Robotica (2024), **42**, pp. 911–930
doi:10.1017/S0263574723001893

RESEARCH ARTICLE

Solving the inverse kinematics problem of discretely actuated hyper-redundant manipulators using the multi-module searching method

Alireza Motahari

Department of Mechanical Engineering, Engineering Faculty, Saveh Branch, Islamic Azad University, Saveh, Iran
Email: Alireza.Motahari@iau.ac.ir

Received: 25 August 2023; **Accepted:** 18 December 2023; **First published online:** 22 January 2024

Keywords: hyper-redundant manipulator; discrete actuator; inverse kinematics; multi-module search

Abstract

Hyper-redundant manipulators are produced by cascading several mechanisms on top of each other as modules. The discrete actuation makes their control easier because discrete actuators usually do not need any feedback to control. So far, several methods have been proposed to solve the inverse kinematic problem of discretely actuated, hyper-redundant manipulators. The two-by-two searching method is better than the other methods in terms of CPU time and error. In this article, the mentioned method is generalized by choosing an arbitrary number of modules as pending modules in each step of the solution instead of the necessary two. For validation, the proposed method is compared with nine meta-heuristic searching algorithms: simulated annealing, genetic algorithm, particle swarm optimization, ant colony optimization, gray wolf optimizer, stochastic fractal search, whale optimization algorithm, Giza pyramid construction, and flying fox optimization. Furthermore, the effect of the number of pending modules on CPU time and error is investigated. All the numerical problems have been solved for two case studies, one is planar and the other is spatial.

1. Introduction

Robots typically have actuators with a continuous range of actuation. From here, in this article, these robots are referred to as “continuous robots,” and their actuators will be called continuous actuators. What is expected of these robots is to track a trajectory with high precision. This involves the use of complex controller devices and programs that increase costs.

In some applications, such as picking and placing objects, the path of motion does not require high precision, but the points at the beginning and end of the path are important. Some investigators [1, 2] have proposed that discrete actuators be used instead of continuous ones. Discrete actuators are actuators whose actuation values are limited to several values. For example, a binary prismatic actuator has only two stable states, fully retracted (0) and fully extended (1). Discrete actuators do not need feedback to be controlled. For example, for control of a binary pneumatic cylinder, it is only necessary to determine the direction of flow of compressed air correctly and give sufficient time to achieve a fully extended or fully retracted state. In this case, the piston moves so far to reach the end of its stroke. Simplicity in control, low controller equipment prices, high precision, and repeatability are all advantages of discrete actuators due to the lack of feedback.

When all the actuators of a robot are discrete, it is referred to as a discrete robot. The workspace of a discrete robot is formed from a finite set of discrete points, so one cannot expect a discrete robot to track a path with a high precision. For tracking a path by a discrete robot, it is necessary to select a few points among all points of the workspace so that they approximate the path in the best way. And then the robot moves from one point to another. Otherwise, the path that the robot tracks between two configurations is uncontrollable. Therefore, the number of workspace points must be greater so that access to the points

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

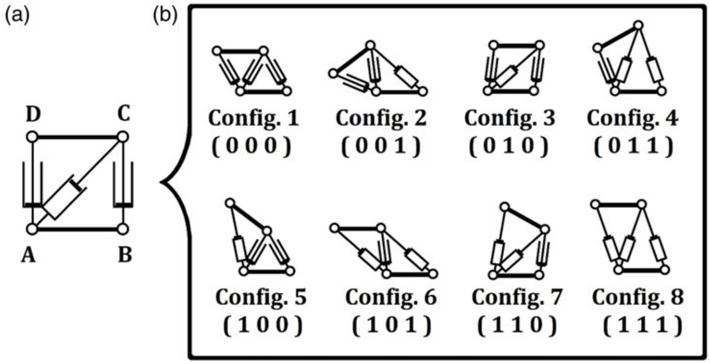


Figure 1. *a) A VGT module. b) Configurations of a VGT module.*

and tracking of the path can be done more accurately. To do this, it is necessary to increase the number of actuation states of the actuators, or, more efficiently, the number of actuators.

