Robotica (2016) volume 34, pp. 1979–2008. © Cambridge University Press 2014
doi:10.1017/S0263574714002690

# A memetic algorithm approach for solving the task-based configuration optimization problem in serial modular and reconfigurable robots

Saleh Tabandeh†, William Melek‡, Mohammad Biglarbegian§ , Seong-hoon Peter Won* ‡ and Chris Clark¶

†*Motion Control, Fanuc robotics, Rochester Hills, Michigan, USA*
‡*Mechanical and Mechatronics engineering, University of Waterloo, Waterloo, Ontario, Canada*
§*School of Engineering, University of Guelph, Guelph, Ontario, Canada*
¶*Harvey Mudd College, Claremont, California, USA.*

(Accepted October 30, 2014. First published online: December 8, 2014)

**SUMMARY**
This paper presents a novel configuration optimization method for multi degree-of-freedom modular reconfigurable robots (MRR) using a memetic algorithm (MA) that combines genetic algorithms (GAs) and a local search method. The proposed method generates multiple solutions to the inverse kinematics (IK) problem for any given spatial task and the MA chooses the most suitable configuration based on the search objectives. Since the dimension of each robotic link in this optimization is considered telescopic, the proposed method is able to find better solutions to the IK problem than GAs. The case study for a 3-DOF MRR shows that the MA finds solutions to the IK problem much faster than a GA with noticeably less reachability error. Additional case studies show that the proposed MA method can find multiple IK solutions in various scenarios and identify the fittest solution as a suboptimal configuration for the MRR.

KEYWORDS: memetic algorithms; reconfigurable robots; configuration optimization; reachability.

## 1. Introduction

While non-modular industrial robots can successfully perform specific tasks, they have a limited ability to adapt their configuration when tasks are changed. In some cases, when tasks change, numerous issues can arise as a result of the robot singularities, saturation of the actuators torques, or mechanical limitations of the joint modules. Consequently, reconfigurable manufacturing systems (RMSs) become a more attractive alternative compared to conventional non-modular industrial robots in these circumstances. According to the Visionary Manufacturing Challenges for 2020 by the USA National Research Council, RMSs are the most vital of all priority technology in manufacturing.[1] Therefore, in recent years, several suppliers of automation tools have begun to adopt MRR[2–4] to develop flexible manufacturing solutions. The term modular refers to robots that are constructed using a limited number of interchangeable standardized modules, which can be assembled in different kinematic configurations (KCs). MRR can be disassembled and rearranged in different configurations rather than being replaced when tasks or changes to the workspace require a new robotic configuration. Although, the advantage of an MRR is apparent, non-modular robots are typically used in industry because of the lack of personnel who can optimize configurations based on task requirements as well as perform complex reconfiguration of existing MRR setups. Hence, the goal of this paper is to develop a novel configuration optimization method for solving the IK solutions for multi degree-of-freedom MRR.

* Corresponding author. E-mail: shwon@engmail.uwaterloo.ca

Typically, a set of modules of an MRR consists of joints, links, and end-effectors. Joints house the actuators that provide the degrees of freedom (DOF) of each robot. Links of varying lengths connect the joints to each other. The end-effector module consists of the tools required to interact with the robot environment. A wide range of MRR designs with distinct architectures and applications are presented in the literature.[5–11] The authors in ref. [5] describe a self-reorganizing robot that consists of small cells with different functions. Cells find each other and assemble autonomously. Manually configurable robots that consist of revolute (both pivot and rotate) joints and links are described in refs. [6] and [7]. The authors use quick coupling connectors to connect two modules which make it possible to reconfigure robots easily and fast. In ref. [8], an MRR is assembled from a module inventory consisting of links, revolute joints, and prismatic joints for a rapid deployment. An MRR for experimental studies in grasping, manipulation, and force control is presented in ref. [9]. The Toshiba Modular Manipulator System that can be assembled with a maximum 3-DOF is presented in ref. [10]. The authors in ref. [11] showed how different configurations can be achieved using revolute joints and various size links. In the existing literature, most MRRs use revolute joints with various size links.[6,7,9–11]

Consider a set of modules consisting of one kind of joint capable of generating rotational and pivotal movements, and three types of links, including straight, L shape, and U shape. Then, the number of configurations is given as follows:[12]

$$x = 2\sum_{k=0}^{n-1} 4^k . \frac{(n-1)!}{k!\,(n-1-k)!}, \tag{1}$$

where $x$ and $n$ are the number of configurations and the number of joints, respectively. According to Eq. (1), up to 6250 configurations can be created for six serial joint modules. For a specific spatial task, only a subset of all the configurations will be capable of reaching all the given set of task points. Moreover, from the set of the robot configurations that can actually reach all task points, some will perform better than others in terms of satisfying a set of performance criteria such as power efficiency, payload carrying capacity, etc. Hence, for evaluating the suitability of each KC, numerous constraints and performance criteria for all the desired task points must be examined. The large number of possible configurations and the complexities of the KC space necessitate employing highly efficient, intelligent, and automated search methods for determining the most suitable KC to perform a given spatial task. Such methods, referred to herein as task-based configuration optimization (TBCO) should search, synthesize, and determine an optimum KC which can be assembled from the available MRR modules. The optimum solution may be found by searching all possible KC. However, since all link lengths are treated as continuous variable, it is unrealistic to determine the optimal solution by enumerative search. Even if the link length is discretized to a finite set of possible values, exploring all possible KC according to Eq. (1) can still be computationally expensive.

Distinct approaches to solve the TBCO problem are proposed in the literature. Chen in refs. [13] and [14] introduced a representation for MRRs where links of a modular robot are considered as squared prisms or cubic boxes with ports on each side. These ports could be used to connect two links to each other through a joint. The joints are considered as connectors which can attach different ports of two neighboring links. A kinematic graph is used to express the configuration of a robot. Based on this graph, an assembly incident matrix (AIM) was extracted. In this works, AIM is translated to a string and used in a GA, with reachability and manipulability as the optimization criteria. In refs. [15] and [16], the same approach is expanded to make a modified AIM which includes the port vectors. The mutation and crossover operators are also modified to be directly applied to the AIMs. This study considers reachability, joint limits, manipulability, mechanical constructability, and the minimum DOF in the optimization objective functions. In refs. [17] and [18], the links and joints of the manipulator are considered to be modular with specific shapes and dimensions.

A reconfigurable robot which uses passive versatile connecting modules is introduced in refs. [19]–[21]. The proposed robot can achieve different configurations by changing the angles of the pseudo joint (passive connecting modules), which makes reconfiguration easy and quick to realize. Ref. [20] shows that an adaptive neuro-fuzzy inference system can be used for fast evaluation of reconfigurable robots. In ref. [21], GA is used to find the optimum angles for the pseudo joints of a reconfigurable robot.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

In another category of approaches, the manipulators are modeled by the DH parameters. Analytic and numerical mappings from the task space to the configuration space for a 2-DOF planar robot was discussed in ref. [22]. In ref. [23], a method is expanded to include obstacle avoidance and was applied to a more general 3-DOF robot. In ref. [24], TBCO is solved with the addition of new constraints and optimization criteria. A multi-population GA is used as the optimization engine. IK problem is solved numerically and provisions are made to prevent the robot from reaching two consecutive task points by passing through singularities. In ref. [25], the same method is used to design a manipulator for polishing ceramic tiles. In ref. [26], the authors upgrade the method by implementing agent-based software. They apply their method to design a fault tolerant redundant manipulator for satellite docking aboard the space shuttle. In ref. [27], a method is presented to identify a relatively optimal configuration for a fault tolerant redundant manipulator. In ref. [5], an algorithm is proposed which searches for KCs by extending a link from the base to the task point and then adding joint modules when the connection between the two points is not possible. A similar approach is used with a GA to find a sub-optimized KC in ref. [28]. In ref. [29], a GA is used to find a sub-optimal KC based on various objectives. A two-level GA approach is used in refs. [30] and [31]. In ref. [30], the upper level GA searches for the most suitable configuration and the lower GA solves the IK problem. In ref. [31], a hybrid genetic-simulated annealing algorithm is proposed; GA is used to find the optimal design and simulated annealing as a local search algorithm. However, the research in refs. [29]–[31] represents the length of a module with limited binary numbers (one to four bits) rather than treating it as a continuous variable. Therefore, these methods could be beneficial for standardizing the link lengths, but the solutions of KCs will most likely not provide a global or local optimal solution.

This paper presents a methodology based on MA to solve the TBCO problem. In order to improve the computational speed and reachability error of the TBCO solver of GA, we propose a MA (the hybrid of GA with local search method), as the core optimization algorithm of the TBCO. The proposed local search method iteratively searches for the dimensions of a manipulator that is capable of performing a certain task. The local search method also uses a proposed Jacobian matrix notation called the task embedded Jacobian to arrive at the optimal result(s). A kinematic-structure-aware elitism and restarting scheme are developed to further increase the resilience of the algorithm when searching for the local optimums. The developed operators significantly improve the chances of the algorithm to find the global optimum of the problem. A priority-based selection operator and local search with adaptive frequency are utilized to further increase the computational efficiency and speed of the algorithm.

Performance of the algorithm is validated by solving the TBCO problem in several distinct test cases. The rest of this paper is organized as follows: In Section 2, the preliminaries and necessary definitions are presented. Section 3 provides a generic overview of MRR kinematics. In Section 4, the mathematical representation of the TBCO is described. In Section 5, MAs are introduced. Section 6 provides a framework of the proposed MA for solving the TBCO problem. A local search method for finding the link dimensions of manipulators enabling them to perform a set of predefined tasks is proposed in Section 7. The proposed TBCO algorithm is explained in details in Section 8. Simulations of the TBCO are presented in Section 9. Section 10 summarizes the conclusions of this work.

## 2. Definitions

In this section, we define the following standard notations and variables that are used throughout the paper:

1. **Task point $T_{\text{des}}$:** A Task point is a desired position and orientation in the Cartesian space that the robot should reach. Task points are defined with respect to a reference frame which is usually positioned at the base of the manipulator. A task point can be a location on the manipulated object, a point in space that the manipulator should reach, or a point that the robot should pass through in order to avoid an obstacle. In this work, the position and orientation of the $i$th task point with respect to the reference is represented by the homogenous transformation, $T^i_{\text{des}}$, as follows:

$$T^i_{\text{des}} = \begin{bmatrix} & R^i_{\text{des}} & & P^i_{\text{des}} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

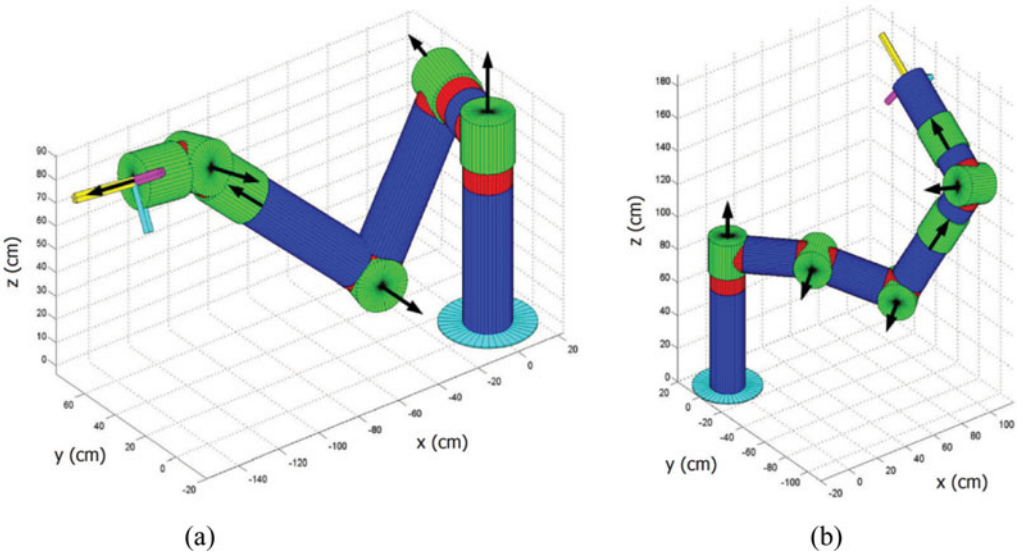



Fig. 1. Two members of the KC space $M_6$: (a) 6 DOF PUMA type configuration and (b) sample 6 DOF configuration.

where $R^i_{\text{des}}$ is a $3 \times 3$ rotation matrix representing the desired orientation, and $P^i_{\text{des}}$ is a $3 \times 1$ translation matrix representing the desired position of the $i$th task point in the Cartesian space.

2. **End-effector's position and orientation $\mathbf{T}_{ee}$:** The position and orientation of the end-effector of an $n$-DOF manipulator is a function of its joint variables. The position and orientation of the end-effector with respect to the reference frame is represented by a homogenous transformation ($T_{ee}$) as a function of the joint variables $[q_1, q_2, \ldots, q_n]$. For a $m_n$ manipulator, the joint variables are all angles and represented by $\Theta = [\theta_1, \theta_2, \ldots, \theta_n]$, as shown in

$$T^{m_n}_{ee} = \begin{bmatrix} R_{ee}(\Theta) & P_{ee}(\Theta) \\ 0\ \ 0\ \ 0 & 1 \end{bmatrix}, \tag{3}$$

where $R_{ee}$ and $P_{ee}$ are the rotation and translation matrices representing the orientation and position of the end-effector in the Cartesian space, respectively.

3. **Task $\mathbb{T}_t$:** A task is a set of the task points which should be reached by the manipulator to accomplish an objective. For example, all the task points that a robot should reach in order to assemble a car seat form a task. In this paper, a Task with $t$ task points is represented by $\mathbb{T}_t$, or simply $\mathbb{T}$, and is defined as

$$\mathbb{T}_t = \left\{\mathrm{T}^1_{des}, \mathrm{T}^2_{des}, \ldots, \mathrm{T}^t_{des}\right\}, \tag{4}$$

where $\mathrm{T}^1_{des}$ represents the first task point of the task $\mathbb{T}$.

