Goal Directed Synthesis of Serial Manipulators Based on Task Descriptions

Sarosh Patel, Tarek Sobh and Ausif Mahmood

Abstract Computing the optimal geometric structure of manipulators is one of the most intricate problems in contemporary robot kinematics. Robotic manipulators are designed and built to perform certain predetermined tasks. There is a very close relationship between the structure of the manipulator and its kinematic performance. It is therefore important to incorporate such task requirements during the design and synthesis of the robotic manipulators. Such task requirements and performance constraints can be specified in terms of the required end-effector positions, orientations and velocities along the task trajectory. In this work, we present a comprehensive method to develop the optimal geometric structure (DH parameters) of a non-redundant six degree of freedom serial manipulator from task descriptions. This methodology is devised to investigate possible manipulator configurations that can satisfy the task performance requirements under imposed joint constraints. Out of all the possible structures, the structures that can reach all the task points with the required orientations selected. Next, these candidate structures are then tested to see if they can attain end-effector velocities in arbitrary directions within the user defined joint constraints, so that they can deliver the best kinematic performance. Finally, the synthesized structures are tested to see if they perform the task under the operating constraints. In this work, we also present a novel approach for computing the inverse kinematics using Particle Swarm Optimization (PSO).

Keywords Global optimization · Manipulator synthesis · Simulated annealing · Task-based design

S. Patel (✉) · T. Sobh · A. Mahmood
University of Bridgeport, Bridgeport, CT, USA
e-mail: saroshp@bridgeport.edu

T. Sobh
e-mail: sobh@bridgeport.edu

A. Mahmood
e-mail: mahmood@bridgeport.edu

A.T. Azar and S. Vaidyanathan (eds.), *Chaos Modeling and Control Systems Design*,
Studies in Computational Intelligence 581, DOI 10.1007/978-3-319-13132-0_13

1 Introduction

The goal of robotics is to automate and delegate real-world tasks to robotic manipulators. Today robots are being applied to wide range of tasks; from the very traditional material handling tasks to the very sophisticated tele-robotic surgery.

Robotic manipulators are designed and built to perform certain predetermined tasks. Ideally speaking, one should easily be able to design a manipulator based on its application. The rapid growth in manufacturing technologies has increased the need for design and development of optimal machinery [1]. No longer is the emphasis on machinery that works but on machinery that works faster, consumes less power, and is more functional.

Even though general-purpose manipulators are commonplace they do not guarantee optimal task performance. Task optimized manipulators are more effective and efficient than general purpose manipulators. There is a great need for task optimized industrial manipulators that can perform a certain set of jobs with the best efficiency, in the shortest time, and with the least operating cost and power requirements. The availability of computing power allows us to design and evaluate multiple structures based on user defined criteria and select the best design.

What is the best manipulator configuration for soldering electronic components? What should be the ideal manipulator structure for a painting job? What is optimal manipulator configuration for a material handling job? Computing the optimal geometric structure of manipulators is one of the most intricate problems in contemporary robot kinematics [32].

Robotics researchers over the years have tried to find answers to these questions. But in this case plenty is the problem; there is no unique solution or definite answer to these questions. Instead, in most cases there can be infinite answers to any of the above questions. Equations describing the kinematic behavior of serial manipulators are highly nonlinear with no closed solutions. And the configuration search space is infinitely large. The difficulty in most cases lies not in finding a solution, but finding the 'best' solution out of the numerous possible solutions, or in other words, an optimal solution. Another big challenge in solving this problem is the number of parameters involved and the high non-linearity of the inverse kinematic equations [16]. There is a very close relationship between the structure of the manipulator and its kinematic performance [15, 16]. Researchers have over the years tried to develop a framework to reverse engineer optimal manipulator geometries based on task requirements [21].

Every robotic manipulator can only perform certain set of a set of tasks, and some more efficiently than others. Deciding the best manipulator structure for a required job at the design stage is done mainly on the basis of experience and intuition. The rigorous analysis of a few widely used manipulator structures and a collection of a few ad hoc analytical tools can be of some help [4, 23]. However, the need for a comprehensive framework to reverse engineer manipulator structures from task descriptions that can guarantee optimal task performance under a set of operating constraints is still lacking [21].

The aim of this work is to develop a goal directed design methodology that can serve as a simple and easy tool for kinematic synthesis of robotic manipulators based on task descriptions. The proposed methodology allows a user to enter the task point descriptions and joint constraints, and generates the optimal manipulator structure for the specified task. In this work we also present a novel approach for calculating the inverse kinematic solutions based on the Particle Swarm Optimization (PSO) algorithm. This approach helps in finding all the inverse solutions that lie within the constrained joint space with one run of the algorithm.

2 Existing Approaches

The research area of robotic manipulator design can be broadly classified into general purpose designs and task specific designs. Even though general purpose manipulators are commonplace, they do not guarantee optimal task execution. Because industrial robotic manipulators perform a given set of tasks repeatedly, task-specific or task-optimized manipulator designs are preferred for industrial applications.

The existing approaches for design and synthesis of serial manipulators can be broadly classified into the following three types:

2.1 Geometric Approach

Serial robotic manipulators are open-loop kinematic chains consisting of interconnected joints and links. There is a great body of research dealing with the mobility issues of closed loop kinematic chains. The principles of closed loop mechanical chains can be applied to design highly dexterous serial manipulators by assuming the distance between the base of the manipulator and the task point as a fixed and imaginary link in the closed mechanical chain.

Grashof [8] proposed a simple rule to judge the mobility of links in four-link closed kinematic chains. This rule was further extended and developed into Grashof's criterion by Paul [27]. Robotic researchers have applied Grashof's criterion to design manipulators with high dexterity at the given task points. Where dexterity refers to the ability of the manipulator to attain any orientation about a given point [37]. In [17, 25], authors proposed a method for the optimal design of three-link planar manipulators using Grashof's criterion. In [25] a simple algorithm for the optimal design of three link planar manipulators with full manipulator dexterity at the given task region or trajectory is proposed. The Grashof's criterion has also been extended by researchers to explain the behavior of longer kinematic chains. Ting introduced the five-link Grashof criterion [34] and later extended it to N-link chains [35, 36]. The main advantage of this method is its independence from the necessity to calculate the inverse kinematic solutions to judge its performance.

2.2 Parametric Optimization Approach

Parametric optimization is a classical way of solving an optimization problem One or more criteria that quantify the performance properties of the manipulator, sometimes with associated weighting factors, are maximized or minimized to arrive at a set of optimal design parameters. Parametric optimization has been one of the widely adopted approaches for the synthesis of serial manipulators. Condition number was used by Angeles and Rojas to obtain optimal dimensions for a three-DoF manipulator and three-DoF spherical wrist [2]. Craig and Salisbury used the condition number of the Jacobian as design criterion to optimize the dimensions of the fingers of the Stanford articulated hand [28].

In [32], optimal kinematic synthesis of the manipulator structure was based on the Yoshikawa manipulability ellipsoid at a given set of task points is presents. An objective cost function incorporating the Yoshikawa manipulability index was optimized using the *steepest-descent* algorithm over the manipulator's task trajectory to derive the optimal geometric structure. This work was implemented as a procedural package in Mathematica®[1] (version 4.1) and used the Robotica[2] version 3.60 (a robotics toolkit for Mathematica®). This work was further extended in [30, 31] to simulate the dynamic behavior of such an optimized manipulator.

Kucuk and Bingul [15, 16], implement a multi-variable optimization. The manipulator workspace was optimized based on a combination of local and global indices: Structural length index, manipulability measure, condition number, and global conditioning index.

These parametric optimization methods are task independent and hence do not guarantee the non-existence of a better manipulator for a specific task [22]. Another limitation of this approach is that it has a very limited scope due to the inherent limitations and general shortcomings of the performance metrics. A comprehensive survey of manipulator performance parameters and their limitations can be found in [26].

2.3 Task-Based Design Approach

Task-based design of manipulators uses the prior knowledge of application of the manipulator to design the best possible structure that can guarantee task completion. Task specifications can either be kinematic or dynamic. The ultimate goal of task-based design model is to be able to generate both the manipulator kinematic and dynamic parameters, using the task description and operating constraints [13]. This task-based design approach has seen considerable interest from researchers dealing

[1] [© 2002] Wolfram Research Inc.

[2] [© 1993] Board of Trustees, University of Illinois.

with re-configurable modular manipulators that can be easily re-configured depending on the task at hand.

Paredis and Kholsa [22], use the task requirements to find the optimal structure of a manipulator. They developed a numerical approach for determining the optimal structure of a six degree of freedom non-redundant manipulator. Their proposed method involves generating the DH parameters by minimizing an objective function using numerical optimization. This method does not check for non-singular positions at task points and the ability of the manipulator to generate effective velocities.

In [1], Al-Dios et al., developed a method for optimizing the link lengths, masses and trajectory parameters of a serial manipulator with known DH table using direct non-gradient search optimization. This work was focused to optimize the task time and joint torques for a specific manipulator task.

In [12, 13], authors propose the concept of Progressive Design as a frame work for the general design of manipulators and reconfigurable modulator manipulator systems, using task descriptions. The framework consists of three modules: kinematic design, planning and kinematic control. The kinematic design module encapsulates the task specifications, manipulator specifications and dexterity measure. In [12, 13], authors apply the proposed framework to develop an optimal manipulator for Space Shuttle tile changing operation, using dexterity as the optimizing criterion.