Today researchers use discrete actuators in a variety of fields and devices such as grippers [3–9], switches [10], medical imaging devices [11–13], exoskeleton rehabilitation robots [14, 15], solar centralizing mirrors [16, 17], millirobots [18], microrobots [19–21], and soft robots [22–25]. Some investigators use discrete actuators in hyper-redundant manipulators [2, 26–29]. A hyper-redundant manipulator is typically made by cascading parallel mechanisms on top of each other as modules. For example, a VGT[1] module is illustrated in Fig. 1-a. Each module of VGT is a planar close chain mechanism consisting of three binary prismatic actuators (AC, AD, and BC links in Fig. 1-a) and two constant-length links (AB and CD links in Fig. 1-a). These links are passively joined in A, B, C, and D. The VGT module can be actuated into $\mathrm{Ncon} = 2^3 = 8$ different states (configurations) by combining the actuation states of its actuators, as shown in Fig. 1-b. A VGT manipulator can be made by cascading VGT modules on top of each other.

A four-module VGT manipulator ($\mathrm{Nmod} = 4$) is shown in Fig. 2. The configuration number of the first module (the module that is connected to the base) is 4 ($c_1 = 4$). The configuration numbers of the second, third, and last modules are 8, 3, and 6, respectively ($c_2 = 8$, $c_3 = 3$, $c_4 = 6$). These numbers correspond to what is shown in Fig. 1-b. Based on what has been discussed earlier, a four-module VGT manipulator can have $\mathrm{Ncon}^{\mathrm{Nmod}} = 8^4 = 4096$ different configurations. Each configuration places the manipulator's end-frame in a specific location. So, the workspace of this manipulator will include the same number of frames as illustrated in Fig. 3.

The hyper-redundant manipulators have the benefits of parallel robots, such as a high payload and high precision, combined with the benefits of serial robots, such as a large workspace and the ability to avoid obstacles. The use of discrete actuators instead of continuous ones eliminates the complex control problem of the hyper-redundant manipulators. From now on, this type of manipulator will be called "discretely actuated hyper-redundant manipulators (DAHRMs)," or in short, the "discrete manipulators." In spite of the valuable capabilities described for discrete manipulators, they are practically underutilized. This may have been due to a lack of research. As for the inverse kinematics (IK) of these types of manipulators, as discussed below, there were deficiencies in existing publications, which was an incentive to present this article.

The IK problem is to determine the actuation amount of all the manipulator actuators so that the manipulator end-frame reaches a predetermined frame in space called the target-frame. As discussed later, in the case of DAHRMs, there are only a few choices for the actuation amount of each actuator. Therefore, the number of possible configurations for each module will be limited. As a result, the IK problem of DAHRMs becomes the choice of the best configuration from among all possible configurations, which minimizes the distance. In other words, the mentioned problem will be a combinatorial

[1] Variable Geometry Truss.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

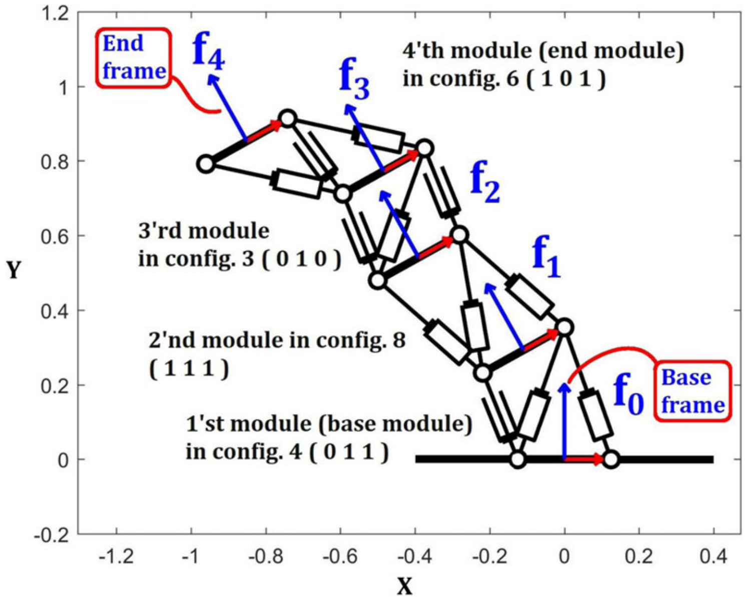

End frame
f_4
f_3
4'th module (end module) in config. 6 (1 0 1)
f_2
3'rd module in config. 3 (0 1 0)
f_1
2'nd module in config. 8 (1 1 1)
f_0
Base frame
1'st module (base module) in config. 4 (0 1 1)
Y
X
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4

Figure 2. *A 4-module VGT manipulator in configuration 4836 ($\mathbb{C}$ =[c_1 c_2 c_3 c_4] = [4 8 3 6]).*