4. **KC**: It refers to the chained arrangement of the joints in a robot. KC is defined by the number of DOFs, the joint types, dimension of the links, and the angle between the rotation axes of two consecutive joints. KCs are categorized as: Serial, parallel, and hybrid. For an MRR, a configuration can also be described as the sequence of the assembled modules and their relative orientation. An $n$-DOF KC assembled only from the standard modular joints is represented by $m_n$. Figure 1 illustrates two distinct $m_6$ KCs and the corresponding axes of the joints.
5. **$n$-DOF KC Space**: The space including all the distinct KCs, which can be assembled by using $n$ joint modules of the standard modular joints, is called an $n$-DOF KC space, i.e., each member of such a space corresponds to an $m_n$. The degree of freedom has a direct impact on the size, dimension, and characteristics of this space. For MRRs with a set of fixed links of different sizes, this space is a discrete space. If the length of the links is considered continuous, this space is a hybrid continuous discrete space (continuous with respect to the link lengths and discrete with respect to the joint types and relative orientation of the joints). In this paper, an $n$-DOF KC space

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

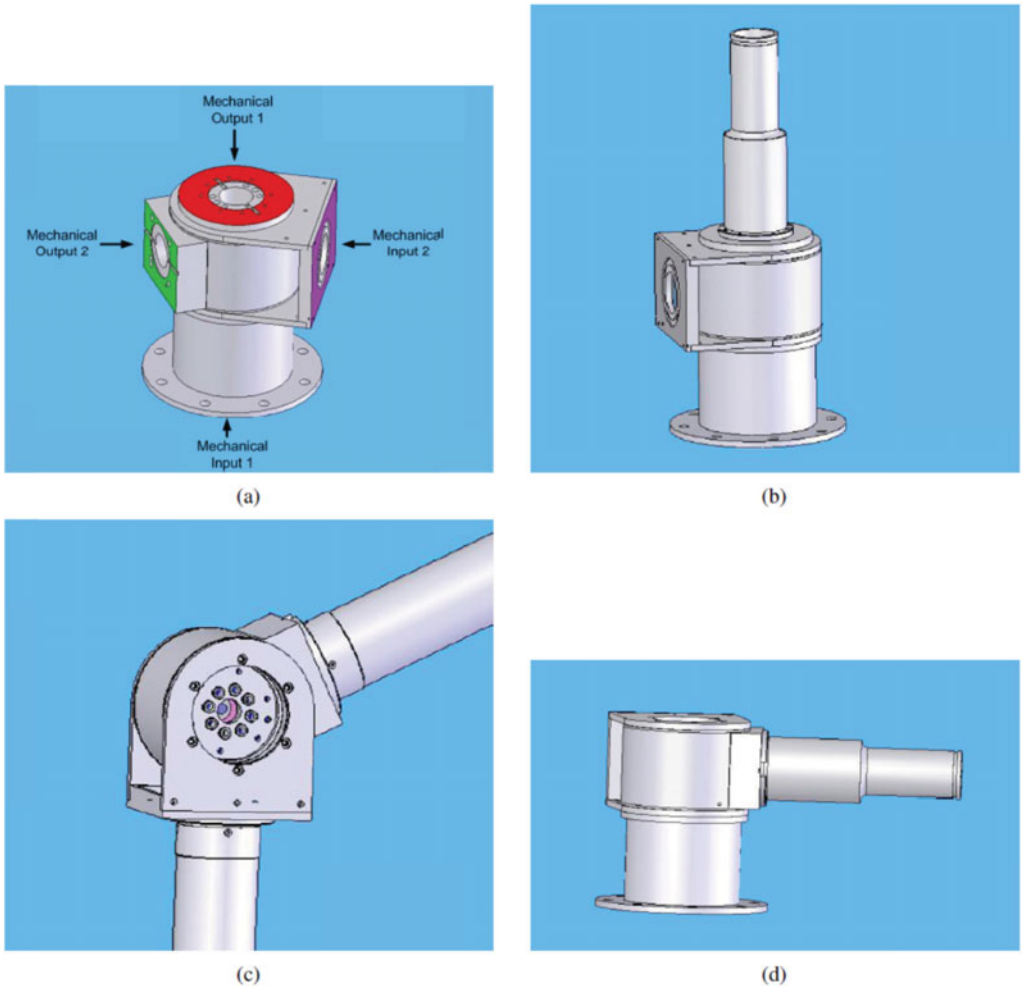



Fig. 2. MRR mechanical design principles: (a) mechanical input and output ports on the joint modules, (b) joint in the rotational configuration (c) joint in the pivotal configuration, and (d) joint in the perpendicular rotational configuration.

is represented by $\mathbb{M}_n$ as:

$$\mathbb{M}_n = \cup_{I=12,\ldots} m_n^i. \tag{5}$$

6. **Manipulator Workspace**: The workspace of a manipulator is the region in the Cartesian space that can be reached by the manipulator end-effector. The dexterous primary workspace is the region that the manipulator can reach with any orientation of the end-effector. The secondary workspace is the volume, reachable by the manipulator from a limited number of orientations.[32] In this paper, the secondary workspace is simply called the manipulator workspace, $W_T$ and is defined as

$$W_T(\boldsymbol{m}_n) = \left\{ \mathbf{T}_{ee}^{\mathbf{m}_n}(\theta) : \theta_{min} \leq \theta \leq \theta_{min} \right\}. \tag{6}$$

Here, the workspace of a manipulator includes all the positions and orientations of the end-effector that can be reached with joint angles within the feasible range.

## 3. Generic Serial Modular and Reconfigurable Robot with $n$-DOF

An MRR which consists of three distinct classes of joint modules, namely rotational, pivotal, and perpendicular-rotational has been developed as shown in Fig. 2. Links are connected to the joint modules by their different mechanical connections, providing distinct types of DOF. Figure 3

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table I. The permissible values for the variables of the manipulator representation matrix.

| Name | Permissible value | Description |
|---|---|---|
| 1 $\phi_i$ | 0 | Radian |
| | $\frac{\pi}{2}$ | |
| 2 $m_i$ | P | Pivotal joint |
| | R | Rotational joint |
| | PR | Perpendicular-rotational joint |
| 3 $l_i$ | $l_i \varepsilon [l_{min}, l_{max}]$ | Length of the link (cm) |

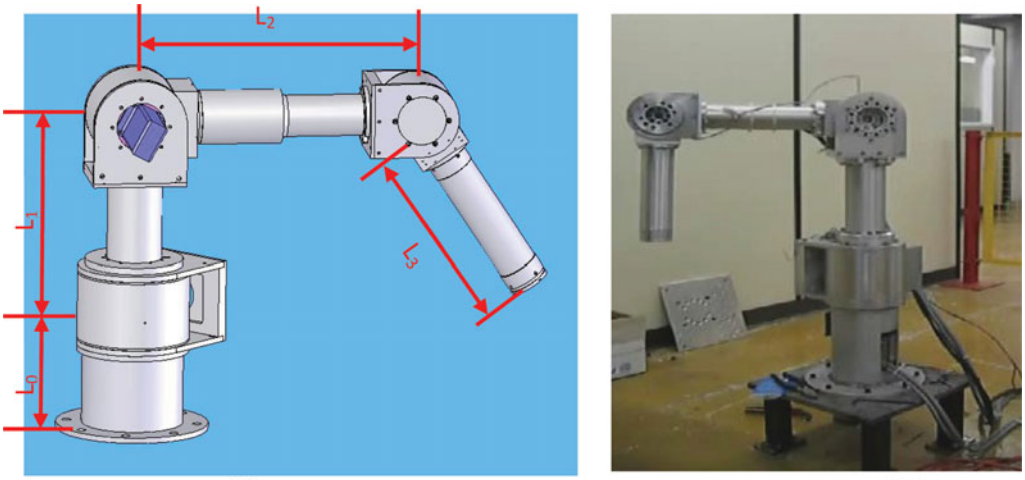



Fig. 3. MRR assembled in a 3 DOF PUMA configuration.

illustrates an example of the MRR when it is assembled in a 3 DOF PUMA arm. A wide range of standard industrial manipulators configurations (PUMA, Scara, Scorbot, etc.) can be represented by a KC consisting only of these standard modular joints. An $n$-DOF manipulator which only includes standard modular joints is represented by $m_n$. A manipulator $m_n$ can be fully described by the joint types, the relative assembly orientation of two consecutive joints, and the length of the links. These characteristics can be conveniently expressed in $(n + 1) \times 3$ array as follows:

$$m_n = \begin{bmatrix} 0 & 0 & l_0 \\ \phi_1 & m_1 & l_1 \\ \phi_2 & m_2 & l_2 \\ \vdots & \vdots & \vdots \\ \phi_n & m_n & l_n \end{bmatrix}. \tag{7}$$

The elements of the first column, $\phi_i$, represent the orientation of joint $i$ relative to joint $i - 1$. This orientation is determined by the angle between the $X_{out}$ axis of joint $i - 1$ and $X_{in}$ axis of joint $i$. In the second column, the joint types $m_i$ are stored. The third column represents the length of the links $l_i$. The link lengths are treated as continuous variables such that $l_i \varepsilon [l_{min}, l_{max}]$. The permissible values for each variable are shown in Table I.

The first row of the matrix in Eq. (7) represents the first link of the robot. This link is perpendicular to the ground and can connect the first joint to the base of the robot. For instance, the matrix representation of the 3 DOF PUMA configuration in Fig. 3 is given as follows:

$$m_3^{sample} = \begin{bmatrix} 0 & 0 & L_0 \\ 0 & R & L_1 \\ 0 & P & L_2 \\ 0 & P & L_3 \end{bmatrix}. \tag{8}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

## 4. Analysis and Mathematical Representation of TBCO

The goal of the TBCO is to find the optimum KC to perform a certain task $\mathbb{T}_t$, consisting of $t$ task points in the Cartesian space. Therefore, the ability to perform $\mathbb{T}_t$ is a necessary condition for a manipulator to be the solution of the TBCO. More specifically, TBCO determines an $m_n$, which is capable of performing the desired task $\mathbb{T}_t$, while simultaneously optimizing a performance measure $f_{obj}$. Therefore, TBCO can be expressed using the following alternative formulation:

$$\left\{ \begin{array}{c} \arg\min\limits_{m_n} f_{obj}(m_n, \mathbb{T}_t) \\ \text{subject to : } \left\{ \exists \theta_i^s \text{ for } m_n | T_{ee}^{\mathbf{m}_n}\left(\theta_i^s\right) = T_{des}^1 \right\}, \text{ for } i = 1, \ldots, t \end{array} \right\}. \tag{9}$$

The TBCO problem can be considered as a mapping from the Cartesian task space, a space that consists of positions and orientations of task points, to$\mathbb{M}_n$. Depending on $\mathbb{T}$ and $n$, a mapping can exist which maps every task point in the Cartesian space to KCs capable of reaching the task point (Fig. 4(a)). In general, this mapping can have more than one solution for any task point in the Cartesian space, i.e., more than one KC exists that is capable of reaching a certain task. Three possible cases can occur if the mapping is applied to the set of task points:

- **No common point in the configuration space exists**: In this case, no single KC exists that can reach all of the task points. Figure 4(b) describes this case. To solve the TBCO, DOFs should be increased by one. Since the configurations with higher DOFs are capable of performing more complex tasks, the chance of finding a KC for such a task increases. With the increased DOF, the search for KC using Eq. (9) is repeated.
- **One common point in the configuration space exists**: This case is shown in Fig. 4(c). In this case, since only one KC capable of reaching all the task points exists, the single KC capable of executing the task is the solution of the TBCO problem. It is assumed that the given configuration is the most efficient solution in comparison to other solutions involving higher DOF.[33]
- **More than one common point in the configuration space exist***:* In this case, among all the configurations that can achieve the task the one with the highest efficiency, according to the set of considered optimization criteria, is the solution of TBCO. Figure 4(d) depicts this case.

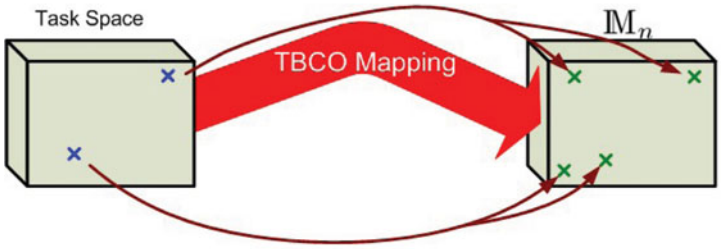
By obtaining a mapping from the Cartesian space to the KC space, the most suitable KC for a task can be found. This mapping is highly nonlinear with a large number of mixed discrete and continuous variables. Furthermore, the mapping varies as the number of DOF $n$ changes. Therefore, the general approach to solve TBCO is not to find the mapping itself, but rather conduct an extensive search of $M_n$ for KCs capable of performing the task. Among these configurations, the one that can minimize the considered optimization criteria in Eq. (9) is selected as the solution.