Dash et al. [6], propose a two stage methodology for structure and parameter optimization of reconfigurable parallel manipulator systems. They propose a '*TaskToRobot Map*' database that maps task description to a suitable manipulator configuration depending on the degrees of freedom required for a given task.

The manipulator configuration search space for all possible manipulator configurations is prohibitively large for evaluating all possible solutions, even if unacceptable solutions are eliminated early in the evaluation process. Two of the most applied approaches to search the configuration space are Random line search and Generic algorithms.

Authors in [5, 9, 11] recommend the use of Genetic Algorithms (GA) for designing the structure of self-organizing and modular robotic systems. In [29], authors Shiakolas et al. use evolutionary optimization approaches to optimize the design of a SCARA manipulator.

3 Problem Statement

The task descriptions can be given in terms of the task points p that the manipulator is supposed to reach with a specified orientation. Let P be the set of m task points that define the manipulator's performance requirements.

$$P = \{p_1, p_2, \ldots, p_m\} \in TS \tag{1}$$

All these points belong to the six-dimensional Task Space (*TS*) that defines both the position and orientation of the manipulator's end-effector. Each point in the Task Space (*TS*) can be given as:

$$p_i = \{x, y, z, \varphi, \theta, \psi\} \forall i = 1, 2, \ldots, m \in TS \tag{2}$$

where x, y, z are the real-world coordinates, and φ, θ, ψ are the roll, pitch and yaw angles about the standard Z, Y and X-axis. Figure 1 shows an example of a manipulator doing multiple tasks that require specific positioning and orientation of the manipulator at different points in the workspace.

In this work, we use the standard DH (Denavit-Hartenberg) notation to represent the manipulator structures [7]. The standard DH notation uses four parameters to define each link in the serial manipulator:

1. Link length (a)
2. Link twist (α)
3. Link offset (d)
4. Joint angle (θ)

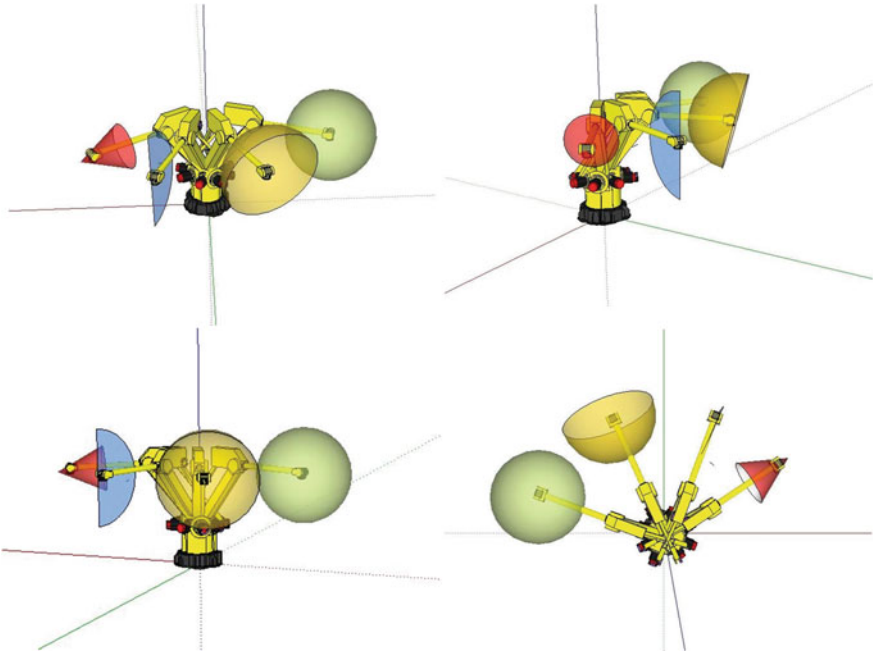

Fig. 1 Manipulator with different orientations at a set of task points

Depending on whether the link is revolute or prismatic, the joint angle (θ) or the link offset (d) is variable while the other three parameters remain constant for any given link. Therefore, each link will have three design parameters that describe it. In the case of a revolute link the design parameters are $\{a, \alpha, d\}$, and in the case of a prismatic link the design parameters are $\{a, \alpha, \theta\}$. A n degree serial manipulator configuration set (DH) can be given as:

$$DH = \{a_0, \alpha_0, \theta_0 \; or \; d_0, a_1, \alpha_1, \theta_1 \; or \; d_1, \ldots, a_{n-1}, \alpha_{n-1}, \theta_{n-1} \; or \; d_{n-1}\} \quad (3)$$

Therefore, an n-link serial manipulator will have $3n$ design parameters. Every set of manipulator configuration parameters can be said to be a point in the Configuration Space (C). Each set of values of the DH vector represents a unique manipulator configuration and a distinct point in the $3n$ dimensional Configuration Space (C).

$$DH = \{a_0, \alpha_0, \theta_0 \; or \; d_0, a_1, \alpha_1, \theta_1 \; or \; d_1 \ldots, a_{n-1}, \alpha_{n-1}, \theta_{n-1} \; or \; d_{n-1}\} \in C \quad (4)$$

Similarly, for an n degree of freedom manipulator, the joint vector q can be a said to be a point in the n dimensional Joint Space (Q), such that:

$$q = [q_1, q_2, \ldots, q_n] \in Q \quad (5)$$

Each joint vector q represents unique manipulator posture and a distinct point in the n dimensional Joint Space (Q). The Joint Space assumes there are no joint limitations (fully revolute ideal joints). But in practice the joints are not fully revolute and are bounded by lower and upper bounds. The values of the joint angles are range bound by user defined joint limits (upper and lower bounds). Hence, we define Q_c as the Constrained Joint Space, such that the joint displacements always satisfy the constraints:

$$q_{i,\min} \leq q_i \leq q_{i,\max} (q_i \in Q_c) \text{ and } Q_c \subset Q \quad (6)$$

Similarly, the manipulator's Reachable Workspace (WS) is defined as the set of points in the world coordinate system that the manipulator's end-effector can reach when no joint constraints are imposed. The manipulator's forward kinematic equations form a mapping $f(C) : Q \rightarrow WS$ between these three spaces: the Configuration Space (C), the Joint Space (Q) and the Workspace (WS).

When the manipulator's joint motion is restricted between joint limits the manipulator can only reach a part of the Reachable Workspace, known as the Constrained Reachable Workspace (CWS), such that $CWS \subset WS$. Constrained Reachable Workspace is defined as the set of points in the real coordinate system that the manipulator's end-effector can reach when joint constraints are imposed. This is given by the forward mapping: $f(C) : Q_c \rightarrow CWS$ and $Q_c \subset Q$.

Figure 2 shows the Reachable Workspace (WS) and Constrained Reachable Workspace (CWS) for a simple planar two-link manipulator as an illustrative example.

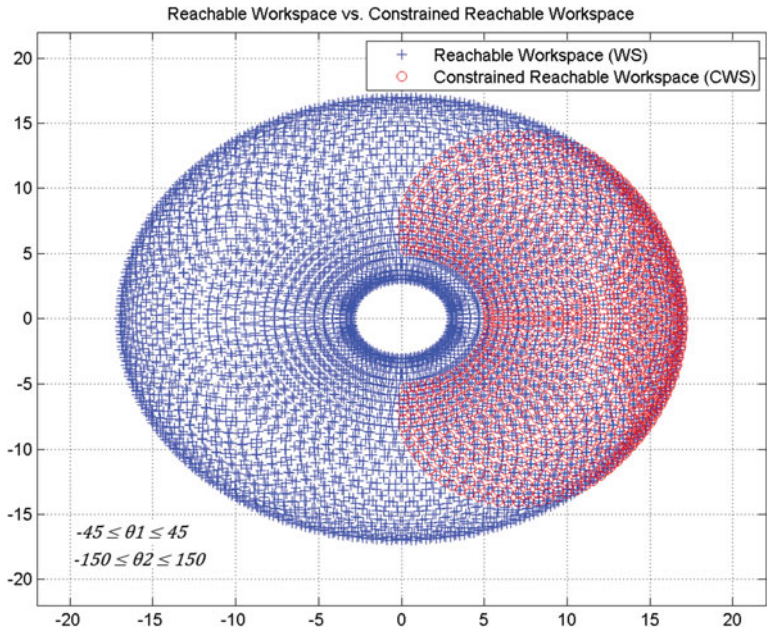

Fig. 2 Reachable workspace (WS) compared with constrained reachable workspace (CWS)

When a given manipulator of configuration set DH, with joint vector q can reach a specific task point p, the mapping can be represented as:

$$f(DH, q) = p \tag{7}$$

Therefore, the problem can be stated as: Find a solution set DH in the $3n$ dimensional Configuration Space such that there exists at least one q in the Constrained Joint Space that can reach the required position and orientation of the end-effector. i.e.

$$\text{Find all } DH \text{ such that } \forall p \in TS; \exists q \in Q_c | f(DH, q) = p$$

Even though this might seem to be a necessary and sufficient condition required for designing a manipulator, simulations and experience will suggest that this solution set might include a few manipulators that are able to reach the one or more of the task points only in singular positions. Such manipulators, if constructed, will not be able to attain good end-effector velocities in one or more directions due to their singular postures at the task point(s). Such manipulators will have very limited mobility at the required task point(s). Infinite forces have to be applied in order to generate motion along one or more directions at singularities. Therefore such manipulator configurations should be removed from the solution set. The test for

singularity is the determinant of the Jacobian matrix, which for a square Jacobian also happens to be the Yoshikawa manipulability index [38].

