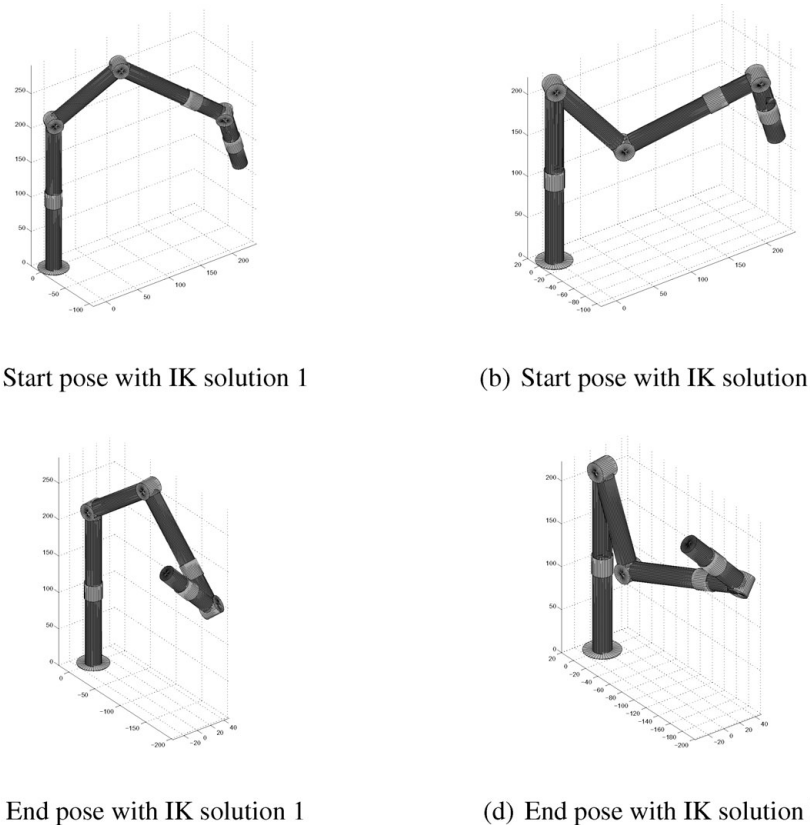
For each task point in the Euclidean space, a 6-DOF PUMA manipulator shown in Fig. 1 has eight distinct IK solutions. For the sake of comparison, a task for the manipulator consisting of moving from task point 1 ($TP1$) in the Euclidean space to task point 2 ($TP2$) is assumed. Figures 1(a) and 1(b) show two out of the eight distinct IK solutions for $TP1$ called elbow-up and elbow-down poses. Figures 1(c) and 1(d) show the elbow-up and elbow-down poses for $TP2$. It can be observed that four cases could happen when a trajectory from $TP1$ to $TP2$ is generated:

(1) The manipulator moves from the elbow-up pose in $TP1$ to the elbow-up pose in $TP2$.
(2) The manipulator moves from the elbow-down pose in $TP1$ to the elbow-down pose in $TP2$.
(3) The manipulator moves from the elbow-up pose in $TP1$ to the elbow-down pose in $TP2$.
(4) The manipulator moves from the elbow-down pose in $TP1$ to the elbow-up pose in $TP2$.

For each of these cases, a 7-degree polynomial trajectory with 50 intermediate points was created. The required joint torque and power consumption of each trajectory was calculated for the PUMA560 manipulator using the Matlab Robotics Toolbox.[19] This toolbox utilizes the recursive Newton–Euler formulation to solve the inverse dynamics problem.

Table I and Table II show the maximum and the average required torque for each joint of the PUMA560 to follow the produced trajectory. Table III shows the average required power for each of the joints and the total required power for the manipulator.

It can be seen that the maximum and average required torque, and the average required power, change drastically from case to case. This change in the performance measure



(a) Start pose with IK solution 1

(b) Start pose with IK solution 2

(c) End pose with IK solution 1

(d) End pose with IK solution 2

Fig. 1. Start and end point of a certain task considering two distinct solutions of the IK for a 6-DOF spatial manipulator.

Table I. Average required torque for the PUMA560 to follow the trajectories (N·m).

Torque (N·m)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Case 1	26.2323	28.0785	10.9776	0.8712	1.8214	1.6948
Case 2	26.2350	25.0806	1.1347	1.9331	0.5412	0.8107
Case 3	26.4836	35.4458	5.2365	2.0909	1.2226	0.8327
Case 4	26.2316	53.9799	12.3313	0.9127	2.6467	1.9210

Table II. Maximum required torque for the PUMA560 to follow the trajectories (N·m).

Torque (N·m)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Case 1	49.8973	40.4988	12.9466	1.7768	3.4713	3.3710
Case 2	48.5416	39.3951	13.0186	4.1371	1.1043	1.6374
Case 3	55.9225	71.1438	9.9031	4.4306	1.8467	1.2812
Case 4	56.5868	68.1273	24.6538	1.8695	4.7908	3.7374

caused by choosing different IK solutions for a unique task suggests that the performance of the manipulator can be improved by a suitable selection of the IK solutions for each task. The selection of the best IK solutions depends on the application and whether the goal is minimizing power, maximizing payload, maximizing speed, identifying an optimal sequence to pass through intermediate points of a given task, or a combination of the above. In any scenario, the dynamic model of the manipulator should be evaluated and multiple options for path planning should exist, i.e., more than one solution to the robot IK problem. In general, if the manipulator has s IK solutions for each of t task point, s^t different cases exist. Hence, a search algorithm is required to examine those cases in order to find the IK solution(s) that optimize the trajectory for a given set of goals.

For this particular example, since only four different cases exist, an exhaustive search where all of the cases one by one are examined is used. If minimizing the power consumption of the manipulator is the goal, as can be seen from Table III, by choosing the trajectory made in Case 2 the best improvement is achieved. Hence, having multiple solutions of the IK can be seen as a first step in creating trajectories

Table III. Average required power for the PUMA560 to follow the trajectories (W).