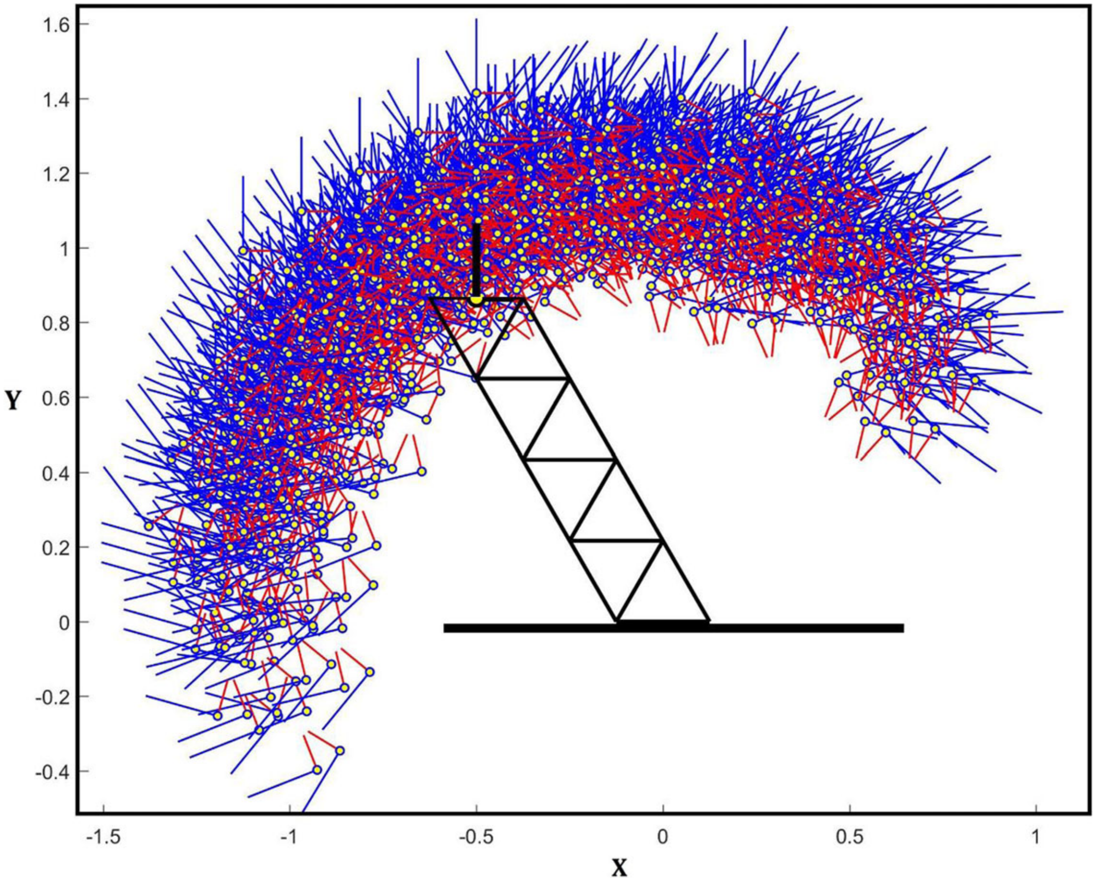


Figure 3. *Workspace of a 4-module VGT manipulator containing 4096 discrete frames.*

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

optimization problem, and it needs a different solution than what is proposed for continuous problems. The distance of the manipulator's end-frame from the target-frame is considered the error of the answer to the IK problem. Here, the distance between two frames involves two concepts: location and direction.

Solving the forward kinematics problem of DAHRMs can be done without any problems. If the forward kinematic solution of each module is known, the transformation matrix of the manipulator can be calculated simply by multiplying the transformation matrix of the modules in sequence. But there are some problems with the methods presented in the field of IK. First, the usual methods used for continuous manipulators are not suitable for discrete manipulators because these methods do not necessarily lead to discrete values of actuation, and if the discrete values close to the values of the result are chosen instead, large errors[2] are created. For example, Chirikijian and Lees [30] approximated the overall shape of the manipulator to a spatial curve. And then they tried to close the end-frame to the target-frame by grasping the manipulator on this curve. But because the actuators are discrete, the modules have a limited number of configurations and could not sit well on the curve. The deviation of the modules from this curve, especially in the case of close-to-base modules, caused large errors in reaching the manipulator's end-frame to the target-frame.