## 5. Memetic Algorithms

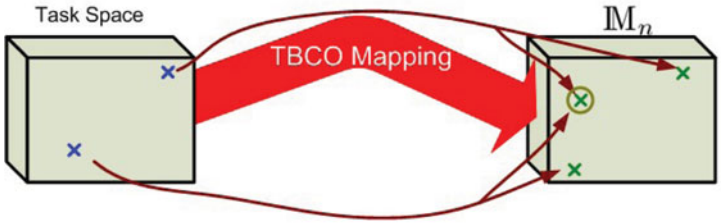
GAs work on the principle of natural evolution. Specifically, the individuals which are the coded variables of the problem evolve to new individuals with better fitness values due to application of GA operations which are crossover, mutation, and selection. When GAs are augmented with local search algorithms create a new class of metaheuristics algorithm called MAs.[34] The word, *Memetic* comes from the term, *meme*, which was coined by Dawkins[35] to denote an analogous to *gene* in the context of cultural evolution.[36] The central philosophy of MAs is individual improvement as well as population cooperation and competition as they are present in many social and cultural systems. In the literature, MAs can be found under a large variety of names such as hybrid GAs, genetic local searchers, Lamarckian GAs to name a few. In numerous studies, it has been suggested that the efficiency of a search in pure GAs can be significantly improved when they are combined with other techniques.[37–39] The no-free-lunch (NFL) theorem,[37] states that a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge it incorporates. Consequently, an MA which incorporates the information of the landscape of the proximity of each individual through a local search performs better than a pure GA without a local search. In essence, the success of MAs can be seen as a tradeoff between the exploration abilities of the GA and the exploitation abilities of the local search algorithms.[40] Figures 5 and 6 show pseudo codes of pure GA and an MA in the simplest form, respectively.[41] One main difference between the two algorithms lies in the

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

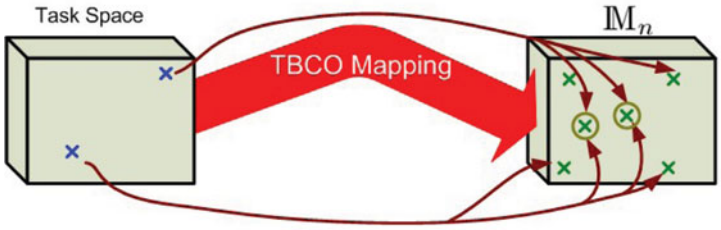

(a) Mapping for one task point




(b) No common solutions in the configuration space




(c) One common solution in the configuration space




(d) Numerous common solutions in the configuration space

Fig. 4. Mapping from the Cartesian task space to the robot configuration space. (a) Mapping for one task point. (b) No common solutions in the configuration space. (c) One common solution in the configuration space. (d) Numerous common solutions in the configuration space.

fact that MAs have a *local search* stage. It accepts an individual as the input and produces a new individual in the neighborhood of the original solution provided that the new solution has a better fitness value. In the literature, a wide range of distinct local search algorithms have been reported. The application requirements and the problem characteristics affect and sometimes dictate the algorithms for the local search. Another difference between the GA and MA is the *restart population* element in the later. Consider a case in which the population is not able to produce new individuals through the genetic operators. This might occur if the individuals of the current population are very similar to each other. In such case, the restart stage introduces new individuals into the population. The method to detect such a situation and the process of introducing new individuals into the population are application-dependent.[36]

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

```
Randomly generate a population   // Initialization
while termination criteria is not reached do
   // Genetic operators
     Calculate fitness value
     Selection
     Crossover
     Mutation
end
```

Fig. 5. Pseudo-code of a genetic algorithm.

```
Randomly generate a population   // Initialization
while termination criteria is not reached do
   // Genetic operators
     Calculate fitness value
     Selection
     Crossover
     Mutation
     // Local search operator
     if population converged then Restart population
     Local search
end
```

Fig. 6. Pseudo-code of a memetic algorithm.

MAs have proven to be effective tools in solving some optimization problems. Yet, the process of designing efficient MAs currently remains fairly *ad hoc* and application-dependent.[40] Computing the fitness of a solution given the fitness of another solution that is close to it is significantly less computationally expensive in comparison to computing the fitness of a solution from scratch. This class of problems is considered suitable for exploration via MAs. The measure of closeness between two individuals can informally be defined as the number of common genetic materials they share. Since the calculation of the fitness function is the most time consuming step in the generation of a GA; MAs are more suitable when the fitness function is decomposable. This holds true if the improvement of the individuals is performed gradually in small steps.[42]

## 6. Framework of the Proposed MA for Solving TBCO

As mentioned, the TBCO can be formulated as a minimization problem in which all the elements of the matrix representation of a manipulator $m_n$, should be determined. The solution $m_n$ should minimize an objective function and simultaneously satisfy a nonlinear constraint. Therefore, each individual of MA which represents an $m_n$ manipulator can be expressed with an MRR matrix representation as shown in Eq. (7).

The MRR matrix representation shown in Eq. (7) can be decomposed into two characteristically distinct, but interconnected, segments to further investigate the most suitable elements of the matrix representation to undergo a local search. Consequently, the elements which should be identified through the genetic operators can be determined. The matrix in Eq. (7) can be expressed as follows:

$$m_n = (\Sigma | \Delta)\,, \tag{10}$$

where $\Sigma$ is the first two columns of Eq. (7) which is the kinematic structure matrix ($\phi_i$ and $m_i$) and $\Delta$ is the third column of Eq. (7) which represents the length of the links ($l_i$). In the TBCO, the primary

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

goal is to find both kinematic structure and link dimension matrices ($\Sigma$ and $\Delta$). Due to the distinctive characteristics of $\Sigma$ and $\Delta$, the TBCO can be decomposed into a search in two interconnected spaces with distinct characteristics as follows:

1. $\Sigma$ **search:** If one element of kinematic structure matrix $\Sigma$ changes, the resulting manipulator becomes a fundamentally different kinematic characteristics. The reason can be explained by using the concept of manipulator workspace. When the orientation of two consecutive joints or the type of the joints is altered, the workspace of the manipulator might change its shape. Since the kinematic parameters of the manipulator change when $\Sigma$ changes, the IK of the manipulator as well as the optimization criteria and constraints should be computed from scratch.
2. $\Delta$ **search:** The link dimension matrix $\Delta$ are bounded continuous variables within the range, $l_i \in [l_{\min}, l_{\max}]$. When $\Delta$ of a manipulator is changed, the shape of the workspace is preserved and only the volume is altered, i.e., if $\Delta$ is slightly altered, the change in the kinematic parameters of the manipulator is small. As a result, the change in the solutions of the IK problem is small. Therefore, the IK solution of the pre-change manipulator can be used as an initial guess for the IK of the post-change manipulator enabling the solver to converge faster.

Computing the fitness values including optimization criteria and constraints requires solutions to the IK problem. Small changes in $\Delta$ result in small changes in the IK solutions. Therefore, when $\Delta$ changes in a robot, the IK problem of the new manipulator is solved relatively fast by utilizing the IK solutions of the original manipulator as an initial guess. That is, solving the IK problem of a manipulator given the IK solutions of another manipulator which has a similar $\Sigma$ but different $\Delta$ is less computationally expensive than computing the IK solutions of the manipulator from scratch. This implies that the fitness function in the TBCO can be considered *decomposable*. The decomposable fitness function entails that the efficiency of GAs can be improved in solving the TBCO when they are combined with local search algorithms.

In summary, a local search can provide an effective means for conducting the search in the continuous part of the TBCO, which is $\Delta$. Therefore, in the proposed TBCO the principal burden of the search in the continuous space of $\Delta$ is carried out by the local search operators. The intention of the $\Delta$ search is to enhance the reachability error of the manipulator enabling it to satisfy the constraints. Primarily, the $\Sigma$ search is conducted by the genetic operators. In the next section, $\Delta$ search is mathematically formulated and a method for solving it is proposed.

## 7. Local Search Operator

In MAs, the local search gradually improves the fitness value of an individual. In the TBCO, the local search modifies the dimension of the links of an individual (manipulator) to decrease its reachability error. Since the local search operates on the $\Delta$ part of the individuals, it is conducted in the continuous search space. The $\Delta$ search is mathematically expressed as follows:

$$\left\{ \begin{array}{c} \Delta_s = \arg\min_{\Delta} f_{rch,\mathbb{T}}(\boldsymbol{m}_n, \mathbb{T}) \\ \boldsymbol{m}_n = (\Sigma|\Delta) \end{array} \right\}, \tag{11}$$

where $\Delta_s$ represents the solution of the $\Delta$ search problem, $\Delta$ is the variable of the problem, and $\Sigma$ and $\mathbb{T}$ are the parameters. In this section, a method for solving the $\Delta$ search problem is proposed.

### *7.1. $\Delta$ local search*

The reason for applying this search to a manipulator is to determine $\Delta$ when $\Sigma$ is fixed such that the resulting manipulator shows enhanced capability in satisfying the reachability constraints for the desired task $\mathbb{T}$. The reachability error for the $i$th task point is a function of the kinematic structure $\Sigma$, the link dimensions $\Delta$, and the joint angles of the manipulator $\Theta_i$ when the manipulator reaches for the task point. The search for link dimensions is always entangled with the search for the set of joint angles required to reach the task points. Therefore, with the assumption that $\Sigma$ is fixed, the variable

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

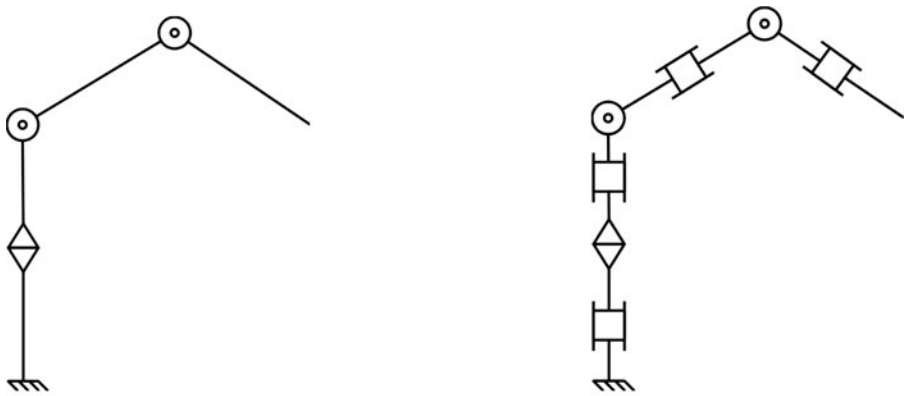



Fig. 7. Sample $m_3$ manipulator and the corresponding 7 DOF revolute-prismatic joint manipulator, where the links are considered as prismatic joints: (a) original manipulator, and (b) resulting 7 DOF manipulator with 3 revolute joints and 4 prismatic joints.

vector of the $\Delta$ search, represented by $q_{\mathbb{T}}$, is defined as follows:

$$q_{\mathbb{T}} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_t \\ \Delta \end{bmatrix}. \tag{12}$$

To solve for $q_{\mathbb{T}}$, each link of the manipulator is modeled as a prismatic joint. Next, the prismatic and rotational joint variables for the new manipulator, which allows it to reach all the task points, are derived. With this assumption, an $m_n$manipulator consisting of $n$ revolute joints is converted to a $(2n + 1)$ DOF manipulator with $n$ revolute and $(n + 1)$ prismatic joints. In Fig. 7, an $\boldsymbol{m}_3$ manipulator and the corresponding post conversion 7-DOF manipulator with 3 revolute and 4 prismatic joints are shown. Applying this transformation effectively converts the problem of finding the link dimensions and the corresponding $t$ sets of joint angles in an $n$-DOF manipulator to the IK problem of a redundant $(2n + 1)$

DOF manipulator for $t$ tasks. Although a wide range of approaches for solving the IK of redundant manipulators exists,[43–49] none involves solving the IK problem for numerous task points simultaneously. The majority of the existing methods rely on solving the first order differential kinematic equations of the manipulator. In the next section, two of the most common approaches to solve this problem when one task point is considered, are reviewed.

### *7.2. Solving the inverse kinematics of redundant manipulators*

Obtaining the IK solutions for robot manipulator is necessary for motion planning. As the number of the links increases (three or more), obtaining an analytical IK solution becomes difficult, and in many cases a closed-form solution does not exist. Therefore, utilizing numerical method to address the IK problem will be required in this case. If a task is represented by the desired position of the end-effector in the Cartesian coordinates and a minimal representation of the orientation (such as the Euler angles) as follows:

$$x_{des} = \begin{bmatrix} P_{x,des} \\ P_{y,des} \\ P_{z,des} \\ \emptyset_{x,des} \\ \emptyset_{y,des} \\ \emptyset_{z,des} \end{bmatrix}. \tag{13}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

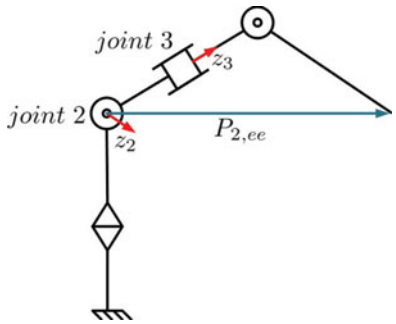



Fig. 8. $z_2$, $z_3$, and $P_{2,ee}$ vectors illustrated on a 4 DOF manipulator with revolute and prismatic joints for the second and third joint.

The mathematical formulation of the forward kinematics (FK) is written as follows:

$$x = K(q), \tag{14}$$

where $q$ represents the joint variables of a manipulator with both prismatic and revolute joints. By differentiating Eq. (14) with respect to time, the first order differential kinematics equations are obtained and expressed as

$$\dot{x} = J(q)\dot{q}, \tag{15}$$

where $\dot{x}$ is the task space velocity vector, $\dot{q}$ is the joint-space velocity vector, and $J(q) = \partial K/\partial q$ is the $6 \times n$ Jacobian matrix of the manipulator, where $n$ is the number of the joints of the manipulator. For serial manipulators, the $i$th and $j$th columns of $J(q)$ corresponding to a revolute and a prismatic joint, respectively, and are calculated as follows:

$$J(q) = \begin{bmatrix} \cdots & z_i \times P_{i,ee} & \cdots & z_j & \cdots \\ \cdots & z_i & \cdots & \vec{0} & \cdots \end{bmatrix}, \tag{16}$$

where $z_i$ and $z_j$ represent the axes of the $i$th and $j$th joint. $P_{i,ee}$ represents the vector which connects the coordinate frame of joint $i$ to the end-effector. In Fig. 8, $z2$ and $z3$ are the joint axes of the second and third joint, rotational and prismatic, respectively, with $P_{2,ee}$ illustrated.

If $J$ is non-singular, then Eq. (16) can be solved for $\dot{q}$ using the following equation:

$$\dot{q} = J(q)^{-1}\dot{x}. \tag{17}$$

Under the assumption that the manipulator is kinematically redundant, Eq. (15) can be solved by resorting to the pseudo-inverse $J^{\dagger}$ of the Jacobian matrix defined as follows:[44]

$$\dot{q} = J^{\dagger}(q)\,\dot{x} + \left(I - J^{\dagger}(q)\,J(q)\right)\dot{q}_0 \tag{18}$$

The pseudo-inverse, $J^{\dagger}$, is a unique matrix which satisfies the Moore-Penrose conditions.[50] The term $(I - J^{\dagger}(q)\,J(q))$ represents the orthogonal projection matrix into the null space of $J$, and $\dot{q}(0)$ is an arbitrary joint space velocity. Consequently, the second part of solution is a null space velocity. The particular solution, in which $\dot{q}(0) = 0$ produces the pseudo-inverse solution of Eq. (15) which is[43]

$$\dot{q} = J^{\dagger}(q)\dot{x}. \tag{19}$$

To solve Eq. (19), a numerical method which updates the value of $q$ at each iteration with the following rule is applied

$$q_k = q_{k-1} + J^{\dagger}(q_{k-1})\Delta x_{k-1}, \tag{20}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

where $J^{\dagger}(q_{k-1})$ is the pseudo-inverse of the Jacobian matrix at $q_{k-1}$, and $\Delta x$ is the distance of the position/orientation of the end-effector at $q_{k-1}$ from the desired task.