Power (W)	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	$\sum$
Case 1	24.9752	12.3913	2.6832	0.7107	5.3727	3.7535	49.8866
Case 2	26.4819	16.7639	0.1891	4.8971	0.3554	0.6260	49.3135
Case 3	24.1281	73.1677	11.4001	5.4981	0.6969	0.3513	115.2422
Case 4	24.5108	48.0250	37.0315	0.8285	10.9475	4.8569	126.2002

capable of optimizing the operation of a robotic manipulator. The proposed IK solver is used for efficient path planning for the Waterloo Modular and Reconfigurable (WMRR) Manipulator.[31]

3. Kinematics and Objective Function

3.1. 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 FK 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 the homogenous transformation of a coordinate frame attached to the end of the link with respect to a fixed frame that is connected to the origin of the frame.[1] The homogenous transformation has the form:

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

in this equation $\mathbf{R}_{i-1,i}(\theta_i)$ and $\mathbf{P}_{i-1,i}(\theta_i)$ describe the relative orientation and position of frame i with respect to frame $i-1$. The parameters of these matrices can be extracted from the physical shape and KC of a robot.

To calculate the position and orientation of the end-effector ($T_{oe}(\theta_1,\ \theta_2,\ \ldots,\ \theta_n)$) with respect to the base of the robot for an arbitrary set of joint angles $[\theta_1,\ \theta_2,\ \ldots,\ \theta_n]$ the transformation will be

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

The inverse problem of the FK, the IK, is the problem of finding $[\theta_1\ \theta_2\ \cdots\ \theta_n]$ from an arbitrary $\mathbf{T}_{oe}$. This problem is a mapping from the 3D task space, usually presented in a Homogenous Transformation, to the joint angle space and usually has more than one solution. For instance, a 6-DOF PUMA560 robot may have four or eight IK solutions[1] depending on the mechanical limitations of the manipulator joints.

3.2. Objective function

In this paper, the approach to solving the IK is based on converting it to a minimization problem and then utilizing a modified niching GA to calculate the global minimums of the problem. In the proposed algorithm, each GA individual consists of a string that represents a joint angle vector of the robot ($[\theta_1\ \theta_2\ \cdots\ \theta_n]$). That is, each GA individual represents a posture of the robot manipulator.

In GAs, a measure of the fitness of each individual, an objective function, is required to select the most potent individuals for crossover operation. In the IK solver GA, this measure can be defined as the difference between the end-effector position and orientation of each individual and those of the desired task point. In other words, the positioning and orienting error of the end-effector produced by the joint angles of each individuals with reference to the task point in the cartesian space can be used as the fitness measure. If the homogenous transformation of the task point in the cartesian space is represented by

$$\mathbf{T}_t = \left(\begin{array}{ccc|c} & \mathbf{R}_t & & \mathbf{P}_t \\ \hline 0 & 0 & 0 & 1 \end{array}\right), \tag{3}$$

and the homogenous transformation of the end effector of each of the individuals of the GA is represented by

$$\mathbf{T}_{\text{ind}} = \left(\begin{array}{ccc|c} & \mathbf{R}_{\text{ind}} & & \mathbf{P}_{\text{ind}} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{4}$$

the Euclidean norm of the difference between the end-effector position of each individual and that of the desired point in the cartesian space can be used as a measure of the positioning error, i.e.,

$$\mathbf{E}_P = \|\mathbf{P}_t - \mathbf{P}_{\text{ind}}\|. \tag{5}$$

The error in end-effector orientation is defined by the rotation matrix $\mathbf{R}_{\text{error}}$ that rotates the end-effector of each individual to the desired orientation. $\mathbf{R}_{\text{error}}$ can be calculated by

$$\mathbf{R}_{\text{error}} = \mathbf{R}_t \,.\, \mathbf{R}_{\text{ind}}^{-1} = \mathbf{R}_t \,.\, \mathbf{R}_{\text{ind}}^{T}. \tag{6}$$

In this work, the quaternion representation of frames rotation is used for calculations of the orientation error in the final fitness value of each individual. Unit quaternions provide a convenient mathematical notation for representing orientations of objects in three dimensions. Compared to Euler angles they are simpler to compose, more numerically stable, and more efficient in some situations.[20,21]

If the coordinate of a point in the Cartesian space is represented by $\mathbf{V}$ in the quaternion representation, it can be shown that the quaternion product $\mathbf{QVQ}^{-1}$ yields the vector $\mathbf{V}$ rotated by an angle α around axis of the quaternion vector $\mathbf{U}$. Where $\mathbf{Q}$ is the rotation vector and can be represented as $\mathbf{Q} = [\cos(\alpha/2) \quad \sin(\alpha/2).\mathbf{U}]$.

For our application, the rotation matrix needed to rotate the end-effector homogenous transformation of each GA individual to the desired location ($\mathbf{R}_{\text{error}}$) can be conveniently converted to the quaternion format and its α can then be

extracted from the quaternion to represent the orientation error.

If $\mathbf{R}_{\text{error}}$ is shown in the quaternion representation by $[q_w\ q_x\ q_y\ q_z]$. where $[q_w\ q_x\ q_y\ q_z] = [\cos(\alpha/2)\ \sin(\alpha/2).\mathbf{U}]$. The orientation error can then be written as

$$\mathbf{E}_O = \alpha. \tag{7}$$

Using Eqs. (5) and (7), the objective function of individual ind with reference to task point T_t can be written as:

$$f(ind) = F_{\text{obj}}(T_{\text{ind}}, T_t) = w_p\,\mathbf{E}_P + w_o\,\mathbf{E}_O, \tag{8}$$

where w_p and w_o are the positioning and orienting weighting factors. Those weighting factors can be used to normalize the corresponding values. To guarantee both the orienting and positioning error contribute equally to the total error expression in 8, we propose to choose w_p and w_o as follows:

$$\begin{aligned} w_p &= 1, \\ w_o &= \frac{\sqrt{\lambda_{KC}.\|\mathbf{P}_d\|}}{\pi}, \end{aligned} \tag{9}$$

where λ_{KC} is a coefficient that depends on the dimensions of the robot. We use the total length of all the links of the robot as λ_{KC}. w_p and w_o are calculated by scaling the orientation error such that the maximum possible orientation error, which occurs in $\alpha = \pi$, will become large enough to compete with the positioning error which is a function of the size of the robot (λ_{KC}) and the distance of the task point from the base of the robot ($\|\mathbf{P}_d\|$).

Our aim in this paper is utilizing a GA for finding individuals that can minimize $f(ind)$. The individuals consist of the joint angle vectors of the robot.

4. Background on the Niching Techniques

A GA, through selection, crossover 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 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.[22] 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.

4.1. Conventional techniques

The sharing method,[23] 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.[22] 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, the 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.[22]

4.2. Adaptive niching via coevolutionary sharing

As mentioned, 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 proximity and distances between the solutions of the problem, a luxury which is not available in IK problems.