Secondly, the brute-force search method is not suitable. In this method, all configurations of the manipulator should be considered in the search space. The manipulator's end-frame location should be calculated for all configurations of the manipulator (forward kinematics). These items are done off-line, and then the information is saved in the computer's memory. After that, for solving an IK problem, it is sufficient to search among the saved information. This method is not feasible for the manipulators that have too many modules because the number of configurations grows exponentially with the increasing number of modules, and as a result, the CPU time and the required memory size grow exponentially too. In other words, the mentioned problem is an NP-hard problem, so brute-force searching is not feasible for it.

Ebert-Uphoff and Chirikjian [31, 32] used the concept of workspace density to solve the IK. When the workspace density of one point is higher than another, the manipulator has a better chance of reaching it. Their proposed method was a single-module searching algorithm. Here, the term "search" refers to the process of selecting the appropriate configuration among all possible configurations. And the term "single-module searching" refers to a searching process in which the searching space includes the configurations of a single module. Simply put, in a single-module searching process, the configuration of the manipulator was determined module by module. There are three types of modules in each step of this process:

1. Decided module: The module that has been configured in the previous steps.
2. Pending module: The module that is currently being configured.
3. Undecided module: The module that has not been configured.

In the first step, the pending module is the base module (the first module of the manipulator), and the rest of the modules are undecided. In the second step, the base module is a decided module, the second module is a pending module, and the other modules are undecided ones.

The searching process will continue until the last step, in which the pending module is the end module of the manipulator and the rest of the modules are decided modules. In the method discussed above, there is a concept called a "sub-workspace." The sub-workspace is a subset of the manipulator workspace. To produce the sub-workspace, it is only required to freeze the pending and decided ones and allow only the undecided modules to move. At each step of the searching process, the criterion for selecting the pending module configuration is the greater than the target point density in the related sub-workplace.

[2] Here, it is necessary to explain the term "error." The answer to the IK problem is a configuration of the manipulator. Now, when the answer is an exact one, the manipulator's end-frame exactly matches the target-frame. But if the answer is an approximate one, the end-frame will have a distance from the target-frame. This distance is considered an error, provided that there is an exact solution. More detailed descriptions of the error will be inserted later.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

The problem with this method is the large amount of computation needed to calculate the sub-workspace density, especially in three-dimensional cases.

Suthakorn and Chirikjian [33] proposed another single-module searching method to solve this problem. Their searching algorithm was simpler and faster. They used the concept of sub-workspace mean-frame instead of sub-workspace density. The mean-frame of a sub-workspace was something like the mass center for a set of discrete masses. They provided an effective, simple, and fast way to calculate this frame and assumed that it had the highest density and that the density of each frame decreased by increasing its distance from the mean-frame. Therefore, their criterion for selecting the configuration of the pending module was to minimize the distance between the mean-frame of the corresponding sub-workspace and the target-frame. This method was very desirable in terms of CPU time and memory required, but its error was large.

Some investigators [28, 34] utilized genetic algorithms (GAs) to solve the IK problem of DAHRMs. In their method, the population size and the number of generations should be increased in order to obtain an acceptable precision, and with this, the CPU time will increase improperly. Kim et al. [29] proposed a continuous variable optimization method to solve the IK problem. They use a special cost function in order to lead the answers to the discrete values. This method was desirable in terms of CPU time and memory required, but the closeness of the results to discrete values was a challenging issue and depended on the subtle definition of the cost function.

Other meta-heuristic search methods, such as simulated annealing (SA) and particle swarm optimization (PSO), can also be used to solve the IK problem of DAHRMs. But as presented in the numerical results section, their results are weaker than the method presented in this article. In meta-heuristic search methods, two issues are important: exploration and exploitation. Exploration refers to the ability of the search to explore new and diverse regions of the solution space, while exploitation refers to the ability of the search to exploit the best solutions found so far and improve them locally. The exploration is done well, but the exploitation is not done well due to the lack of a suitable method to define neighbors in a controllable neighborhood radius. It causes a waste of time and inaccuracy. Anyway, solving this problem can lead to effective methods to solve the IK problem of DAHRMs. This issue can be investigated by researchers in the future.

Motahari et al. [35–38] proposed two new ideas in their search method: 1) Two-by-two searching or two-module searching, which means that two modules were configured in each step of the searching instead of one. These two were named “pending pair modules.” These two did not need to be adjacent. 2) Iteration, which means that when all modules are configured, repeating the search steps in a new order can improve the results and reduce errors. Their method did not necessarily find the exact solution, but it resulted in less error than the previous methods, and at the same time, it had a satisfactory CPU time. But the question that arises here is whether increasing the number of pending modules will improve the results. In other words, why not do three- or more-module searching instead of two-module searching? In this article, a generalized algorithm called “multi-module search (M-MS)” is presented to solve the IK problem of DAHRMs. Also, the effect of increasing the number of pending modules on CPU time and errors will be investigated.