The Jacobian mapping from joint velocities to end-effector velocities for a manipulator is given as:

$$\xi = J(DH, q)\dot{q} \tag{8}$$

where $\xi = \left[\dot{x}\ \dot{y}\ \dot{z}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]$ is the end-effector velocity vector.

The Jacobian matrix is posture dependent matrix. It is also important to evaluate the Jacobian of the manipulator because the Jacobian matrix maps joint velocities to end-effector velocities, according to the mapping $J(DH, q) : \dot{q} \rightarrow \xi$. Hence, it is important to check if the Jacobian of manipulator at a given task point is well conditioned, and not in a singular posture. A manipulator with well-conditioned Jacobian at the task points will easily be able to transform joint velocities into end-effector velocities in any required direction, however the opposite cannot be said to be true on the basis of just the Jacobian determinant.

$$\xi = J_1\dot{q}_1 + J_2\dot{q}_2 + \cdots + J_n\dot{q}_n \tag{9}$$

Therefore, we modify the problem statement as follows:
Find all DH such that $\forall p \in TS; \exists q \in Q_c | f(DH, q) = p$ *and* $det(J(q)) \neq 0$

4 Solution Methodology

In this section we define two functions for evaluating the reachability and kinematic performance of the manipulator. To solve the problem we make the following assumptions:

1. The robot base is fixed and located at the origin O.
2. The task points are specified with respect to the manipulator's base frame.
3. The joint limitations are known to the designer.
4. If a joint is prismatic, the joint angle (θ) can assume values in the interval [−180, 180].
5. If a joint is revolute, the joint twist angle (α) can assume values [−180, 180].
6. The last three axes of the six degree of freedom manipulator intersect at a point to form a spherical wrist.
7. To limit the number of inverse kinematic solutions only non-redundant configurations are considered.

Let the task points be represented as $p = [x, y, z, \phi, \theta, \psi]$. The position of the operating point (OP) on the end-effector is given by $p_P = [x, y, z]$ and its orientation by $p_0 = [\phi, \theta, \psi]$.

$$p_i = [p_P \; p_0] \in TS \quad \forall i = 1, 2, 3 \ldots, m \tag{10}$$

In cases where multiple orientations are required at the same point the vector p_P remains same while the orientation vector p_O will assume different values.

The first criterion that needs to be satisfied is that all the points in the Task Space should be a part of the manipulator's Constrained Reachable Workspace. We define the Constrained Reachable Workspace as the set of points that the manipulator is able to reach under constrained joint limitations Q_c, while the normal reachable workspace (*WS*) is the set of points that the manipulator can reach with no joint limits, such that $CWS \subset WS$. Hence, given a set of task points P, the first objective is to find all possible manipulator configurations such that all task points in P are a part of the manipulator's Constrained Reachable Workspace (*CWS*).

$$Find\ all\ DH\ such\ that\ \forall p \in TS; p \in CWS$$

The Constrained Reachable Workspace (*CWS*) of the manipulator is given by the forward kinematic mapping $f(C) : Q_c \rightarrow CWS$. With the help of the standard DH notation parameters, the forward kinematic relationship is given as:

$$f(DH, q) = p \tag{11}$$

Due to the highly non-linear nature of the kinematic equations describing this forward kinematic mapping from the Joint Space to the Task Space, multiple manipulator postures or points in the Joint Space can lead to the same point in the Task Space. In such cases, point(s) in the Task Space will have more than one inverse kinematic solution.

$$q = f^{-1}(DH, p) \tag{12}$$

The inverse kinematic equations often have no unique solution. Depending on the manipulator's structure (*DH*) and location of the task points (p), the number of solutions might range from zero to infinite. And, even in the case where there are multiple known solutions to the above equations, it is still possible that none of them lie within the Constrained Joint Space (Q_c).

$$q = f^{-1}(DH, p) | q \in Q_c \tag{13}$$

In this work we use Particle Swarm Optimization based inverse kinematic approach for finding the inverse kinematic solutions within the constrained joint space. This numerical approach finds all possible inverse kinematic solutions within the specified joint constraints. This is discusses in detail in the following section.

To determine if the structure manipulator is able to reach a given task point with required orientation we construct a reachability function. The reachability function determines if the manipulator can reach and orient the end-effector at the task point within the set joint limitations.

$$reachability(DH) = \max\left[\min\left(\frac{(q_{i,\max} - q_i)(q_i - q_{i,\min})}{(0.5(q_{i,\max} - q_{i,\min}))^2}\right)_{i=1}^{n}\right]_{j=1}^{g} \tag{14}$$

where g is the number of inverse kinematic solutions.

When the joint angle displacements required to reach a task point are within the joint constraints the reachability function is bounded with in zero and unity. And, if the maipulator reaches the task point with at least one joint angle at it maximum displacement, the reachability function will have value of zero. The reachability function will have a maximum value of unity if the manipulator reaches the task point with all joint displacement being mid-range of their joint limits. A reachability value of unity is the ideal case and is only possible with one task point. If the one of the bounds is violated by any given joint out of the n manipulator joints the function will have a negative value. The reachablility function value for different locations of the task point is shown in Table 1.

Since we take a minimum of all the n joints, the reachability indicates the worst joint performance. This reachability function can help in the design of optimal manipulator structures by checking if they can reach the task point with proper joint displacements. To find the best reachable configurations the reachability function needs to be maximized.

Next, to select the best manipulator out of this set of manipulator configurations based their kinematic performance and manipulability. For this we write an objective function that can be maximized or minimized to obtain the optimal manipulator configuration.

$$f(DH)_{velocity} = \max\left[\det\left(J\left(q^1\right)\right), \det\left(J\left(q^2\right)\right), \ldots, \det(J(q^g))\right] \tag{15}$$

where g is the number of inverse kinematic solutions.

This objective function should be maximized to find the optimal manipulator structure that has the best conditioned Jacobian a task points. Such a manipulator will be able to easily transform joint velocities into needed end-effector velocities.

We extend the above formulation for reachability and kinematic performance to include all m points that define the Task Space, as a summation of the function values at the individual task points.

Table 1 Reachability function values

Location of the task point 'p'	Reachability function value
When p is inside the workspace and at least one solution is within joint constraints	[0, 1]
When p is inside the workspace and the best solution has at least one of the joint angles at its extreme position	0
When p is inside the workspace and the best solution is one with all joints displacements mid-range	1

$$reachability(DH) = \sum_{\forall p \in TS} \left(\max \left[\min \left(\frac{\left(q_{i,\max} - q_i\right)\left(q_i - q_{i,\min}\right)}{\left(0.5\left(q_{i,\max} - q_{i,\min}\right)\right)^2} \right)_{i=1}^{n} \right]_{j=1}^{g} \right) \quad (16)$$

$$f(DH)_{velocity} = \sum_{\forall p \in TS} \left(\max \left[\det\left(J\left(q^1\right)\right), \det\left(J\left(q^2\right)\right) \ldots, \det(J(q^g)) \right] \right) \quad (17)$$

To convert these functions into general optimization problems, such that minimizing them will yield optimal solutions we add a negative sign. The functions then become:

$$reachability(DH) = -\sum_{\forall p \in TS} \left(\max \left[\min \left(\frac{\left(q_{i,\max} - q_i\right)\left(q_i - q_{i,\min}\right)}{\left(0.5\left(q_{i,\max} - q_{i,\min}\right)\right)^2} \right)_{i=1}^{n} \right]_{j=1}^{g} \right) \quad (18)$$

$$f(DH)_{velocity} = -\sum_{\forall p \in TS} \left(\max \left[\det\left(J\left(q^1\right)\right), \det\left(J\left(q^2\right)\right) \ldots, \det(J(q^g)) \right] \right) \quad (19)$$

When multiple task points constitute a task goal these functions will have many local minima. This should be kept in mind while selecting a proper optimization algorithm. Using local minimization routines to find optimal solutions will yield acceptable solutions but not global solutions. Only global minimization routines will be able to deliver an optimal solution for the problem. The choice of the global minimization algorithm to be used depends on the number of iterations required, number for function evaluations and the speed of convergence.

4.1 Methodology Flowchart

The presented mathematical formulation and methodology can be represented in the form of a flow chart shown in Fig. 3. Random configurations are generated and tested for the existence of the inverse solutions within the joint limits range. In case a solution exists within the joint constraints, we further test the configurations for good manipulability and other additional performance criteria. Every reachable configuration is saved so that it can be used for further analysis and testing. Some of these configurations can also be used as initial starting points to optimization search algorithms. The final stop criteria can be set in terms of either the number of iterations, number of functional evaluations, or desired objective function value limit or a time limit.