To address this drawback in the sharing methods, Goldberg and Wang introduced an adaptive niching algorithm via coevolutionary sharing (CSN).[24] 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 who 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

$$f'(c) = \left.\frac{f(c)}{m_{b,t}}\right|_{c\in C_b}, \tag{10}$$

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.[24]

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:

$$\phi(b) = \sum_{c\in C_{b,t}} f(c). \tag{11}$$

Table IV. The IK algorithm.

Step	Description
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
4	Transferring the elite customers to the next generation
5	Customer crossover
6	Businessmen imprint
7	Updating $d_{\min}$
8	If the termination criterion is not reached return step 2

In ref. [24], 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.[25] Rank-based selection, elitist recombination, and nondominated sorting are some of the prominent features of that work.

5. 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 6D space for a PUMA or general purpose articulated 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.

5.1. The algorithm

An overview of the proposed algorithm can be seen in Table IV. 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 consists of n joint angles corresponding to the n DOF of the robot:

$$ind_i = [q_1\ q_2\ \cdots\ q_n]^T, \tag{12}$$

where $q_1,\ q_2,\ \ldots,\ q_n$ 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. $q_{\min}$ and $q_{\max}$ are the joints' rotational limitations which are usually dictated by the mechanical design and manufacturing.
After the Customer and Businessmen populations are randomly generated, the initial $d_{\min}$ is calculated using the following equation:

$$d_{\min_{start}} = \frac{\kappa(q_{\max}-q_{\min})}{1+\sqrt[n]{b}} \tag{13}$$

where n, b, and κ correspond to the DOF (i.e., number of joints), the number of businessmen, and the fitting index respectively. Equation (13) 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 Eq. (8). Then, customers are assigned to the closest businessman, where closeness is measured using the Euclidean distance between customers and businessman. 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 ($f'(c)$) is calculated using the following equation:

$$f'(c) = f(c)\cdot m_{b,t}. \tag{14}$$

This equation is the counterpart of Eq. (10), in the original maximization CSN algorithm, which has been modified for minimization. By using the niched fitness value $f'(c)$ in the GA, the tendency of the selection operator will be towards forming the parent pool from individuals with smaller $m_{b,t}$, i.e., individuals in less dense locations. The niched selection scheme preserves the diversity of the individuals and prevents the individuals from converging on one single solution of the problem.

(3) Selection: Tournament selection is adapted to create the parent pool, because it does not require a priori knowledge of the problem.
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.

(4) Elitism: Results in refs. [29, 30] show that elitism can speed up the performance of the GA significantly. It can also help prevent the loss of good solutions once they have been found. In simple GAs the elitism is performed

by transferring a number of the customers with the best fitness values to the next generation directly. But in niching GAs, by applying the same elitism method, there is the chance of all the elites be chosen from just the few most developed niches. This, can decrease the diversity of the population very fast and cause premature convergence of the algorithm. In our proposed algorithm, to profit from elitism while avoiding its potential drawbacks, the elitism is performed as the following; Instead of simply choosing the best customers from the entire pool of customers, one customer belonging to each businessman will be chosen. This customer will be that with the lowest fitness value over all other customers belonging to the same businessman. Hence, each businessman will contribute one Elite unless it does not have any customers. If the number of the elites was odd, after adding a randomly selected customer from the customer population to it, the Elites list is transferred to the next generation population. The crossover operator will be used to find the rest of the customers in the next generation population.

(5) Customer crossover: In this step, the new generation of customers is produced from the parent pool of the previous step. Simulated Binary Crossover (SBX)[26] is used for the crossover operation. In Section 5.2 SBX and the proposed modifications that has been applied to it to accommodate joint angles with physical limits are presented.

(6) 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. For each businessman, the process of comparison 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.

(7) Updating $d_{\min}$: The value of $d_{\min}$ 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. The drawback of a having a small $d_{\min}$ is an increase in the probability of losing some niches because of the tendency of the businessmen to converge around the regions with high customer concentration.

In initialization of the algorithm, $d_{\min}$ is 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 and the number of individuals in different niches decreases.

In this step of the algorithm, $d_{\min}$ is decremented in small step sizes until it reaches a certain lower limit. In the GA proposed here, the following function is used for updating $d_{\min}$:

$$d_{\min} = d_{\min_{\text{start}}}\left(1 - \lambda\frac{t}{t_{\max}}\right), \tag{15}$$

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.

(8) Checking the termination criterion: In this stage, the output of the algorithm is checked against a termination criterion. If the average fitness value of the businessmen population ($f_{(b)\text{avg},t}$) in generation t could satisfy the following criterion, the algorithm is interrupted and the results are entered into the postprocessing phase.

$$\begin{aligned} f_{(b)\text{avg},t} &\leq \mu F_{\text{limit}} \\ F_{\text{limit}} &= F_{\text{obj}}(T_t, \mathbf{I}) \end{aligned} \tag{16}$$

where F_{obj} is defined in Eq. (8). T_t and $\mathbf{I}$ are respectively the Homogenous Transformation of the task point and the base of the robot which is the Identity matrix. $F_{\text{obj}}(T_t, \mathbf{I})$ can be considered as an upper bound on the fitness value $f(ind)$ that depends only on the position and orientation of the task point. $0 \leq \mu \leq 1$ is a coefficient that represents how small the average businessmen fitness value should become before terminating the algorithm. According to the termination criterion, the algorithm is stopped when all the businessmen in the GA have produced robot postures with end-effector positions/orientations that can reach on average a distance of $\mu F_{\text{obj}}(T_t, \mathbf{I})$ from the task point.

In the first generations of the algorithm, the businessmen are randomly spread over the search space and their average fitness value is high. With the progress of the algorithm and the decrease in $d_{\min}$, the businessmen start converging on the locations with better fitness values while keeping their distance from each other, resulting in the decrease in their average fitness value. Further decrease in $d_{\min}$ and discovery of better locations decreases the average fitness value of the businessmen even more. Hence, with the termination criterion of Eq.(16) it is guaranteed that the algorithm stops only when the businessmen have positioned themselves in the vicinity of the solutions of the IK problem.