In the M-MS method, the manipulator is initially configured randomly. Then, in each step, a certain number of modules (Npen) are randomly selected from all the modules of the manipulator as the pending modules. The search space in each step will contain all possible combinations of the configuration of the pending modules, and the best of them will finally be selected as the new configuration of these modules. This step is repeated by selecting the other pending modules randomly to reduce the error. The novelty of this method is the use of several modules instead of two in each step of the search. With this, a generalized method with the desired number of pending modules is presented.

The remainder of the article is organized as follows: in Section 2, the problem is stated clearly; after that, in Section 3, the proposed algorithm for solving the problem is described. Next, in Section 4, numerical results are presented. Finally, the conclusions are presented in Section 5.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

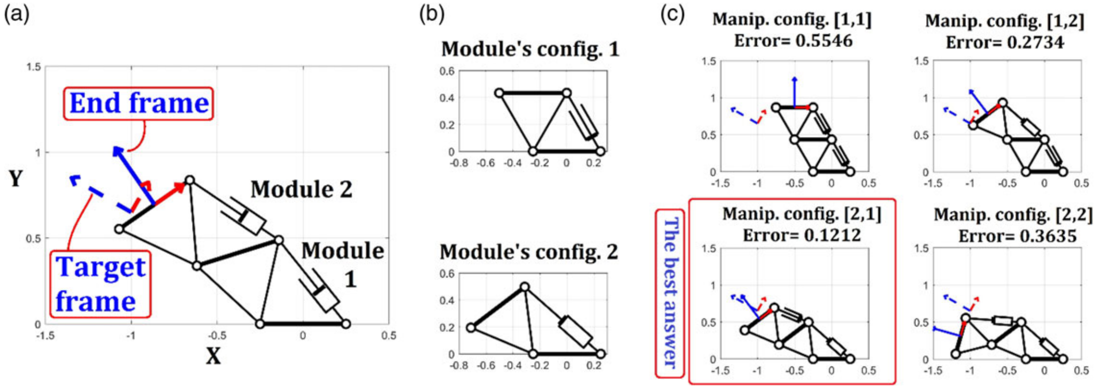


Figure 4. *An example of the IK problem of DAHRMs.*

2. Statement of the problem

Consider a prescribed frame as the target-frame (F_{tar}) and the end-frame of the manipulator (F_{end}). The end-frame location and orientation are determined if and only if the configuration of the manipulator ($\mathbb{C}$) is determined. To do this, all modules should be configured ($\mathbb{C}$= [c_1 c_2 . . . c_{Nmod}]). The IK problem of a DAHRM that contains Nmod modules, each with Ncon configurations, can be described as a discrete variable optimization problem as below:

$$\underset{\mathbb{C}}{\textbf{Minimize}}\,(\textbf{Error}(\mathbb{C}) = \textbf{Distance}\,(\textbf{F}_{\textbf{end}}(\mathbb{C}), \textbf{F}_{\textbf{tar}}))$$

where
$\mathbb{C} = [c_1\ c_2\ \ldots\ c_{Nmod}]$
subjected to
$c_i \in \{1, 2, \ldots, Ncon\}$ for $i = 1, 2, \ldots, Nmod$

In the above expression, Error, which equals the distance between F_{end} and F_{tar}, is considered an objective function that is a function of the manipulator configuration ($\mathbb{C}$). $\mathbb{C}$ can be selected from a set of all possible configurations of the manipulator. In other words, the modules should be configured so that the end-frame is as close to the target-frame as possible.

Figure 4 is presented for more familiarity with the IK problem of DAHRMs. Figure 4-a shows a VGT manipulator consisting of two modules (Nmod = 2) that must reach the target-frame (F_{tar}) represented by the dashed-line. Each module has a binary prismatic actuator, so it has two configurations (Ncon = 2), as shown in Fig. 4-b. By combining the possible configurations of the modules, the manipulator will have $Ncon^{Nmod} = 2^2 = 4$ configurations, as shown in Fig. 4-c. As can be seen in Fig. 4-c, among all configurations, the end-frame (F_{end}) of configuration No. 21 ($\mathbb{C}$ = [2 1]) is closer to the target-frame (F_{tar}). Therefore, this configuration will be the best answer to the problem.