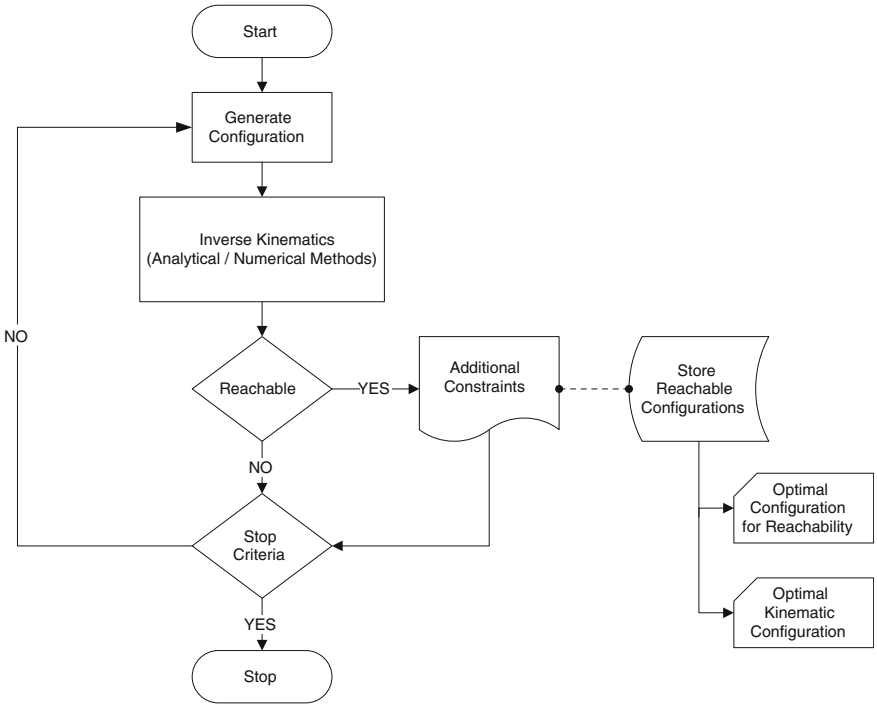

Fig. 3 Proposed methodology flowchart

4.2 Simulated Annealing

There are many approaches to solve a given global optimization problem. The choice of the algorithms greatly depends on factors such as the dimensionality of the problem, the nature of the variables (discrete or continuous), availability of a function derivative. A good global optimization method for a given problem can only be found by matching the features of the problem to the algorithm characteristics and its problem handling capabilities.

In this case, the objective or cost function—which is the reachability function—does not have a direct analytical expression, and is computationally expensive to calculate as it depends on the inverse kinematic solutions. It is also important to note here that this problem does not have a formulation for a function derivative or any function gradient data. The objective function will have multiple local and global minima points where the function value attains the desirable value. The search space is also very exhaustive. Keeping in mind the above factors we chose to implement the problem using Simulated Annealing algorithm. The simulated annealing method is a heuristic algorithm.

Simulated annealing was developed in the 1980s by Scott Kirkpatrick [14] based on a statistical algorithm developed much earlier by Metropolis [20], to improve designs of Integrated Circuit (IC) chips by emulating the actual process of annealing.

Simulated Annealing (SA) is a generic probabilistic meta-heuristic algorithm for finding the global minimum of a cost function that has many local minima. The SA algorithm uses random generated inputs based on a probabilistic model. Only under certain conditions is a change in the objective function due to a new random input accepted. The acceptance condition for a new input is given as follows:

$$\Delta f_{obj} \leq 0 \tag{20}$$

$$\exp\left(-\frac{\Delta f_{obj}}{T}\right) > random[0,1) \tag{21}$$

where Δf_{obj} is the change in the objective function and T is the temperature of the algorithm.

Beginning with a high temperature the algorithm with every iterative step gradually lowers the temperature simulating the annealing process. And, after every fixed number of iterations, known as the annealing period, the temperature is back raised again. Higher temperatures mean greater randomization of the input variables. Therefore, a slow annealing method that lowers the temperature gradually will explore the search space to a greater extent that a fast annealing method that lowers the temperature quickly. At lower temperatures the search space is exploited while at high temperature the algorithm explores the search space.

The algorithm stops when there is no change in the objective function for a certain number of consecutive inputs. SA algorithm remembers the best inputs throughout its run. SA works well with high dimensionality problems even when the search space is extensive.

The Simulated Annealing Method first generates random manipulator configurations that are then tested for reachability using the inverse solutions found by the Particle Swarm Optimization. The PSO based inverse kinematic module only searches for solutions within the user specified joint constraints. All the configurations that are found to be reachable are then further tested based on additional criteria. We keep re-annealing, by raising the temperature of the simulated annealing algorithm when the temperature of the algorithm reaches a minimum. The best reachability table is updated every time a better configuration is found.

4.3 Inverse Kinematics Using Particle Swarm Optimization

The Particle Swarm Algorithm (PSA) was designed to simulate the social behavior of organisms that behave in groups, commonly referred to as Swarm Intelligence. PSA mimics the population behavior followed by groups of animals such as a flock

of birds or a school of fish. PSA over the years has developed from a social behavior simulator to a global optimization algorithm. It was first proposed as a method for global optimization by Kennedy in [10]. Particle Swarm Optimization (PSO) belongs to the family of algorithms commonly referred to as Swarm Intelligence algorithms. The PSO method optimizes iteratively, by having particles learn from each other's position and move accordingly in the search space. With the help of a set of control parameters the algorithm maintains a good balance between the exploration and exploitation of the search space, by controlling the swarm population accordingly.

PSO is a meta-heuristic algorithm that can be used for problems with large search spaces. PSO is very easy to implement and it does not require large computational capabilities. PSO also does not require the objective function to have a gradient or to be differential. It determines the global minima points through cooperation and competition among the individuals of the population or agent particles [24]. PSO can solve complex problems faster than traditional algorithms due to its inherent characteristic of learning from the swarm population.

A very good survey of different applications of PSO to global optimization problems and its variants can be found in paper [24]. The inertia weight of the particles is one of the important factors that determines the how fast the PSO algorithm converges, and therefore has been a topic of interest and research. In [3], authors investigate different inertia weight strategies for PSO when applied to different problems to see which inertia update strategies work the best.

The PSO algorithm has attracted researchers from various backgrounds who have tried to improve its features by applying it to a wide range of problems. In [18, 19, 39] authors propose the use of dynamic multi-swarm methodologies for certain optimization problems.

Though PSO is originally meant to find a global minimum within the search space, it can be adapted to find multiple global minima, as required in this case. In this work the PSO has been used to find the inverse kinematic solutions for six degree of freedom manipulators. One main advantage of using PSO is that the search space can be limited to constrained joint space. Therefore, all solutions found will automatically lie within the constrained joint space, as opposed to the previous approach where all the solutions have to be found and then solutions that are outside the joint limits had to be rejected.

4.3.1 Example of a Two DoF Planar Manipulator

A two degree of freedom planar manipulator has a maximum of two inverse solutions for all points except when the arm is fully extended. These two are commonly referred to as 'elbow up' and 'elbow down' postures. Figure 4 below shows the two postures for a given point in the reachable workspace.

Consider a simple two-link planar manipulator, with link lengths $l_1 = l_2 = 1$ and the desired point $P(x, y) = (1, 1)$. We construct a simple error function in terms of the two joint angles, as follows:

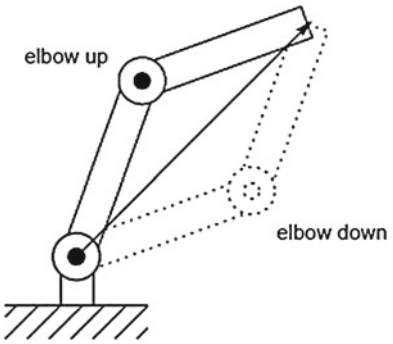

Fig. 4 Inverse kinematic solutions of a two link arm [33]

$$err = |x - l_1 \cos\theta_1 - l_2 \cos(\theta_1 + \theta_2)| + |y - l_1 \cos\theta_1 - l_2 \cos(\theta_1 + \theta_2)| \quad (22)$$

The plot for the error function over the range of θ_1 and θ_2 is shown in Fig. 5. The two minima points are the required inverse kinematic solutions. Figure 6 below shows the contour plot of the error function and the location of the two solutions.

The error function for such a simple manipulator can easily be visualized in a three dimensional plot, but this cannot be done for higher order manipulators that require more than three dimensions. This position error function is given as an input to the PSO algorithm. The different stages of the swarm optimization are shown below in Fig. 7.

The swarm particles/agents can be seen as red dots on the function surface. The swarm particles finally converge at the two minima points in the final plot.

To identify the two global solutions after a specified number of iterations of the PSO algorithm, the following steps are implemented:

(1) Sort the particles in ascending order of the error function value.
(2) Eliminate particles that have an error function value greater than a specified threshold.
(3) Group particles that lie with a specified radius.