5.2. 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.[26] In this paper, we use a SBX[26] 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:

$$\beta(i) = \left|\frac{C_2(i) - C_1(i)}{P_2(i) - P_1(i)}\right|. \tag{17}$$

In order to incorporate the joint angle's mechanical limitations, the crossover is carried out using 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 $l = 0.5n$.[27]

(2) For each of the l joint angles selected in the last step, a random number, $u_\beta(i)$, is generated. The expansion ratio, β, is then calculated using:

$$\beta(i) = \begin{cases} (2u_\beta(i))^{\frac{1}{\eta+1}} & \text{if } u_\beta(i) \leq 0.5 \\ \left(\frac{1}{2(1-u_\beta(i))}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases}, \quad (18)$$

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.[27] 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 in industrial robots), Eq. (18) must be modified to produce offsprings that are located inside the joint limits. To accomplish this, the following method has been proposed. First, the lower and upper bound of the expansion ratio β_L and β_H are calculated from the following equation for each of the joint variables:

$$\beta_L(i) = \frac{0.5(P_1(i) + P_2(i)) - q_{\min}}{|P_1(i) - P_2(i)|},$$
$$\beta_H(i) = \frac{-0.5(P_1(i) + P_2(i)) + q_{\max}}{|P_1(i) - P_2(i)|}, \quad (19)$$

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:

$$\beta_{\text{limit}}(i) = \begin{cases} \beta_L(i) & \text{if } \beta_L \leq \beta_H \\ \beta_H(i) & \text{otherwise} \end{cases}, \quad (20)$$

$$k(i) = \frac{1}{1 - \frac{0.5}{\beta_{\text{limit}}(i)^{\eta+1}}},$$
$$u_\beta(i) = \frac{u_\beta(i)}{k(i)}. \quad (21)$$

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, Eq. (19) calculates the maximum allowable value of this parameter corresponding to each limit. Equations (20) and (21) modify u_β in a way to set the probability of choosing a β less than $\beta_{\text{limit}}(i)$, 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 Eq. (18) using the updated $u_\beta(i)$.

Fig. 2. Block diagram of the proposed GA postprocessing algorithm.

(3) In the last step, children are produced from the following equation:

$$C_1(i) = 0.5[(1 + \beta(i))\, P_1(i) + (1 - \beta(i))\, P_2(i)], \quad (22)$$
$$C_2(i) = 0.5[(1 - \beta(i))\, P_1(i) + (1 + \beta(i))\, P_2(i)], \quad (23)$$

and are then placed in the new generation population.

6. Processing the Output

The output of the GA is a set of n dimensional vectors, representing the manipulators joint angles, with high population density around the regions with high fitness values and lower concentration in the rest of the search space. Since the local optimums have higher fitness values compared to the regions around them, they also attract a concentration of the individuals. In order to distinguish solution regions from this output, a mechanism to detect the regions with high concentration of individuals and low error is required.

If the robot had 2-DOF, identifying these results in the 2D space could be accomplished by observation, which is not convenient if the GA is supposed to be used as a block in a larger algorithm or software. For robots with more DOFs (for example a 6-DOF PUMA), identifying these solution regions must be done in a 6D space, which is not possible by visual inspection. Hence, a robust algorithm for clustering the results is required. Furthermore, a method to increase the accuracy and resolution of the solutions for more precise applications is required. When the solutions converged within the desired tolerance, the global optimums could easily be identified from the global ones.

In this section, the filtering, clustering, and the numerical IK postprocessing stage are explained. Figure 2 shows a block diagram of the proposed postprocessing algorithm.

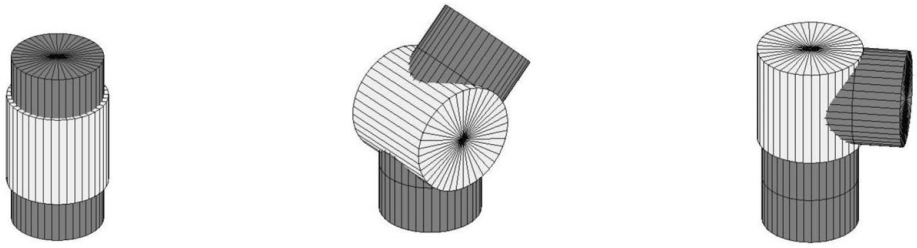
6.1. Filtering

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. The threshold of the filtering is represented by ε. It should be noted that although choosing a small ε can lead to selecting fewer more accurate individuals, some of the less established niches might also be lost as a result.

6.2. Clustering

Since no priori knowledge of the number of solutions of IK exists, the number of solution niches or clusters is also



(a) Rotational Joint Module (b) Pivotal Joint Module (c) Perpendicular Rotational Joint Module

Fig. 3. Joint modules of the WMRR.

unknown. To deal with this issue, subtractive clustering[28] is utilized 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.

In the GA, all the joint angles are mapped into $[-\pi, \pi]$. If some of the solutions of the IK are close to the border limits $-\pi$ or π two concentrations of the individuals representing just one solution will form close to both of the border limits. The reason is that although the angles close to these border are at the two limits, from a mechanical point of view they are close to each other. As a result, the clustering stage will detect two different clusters close to the borders while they both belong to the same cluster. To overcome this problem, in the proposed algorithm, first the clustering is applied to the joint angles when they are mapped into $[-\pi, \pi]$. Then the solutions of the previous clustering is mapped into $[0, 2\pi]$ and will undergo subtractive clustering again.

6.3. Numerical inverse kinematics

The outputs of the previous two stages are the centers of the niches detected by the clustering algorithm. Although these centers represent the location of the solutions in the joint angle space, they might not have the required accuracy and resolution for a precision control of the robotic manipulator. Hence, the necessity of improving the accuracy of the solutions arises.

A quasi-Newton algorithm is utilized in order to increase the accuracy of the results to any desired value. The numerical IK uses the outputs of the clustering stage as the initial search point and then converges to within the desired positioning/orienting tolerances.

If a solution in the output of the numerical error still causes an error greater than the admitted tolerances, it could be identified as a local optimum and eliminated from the final set of the solutions. However, in the results section, to clarify the results achieved the local optimums were not eliminated and were only identified.

Using numerical methods after the GA has the added benefit that the GA, which is computationally demanding, can be stopped even when the niches were formed on the approximate location of the solutions. Meaning that the GA can be stopped earlier and the responsibility of further convergence to the solutions can be transferred to the much faster numerical algorithm. Hence enhancing the overall speed of convergence of the IK algorithm.