Another method for solving the kinematic equations for redundant manipulators, which is more resilient to singular poses, is the damped-least-square method.[46,51] In this method, instead of solving Eq. (19), the following first order differential equation is solved:

$$\dot{q} = J^T \left(J^T J + \lambda^2 I\right)^{-1} \dot{x}. \tag{21}$$

where $J^T$ is the transpose of $J$, and $I$ is the identity matrix. $\lambda$ is called the damping factor. If $\lambda$ is zero, then Eqs. (17) and (21) become identical. The update formula for solving Eq. (21) iteratively is as follows:

$$q_k = q_{k-1} + \left(J^T(q_{k-1})J(q_{k-1}) + \lambda^2 I\right)^{-1} \Delta x_{k-1}. \tag{22}$$

When the links of a manipulator are converted to prismatic joints, their lengths are transformed into prismatic joint variables. The lengths of the links of the manipulator remain constant regardless of the task point the robot is attempting to reach. Consequently, the prismatic joint variables which represent the length of the links should remain equal regardless of the task point. Therefore, in the IK solver, an equality set of constraints for the prismatic joints should be considered. Furthermore, the pseudo-inverse and damped-least-square methods are developed to solve the IK problem, when only one task point is considered. In the next section, a method for solving the multitask IK problem with an implicit implementation of the prismatic joint constraints is described.

### *7.3. Task embedded Jacobian matrix*

When the link dimensions are considered as prismatic joints, the joint variables of the IK of the resulting manipulator while reaching the *i*th task point is expressed as

$$q_i = \begin{bmatrix} \Theta_i \\ \Delta_i \end{bmatrix}, \tag{23}$$

where $\Theta_i$ and $\Delta_i$ represent the revolute and prismatic joint angles of the converted manipulator, respectively. When only the *i*th task point is considered, the pseudo-inverse solution of Eq. (23) is written as

$$\dot{q}_i = J_i^{\dagger}(q_i)\dot{x}_i, \tag{24}$$

where $J_i$ is the Jacobian of the converted manipulator at $q_i$, and is written as follows:

$$J_i(q_i) = \left[\, J_i^R(\Theta_i, \Delta_i) \quad J_i^P(\Theta_i, \Delta_i) \,\right], \tag{25}$$

where $J_i^R$ represents the Jacobian matrix of the manipulator at $q_i$ where only the revolute joints are considered. $J_i^R$ is equal to the Jacobian matrix of the original matrix of the manipulator at $q_i$, when only the $(n+1)$ prismatic joints representing the links of the manipulator are considered. To obtain a solution to the $\Delta$ search problem, Eq. (24) should be concurrently solved for all $q_i$, when $i = 1,\ldots,t$. In other words, the set of the following equations should be simultaneously solved

$$\begin{cases} \dot{q}_i = J_i^{\dagger}(q_i)\,\dot{x}_i & \text{for all } i = 1..t \\ \Delta_i = \Delta_j & \text{for all } i, j = 1..t \end{cases}, \tag{26}$$

where $\dot{q}_i$ and $\dot{x}_i$ represent the velocity vectors in the joint space and the task space, respectively. The second set of equations stipulates that the prismatic joint variables which represent the link dimensions of the manipulator should be the same for all the tasks. To solve Eq. (26), a Jacobian

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

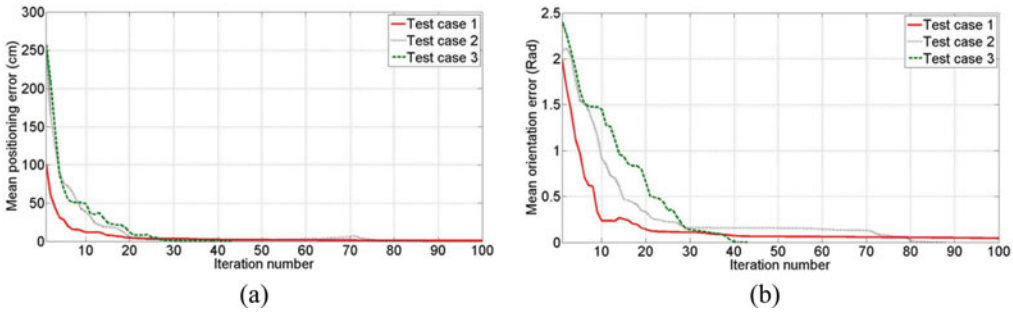



Fig. 9. Mean position and orientation reachability errors of the test cases plotted with respect to the iteration number: (a) positioning reachability error (cm) with respect to the iteration number and (b) orientation reachability error (rad) with respect to the iteration number.

matrix, called a task embedded Jacobian matrix, $J_{\mathbb{T}}$, is formed as follows:

$$J_{\mathbb{T}} = \begin{bmatrix} J_1^R(\Theta_1, \Delta) & \bar{0} & \cdots & \bar{0} & | & J_1^P(\Theta_1, \Delta) \\ \bar{0} & J_2^R(\Theta_2, \Delta) & \cdots & \bar{0} & | & J_2^P(\Theta_2, \Delta) \\ \vdots & \vdots & \ddots & \vdots & | & \vdots \\ \bar{0} & \bar{0} & \cdots & J_t^R(\Theta_t, \Delta) & | & J_t^P(\Theta_t, \Delta) \end{bmatrix}. \tag{27}$$

By using $J_{\mathbb{T}}$, Eq. (26) is written as follows

$$\dot{q}_{\mathbb{T}} = J_{\mathbb{T}}^{\dagger}(q_i)\dot{x}_{\mathbb{T}}, \tag{28}$$

where $q_{\mathbb{T}}$ is defined by Eq. (12). $\dot{x}_{\mathbb{T}}$ includes all the minimal representation task velocities of the $t$ task points and is defined as follows:

$$\dot{x}_{\mathbb{T}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_t \end{bmatrix}. \tag{29}$$

By solving Eq. (28) for $\dot{q}_{\mathbb{T}}$, the link dimensions $\Delta$ and the joint angles of the manipulator $\Theta_i$ when the $i$th task is reached, are computed concurrently. Although, the task embedded Jacobian matrix is explained by applying the pseudo-inverse method, to solve Eq. (28), any of the existing methods in the literature can also be utilized. In this paper, the damped-least-square method is chosen to solve Eq. (28) due to its robustness to matrix singularities. $\Delta$, which is found through the iterative task embedded Jacobian matrix method, is the closest solution to the initial guess with which the algorithm is initialized. In more specific terms, depending on $\mathbb{T}$, $\Delta$ search might not generate a unique solution. In such a case, the proposed $\Delta$ search algorithm converges to a solution that is closest to the initial guess.

### *7.4. Applying the task embedded Jacobian method for solving the $\Delta$ search*

To validate the proposed approach in Section 7.3, the dimensions of a 6 DOF PUMA type manipulator are determined by using the proposed $\Delta$ search. Three distinct tasks, each consists of 50 task points, are randomly generated for a manipulator with $\Sigma$ of a PUMA and a random $\Delta$. The task embedded Jacobian method is selected to determine the dimensions of a manipulator capable of performing the defined tasks.

The positioning and orienting reachability errors are plotted with respect to the iteration number in Fig. 9(a) and (b), respectively. All the test cases show that the positioning and orientation errors converge close to zero in the first 40 iterations. To converge to the desired accuracy of 0.1 cm and 0.05 radians, further iteration may be needed.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

$R_{pop}$ = Random_Population_Generation
$F_{fv}$ = Fitness_Calculation ($R_{pop}$)
**while** *termination criteria is not reached* **do**
  $E_{pop}$ = Elitism ($R_{pop}, F_{fv}$)
  $R_{pop}$ = Restart_Population ($R_{pop}$)
  $P_{pop}$ = Selection ($R_{pop}, F_{fv}$)
  $C_{pop}$ = Crossover ($P_{pop}$)
  $C_{pop}$ = Mutation ($C_{pop}$)
  $R_{pop} = C_{pop} \bigcup E_{pop}$
  $N_{pop}$ = Selection_For_Local_Search ($R_{pop}$)
  $R_{pop} = R_{pop} - N_{pop}$
  $N_{pop}$ = Local_Search ($N_{pop}$)
  $R_{pop} = R_{pop} \bigcup N_{pop}$
  $F_{fv}$ = Fitness_Calculation ($R_{pop}$)
**end**

Fig. 10. Pseudo-code of the proposed TBCO memetic algorithm.

## 8. Task-Based Memetic Algorithm Configuration Optimization

This section presents the proposed MA algorithm for TBCO. The overview of the algorithm is first provided. The specific details of the algorithm will follow. The pseudo-code of the proposed algorithm is shown in Fig. 10.

The algorithm starts by randomly initializing the population, $R_{pop}$, in Line 1. In Line 2, the high priority fitness values of the individuals which consists of the constraints $F_{cons}$, are computed. The low priority fitness values which are the optimization criteria $F_{obj}$, are calculated in the selection operator as needed. The *while* loop of Line 3 causes the algorithm to iterate between Lines 4 and 14 when the termination criteria is not satisfied. In Line 4, a kinematic structure-aware elitism method selects a fraction of the fittest individuals to form the elite population $E_{pop}$. The elite population is added to the next generation of the population without any change. In Line 5, the Restart Population subroutine computes a measure of the occurrence of each individual. Then, the individuals with excessive occurrence measures are replaced with fresh randomly generated individuals in order to preserve the diversity of the population. In Line 6, the parent pool, $R_{pop}$, is formed through the selection operator. A new generation is created in Lines 7 and 8 from the parent pool by using the crossover and mutation operators. The elite population is then added to the new population in Line 9. In Lines 10–13, a part of the population represented by $N_{pop}$ is selected to undergo the numerical improvement stage. The individuals, after being improved by the local search ($\Delta$ search) are added to the original population $R_{pop}$. In Line 14, the fitness value of the new population is computed. Each stage of the pseudo-code is explained in more details in the following section. Table II lists the names and descriptions of the variables used. Each individual in the MA population represents a manipulator coded into the matrix representation.

### *8.1. Random population generation*

In this stage, a population of $\boldsymbol{m}_n$manipulators is created randomly. This population, represented by $R_{pop}$, acts as the initial population of the MA. The number of individuals in $R_{pop}$, which is the population size $r$, for search in the space of $n$-DOF manipulators is selected using the following equation:

$$r = \max\{r_{min}, \kappa r \text{ size}(\mathbb{M}_n)\}, \tag{30}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table II. Description of the variables of the proposed MA.

| Variables | Description |
|---|---|
| $R_{pop}$ | Population of manipulators |
| $r$ | Size of population $R_{pop}$ |
| $E_{pop}$ | Population of elites |
| $e$ | Number of elites |
| $P_{pop}$ | Population of parents |
| $p$ | Size of the parent pool |
| $N_{pop}$ | Population of the selected individuals to undergo local search |
| $C_{pop}$ | Population of children (offspring) |
| $g_{max}$ | Maximum number of iterations after satisfying the constraints |
| $c_i$ | Number of individuals with the same as the $i$th individual |
| $\Lambda_i$ | Probability of the $i$th individual to undergo local search |
| $P_m$ | Mutation probability |
| $P_e$ | Crossover probability |
| $f^i_{cons}$ | Optimization constraint of the $i$th individual |
| $f^i_{obj}$ | Optimization criteria of the $i$th individual |

where the population size is set at a fraction $\kappa r < 1$ of the total number of KCs in $\mathbb{M}_n$, stipulating that the population size should be at least, $r_{min}$. The initial population is formed such that it has the following characteristics: (1) Each individual represents a feasible $\boldsymbol{m}_n$ manipulator. The relative orientation of the joints and the link dimensions are within the permissible ranges, and the joints are selected from the standard modular joint set. (2) The population consists of only one individual from each kinematic structure, i.e., two individuals with the same kinematic structure, $\Sigma$, cannot be found in the initial population. This feature provides the initial population with a more diverse sampling of the KC space. (3) The initial individuals are created such that all are non-redundant. For instance, in an initial population consisting of $m_4$ manipulators, no manipulator with four parallel joints axes can exist.

### *8.2. Fitness calculation*

In this stage, the fitness values of all the individuals are computed. Since, in the proposed algorithm, a priority-based selection scheme is adopted, the computation of the optimization criterion $f_{obj,\mathbb{T}}$, with lower priorities can be postponed until they are required by the selection scheme. The optimization constraints, $f_{\text{cons},\mathbb{T}}$ on the other hand, should be calculated in each iteration of the algorithm.

### *8.3. Elitism*

Elitism is the process in which the fittest individual(s) of a population are directly transferred to the next population. It has been shown that elitism can improve the efficiency of GAs significantly.[52,53] Furthermore, for optimization problems in which prior information about the fitness value does not exist, the use of an elitism scheme in MAs has been recommended.[54] In our proposed TBCO algorithm, a kinematic structure-aware elitism scheme is employed. In the elitism scheme, a population $E_{pop}$ with size $e$, consisting of the fittest individuals is formed. The individuals of $E_{pop}$ are selected such that the population has the following characteristics:

- The members of $E_{pop}$ are the fittest individuals of the population. That is, the individuals of $E_{pop}$have the lowest $f_{cons,\mathbb{T}}$ among the population. If two individuals satisfy the TBCO constraints, i.e., $f_{cons,\mathbb{T}} \leq \varepsilon_{cons}$, the one with a lower $f_{obj,\mathbb{T}}$is adopted for $E_{pop}$.$\varepsilon_{cons}$represents the permissible tolerance for the constraint violation.
- $E_{pop}$ consists of individuals with different kinematic structure $\Sigma$. The fittest $e$ individuals with distinct kinematic structure are preserved for the next population. This feature prevents individuals with the same $\Sigma$ from being transferred to the next population. Consequently, the chance of the population being dominated by a few kinematic structures decreases.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

**Input**: $R_1$ and $R_2$: Randomly selected individuals for tournament
**Output**: $R_{out}$: Winner of the tournament selection
**if** $\left(|F^1_{cons,\mathbb{T}} - F^2_{cons,\mathbb{T}}| \leq \delta_1\right)$ **then**
  **if** $F^1_{obj,\mathbb{T}} \leq F^2_{obj,\mathbb{T}}$ **then** $R_{out} = R_1$
  **else** $R_{out} = R_2$
**else if** $\left(|F^1_{cons,\mathbb{T}} - F^2_{cons,\mathbb{T}}| \leq \delta_2\right)$ **and** $rand \leq \zeta$ **then**
  **if** $F^1_{obj,\mathbb{T}} \leq F^2_{obj,\mathbb{T}}$ **then** $R_{out} = R_1$
  **else** $R_{out} = R_2$

Fig. 11. Pseudo-code of the proposed priority-based selection scheme.