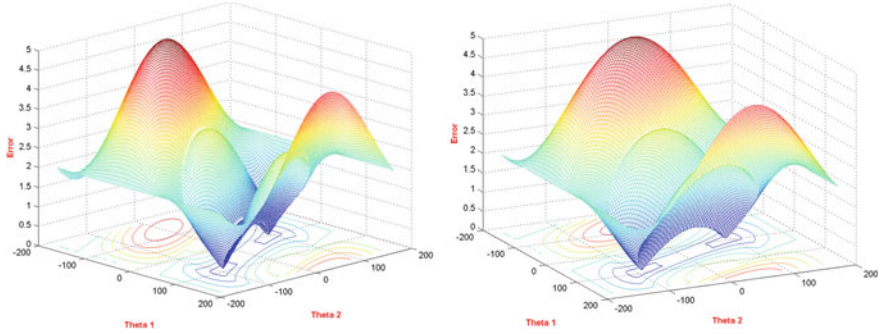

Fig. 5 Position error plot and contour

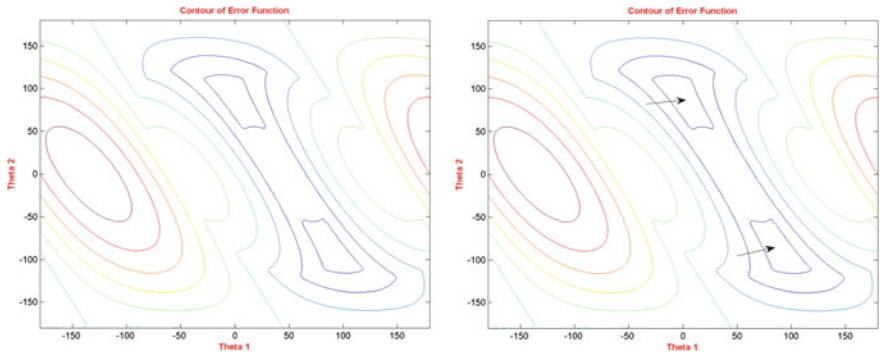

Fig. 6 Position error contour showing the solution points

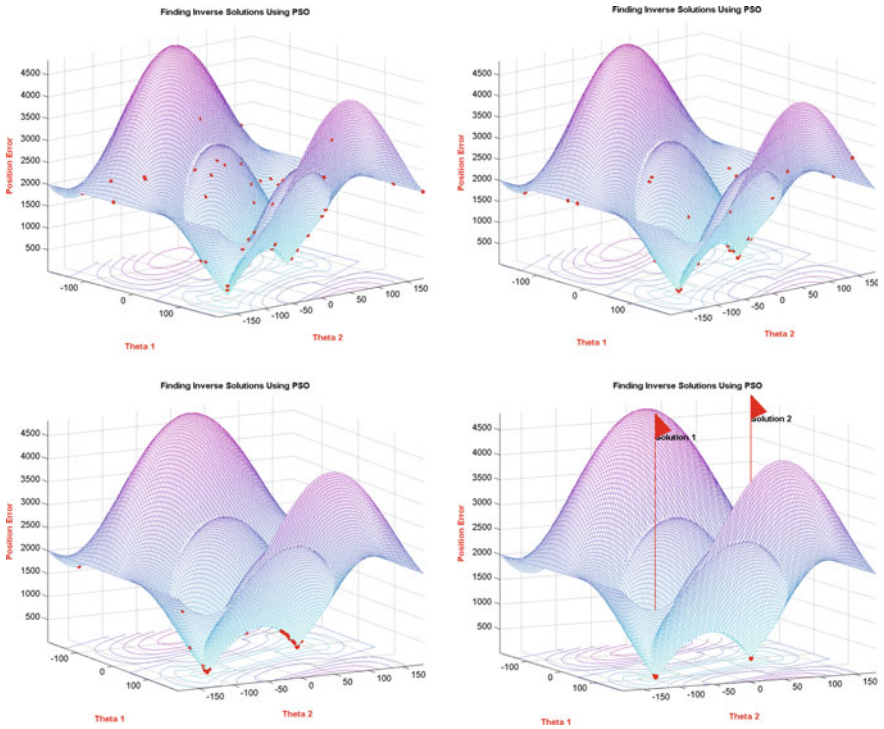

Fig. 7 Stages of the PSO while finding the inverse kinematic solutions

The number of PSO iterations, the threshold, and the grouping radius have to be carefully selected so that all possible solutions can be identified. Fewer iterations will not allow enough time for all the particles to reach the multiple minima points. Choosing a higher threshold can give false solutions that are at a distance from the

goal, while using a very low threshold can eliminate real solutions too. Furthermore, having a large grouping radius can lead to two or more solutions being grouped together.

4.3.2 Six Degree-of-Freedom Example

Next we apply the same PSO bases methodology to compute the inverse kinematic solutions of the six link manipulator with a spherical. When PSO is applied directly to compute all six joint angles for a given reachable point, the PSO has a hard time converging on the solutions due to the high dimensionality of the problem. To do this we approach the problem using the *Greedy Optimization* philosophy, according to which, the global optimum to a large problem consists of the global optimum to each of its sub-problems. Due to the presence of the spherical wrist we can decouple the positioning and the orienting equations as two sub-problems. Finding the global solutions to these two sub-problems will automatically solve the larger inverse kinematics problem.

In this work we have implemented the inverse kinematic solution algorithm in two stages. The first run of the PSO finds all possible joint angles for the first three joints such that the manipulator is able to reach the desired point. Next, for each of the set of solutions for the first three joints the PSO is run to find the possible set of joint angles for the wrist joints such that the desired end-effector orientation can be achieved. This approach also saves precious computation time as there is no need to find the wrist solutions if the arm cannot reach the desired position. Hence, the wrist angles are only calculated if the arm is able to position itself at the desired point. For the class of manipulators with six degrees of freedom and a spherical wrist, below is the algorithm to find the inverse kinematic solutions.

Let P be the target point in the task space such that P can be decoupled into positioning and orienting terms as:

$$P = [p_P \; p_o] = [x, y, z, \emptyset, \theta, \psi] \tag{23}$$

The angles $\emptyset, \theta, \psi$ are such that successive transformations about the respective axes by these angles should lead to the required end-effector orientation, such that:

$$R_6^0 = R(\phi, \theta, \psi) = R_{z,\phi} R_{y,\theta} R_{x,\psi} \tag{24}$$

This final end-effector orientation matrix can also be represented in terms of the normal, sliding and approach vectors as follows:

$$R_6^0 = R(\phi, \theta, \psi) = [n \; s \; a] \tag{25}$$

The following algorithm is used to find the inverse kinematic solution for a given point. First, the wrist center (c) is found by using the approach vector (a):

$$c = p_P - d_6 a \tag{26}$$

Next, we solve the forward kinematic equations for positioning the wrist center (c) at the desired point, using PSO. This yields sets of first three joint angles q_1, q_2, q_3 that can place the wrist center at the desired point. To solve the position of the manipulator using PSO we set up a position error function (*err*):

$$err = \sum |p_P - f(q_1, q_2, q_3)| \tag{27}$$

The multiple minima points of this position error function are the possible sets of the first three joint angles:

$$[q_1, q_2, q_3] = pso(err) \tag{28}$$

Next, for each set of possible q_1, q_2, q_3 we need to find q_4, q_5, q_6 such that the desired orientation is possible. To do this we first calculate the desired orientation due to the last three joints.

$$R_6^3 = \left(R_3^0\right)^T R_6^0 \tag{29}$$

Using Particle Swarm Optimization we solve R_6^3 to get the last three joint angles:

$$[q_4, q_5, q_6] = pso\left(R_6^3 - f(R)\right) \tag{30}$$

The PSO algorithm is configured to search for solutions only within the joint limits; this eliminates the need to check for solutions lying outside the joint limits.

4.3.3 Puma560 Inverse Kinematics

In this section the above algorithm is applied to a PUMA 560 robotic arm to find its inverse kinematic solutions. For most points in the reachable workspace the PUMA 560 arm has four solutions for the inverse position kinematics (unless they violate the joint constraints) as shown in Fig. 8.

The upper bound (UB) and lower bound (LB) for the six revolute joints for a Puma560 manipulator arm in degrees are as follows:

$$\begin{aligned} \text{LB} &= [-160, -45, -225, -110, -100, -266] \\ \text{UB} &= [160, 225, 45, 170, 100, 266] \end{aligned}$$

Below are the example test runs of the PSO-based inverse kinematics method.

a. **Home position**—Here we find the inverse kinematic solutions for the PUMA 560's home position which is given by the point P. The orientation angles are all

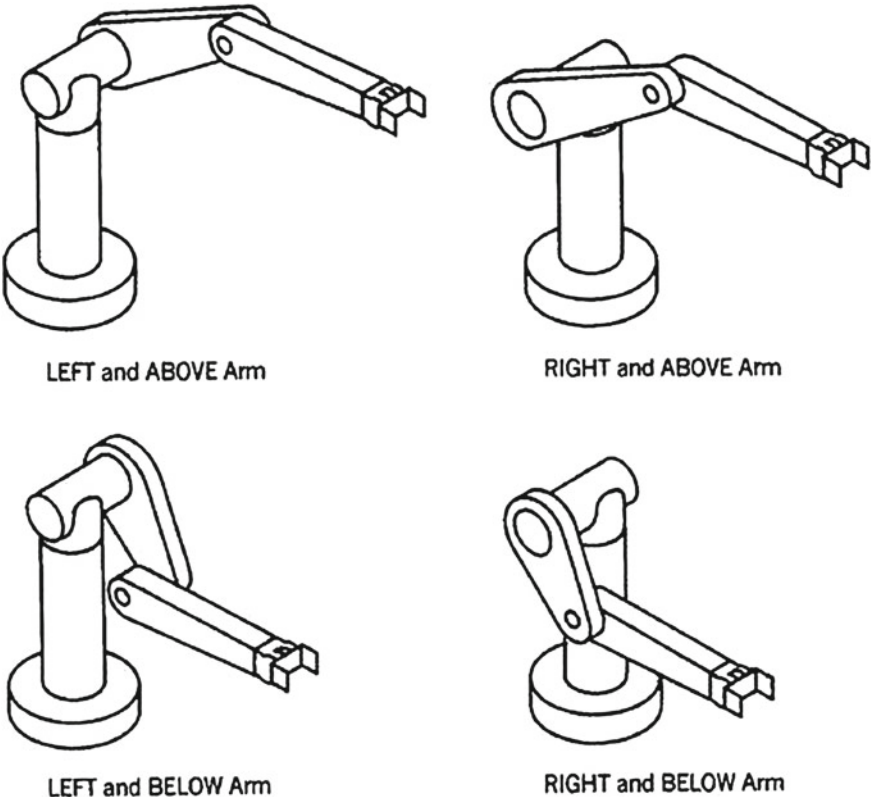

Fig. 8 Four solutions of the inverse position kinematics for the PUMA arm [33]

zero in this case. Figure 9 shows the four inverse position solutions that lead to the same point. Figure 10 shows the end-effector orientation. As seen in Fig. 10 all the normal, sliding, and approach axes of the end-effector perfectly coincide for all four solutions.