7. Results

Three distinct KCs, each with different number of joints have been used for validation of the proposed algorithm. These configurations were created based on the joint module set of the WMRR.[31] MRRs, a breed of industrial manipulators, are assembled from a variety of modular components and can be physically configured to meet the requirements of the workspace and the task at hand. The set of modules may consist of joints, links, and end-effectors. Different KCs will be achieved by using different joint, link, and end-effector modules and by changing their relative assembly orientation. The number of distinct KCs attainable by a set of modules can vary with respect to the size of the module set from several tens to several thousands. Hence, in MRRs to solve for multiple solutions of the IK a robust algorithm capable of handling a large set of KCs is required. Furthermore, the algorithm should perform well, without any change in parameters even after the robot is reconfigured. Hence, an MRR is an ideal platform for testing the functionality and practicality of the proposed algorithm. Figure 3 shows the kinematic schematic of the joint modules of the WMRR.

In the rest of this section, the results of running the algorithm for the tested KCs are presented. First, the results of running the algorithm for a 4-DOF and a 6-DOF PUMA robot for two different task points are presented. Then, the solutions that were found by the algorithm for a 7-DOF robot has been shown and discussed in the rest of this section. The results were achieved on Matlab on a 2GHz AMD64 processor.

Table V. Parameters used in the GA.

Parameter	Value
Businessman population (b)	$2n.2^{n-2}$
Customer population (c)	$2n.b$
κ	1.2
λ	0.5
$q_{\min}$	$-\pi$
$q_{\max}$	π
ψ	π
η	2-5
R_{cluster}	0.0796
ε	μF_{limit}
n_{limit}	$3.b$
n_t	2
μ	0.5
$t_{\max}$	500

GA is a stochastic method, therefore numerous runs of the algorithm are required in order to examine the performance and the repeatability of the algorithm. Hence, for each of the selected task points the algorithm was executed five times. The parameters that were used in the algorithm were kept fixed for all the runs and all of the configurations. The parameters are shown in Table V. It should be noted that the expression for the population size of the Businessmen and the Customers represent an approximate upper bound. Therefore, the algorithm performs as well even with smaller sizes of the population in most of the cases. In the table, n represent the number of the joints of the robot.

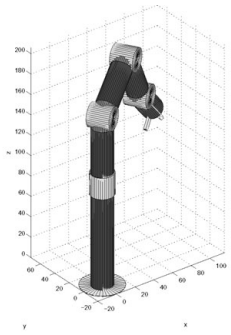
7.1. 4-DOF spatial manipulator

The tested 4-DOF KC has two distinct IK solutions for any reachable task point. The 4-DOF manipulator is essentially a spatial manipulator that is attached to a rotational joint and can produce spatial movements. Figures 4(a) and 4(b) show the 4-DOF spatial manipulator when it reaches task point 1 with the two distinct IK solutions, while Fig. 4(c) and Fig. 4(d) show the distinct solutions for task point 2. Table VI and Table VII shows the results of 5 runs of the algorithm for each of the task points. In the tables, N_{GA} and N_{NM} are the number of solutions before and after numerical IK respectively. E^P_{GA} and E^P_{NM} are the minimum positioning error (cm) before and after numerical IK, while E^O_{GA} and E^O_{NM} are the minimum orientation error(degrees) before and after numerical IK, respectively. E^J is the per joint difference of the joint angles before numerical IK to the respective value after applying the numerical method and reaching a solution.

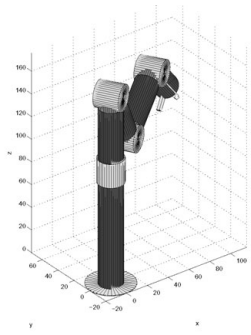
For the first task point, the algorithm ran for an average 82 iteration and 40 s before convergence. The corresponding values for the second task point were 24 iterations and 19 s. As can be seen, in all of the runs all the solutions of the IK have been found. The GA usually ends up with more niches than the actual number of the solutions. After undergoing the numerical method, more than one of these niche centers converge on the same solution. Hence, at the output of the numerical stage the final number of the solutions is typically less than the number of niche centers.

7.2. 6-DOF PUMA

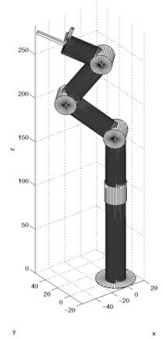
A 6-DOF manipulator resembling the KC of a PUMA manipulator is chosen as the next test bed of the algorithm.



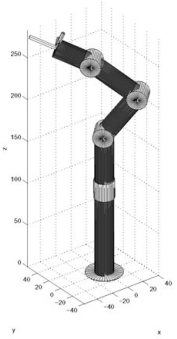
(a) Task point 1 with IK solution 1



(b) Task point 1 with IK solution 2



(c) Task point 2 with IK solution 1



(d) Task point 2 with IK solution 2

Fig. 4. Solutions of the 4-DOF spatial KC.

Table VI. Output of the algorithm in reaching task point 1 for the 4-DOF spatial. N represents the number of niches. E^P, E^O, and E^J represent the positioning error, orienting error, and error in the joint angle space, respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	18	30	15	17	21
N_{NM}	2	2	2	2	2
E^P_{GA}(cm)	1.2184	10.4490	12.5175	15.5401	23.3084
E^O_{GA}(degree)	1.9098	0.4613	0.3554	1.0181	1.9905
E^P_{NM}(cm)	0.0000	0.0001	0.0001	0.0001	0.0000
E^O_{NM}(degree)	0.0001	0.0001	0.0001	0.0000	0.0000
E^J(degree)	3.7007	3.5954	3.2532	1.9500	7.1817
t(s)	39	30	45	36	48

Table VII. Output of the algorithm in reaching task point 2 for the 4-DOF spatial. N represents the number of niches. E^P, E^O, and E^J represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	47	17	30	19	10
N_{NM}	2	2	2	2	2
E^P_{GA}(cm)	17.6113	7.0731	4.0406	1.0563	34.3000
E^O_{GA}(degree)	0.6621	5.0517	0.5007	2.3897	1.3140
E^P_{NM}(cm)	0.0002	0.0001	0.0002	0.0001	0.0001
E^O_{NM}(degree)	0.0002	0.0001	0.0003	0.0001	0.0000
E^J(degree)	7.8336	3.8397	3.6875	4.8561	10.3207
t(s)	18	14	21	27	16

Figures 5 and 6 show all the solutions of the IK when the robot reaches the two task points. In the figures, visual handles are attached to links connected to output of joints 1, 4 and 6 to differentiate between the two possible values of joint angles q_1, q_4 and q_6 in producing multiple IK solutions. As can be seen from Fig. 5, since the only aim of running the algorithm was finding multiple solutions of the IK, all of the solutions of IK, regardless of self collision, are shown. The results of five runs of the algorithm have been summarized in Table VIII and Table IX. The algorithm reached these results after 61 and 86 iterations (36 and 42 minutes) for task points 1 and 2, respectively.