### *8.4. Restart population*

In the proposed TBCO, the inverse of the number of manipulators with a similar kinematic structure $\Sigma$ is adopted as a diversity measure, i.e., for a given population, when the number of the manipulators with the same $\Sigma$ increases, the diversity of the population decreases. Therefore, the following restart scheme is devised, where $c_i$ (for $i = 1, \ldots, r$) represents the number of manipulators with similar kinematic structures to that of the $i$th individual:

1. A list of the individuals with $c_i \geq \kappa_c.r$ (where $\kappa_c \leq 1$ is a constant) is created. Therefore, the list consists of the manipulators which have repeated occurrences in terms of the kinematic structure $\Sigma$ with a total number of occurrences of $\kappa_c.r$ or more.
2. Half of the individuals on the list which are less fit are replaced by individuals created from scratch (using random population generation) and the other half which are fitter are returned to the population.

### *8.5. Selection*

In the selection phase, a group of individuals are chosen to form a parent pool. The members of the parent pool undergo crossover, mutation, and local search manipulation to create the new population. In TBCO, the optimization criteria and the constraints are mapped into two sets of fitness values with different priorities. Then, the selection can be formulated into a priority-based selection in which two individuals are compared such that the constraints have more impact than the optimization criterion. To implement a priority-based selection scheme a binary tournament selection is chosen. In the tournament selection, to select a parent, $n_{tour}$ individuals are selected randomly. From these individuals, the fittest individual is transferred to the parent pool. If the parent pool $P_{pop}$, consists of $p$ individuals, a total of $p$ tournaments should be performed for forming $P_{\text{pop}}$. An advantage of the binary tournament, in which $n_{tour} = 2$, is that it produces a larger selection variance than that of the ranking selection scheme which is a more commonly used selection method in the GA literature. Choosing $n_{tour} \geq 2$ decreases the chance of weaker individuals being selected and subsequently, decreases the diversity of the parent pool but increases the convergence speed.[55] Another advantage of using the binary tournament selection in the TBCO is that it reduces the process of selection into a simple comparison between two individuals. The winner of the tournament can be decided by comparing any characteristic or measure of the individuals involved in the tournament. This feature can be exploited for developing priority-based selection operators. In such operation, the individuals are compared according to a set of fitness values with different priorities.

Figure 11 provides the pseudo-code of the proposed priority-based selection scheme where $\delta_1$, $\delta_1$, and $\xi$ are constants such that $\delta_1 < \delta_2$. $R_1$ and $R_2$ are two individuals randomly selected from the population for the binary tournament. $R_{out}$ is the winner of the tournament which is added to the parent pool. $F^1_{cons,\mathbb{T}}$ and $F^2_{cons,\mathbb{T}}$ represent the optimization constraint computed for the task, $\mathbb{T}$, and $F^1_{obj,\mathbb{T}}$ and $F^2_{obj,\mathbb{T}}$ represent the computed optimization criteria for $R_1$ and $R_2$, respectively. According to the pseudo-code, when $R_1$ and $R_2$ are being compared, one of three distinct cases results:

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

**Input**: $P_1$ and $P_2$: Randomly selected individuals for crossover
**Output**: $C_1$ and $C_2$: The offsprings

**forall** $i = 1..n, j = 1..3$ **do**
  **if** $rand \leq 0.5$ **then**
    **switch** $j$ **do**
      **case** $j = 1$ // $X(i,j)$ is joint relative orientation
        $[C_1(i,j) \quad C_2(i,j)] = \phi$_crossover( $P_1(i,j) \quad P_2(i,j)$ )
      **case** $j = 2$ // $X(i,j)$ is joint type
        $[C_1(i,j) \quad C_2(i,j)] = m$_crossover( $P_1(i,j) \quad P_2(i,j)$ )
      **case** $j = 3$ // $X(i,j)$ is link length
        $[C_1(i,j) \quad C_2(i,j)] = l$_crossover( $P_1(i,j) \quad P_2(i,j)$ )

Fig. 12. Pseudo-code of the GeneAS-based crossover scheme.

1. If the optimization constraints of $R_1$ and $R_2$ are very close to each other within $\delta_1$, the one with the better $F_{obj,\mathbb{T}}$ is selected.
2. If the optimization constraints of $R_1$ and $R_2$ are fairly close within $\delta_2$, the one with the better $F_{obj,\mathbb{T}}$ with a certain probability $\zeta$ is selected.
3. If the difference between the optimization constraints of $R_1$ and $R_2$ is large, the one which can better satisfy the optimization constraints with a smaller $F_{cons,\mathbb{T}}$ is selected.

It can be observed that the priority-based scheme considers $f_{obj,\mathbb{T}}$, only if $R_1$ and $R_2$ have approximately the same $f_{cons,\mathbb{T}}$. Moreover, since knowledge of the value of $f_{obj,\mathbb{T}}$ is not required in the third case, its computation is postponed until necessary.

### *8.6. Crossover*

To produce two new individuals, called off springs, two parents are randomly selected from the parent pool to undergo crossover operation. The two parents are crossed with a probability of $P_c$, and otherwise, are transferred unchanged to the new population. In the TBCO, each individual consists of two segments, kinematic structure $\Sigma$ and link dimension $\Delta$, or three types of variables corresponding to the three columns of the MRR matrix representation, namely the relative orientation of the joints $\phi_i$, joint types $m_i$, and link lengths $l_i$. The method by which the crossover is applied to each of the variables, differs due to the differences in the variable type and the permissible bounds. For $\phi_i$ and $m_i$, the crossover is applied in a discrete space with two distinct permissible bounds. For $l_i$, the crossover is applied in a bounded continuous space. We adopt Gene AS framework[56] for the crossover operation. Figure 12 denotes the pseudo-code of the crossover scheme. $P_1$ and $P_2$ are the selected parents from the parent pool, whereas $C_1$ and $C_2$ are the off springs produced from crossing $P_1$ and $P_2$. The variable at the $i$th row and $j$th column of the MRR matrix representation is expressed by $X(i, j)$, for all $i = 1, \ldots, n$ and $j = 1, \ldots, 3$, where $X$ can be $P_1, P_2, C_1, C_2$. Each $X(i, j)$ has a 50% chance of undergoing crossover. The crossover is applied to each element according to the variable type of the element. The crossover operator used for the link lengths is the bounded SBX operator,[57] and $\phi$-crossover and $m$-crossover use a discrete version of the bounded SBX which creates only the permissible values of $\phi$ and $m$.

### *8.7. Mutation*

With a probability of $P_m$, the crossover results may undergo mutation operation. In the proposed TBCO, the mutation is performed as depicted in Fig. 13, where $R_{in}$ and $R_{mutated}$ represent a selected individual that undergoes mutation and the mutated individual, respectively. The pseudo-code shows that an element of individual $R_{in}$ is randomly selected. Based on the type of the selected element, the appropriate type of mutation is employed. If the selected element is a link length, $l_i$, a continuous

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

**Input**: $R_{in}$: Individual to undergo mutation
**Output**: $R_{mutated}$: Mutated individual

$i$ = Random_integer(1..$n$)
$j$ = Random_integer(1..3)

**switch** $j$ **do**
  **case** $j = 1$ // $X(i,j)$ is joint relative orientation
    $R_{mutated}(i,j) = \phi$_mutation($R_{in}(i,j)$)
  **case** $j = 2$ // $X(i,j)$ is joint type
    $R_{mutated}(i,j) = m$_mutation($R_{in}(i,j)$)
  **case** $j = 3$ // $X(i,j)$ is link length
    $R_{mutated}(i,j) = l$_mutation($R_{in}(i,j)$)

Fig. 13. Pseudo-code of the mutation operator.

mutation operator[56] is performed. In order to mutate the link lengths such that the mutated value $l_{i,mutated}$ remains in the bound, $[l_{min}, l_{i\,max}]$, the mutation is carried out using the following steps:

1. A random number $u$ in the range [0, 1] is created.
2. $\Delta_{max}$ is computed by using the following equation:

$$\Delta_{max} = \begin{cases} |l_i - l_{min}|, & \text{if } u \le 0.5 \\ |l_i - l_{min}|, & \text{if } u \ge 0.5 \end{cases}. \tag{31}$$

3. $\bar{\delta}$ is calculated as follows:

$$\bar{\delta} = \begin{cases} (2u)^{\frac{3}{1+\eta}}, & \text{if } u \le 0.5 \\ 1 - [2(1-u)]^{\frac{1}{1+\eta}}, & \text{if } u \ge 0.5 \end{cases}. \tag{32}$$

4. The mutated link length is determined by computing:

$$l_{i,mutated} = l_i + \bar{\delta}\Delta_{max}. \tag{33}$$

The aforementioned process produces a mutated link length using a polynomial probability distribution with the mean as the original link length and the variance a function of $\eta$. The maximum and minimum of the mutated value are bounded as $\Delta_{min}$ and $\Delta_{max}$. An $\eta$ between two and five produces a mutation process which simulates the binary mutation.[56] If the selected variable is a joint relative orientation $\phi_i$ or a joint type $m_i$, a discrete version of the mutation operator is used.

### *8.8. Selection for local search*

In this stage, $N_{pop}$, the list of the individuals that are selected to undergo the local search is formed. The number of individuals in $N_{pop}$ depends on the desired frequency which the local search should be applied to the population. The $i$th individual is added to $N_{pop}$ with a probability $\Lambda_i$ which is computed as

$$\Lambda_i = \frac{\Lambda}{N_i}, \tag{34}$$

where $N_i$ is the number of individuals in the population which represent identical solutions to the problem. In the proposed TBCO, $N_i$ represents the number of individuals with the same $\Sigma$ as the $i$th individual and therefore, $N_i = c_i$. $\Lambda$ is a constant which represents the probability of the local search when $N_i = 1$.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table III. MA parameters used in the numerical analysis of the result section.

| Parameter | Value |
|---|---|
| $r_{min}$ | 30 |
| $\rho$ | $0.5r$ |
| $e$ | $0.1p$ |
| $\kappa_r$ | $\frac{1}{8}$ |
| $g_{max}$ | 20 |
| $\varepsilon_{max}$ | $0.1t$ |
| $\kappa_e$ | $\frac{1}{3}$ |
| $\delta_1$ | $10^2$ |
| $\delta_2$ | $10^3$ |
| $\xi$ | 0.2 |
| $\eta$ | 2 |
| $\Lambda$ | 1 |
| $\rho_l$ | 20 |
| $\rho_H$ | 50 |
| $\delta_{LS}$ | $10^3\varepsilon_{cons}$ |

### *8.9. Local search*

In this stage, the $\Delta$ search is applied to the members of $N_{pop}$. A method for limiting the number of iterations of the $\Delta$ search is employed. A gradual local improvement of the individuals' fitness decreases the computational cost of the local search in each generation and simultaneously increases the resilience of the algorithm against premature convergence. To achieve a tangible decrease in the reachability error, if the reachability error is small, more iterations are required than if the error is high. Therefore, the number of iterations $\rho$ is determined, adaptively, for each individual according to its reachability error as follows:

$$\rho_i = \begin{cases} \rho_L, & \text{if } F^i_{rch,T} \leq \delta_{LS} \\ \rho_H & \text{else} \end{cases}, \tag{35}$$

where the number of iterations of the local search for individual $R_i$ is equal to constant $\rho_L$, if the reach- ability error is less than or equal to a threshold, $\delta_{LS}$. Otherwise the number of iterations is equal to $\rho_H$, where $\rho_L \leq \rho_H$. By using this scheme, the local search is applied to individuals with smaller reachability errors more aggressively.

## 9. Results

In this section, the results of the numerical analysis of the proposed TBCO algorithm are presented. In all the test cases, the total reachability error is considered as the optimization constraint. When the reachability errors are within the tolerance of the system, optimization criterion is used to select the optimal solution. The optimization criterion depends on the application, what needs to be maximized or minimized such as required power, required torque, task duration, maximizing the payload, or a combination of them. Therefore, the algorithm in fact aims to find the optimum IK solution for the given goal. For this paper, the optimization criterion is the required power to execute the task. To calculate the required power, seven degree polynomial trajectories are used to connect the starting pose to the end pose. Each trajectory is created such that the duration of the motion from the initial point to the end point is constant for all the task point pairs. With the recursive Newton–Euler formulation, the inverse dynamic problem is solved to reach the required torque by the manipulator to follow the trajectory. Using the required torque and the angular velocity, the maximum power of each joint is calculated. Then, the sum of required power is used as a measure of the optimization criterion. Table III lists the value of the parameters of the MA used in numerical analysis of this section, where $[x]$ represents the nearest integer less than or equal to $x$.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table IV. Performance measures of the MA and GA.

| Method | Test run number | Final generation number | Minimum reachability error | Average generation time (minutes) | Total time (minutes) |
|---|---|---|---|---|---|
| GA | 1 | 40 | 111.7028 | 4.93 | 197.41 |
| | 2 | 40 | 46.9296 | 4.38 | 175.40 |
| | 3 | 40 | 8.7758 | 2.99 | 119.62 |
| | 4 | 40 | 54.0351 | 4.48 | 179.40 |
| | 5 | 40 | 48.6138 | 3.98 | 159.22 |
| MA | 1 | 1 | 0.0643 | 14.66 | 14.66 |
| | 2 | 1 | 0.0254 | 13.52 | 13.52 |
| | 3 | 1 | 0.0204 | 13.57 | 13.57 |
| | 4 | 8 | 0.0334 | 11.43 | 91.45 |
| | 5 | 5 | 0.0369 | 12.73 | 63.67 |