$$\mathrm{P} = [\mathrm{x},\ \mathrm{y},\ \mathrm{z}, \emptyset, \theta, \psi] = [0.4521, -0.1500, 0.4318, 0, 0, 0]$$

b. **Top position**—In this example another point P is chosen with arbitrary orientation angles. Figure 11 shows the four inverse position solutions that lead to the same point. Figure 12 shows the end-effector orientation. Again, all the orientation axes of the end-effector perfectly coincide for all four solutions .

$$\mathrm{P} = [\mathrm{x},\ \mathrm{y},\ \mathrm{z}, \emptyset, \theta, \psi] = [0.4521, -0.1500, 0.4318, 0, 0, 0]$$

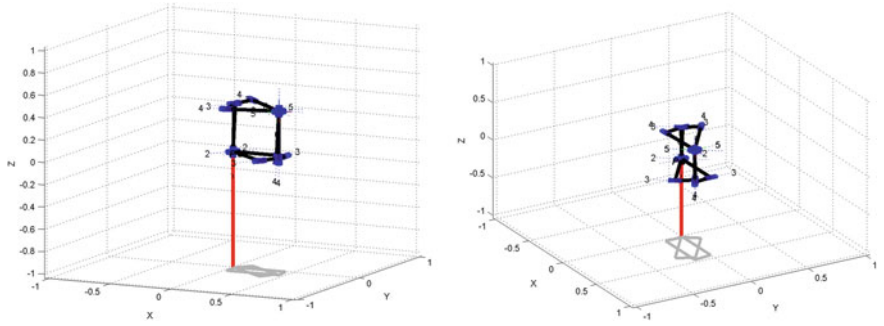

Fig. 9 Inverse position kinematics solutions

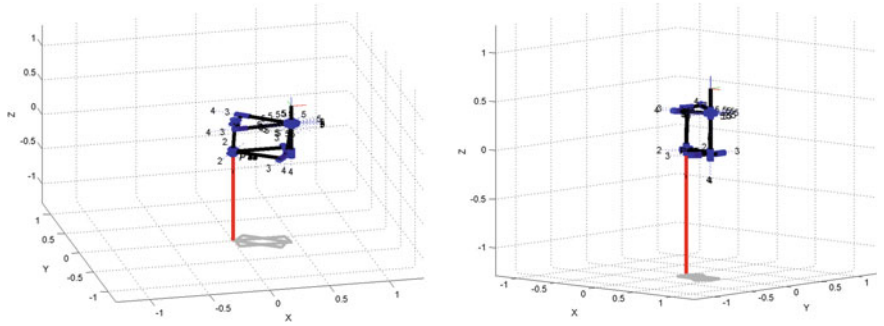

Fig. 10 Inverse position and orientation solutions

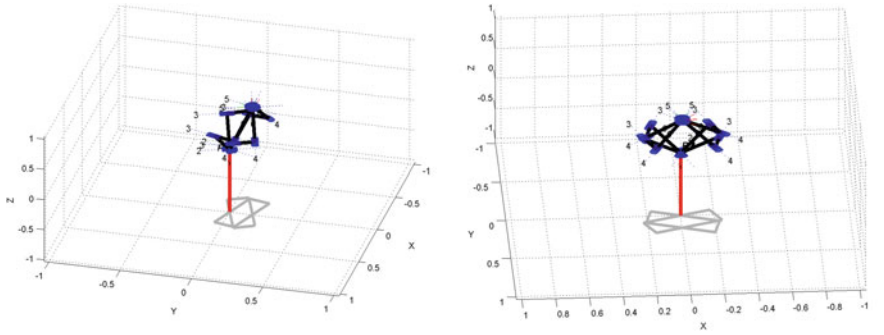

Fig. 11 Inverse position kinematics solutions

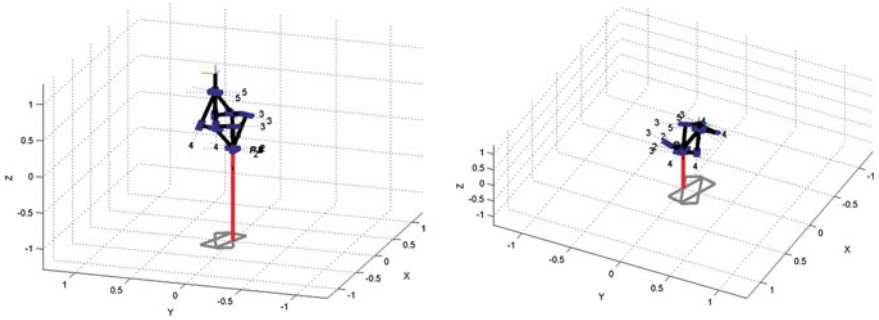

Fig. 12 Inverse position and orientation solutions

5 Experimental Results

In this section we test the proposed methodology to design manipulators based on task point descriptions. The task goals differ in the number of task points and also in the orientation required at these task points. For a prismatic link the joint limit is constrained between zero and unity. The joint limit constraints for the revolute joints are set as follows:

$$\text{Lower Bound} = [-160,\ -45,\ -225,\ -110,\ -100,\ -266]$$
$$\text{Upper Bound} = [160, 225, 45, 170, 100, 266]$$

5.1 Spherical Goal

In this task the manipulator is required to have the ability to reach a task point from all possible approaches or angles. This task involves approaching a point from six different angles separated by 90°, such that they represent the three diagonals of a sphere perpendicular to each other. The task points for a sphere goal are given below.

```
Sphere goal = [
                0 0.75 0 0 0 0;
                0 0.75 0 -3.142 0 -3.142;
                0 0.75 0 0  1.565 0;
                0 0.75 0 0  -1.565 0;
                0 0.75 0 -1.372 1.541 -3.142;
                0 0.75 0 1.784 -1.571 -0.213
            ];
```

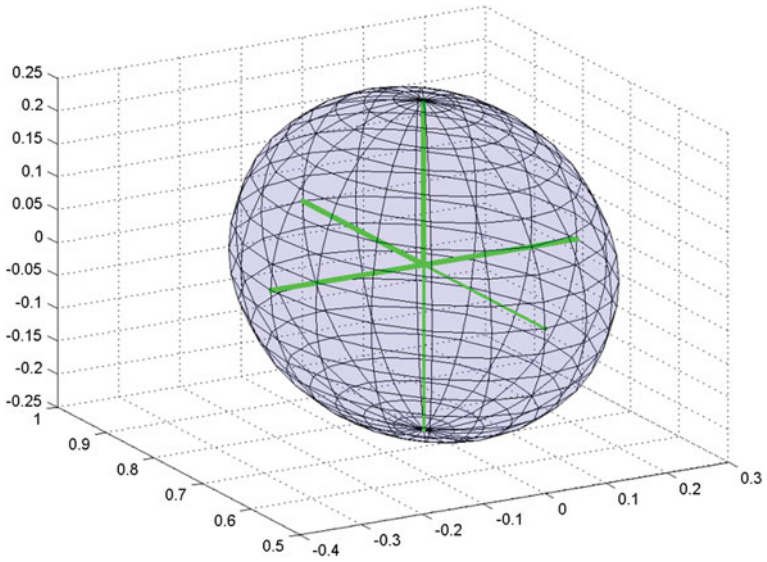

Fig. 13 Task description for the spherical goal

The task visualization is shown in Fig. 13.

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability value for this set of points of the sphere is an RRR-RRR manipulator. This configuration has a reachability value of −0.5441.

The DH parameters of the manipulator are:

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|     0.9979|     0.8345|     -2.375|
|  2|         q2|     0.7467|     0.9979|      3.101|
|  3|         q3|     0.0025|     0.9978|      2.269|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

Figure 14 shows superimposed manipulator positions at the required task points.