For the first task point, while the correct number of the solutions for the IK is eight, the algorithm reports nine distinct solutions in one of the runs. Further investigations reveals that two of the solutions are the same where in one the third joint angle is $\theta_3 = 0$ while in the other $\theta_3 = 2\pi$.

7.3. 7-DOF spatial robot

A 7-dof spatial manipulator was chosen for further test of the algorithm in finding multiple solutions of IK. As can be seen from Fig. 7, the KC of the robot closely resembles that of the Canada Arm 2 of the International Space Station. The length of the links 1, 2, 5, and 6 are assumed to be zero, meaning

Table VIII. Output of the algorithm in reaching task point 1 for the 6-DOF PUMA. N represents the number of niches. E^P, E^O, and E^J represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm in each case.

	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	216	214	214	219	229
N_{NM}	8	9	8	8	8
E^P_{GA}(cm)	6.2119	9.4829	9.3211	7.6261	12.0384
E^O_{GA}(degree)	3.3087	3.9755	3.9423	3.4848	1.8421
E^P_{NM}(cm)	0.0001	0.0000	0.0000	0.0001	0.0000
E^O_{NM}(degree)	0.0000	0.0001	0.0001	0.0001	0.0000
E^J(degree)	7.2211	4.7227	7.6433	9.7832	5.5340
t(s)	1638	2240	2295	3263	1536

Table IX. Output of the algorithm in reaching task point 2 for the 6-DOF PUMA. N represents the number of niches. E^P, E^O, and E^J represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method respectively. t represents the run time of the algorithm in each case.

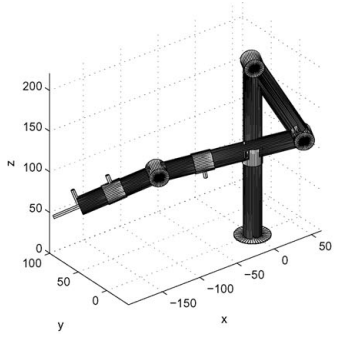
	Run 1	Run 2	Run 3	Run 4	Run 5
N_{GA}	107	154	146	59	124
N_{NM}	8	8	8	8	8
E^P_{GA}(cm)	3.9693	3.6407	3.1639	4.1705	4.8595
E^O_{GA}(degree)	2.4588	6.8048	3.8925	1.7206	3.0407
E^P_{NM}(cm)	0.0000	0.0006	0.0000	0.0001	0.0004
E^O_{NM}(degree)	0.0000	0.0001	0.0001	0.0001	0.0001
E^J(degree)	1.4597	3.9952	4.7653	5.8982	6.4299
t(s)	5421	5453	5276	5381	5695

the joints are directly connected without any links between them. The algorithm reached the termination criterion in iteration 530 after running for about 42 h. The algorithm converged slowly in this example due to the fact that the 7-DOF spatial manipulator is a redundant robot. Hence, the objective function has numerous local and global optima. Furthermore, the complexity of searching the solution space of 7-DOF manipulator increases exponentially compared to that of the 6-DOF system presented in the previous example.

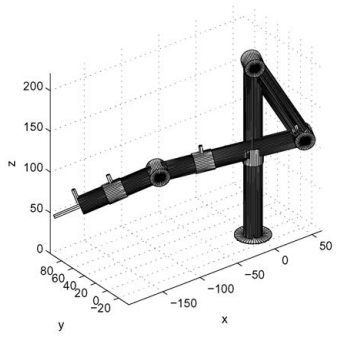
The output of the algorithm after the numerical stage consisted of 28 niches. From these 28 niches, 16 were local solutions of the problem and 12 were the global and correct solutions of the IK. The summary of the results of the run for the 7-DOF KC is shown in Table X.

8. Conclusion and Discussion

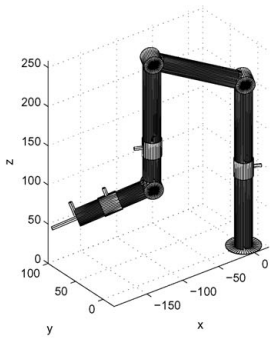
An adaptive niching strategy was proposed and used to extract multiple solutions of the IK problem for spatial robots. Since this algorithm uses few preset parameters, it can be generalized to solve IK of a robot with an unknown number of degrees of freedom and manipulator configuration. The algorithm combines real coding, adaptive minimum distance setting of businessmen, elitism, and adaptive real coding distribution index. To process the results, a subtractive