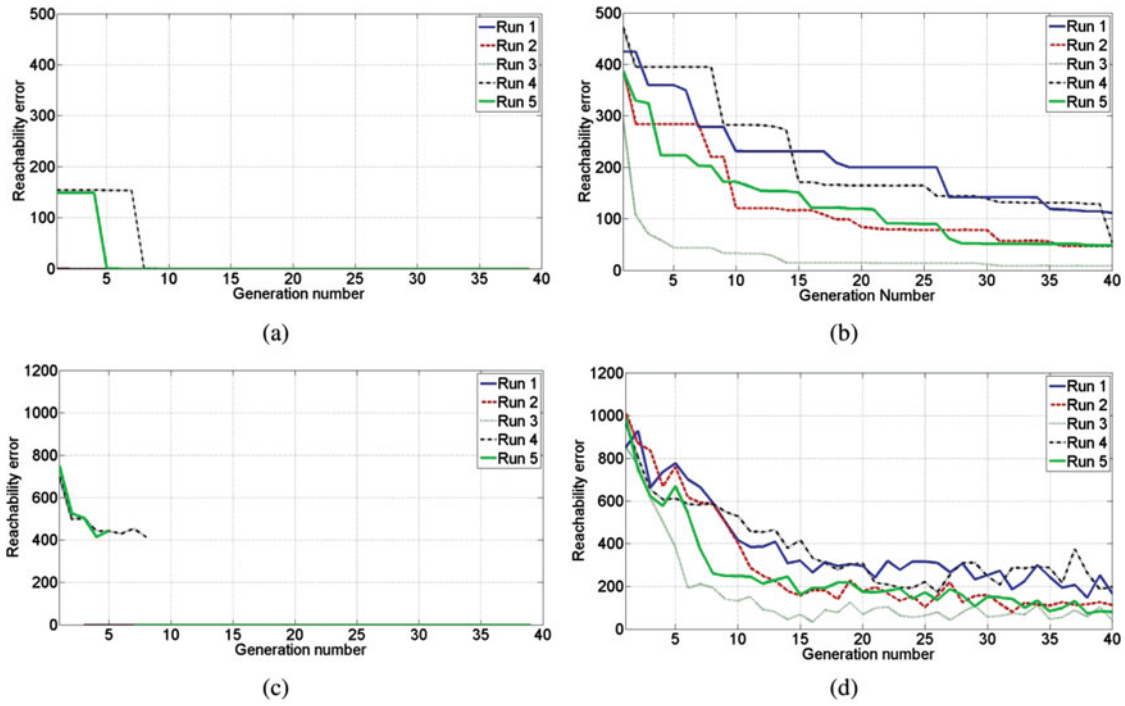
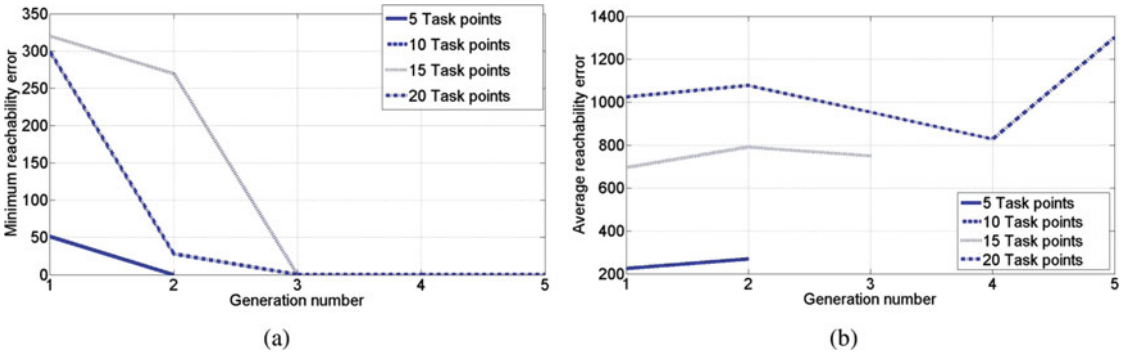



Fig. 14. Comparison of the MA and GA: (a) MA minimum reachability error, (b) GA minimum reachability error, (c) MA average reachability error, and (d) GA average reachability error.

### *9.1. TBCO in* $\mathbb{M}_3$

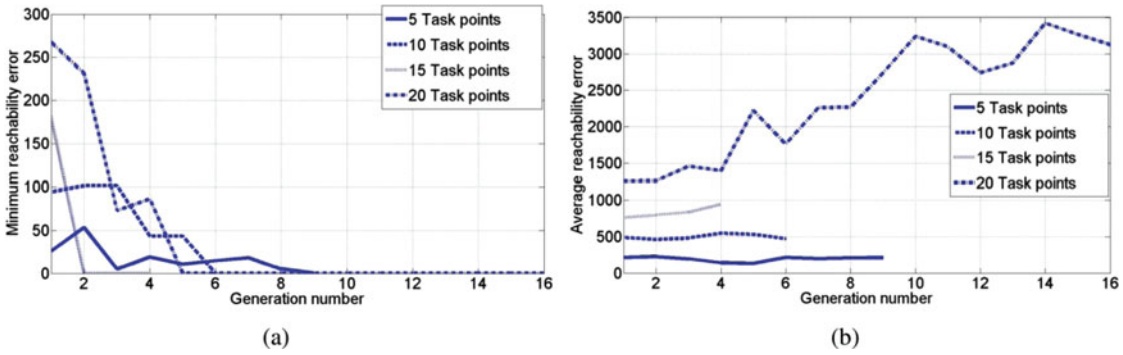
*9.1.1. Test case 1: MA versus GA for solving the TBCO.* In this test case, the TBCO problem is solved in the $\mathbb{M}3$ space for five tasks each consisting of four task points. Both MA and a pure GA version of the algorithm are tested. The GA version shares the genetic operators (Crossover and Mutation), and the elitism (Elitism) subroutines with the MA, but does not include Restart Population, Selection for Local Search, and Local Search subroutines. Since the idea is to compare how fast each algorithm finds a manipulator capable of satisfying the constraints, no optimization criteria are considered, and only the third case of the selection operator, in which the constraints are compared, is implemented. In Fig. 14, the minimum and average reachability errors of the population with respect to the generation number for MA and GA are plotted. In both algorithms, the maximum permissible generation number is 40. The MA can find a manipulator capable of performing the task in all the test runs. In three of the runs, such a manipulator is found in the first generation of the MA. For the GA, none of the runs reaches a manipulator capable of satisfying the constraints. Table IV summarizes the operational measures of the test runs for the MA and GA. For the GA, the lowest reachability error belongs

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press




(a) (b)

Fig. 15. Effect of the number of task points in $\mathbb{M}3$ TBCO: (a) minimum reachability error of the population and (b) average reachability error of the population.




(a) (b)

Fig. 16. Effect of the number of task points in $\mathbb{M}5$ TBCO: (a) minimum reachability error of the population and (b) average reachability error of the population.

to a manipulator in the last generation of the third test run. Since the reachability error for the manipulator is 8.7758, the manipulator cannot perform the task with the desired precision. The MA finds a manipulator with a reachability error within the permissible tolerance, $\varepsilon_{cons}$, in all the test runs. However, the average generation run time of the MA is approximately three times the generation run time of the GA. The reason is that in each generation of the MA, a fraction of the population undergoes the local search algorithm. Although, the generation run time of the GA is smaller than that of the MA at the final generations, the total run time of the MA is less than the GA. The fact that, in the final generation, the MA has already found a good solution, whereas the GA is unable to determine a manipulator capable of performing the task shows the superior performance of the MA in finding manipulators capable of satisfying the constraints, i.e., performing the task.

*9.1.2. Test case 2: The effect of the number of task points.* To investigate the effect of increasing the number of task points on the performance of the proposed MA, the TBCO in $\mathbb{M}3$ and $\mathbb{M}5$ space with varying number of tasks are solved. Figure 15 shows the minimum and average reachability errors, plotted with respect to the generation number for each search in $\mathbb{M}3$, when the number of task points varies from 5 to 20. Since, in the 10 task point case, a solution is reached in the first generation, the minimum and average reachability errors of the test are not visible on the plot. It is obvious that the algorithm finds a KC, capable of performing the task, in all the test cases.

Figure 16 shows the results of the same test for a search in $\mathbb{M}5$. Here, the algorithm has reached a solution in generation 9 for the 5 task point case, and a solution in generation 2 of the 15 task point case.

Table V summarizes the generation numbers in which a KC, capable of performing the task is reached. It seems that the generation, in which a manipulator capable of performing the task is found, is directly affected by the following factors:

- **The closeness of the individuals of the initial population to a KC capable of performing the task.** The measure of the closeness can be considered as the number of genetic operators needed

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table V. MA generation number in which a KC, capable of performing the task, is reached.

| Search space | 5 Task points | 10 Task points | 15 Task points | 20 Task points |
|---|---|---|---|---|
| $\mathbb{M}_3$ | 2 | 1 | 3 | 5 |
| $\mathbb{M}_5$ | 9 | 6 | 4 | 16 |

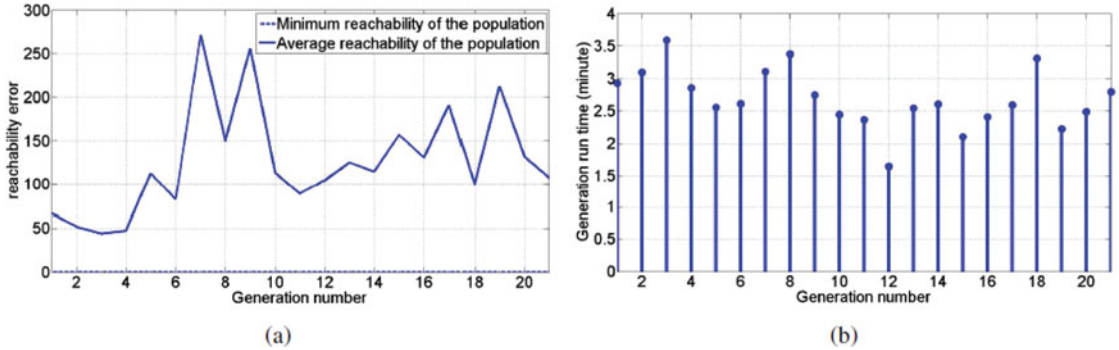



Fig. 17. Minimum and average reachability errors of the population and the generation run time for a M5 search using the proposed MA, test case 1: (a) minimum and average population reachability errors and (b) generation run time.

to produce a KC with a kinematic structure which can be improved by the $\Delta$ search to perform the task.

- **The generation number in which the local search is applied to a potential solution.** In each generation and for all the individuals, the probability for undergoing the $\Delta$ search is computed. Even though an individual close to a solution might exist in the population, it might not have the chance to undergo the $\Delta$ search until further generations are completed.

### *9.2. TBCO in* $\mathbb{M}_5$

*9.2.1. Test case 1.* In this test case, the proposed TBCO algorithm is applied searching in the $\mathbb{M}5$ space for manipulators that are capable of performing a T2 task. To ensure that the all the task points are reachable with an $\boldsymbol{m}_n$ manipulator, the task points are selected randomly from the workspace of the following $\boldsymbol{m}_5^{\text{ref}}$ manipulator:

$$\boldsymbol{m}_5^{\text{ref}} = \begin{bmatrix} 0 & 0 & 60 \\ 0 & R & 30 \\ 0 & P & 30 \\ 0 & R & 10 \\ 0 & P & 10 \\ 0 & R & 10 \end{bmatrix}. \tag{36}$$

The minimum and average reachability errors, and the generation run time of the TBCO, applied to this problem are shown with respect to the generation number in Fig. 17. According to Fig. 17(a), the first manipulator capable of performing the task is found in the first generation, but the run is not terminated in order to search for KCs with lower requirements for executing the task. Figure 17(b) illustrates the run time of the algorithm in each generation. The variation, which can be observed in the run time from one generation to another, is due to the stochastic nature of the genetic selection operator and the selection for numerical operators. The generations, in which the selection operator needed to compute $F_{obj}$ often require more time. Another factor affecting the run time is the number of individuals in the generation which undergo the $\Delta$ search operator. In diverse populations, in which $c_i$ is relatively small, $\Lambda_i$is higher, and the numerical search is applied to a higher fraction of the individuals, the run time of the generation increases.

In the last generation of MA, two KC, $\boldsymbol{m}_5^{KC1}$ and $\boldsymbol{m}_5^{KC2}$, capable of performing the task, exist. The KC of $\boldsymbol{m}_5^{ref}$ with $\boldsymbol{m}_5^{KC1}$ and $\boldsymbol{m}_5^{KC2}$, when they reached the first task point, are signified in Fig. 18.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table VI. Results of the MA-based TBCO in $\mathbb{M}_5$, test case 1.

| Kinematic configuration | Position reachability error (cm) | Orientation reachability error (rad) | Required power (KW) |
|---|---|---|---|
| $\boldsymbol{m}_5^{KC1}$ | 0.0006 | 0.0000 | 0.143 |
| $\boldsymbol{m}_5^{KC2}$ | 0.0007 | 0.0000 | 0.301 |

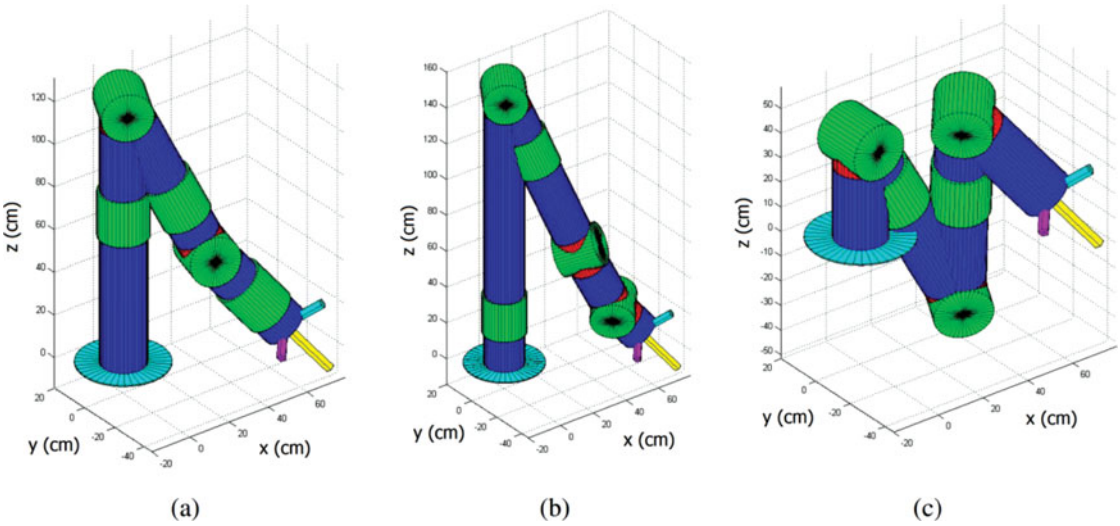



Fig. 18. KCs of $\boldsymbol{m}_5^{\text{ref}}$ and the two outputs of the MA-based TBCO algorithm in test case 1: (a) $\boldsymbol{m}_5^{\text{ref}}$, (b) $\boldsymbol{m}_5^{KC1}$, and (c) $\boldsymbol{m}_5^{KC2}$.