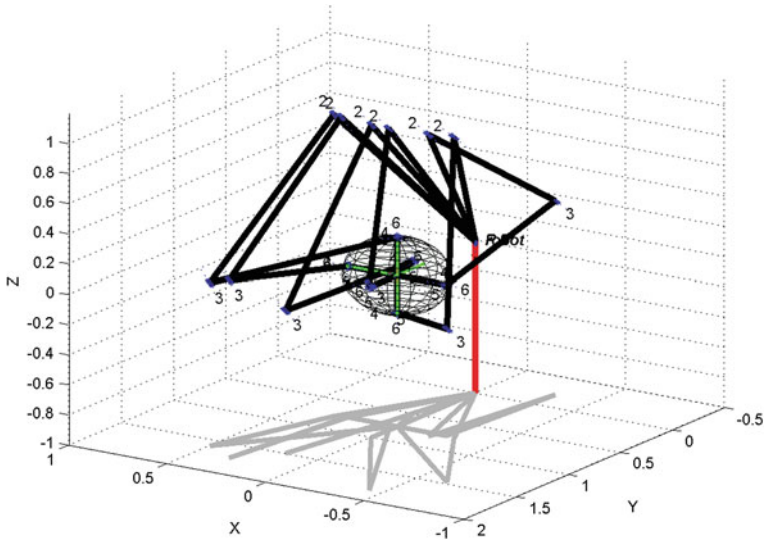

Fig. 14 Designed manipulator reaching all the task points of the spherical goal

For this goal the best kinematic performance structure was found to be:

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|     0.9956|     0.6492|     -2.342|
|  2|         q2|     0.9973|     0.9956|      3.057|
|  3|         q3|   0.005252|     0.9954|      2.167|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

5.2 Circular Ring Goal

In this task the manipulator is required to reach eight points on the circumference of a circle with the same orientation at all the task points. The task points for the ring goal are given below.

```
Ring Goal = [
   0.7000   0.5000      0        -3.142 0 -3.142
   0.6414   0.6414      0        -3.142 0 -3.142
   0.5000   0.7000      0        -3.142 0 -3.142
   0.3586   0.6414      0        -3.142 0 -3.142
   0.3000   0.5000      0        -3.142 0 -3.142
   0.3586   0.3586      0        -3.142 0 -3.142
   0.5000   0.3000      0        -3.142 0 -3.142
   0.6414   0.3586      0        -3.142 0 -3.142
];
```

The task visualization is shown in Fig. 15.

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability value for this set of points of the ring task is an RRR-RRR manipulator. This configuration has a reachability value of −0.833

The DH parameters of the manipulator are:

```
robot (6 axis, RRRRRR, stdDH)
```

j	theta	d	a	alpha
1	q1	0.049	0.6576	0.7544
2	q2	0.817	0.908	3.02
3	q3	0.9482	0.6897	1.264
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

Figure 16 shows superimposed manipulator positions at the required task points.

For this goal the best kinematic performance structure was found to be:

```
robot (6 axis, RRRRRR, stdDH)
```

j	theta	d	a	alpha
1	q1	0.04902	0.6576	0.7545
2	q2	0.8169	0.908	3.02
3	q3	0.9482	0.6897	1.264
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

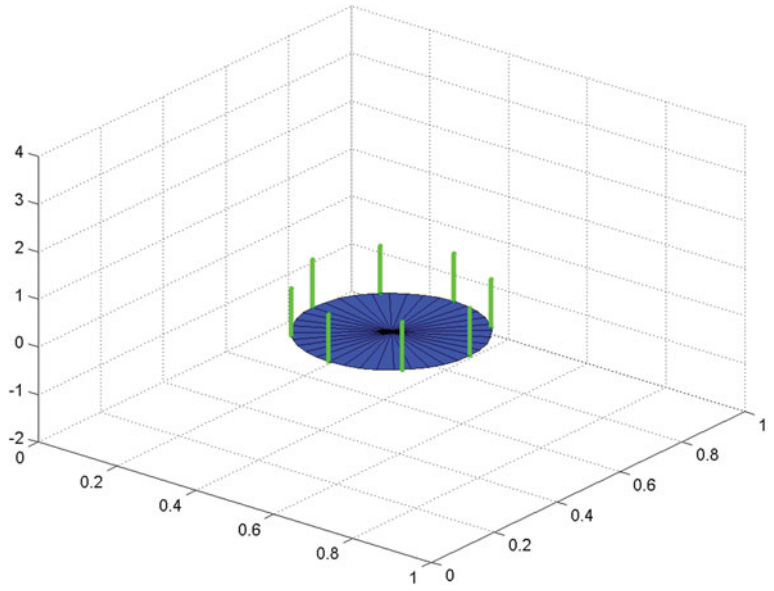

Fig. 15 Task description for the ring goal

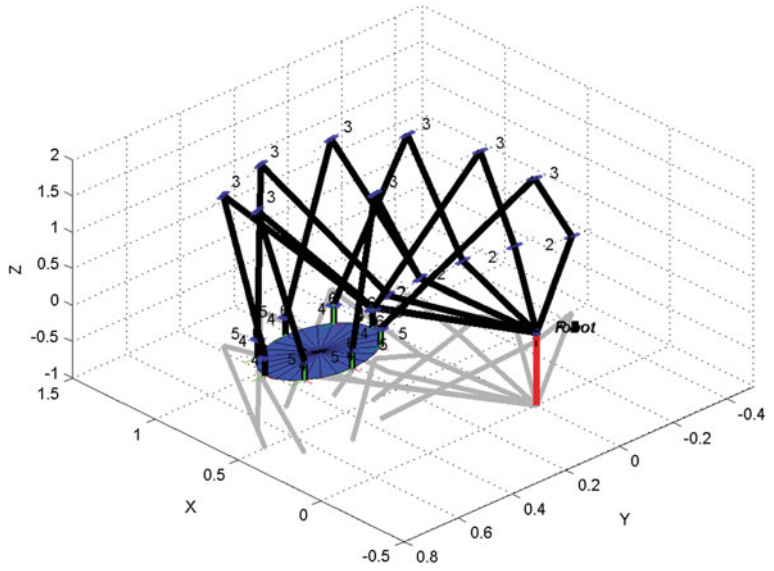

Fig. 16 Designed manipulator reaching all the task points of the ring goal

5.3 Horizontal Plane Goal

This task comprises of nine points that lie in a horizontal plane, the manipulator is supposed to reach all of the task points with the same orientation. This task is similar to the task manipulators execute in the packaging/soldering application. The task points for the horizontal plane goal are given below.

```
Horizontal Plane Goal = [
                        0.9  -0.5  0 -3.142 0 -3.142;
                        0.9   0    0 -3.142 0 -3.142;
                        0.9   0.5  0 -3.142 0 -3.142;
                        0.7  -0.5  0 -3.142 0 -3.142;
                        0.7   0    0 -3.142 0 -3.142;
                        0.7   0.5  0 -3.142 0 -3.142;
                        0.5  -0.5  0 -3.142 0 -3.142;
                        0.5   0    0 -3.142 0 -3.142;
                        0.5   0.5  0 -3.142 0 -3.142;
                    ];
```

The task visualization is shown in Fig. 17.

Based on the evaluations of all possible configurations, the best configuration that has the maximum overall reachability value for this set of points of the horizontal goal is an RRR-RRR manipulator. This configuration has a reachability value of −0.68127.

The DH parameters of the manipulator are:

```
robot (6 axis, RRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
|  1|         q1|     0.2472|     0.6404|      0.104|
|  2|         q2|     0.0019|     0.6147|      1.404|
|  3|         q3|     0.3707|     0.3709|     -1.135|
|  4|         q4|          0|          0|     -1.571|
|  5|         q5|          0|          0|      1.571|
|  6|         q6|       0.25|          0|          0|
+---+-----------+-----------+-----------+-----------+
```

Figure 18 shows superimposed manipulator positions at the required task points.

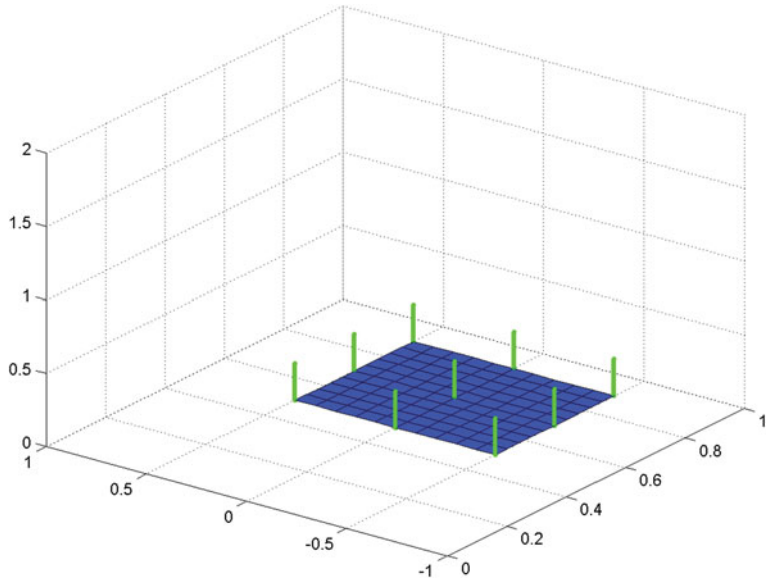

Fig. 17 Task requirements for the horizontal plane goal

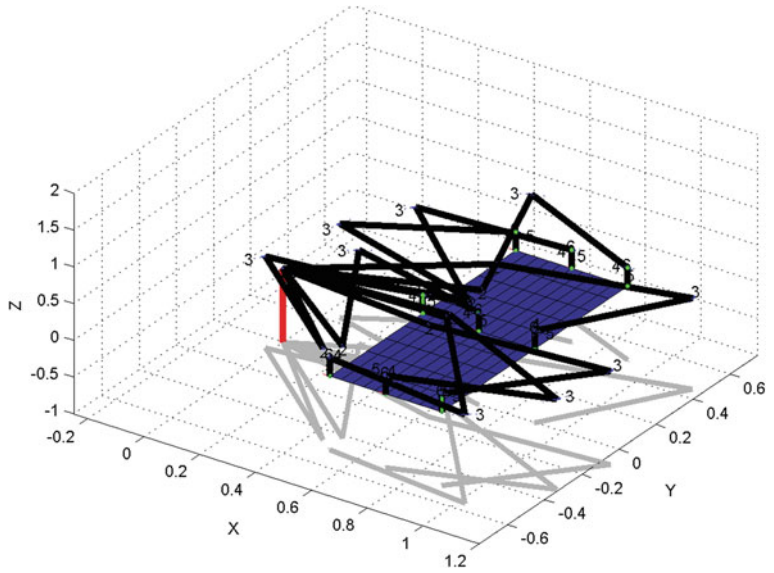

Fig. 18 Designed manipulator reaching all the task points of the horizontal goal

For this goal the best kinematic performance structure was found to be:

robot (6 axis, RRRRRR, stdDH)

j	theta	d	a	alpha
1	q1	0.1855	0.9	2.619
2	q2	0.07636	0.3638	-1.332
3	q3	0.5625	0.3354	-2.649
4	q4	0	0	-1.571
5	q5	0	0	1.571
6	q6	0.25	0	0