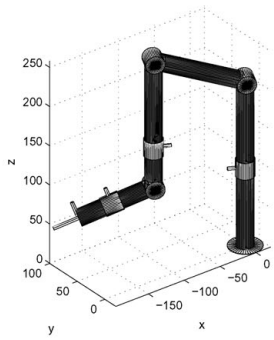
(a) Task point 1 with IK solution 1



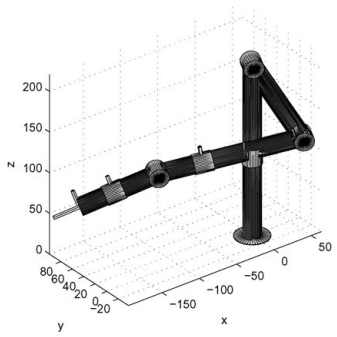
(b) Task point 1 with IK solution 2



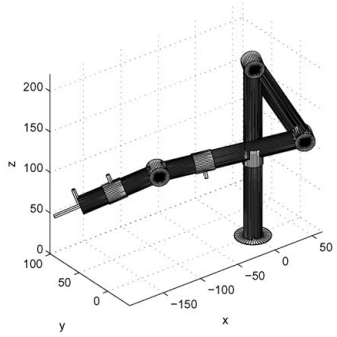
(c) Task point 1 with IK solution 3



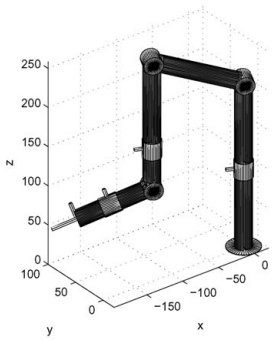
(d) Task point 1 with IK solution 4



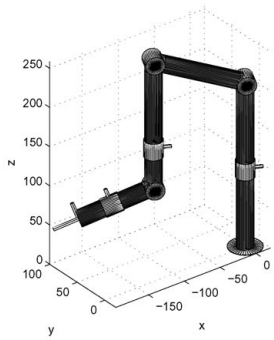
(e) Task point 1 with IK solution 5



(f) Task point 1 with IK solution 6

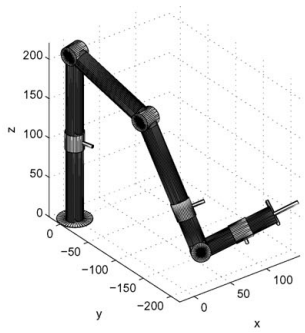


(g) Task point 1 with IK solution 7

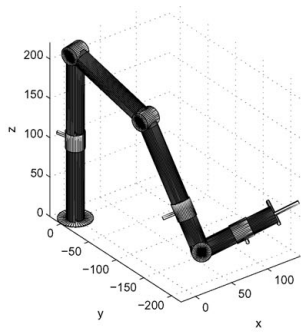


(h) Task point 1 with IK solution 8

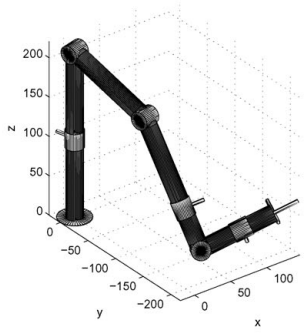
Fig. 5. Solutions of the 6-DOF PUMA KC for task point 1.



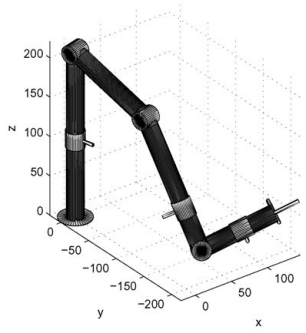
(a) Task point 2 with IK solution 1



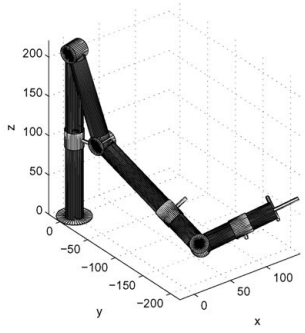
(b) Task point 2 with IK solution 2



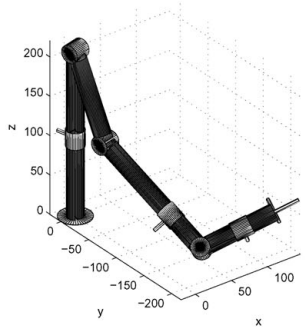
(c) Task point 2 with IK solution 3



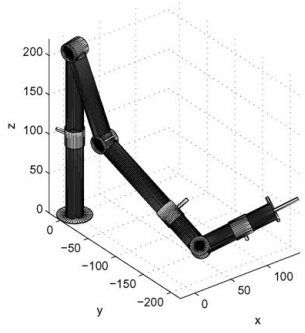
(d) Task point 2 with IK solution 4



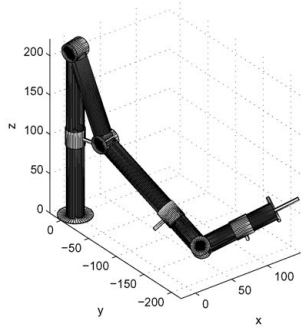
(e) Task point 2 with IK solution 5



(f) Task point 2 with IK solution 6



(g) Task point 2 with IK solution 7



(h) Task point 2 with IK solution 8

Fig. 6. Solutions of the 6-DOF PUMA KC for task point 2.

Table X. Output of the algorithm in reaching the task point for the 7-DOF Spatial Robot. N represents the number of niches. E^P, E^O, and E^J represent the positioning error, orienting error, and error in the joint angle space respectively. Parameters with GA and NM as the subscript represent the values after the clustering phase and after the numerical method, respectively. t represents the run time of the algorithm.

N_{GA}	N_{NM}	E^P_{GA}(cm)	E^O_{GA}(degree)	E^P_{NM}(cm)	E^O_{NM}(degree)	E^J(degree)	t(h)
69	28	2.87	1.33	0.0003	0.0000	7.79	42

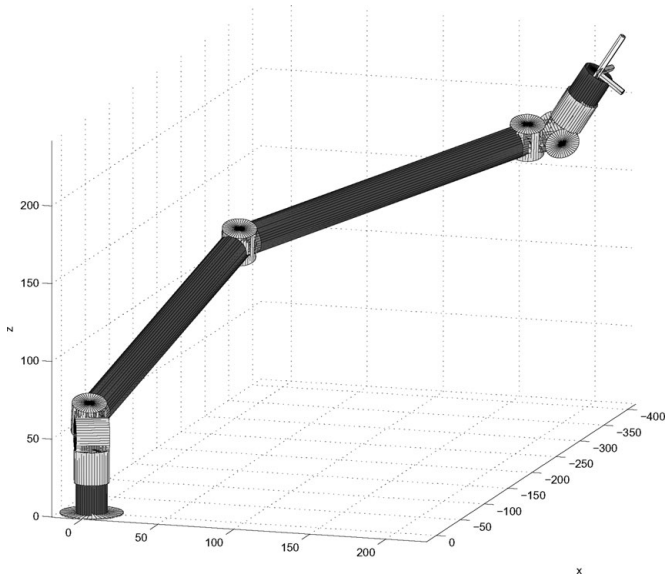


Fig. 7. 7-DOF spatial robot.

clustering algorithm was also modified for the application. It was shown that the algorithm performs with good precision for random task points independent of the KC of the robot.

By using the GA niching algorithm in conjunction with a numerical method the resolution and precision of the results were improved drastically. The niche centers that were detected in the GA were used in the numerical method as the initial search points and the numerical method was then invoked to achieve the convergence of the result to the required precision.

All the experiments were run on Matlab. The speed of the algorithm can drastically be improved by implementing the algorithm with lower level programming languages such as C/C++.

With the proposed algorithm, multiple solutions of the IK can be assessed with the purpose of finding the optimized trajectory for performing a task. In other words, instead of using just one of the solutions of the IK in trajectory planning, different IK solutions are compared and the one that satisfies a set of optimization measures better than the rest is used in the trajectory planning. For instance, IK solutions that can minimize joint torques, required electrical energy, or required time to perform a task can be selected for performing a task from the pool of solutions identified by the proposed algorithm.