The matrix representations of the solutions are as follows:

$$\boldsymbol{m}_5^{KC1} = \begin{bmatrix} 0 & 0 & 17.8 \\ 0 & R & 101.6 \\ 0 & P & 14.5 \\ 0 & R & 41.0 \\ 0 & P & 20.6 \\ \pi/2 & R & 10.0 \end{bmatrix}, \tag{37}$$

and

$$\boldsymbol{m}_5^{KC2} = \begin{bmatrix} 0 & 0 & 28.7 \\ 0 & R & 6.4 \\ 0 & P & 42.1 \\ 0 & R & 32.7 \\ 0 & P & 13.7 \\ 0 & R & 36.8 \end{bmatrix}. \tag{38}$$

Table VI summarizes the operational measures of the two KCs. Although the position and orientation reachability errors of both manipulators indicate that they can reach the task points with a high precision, $\boldsymbol{m}_5^{KC1}$is the final solution of the TBCO due to the lower power requirements in performing the task.

*9.2.2. Test case 2.* In this test case, the proposed TBCO algorithm is used to determine a manipulator capable of performing a T10 task. The task points are randomly selected from the workspace of $\boldsymbol{m}_{ref}$.

In Fig. 19(a) the minimum and average reachability errors of the population are plotted with respect to the generation number. According to the minimum reachability error, the first manipulator, capable of performing the task, is found in generation 6. To perform the search for the manipulators with a higher performance, in terms of the required power, the algorithm is not terminated in 20

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

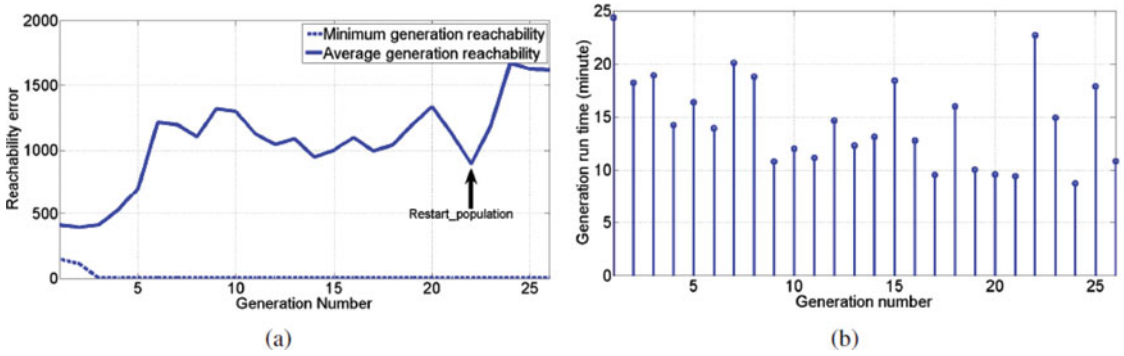



Fig. 19. Minimum and average reachability errors of the population and the generation run time for a $M_5$ search using the proposed MA, test case 2: (a) minimum and average population reachability errors and (b) generation run time.

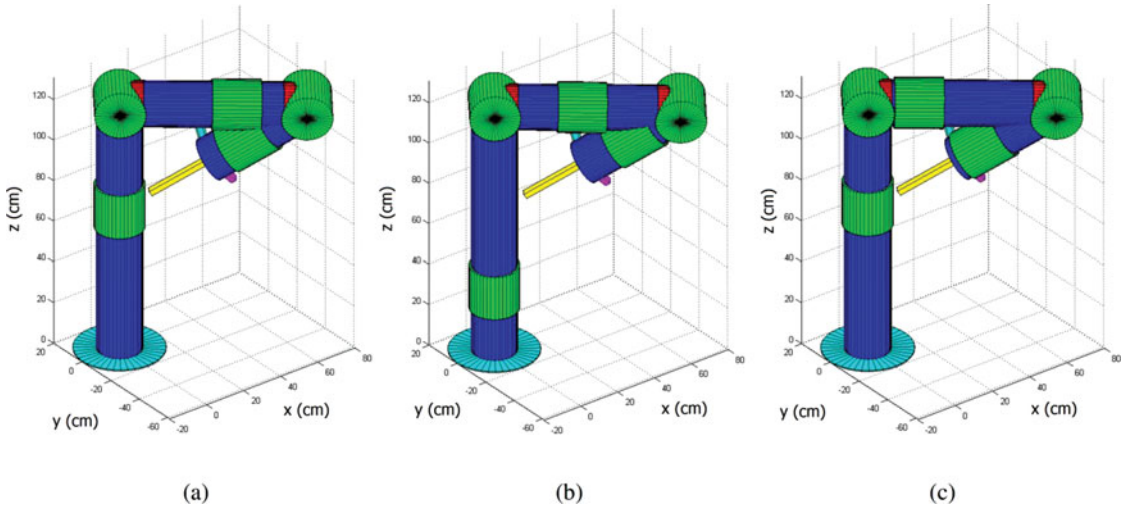



Fig. 20. KCs of $\boldsymbol{m}_5^{\text{ref}}$ and the two outputs of the MA-based TBCO algorithm in test case 2: (a) $\boldsymbol{m}_5^{\text{ref}}$, (b) $\boldsymbol{m}_5^{KC3}$, and (c) $\boldsymbol{m}_5^{KC4}$.

more generations. As illustrated in the figure, in generation 22, the restart population subroutine has substituted a part of the population with new individuals.

Figure 20(a) shows the KC of $\boldsymbol{m}_5^{ref}$ for the first task point. In the final population of the MA, two KCs, $\boldsymbol{m}_5^{KC3}$for $\boldsymbol{m}_4^{KC}$, that are capable of performing the task are found. The matrix representation of $\boldsymbol{m}_5^{KC3}$for $\boldsymbol{m}_4^{KC}$ are

$$\boldsymbol{m}_5^{KC3} = \begin{bmatrix} 0 & 0 & 21.3 \\ 0 & R & 68.6 \\ \pi/2 & P & 17.7 \\ 0 & R & 22.2 \\ 0 & P & 3.8 \\ 0 & R & 16.1 \end{bmatrix}, \tag{39}$$

and

$$\boldsymbol{m}_5^{KC4} = \begin{bmatrix} 0 & 0 & 61.2 \\ 0 & R & 28.7 \\ 0 & R & 1.7 \\ 0 & R & 38.2 \\ \pi/2 & P & 16.6 \\ 0 & R & 3.3 \end{bmatrix}. \tag{40}$$

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table VII. Results of the MA-based TBCO in $\mathbb{M}_5$, test case 2.

| Kinematic configuration | Position reachability error (cm) | Orientation reachability error (rad) | Required power (KW) |
|---|---|---|---|
| $\boldsymbol{m}_5^{KC3}$ | 0.0094 | 0.0004 | 2.2007 |
| $\boldsymbol{m}_5^{KC4}$ | 0.0068 | 0.0006 | 3.0350 |

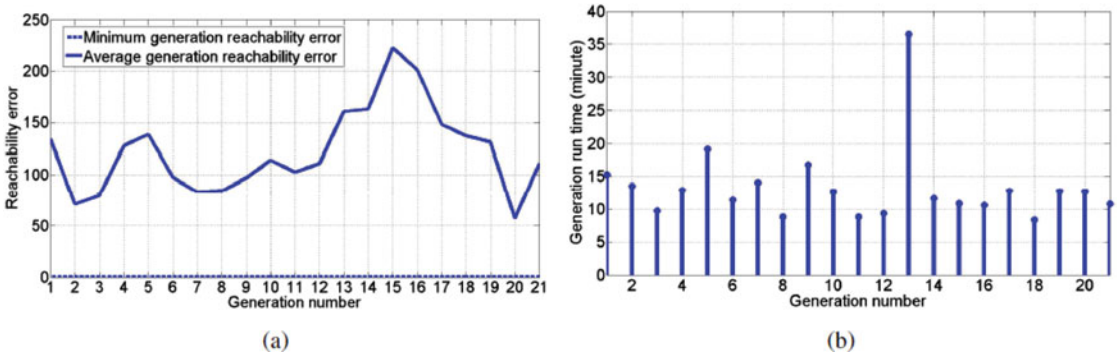



Fig. 21. Minimum and average reachability errors of the population and the generation run time for a $\mathbb{M}6$ search using the proposed MA, test case 2: (a) minimum and average population reachability errors and (b) generation run time.

Table VII lists the reachability errors and the required power to perform the task for $\boldsymbol{m}_5^{KC3}$for $\boldsymbol{m}_4^{KC}$. The position and orientation measure for both manipulators are within the permissible tolerance. Therefore, both manipulators are capable of performing the tested $\mathbb{T}_{10}$. Figure 20(b) and (c) show $\boldsymbol{m}_5^{KC3}$ and $\boldsymbol{m}_4^{KC}$, respectively. Although the kinematic structure matrix of $\boldsymbol{m}_{ref}^5$, $\boldsymbol{m}_5^{KC3}$, and $\boldsymbol{m}_5^{KC4}$ are distinct by changing the joint reference angle of the rotation joints 1 and 3, it can be shown that they all refer to a similar kinematic structure, i.e., the workspace of all the three manipulators has the same shape.

Although the $\Delta$ of the three manipulators are not equal, the volume (size) of their workspace is also equal and that is because the sum of the links connected to the input and output mechanical ports of rotational joints 1, 3, and 5 are equal. This could be an indication that only a KC with a workspace similar to $\boldsymbol{m}_6^{ref}$ can perform the desired $\mathbb{T}_{10}$ task. The required power for performing the task are 2.2 KW and 3.0 KW for $\boldsymbol{m}_5^{KC3}$ and $\boldsymbol{m}_5^{KC4}$, respectively. Therefore, the final solution of the TBCO is $\boldsymbol{m}_5^{KC3}$which can perform the task with the least required power.

### *9.3. TBCO in* $\mathbb{M}_6$

In this test, the TBCO is applied for finding an $\boldsymbol{m}_6$ manipulator for performing a $\mathbb{T}_5$ task. The task points are generated randomly by the following manipulator

$$\boldsymbol{m}_5^{\text{ref}} = \begin{bmatrix} 0 & 0 & 60 \\ 0 & R & 30 \\ 0 & P & 30 \\ 0 & R & 10 \\ 0 & P & 10 \\ 0 & R & 10 \end{bmatrix}. \tag{41}$$

Figure 21(a) depicts the minimum and average reachability errors of the population with respect to the generation number. It is evident that in generation 1, a manipulator that is capable of performing the task is found. The MA iterates for 20 generations in order to search $\mathbb{M}_6$ for manipulators that perform better in executing the task. Figure 21(b) reflects the run time of each generation.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

Table VIII. Results of the MA-based TBCO in $\mathbb{M}_6$.

| Kinematic configuration | Position reachability error (cm) | Orientation reachability error (rad) | Required power (KW) |
|---|---|---|---|
| $\boldsymbol{m}_5^{KC1}$ | 0.0041 | 0.0007 | 1.247 |
| $\boldsymbol{m}_5^{KC2}$ | 0.0032 | 0.0002 | 1.286 |
| $\boldsymbol{m}_5^{KC3}$ | 0.0020 | 0.0001 | 10.558 |
| $\boldsymbol{m}_5^{KC4}$ | 0.0055 | 0.0000 | 3.178 |
| $\boldsymbol{m}_5^{KC5}$ | 0.0070 | 0.0004 | 4.985 |
| $\boldsymbol{m}_5^{KC6}$ | 0.0061 | 0.0010 | 3.681 |
| $\boldsymbol{m}_5^{KC7}$ | 0.0094 | 0.0001 | 7.862 |
| $\boldsymbol{m}_5^{KC8}$ | 0.0013 | 0.0000 | 2.262 |

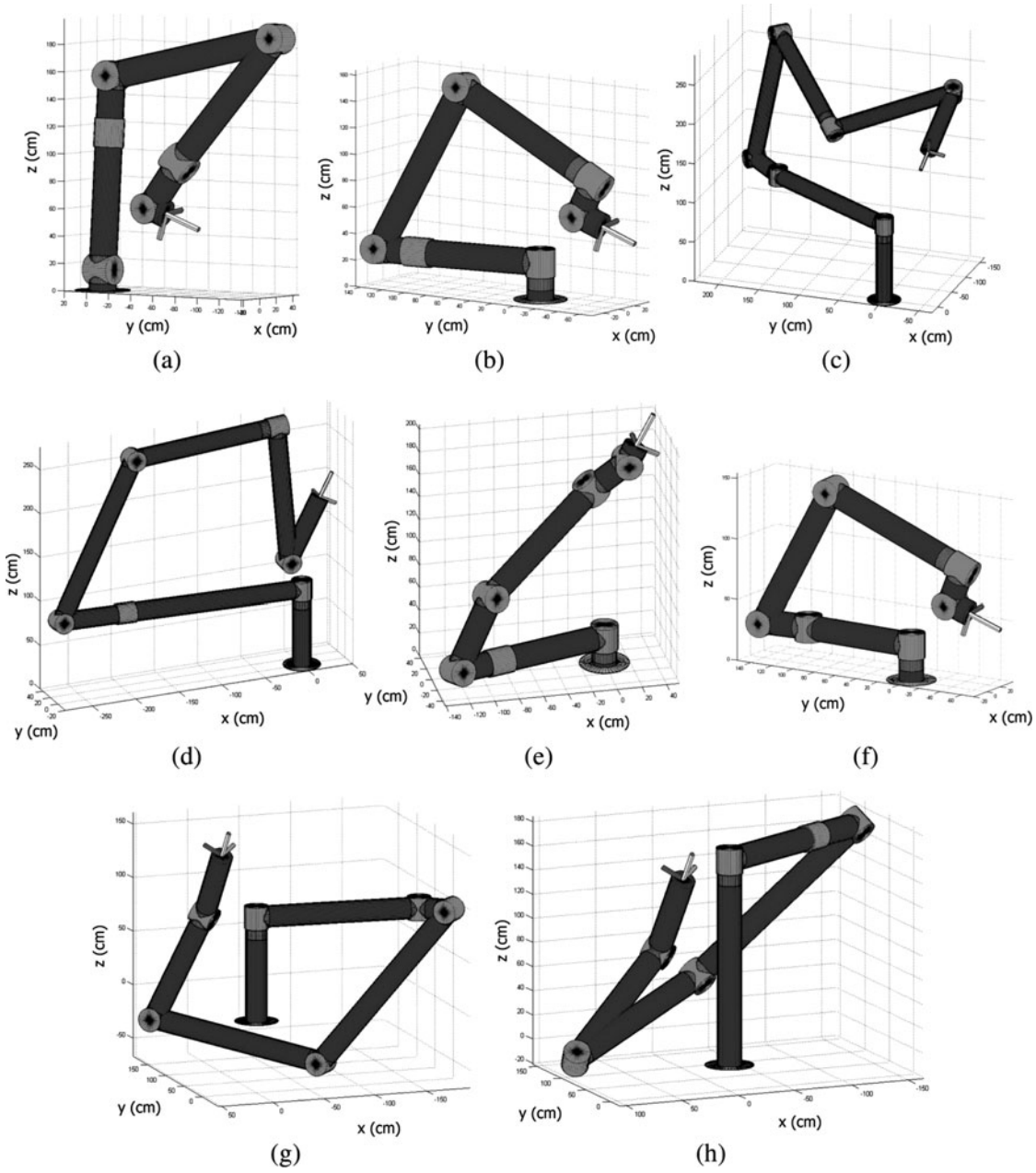



Fig. 22. KCs of the outputs of the MA-based TBCO algorithm in $\mathbb{M}6$: (a) $\boldsymbol{m}_6^{KC1}$, (b) $\boldsymbol{m}_6^{KC2}$, (c) $\boldsymbol{m}_6^{KC3}$, (d) $\boldsymbol{m}_6^{KC4}$, (e) $\boldsymbol{m}_6^{KC5}$, (f) $\boldsymbol{m}_6^{KC6}$, (g) $\boldsymbol{m}_6^{KC7}$, (h) $\boldsymbol{m}_6^{KC8}$.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