6 Discussion

In all the task experiments the initial seed to the algorithm was a set of random values such that the resultant configuration did not constitute an existing structure and did not reach even a single task point. The methodology then iteratively found a set of reachable configurations from which task suitable configurations are selected.

The optimal manipulator structures for the best reachability, and kinematic performance are not always the same. They can be two different manipulators. A manipulator structure having a very good reachability value for a set task may not actually be the most efficient manipulator. Therefore, selecting the right manipulator will involve a certain intelligent trade off with respect to these parameters.

As expected for most of the tasks, the best manipulator structure found happened to be a RRR/RRR manipulator. This supports the fact that most industrial manipulators are RRR robots with spherical wrists as they provide better reachability at the task points and also the ability to orient the end-effector arbitrarily in the workspace.

The manipulator structures that were generated by the methodology for each of the tasks are not ones that would intuitively come to mind for those tasks. Using this task based tool to design manipulators can help the designer in evaluating new and different configurations.

In some cases a few structures failed to reach all the task points with the necessary orientation required for task completion. For example no RPP/RRR configuration could be found that could successfully complete the sphere goal task within the set joint constraints.

7 Conclusion

In this work we have presented a general methodology for task-based prototyping of serial robotic manipulators. This framework can be used to generate specialized goal oriented manipulator structures based on the task descriptions. The framework allows for practical joint constraints to be imposed during the design stage of the manipulator. This methodology incorporates the necessary criteria for the design of a manipulator, such as reachability, orientation and non-singularity. However the sufficient condition can be specified by the user, by incorporating additional constraints. In this work we have used a novel approach based on particle swarm optimization to calculate the inverse kinematic solutions. This work can be viewed as part of a broader program to develop a general framework for the reverse prototyping of robotic manipulators based on task descriptions and operating constraints.

References

1. Al-Dois, H., Jha, A.K., Mishra, R.B.: Task-based design optimization of serial robot manipulators. Eng. Optim. **45**(6), 647–658 (2012)
2. Angeles, J., Rojas, A.: Manipulator inverse kinematics via condition-number minimization and continuation. Int. J. Robot. Autom. **2**(2), 61–69 (1987)
3. Bansal, J.C., Singh, P.K., Saraswat, M., Verma, A., Jadon, S.S., Abraham, A.: Inertia weight strategies in particle swarm optimization. In: Proceedings of the Third World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 633–640 (2011)
4. Bohigas, O., Manubens, M., Ros, L.: A complete method for workspace boundary determination on general structure manipulators. IEEE Trans. Rob. **28**(5), 993–1006 (2012)
5. Chung, W.K., Jeongheon, H., Youm, Y., Kim, S.H.: Task based design of modular robot manipulator using efficient genetic algorithm. In Proceedings of 1997 IEEE International Conference on Robotics and Automation, pp. 507–512 (1997)
6. Dash, A.K., Chen, I.M., Yeo, S.H., Yang, G.: Task-oriented configuration design for reconfigurable parallel manipulator systems. Int. J. Comput. Integr. Manuf. **18**(7), 615–634 (2005)
7. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices. Trans. ASME J. Appl. Mech. **22**, 215–221 (1955)
8. Grashof, F.: Thertische Mshinenlehre. Leipzig, pp. 113–183 (1883)
9. Izumi, K., Tamura, H., Watanabe, K.: Task-oriented optimal configuration structure in a three-dimensional self-organizing robot by genetic algorithms. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (SMC'99), pp. 740–745 (1999)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
11. Kim, J., Khosla, P.K.: A multi-population genetic algorithm and its application to design of manipulators. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 279–286 (1992)
12. Kim, J., Khosla, P.K.: Design of space shuttle tile servicing robot: an application of task based kinematic design. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 867–874 (1993a)

13. Kim, J., Khosla, P.K.: A formulation for task based design of robot manipulators. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2310–2317 (1993b)
14. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
15. Kucuk, S., Bingul, Z.: Robot workspace optimization based on a novel local and global performance indices. In: Proceedings of IEEE International Symposium on Industrial Electronics, pp. 1593–1598 (2005)
16. Kucuk, S., Bingul, Z.: Comparative study of performance indices for fundamental robot manipulators. Robot. Auton. Syst. **54**(7), 567–573 (2006)
17. Li, R., Dai, J.S.: Orientation angle workspaces of planar serial three-link manipulators. Sci. China Ser. E: Technol. Sci. **52**(4), 975–985 (2009)
18. Liang, J.J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer. In: Proceedings of the 2005 IEEE Swarm Intelligence Symposium (SIS 2005), pp. 124–129 (2005a)
19. Liang, J.J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer with local search. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, pp. 522–528 (2005b)
20. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
21. Oetomo, D., Daney, D., Merlet, J.P.: Design strategy of serial manipulators with certified constraint satisfaction. IEEE Trans. Rob. **25**(1), 1–11 (2009)
22. Paredis, C.J.J., Khosla, P.K.: Kinematic design of serial link manipulators from task specifications. Int. J. Robot. Res. **12**(3), 274–287 (1993)
23. Park, F.C., Brockett, R.W.: Kinematic dexterity of robotic mechanisms. Int. J. Robot. Res. **13** (1), 1–15 (1994)
24. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput. **1**(2–3), 235–306 (2002)
25. Patel, S., Sobh, T.: Optimal design of three-link planar manipulators using Grashof's criterion. In: Sobh, T., Xiong, X. (eds.) Prototyping of Robotic Systems: Applications of Design and Implementation, pp. 70–84. IGI Global, Hershey (2012)
26. Patel, S., Sobh, T.: Manipulator performance measures—a comprehensive literature survey. J. Intell. Robot. Syst. 1–24 (2014). doi:10.1007/s10846-014-0024-y
27. Paul, B.: A reassessment of Grashof's criterio. Trans. ASME J. Mech. Design **101**(3), 515–518 (1979)
28. Salisbury, J.K., Craig, J.J.: Articulated hands: force control and kinematic issues. Int. J. Robot. Res. **1**(1), 4–17 (1982)
29. Shiakolas, P.S., Koladiya, D., Kebrle, J.: Optimum robot design based on task specifications using evolutionary techniques and kinematic, dynamic, and structural constraints. Inverse Probl. Eng. **10**(4), 359–375 (2002)
30. Sobh, T., Wang, B., Patel, S.: A mobile wireless and web based analysis tool for robot design and dynamic control simulation from task points description. J. Internet Technol. **4**(3), 153–162 (2003a)
31. Sobh, T., Wang, B., Patel, S.: Web enabled robot design and dynamic control simulation software solutions from task points description. In: Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society (IECON'03), pp. 1221–1227 (2003b)
32. Sobh, T.M., Toundykov, D.Y.: Optimizing the tasks at hand (robotic manipulators). IEEE Robot. Autom. Mag. **11**(2), 78–85 (2004)
33. Spong, M.W., Vidyasagar, M.: Robot Dynamics and Control. Wiley, New York (1989)
34. Ting, K.-L.: Five-bar Grashof criteria. J. Mech. Transmissions Autom. Design **108**(4), 533–537 (1986)
35. Ting, K.-L.: Mobility criteria of single-loop N-bar linkages. J. Mech. Transmissions Autom. Design **111**(4), 504–507 (1989)
36. Ting, K.-L., Liu, Y.-W.: Rotatability laws for N-bar kinematic chains and their proof. J. Mech. Des. **113**(1), 32–39 (1991)

37. Vijaykumar, R., Waldron, K.J., Tsai, M.: Geometric optimization of serial chain manipulator structures for working volume and dexterity. Int. J. Robot. Res. **5**(2), 91–103 (1986)
38. Yoshikawa, T.: Manipulability of robotic mechanisms. Int. J. Robot. Res. **4**(2), 3–9 (1985)
39. Zhao, S.Z., Liang, J.J., Suganthan, P.N., Tasgetiren, M.F.: Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization. In: Proceedings of (IEEE World Congress on Computational Intelligence) IEEE Congress on Evolutionary Computation (CEC 2008), pp. 3845–3852 (2008)