References

1. R. P. Paul, *Robot Manipulator: Mathematics, Programming and Control* (MIT Press, Cambridge, MA, 1981).
2. L. Tsai and A. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods," *J. Mech. Transm. Autom. Des*. **107**, 189–200 (1985).
3. S. Kucuck and Z. Bingul, "The inverse kinematics solutions of fundamental robot manipulators with offset wrist," *IEEE Int. Conf. Mechatronics* 197–202, (Jul. 2005).
4. J. Xie, S. Yan and W. Qiang, "A Method for Solving the Inverse Kinematics Problem of 6-DOF Space Manipulator," *First International Symposium on Systems and Control in Aerospace and Astronautics*, Harbin (Jan. 2006) pp. 379–382.
5. A. A. Goldenberg, B. Benhabib and G. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE J. Robot. Autom*. **RA-1**(1), 14–20 (Mar. 1985).
6. A. A. Goldenberg and D. L. Lawrence, "A generalized solution to the inverse kinematics of robotics manipulators," *ASME J. Dyn. Syst., Meas. Control* **107**, 103–106 (Mar. 1985).
7. G. Z. Grudic and P. D. Lawrence, "Iterative inverse kinematics with manipulator configuration control," *IEEE Trans. Robot. Autom*. **9**(4), 476–483 (Aug. 1993).
8. I.-M. Chen and G. Yang, "Inverse Kinematics for Modular Reconfigurable Robots," *Proceedings of IEEE International Conference on Robotics and Automation*, Leuven, Belgium, vol. 2, (May 1998) pp. 1647–1652.
9. I.-M. Chen and Y. Gao, "Closed-Form Inverse Kinematics Solver for Reconfigurable Robots," *IEEE International Conference on Robotics and Automation*, Seoul, Korea, vol. 3, (2001) pp. 2395–2400.
10. J. K. Parker, A. R. Khoogar and D. E. Goldberg, "Inverse Kinematics of Redundant Robots Using Genetic Algorithm," *IEEE International Conference on Robotics and Cybernetics*, Scottsdale, AZ, USA, vol. 1, (1989) pp. 271–276.
11. K. A. Buckley, S. H. Hopkins and B. C. H. Turton, "Solution of Inverse Kinematics Problems of a Highly Kinematically Redundant Manipulator Using Genetic Algorithms," *Second International Conference on Genetic Algorithms in Engineering: Innovations and Applications*, Glasgow, UK (Conf. Publ. No.449), (Sep. 1997) pp. 264–269.
12. L. Sheng, L. Wan-long, D. Yan-chun and F. Liang, "Forward Kinematics of the Stewart Platform Using Hybrid Immune Genetic Algorithm," *IEEE Conference on Mechatronics and Automation*, Bangkok, Thailand (Jun. 2006) pp. 2330–2335.
13. Y. Zhang, Z. Sun and T. Yang, "Optimal Motion Generation of a Flexible Macro-micro Manipulator System Using Genetic Algorithm and Neural Network," *IEEE Conference on Robotics, Automation and Mechatronics*, Bangkok, Thailand (Dec. 2006) pp. 1–6.
14. F. Chapelle, O. Chocron and Ph. Bidaud, "Genetic Programming for Inverse Kinematics Problem Approximation," *Proceedings of IEEE International Conference on Robotics and Automation*, Seoul, Korea, vol. 4, (2001) pp. 3364–3369.
15. P. Karla, P. B. Mahapatra and D. K. Aggarwal, "On the Solution of Multimodal Robot Inverse Kinematic Function Using Real-coded Genetic Algorithms," *IEEE International Conference on Systems, Man and Cybernetics*, Washington, DC, USA, vol. 2, (2003) pp. 1840–1845.
16. 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 International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, vol. 6, (2004) pp. 5356–5361.

17. S. Tabandeh, C. Clark and W. Melek, "A Genetic Algorithm Approach to Solve for Multiple Solutions of Inverse Kinematics Using Adaptive Niching and Clustering," *Proceedings of IEEE World Congress on Computational Intelligence*, Vancouver, Canada (July 2006) pp. 1815–1822.
18. O. Chocron and Ph. Bidaud, "Evolutionnary Algorithms in Kinematic Design of Robotic Systems," *International Conference on Intelligent Robots and Systems, September 7–11*, (1997), Grenoble, France, pp. 1111–1117.
19. P. I. Corke, "A robotics toolbox for MATLAB," *IEEE Robot. Autom. Mag*. **3**(1), 24–32 (1996).
20. J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality* (Princeton University Press, 1999).
21. S. L. Altmann, *Rotations, Quaternions, and Double Groups* (Dover Publications, 1986).
22. B. Sareni and L. Krähenbühn, "Fitness sharing and niching methods revisited", *IEEE Trans. Evol. Comput*. **2**(3), 97–106 (Sept. 1998).
23. D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization", *Proceedings of the 2nd International Conference Genetic Algorithms*, Cambridge, Massachusetts, United States (1987) pp. 41–49.
24. D. E. Goldberg and L. Wang, "Adaptive Niching via Coevolutionary Sharing", Technical Report, IlliGAL Report No. 97007 (Urbana, IL: University of Illinois at Urbana-Champaign, 1997).
25. M. Neef, D. Thierens and H. Arciszewski, "A Case Study of a Multiobjective Recombinative Genetic Algortihm with Coevolutionary Sharing," *Proceedings of 1999 Congress on Evolutionary Computation*, Washington, DC, USA, vol. 1, (1999).
26. K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst*. **9**(2), 115–148 (1995).
27. K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Inform*. **26**(4), 30–45 (1995).
28. S. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.* **2**(3), 268–278 (Sep. 1994).
29. G. Rudolph, "Evolutioary Search Under Partially Ordered Sets," Technical Report No. CI-67/99 (Dortmund, Germany: Department of Computer Science/LS11, University of Dortmund).
30. E. Zitzler, K. Deb and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput*. **8**(2), 173–195 (Feb. 1999).
31. Z. Li, W. Melek and C. Clark, "Development and Characterization of a Modular and Reconfigurable Robot," *Proceedings of the International Conference on Changeable, Agile, Reconfigurable and Virtual Production*, Toronto, Canada (Jul. 2007).