In the final generation of MA, eight KCs, $\boldsymbol{m}_6^{KCi}$, with $i = 1, \ldots, 8$, capable of performing the task are found. In Fig. 22, the KCs of the aforementioned manipulators are illustrated. Table VIII summarizes the operational performance measures of the manipulators. The reachability errors of all the manipulators are within the permissible tolerance range. However, since $\boldsymbol{m}_6^{KC1}$ can perform the task with lower power requirements, $\boldsymbol{m}_6^{KC1}$ is selected as the solution to the TBCO problem.

## 10. Conclusions

In this paper, a TBCO solving problem for MRRs using an MA was presented. The proposed algorithm combines GA and local search algorithms to exploit the advantages of both algorithms. The proposed algorithm benefits from a kinematic structure-aware elitism scheme, kinematic structure-aware restart scheme, priority-based selection operator, and adaptive local search frequency. It is demonstrated that in an $\mathbb{M}_3$ TBCO, the proposed MA finds a manipulator capable of satisfying the task faster than the GA. Moreover, the search in $\mathbb{M}_5$ shows that when the number of the task points $t$ in the desired task $\mathbb{T}_t$ increases, the set of manipulators, capable of performing $\mathbb{T}_t$, shrinks. For instance, in one of the test cases, it is shown that the algorithm only finds one $\boldsymbol{m}_5$ kinematic structure, capable of performing the desired $\mathbb{T}_{10}$ task. In addition, the MA-based TBCO algorithm was successfully applied to find optimized $\boldsymbol{m}_6$ manipulator for performing a $\mathbb{T}_5$ task. Furthermore, a methodology was proposed for conducting the $\Delta$ search, i.e., for the link dimensions of a manipulator this method enables reaching a set of task points. The developed method is based on converting the $\Delta$ search to an IK problem of a redundant manipulator and solving it for all the task points concurrently. The proposed method was verified for three distinct cases, each consisting of a task with 50 task points. The proposed algorithm can accommodate any number of constraints and optimization criteria. In a case with more criteria, a weighted sum of constraints and optimization criteria can be adopted as the high and low priority fitness values, respectively. The proposed algorithm can be extended to reach task points with obstacle avoidance in future.

## References

1. Board on Manufacturing and National Research Council Engineering Design, *Visionary Manufacturing Challenges for 2020* (National Academy Press, Washington, D.C., 1998).
2. Z. M. Bi and W. J. Zhang, "Modularity technology in manufacturing: Taxonomy and issues," *Int. J. Adv. Manuf. Technol.* **18**, 381–390 (2001).
3. Z. M. Bi, S. Y. T. Lang, W. Shen and L. Wang, "Reconfigurable manufacturing systems: The state of the art," *Int. J. Prod. Res.* **46**(4), 967–992 (2007).
4. Z. M. Bi, S. Y. T. Lang, M. Verner and P. Orban, "Development of reconfigurable machines," *Int. J. Adv. Manuf. Technol.* **39**, 1227–1251 (2008).
5. T. Fukuda and S. Nakagawa, "Dynamically Reconfigurable Robotic System," *Proceedings of the IEEE International Conference on Robotics and Automation*, (1988) pp. 1581–1586.
6. D. E. Schmitz, P. K. Khosla and T. Kanade, "The CMU Reconfigurable Modular Manipulator System," *Proceedings of the International Symposium and Exposition on Robots (designated 19th ISIR)*, Sydney, Australia (1988) pp. 473–488.
7. C. J. J. Paredis, H. B. Brown and P. K. Khosla, "A rapidly deployable manipulator system," *Robot. Autom. Syst.* **21**, 289–304 (1997).
8. I.-M. Chen, "A Rapidly Reconfigurable Robotics Workcell and its Applications for T Engineering," *Innovation in Manufacturing Science and Technology Program, Singapore-MIT Alliance Annual Symposium*, Traders Hotel, Singapore (2003).
9. R. Hui, N. Kircanski, A. Goldenberg, C. Zhou, P. Kuzan, J. Wiercienski, D. Gershon and P. Sinha, "Design of the IRIS Facility - A Modular, Reconfigurable and Expandable Robot Test Bed," *Proceedings of the IEEE International Conference on Robotics and Automation,* **3**, (1993) pp. 155–160.
10. T. Matsumaru, "Design and Control of the Modular Robot System: TOMMS," *Proceedings of the IEEE International Conference on Robotics and Automation*, (May 21–27, 1995) 2, pp. 2125–2131.
11. Z. Li, "Development and Control of a Modular and Reconfigurable Robot with Harmonic Drive Transmission System," *Master's Thesis*, (Ontario, Waterloo: University of Waterloo, 2007).
12. S. Kirchhoff, "Intelligent Configuration Selection and Robust Control for Reconfigurable Robots," *Master's Thesis*, (University of Waterloo and Technical Universitat Hamburg-Harburg, 2005).
13. I.-M. Chen and J. W. Burdick, "Determining Task Optimal Modular Robot Assembly Configurations," *IEEE International Conference on Robotics and Automation*, (1995) pp. 132–137.

https://doi.org/10.1017/S0263574714002690 Published online by Cambridge University Press

14. I.-M. Chen, "On Optimal Configuration of Modular Reconfigurable Robots," *4th International Conference on Control, Automation, Robotics and Vision*, Singapore (1996) pp. 1855–1859.
15. G. Yang and I.-M. Chen, "Reduced DOF Modular Robot Configurations," *5th International Conference on Control, Automation, Robotics and Vision*, Singapore (1998) pp. 1513–1517.
16. I.-M. Chen and G. Yang, "An Evolutionary Algorithm for the Reduction of DOFs in Modular Reconfigurable Robots," *Proceeding of the ASME Dynamic System and Control Division*, DSC-64, (1998) pp. 759–766.
17. C. Leger and J. Bares, "Automated Synthesis and Optimization of Robot Configurations," *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, Atlanta, GA (1998).
18. C. Leger and J. Bares, "Automated Task-Based Synthesis and Optimization of Field Robots," *Proceedings of the International Conference on Field and Service Robotics (FSR99)*, Pittsburgh, PA (1999).
19. H. Valsamos and N. A. Aspragathos, "Design of a Versatile Passive Connector for Reconfigurable Robotic Manipulators with Articulated Anatomies and their Kinematic Analysis," *I*PROMS 2007 Virtual Conference*, (2007).
20. H. Valsamos, V. C. Moulianitis and N. A. Aspragathos, "Rapid Evaluation of Reconfigurable Robots Anatomies Using Computational Intelligence," *Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II*, (Sep. 2010) pp. 341–350.
21. C. Valsamos, V. Moulianitis and N. Aspragathos, "Index based optimal anatomy of a metamorphic manipulator for a given task," *Int. J. Robot. Comput. Integr. Manuf.* **28**, 517–529 (2012).
22. C. J. J. Paredis and P. K. Khosla, "An Approach for Mapping Kinematic Task Specifications into a Manipulator Design," *Proceedings of the 5th International Conference on Advanced Robotics*, (ICAR '91), (1991) 1, pp. 556–561.
23. C. J. J. Paredis and P. Khosla, "Kinematics design of serial link manipulators from task specifications," *Int. J. Robot. Res.* **12**(3), 274–287 (1993).
24. J.-O. Kim and P. Khosla, "A Formulation for Task based Design of Robot Manipulators," *Proceedings of the 1993, IEEE/RSJ International Conference on Intelligent Robots and Systems*, (1993) pp. 2310–2317.
25. J. Kim and P. Khosla, "Design of Space Shuttle Tile Servicing Robot: An Application of Task based Kinematic Design," *IEEE International Conference on Robotics and Automation*, (ICRA '93), (May 1993) 3, pp. 867–874.
26. C. J. J. Paredis and P. K. Khosla, "Agent-Based Design of Fault Tolerant Manipulators for Satellite Docking," *Proceedings of the 1997, IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico (Apr. 1997).
27. Q. Li and J. Zhao, "A universal approach for configuration synthesis of reconfigurable robots based on fault tolerant indices," *Ind. Robot: Int. J.* **39**(1), 69–78 (2012).
28. J. Han, W. K. Chung, Y. Youm and S. H. Kim, "Task based Design of Modular Robot Manipulator using Efficient Genetic Algorithm," *Proceedings of the 1997, IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico (Apr. 1997) pp. 507–512.
29. W. Gao, H. Wang, Y. Jiang and X. Pan, "Task-based Configuration Synthesis for Modular Robot," *International Conference on Mechatronics and Automation (ICMA)*, (Aug. 2012) pp. 789–794.
30. O. Chocron, "Evolutionary design of modular robotic arms," *Robotica* **26**(3), 323–330 (May 2008).
31. B. Dong and Y. Li, "Multi-Objective-Based Configuration Generation and Optimization for Reconfigurable Modular Robot," *International Conference on Information Science and Technology (ICIST)*, (Mar. 26–28, 2011) pp. 1006–1010.
32. A. Ellery, *An Introduction to Space Robotics* (Springer Praxis publishing, UK, 2000).
33. G. Yang and I. Chen, "Task-based optimization of modular robot configurations: Minimized degree-of-freedom approach," *Mech. Mach. Theory* **35**(4), 517–540 (2000).
34. P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," *Technical Report Caltech Concurrent Computation Program 826*, California Institute of Technology, USA (1989).
35. R. Dawkins, *The Selfish Gene* (Oxford University Press, Oxford, 1976).
36. G. C. Onwubolu and B. V. Babu, *New Optimization Techniques in Engineering* (vol. 141 of Studies in Fuzziness and Soft Computing, Springer - Verlag, Berlin, Heidelberg, 2012).
37. D. Wolpert and W. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions in Evolutionary Computation*, (1997) pp. 67–82.
38. J. Culberson, "On the futility of blind search: An algorithmic view of no free lunch," *Evol. Comput.* **6**(2), 109–128 (1998).
39. D. Goldberg and S. Voessner, "Optimizing Global-Local Search Hybrids," *Proceedings of the Genetic and Evolutionary Computation Conference*, (1999) pp. 220–228.
40. N. Krasnogor and J. Smith, "A tutorial for competent Memetic algorithms: Model, taxonomy, and design issues," *IEEE Trans. Evol. Comput.* **9**(5), 474–488 (Oct. 2005).
41. N. Krasnogor, "Memetic Algorithms," *A Tutorial given in the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, (Sep. 2002).
42. N. J. Radcliffe and P. D. Surry, "Formal Memetic Algorithms," In: *Selected Papers from AISB Workshop on Evolutionary Computing*, (Springer-Verlag, London, UK, 1994) pp. 1–16.
43. D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Mach. Syst.* **10**(2), 47–53 (Jun. 1969).

44. T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robot. Res.* **4**(2), 3–9 (1985).
45. A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. J. Robot. Res.* **4**, 109–117 (1985).
46. C. W. Wampler II, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. Syst. Man Cybern.* **16**, 93–101, (1986).
47. W. A. Wolovich and H. Elliot, "A Computational Technique for Inverse Kinematics," *Proceedings of the 23rd IEEE Conference on Decision and Control*, (1984) pp. 1359–1363.
48. J. Baillieul, "Kinematic Programming Alternatives for Redundant Manipulators," *IEEE International Conference of Robotics and Automation*, St. Louis (1985) pp. 722–728.
49. H. Seraji, "Configuration control of redundant manipulators: Theory and implementation," *IEEE J. Robot. Autom.* **5**(4), 472–490 (1989).
50. C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and its Applications* (Wiley, New York, 1971).
51. Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Trans. ASME J. Dyn. Syst. Meas. Control* **108**, 163–171 (1986).
52. E. Zitzler, K. Deb and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.* **8**(2), 173–195 (Feb. 1999).
53. G. Rudolph, "Evolutionary search under partially ordered fitness sets," *Proceedings of the International Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems*, (1999) pp. 818–822.
54. W. E. Hart, Adaptive Global Optimization with Local Search *Ph.D. Thesis* (San Diego, California: University of California, 1994).
55. B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Syst.* **9**, 193–212 (1995).
56. K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Inf.* **26**(4), 30–45 (1996).
57. S. Tabandeh, C. M. Clark and W. W. Melek, "Task-based Configuration Optimization of Modular and Reconfigurable Robots using a Multi-Solution Inverse Kinematics Solver," *Proceedings of the 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production*, Toronto, Canada (Jul. 2007) pp. 4–13.