The distance of the manipulator's end-frame from the target-frame is considered the error of the answer to the IK problem. Here, the distance is not just the distance between the two origins; the difference in the direction of the two frames should also be considered. Formulas for calculating the distance between two frames are presented in the appendix. According to the definition of the error, even the best answer may have some errors, as can be seen in Fig. 4-c. But all the sample problems solved in the numerical results section are chosen to have an exact solution (zero error) to make sure the error in the answer is due to the solution method.

As mentioned in the introduction section, ordinary methods used for solving the IK problem of continuous manipulators are not usable for DAHRMs because they lead to big errors. So, solving this problem requires a special method. On the other hand, due to the large number of the manipulator configurations, it is not feasible to use brute-force searching. Among the methods presented in the articles, the

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

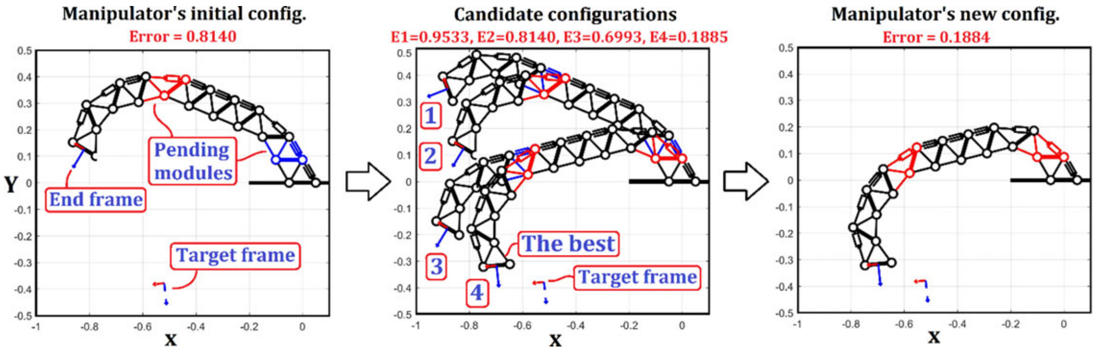


Figure 5. *The process of selecting a new configuration for a ten-module VGT manipulator using two randomly selected pending modules. This process can be repeated for the new configuration to reduce the error.*

two-by-two searching method has the best results in terms of error and CPU time. Two ideas have been used in this method: two-module searching and iteration. In the two-by-two searching method, there are two pending modules in each step of the solution, which are chosen randomly among all manipulator modules. Iterating this process reduces the error. The question that arises here is: Does increasing the number of pending modules lead to better results? To answer this question, a comprehensive method with an arbitrary number of pending modules is designed based on the two-by-two searching method, which is named “multi-module search (M-MS).” Step-by-step explanation of this method is as follows:

Step 1: The manipulator is configured arbitrarily.

Step 2: A certain number of modules (Npen) are arbitrarily selected from among all manipulator modules. These modules are known as pending modules.[3]

Step 3: In this step, a new configuration for the pending modules is selected to reduce the error. The search space for this selection includes all the configuration combinations of the pending modules. The selection criterion is the error value. For example, if the number of pending modules is two (Npen = 2), and each module has eight configurations (Ncon = 8), the search space in this step contains $\mathrm{Ncon}^{\mathrm{Npen}} = 8^2 = 64$ configurations. By changing the configuration of the pending modules, the configuration of the manipulator will also change. This new configuration, called the candid configuration, has a different error. All possible configuration combinations for the pending modules should be tested, and the related error should be calculated. Then, from among the candidate configurations, the configuration that has the least error is considered the new configuration of the manipulator.[4]

Step 4: If the error value is greater than a predetermined threshold, then go back to step 2 and repeat the steps above. Otherwise, terminate the execution of the steps.

Step 3 is illustrated in Fig. 5 for a ten-module VGT manipulator using two randomly selected pending modules. In this figure (as in Fig. 4), each module contains a binary prismatic actuator. Therefore, in each iteration, there will be $\mathrm{Ncon}^{\mathrm{Npen}} = 2^2 = 4$ candidate configurations.

In the second step, modules are not required to be selected in a particular order or adjacently. Numerical results show that if the initial configuration in Step 1 is chosen based on the “middle

[3] The value of Npen remains constant throughout the solution process. For example, in a 3-module search (3-MS method), Npen is equal to three.

[4] In the third step, the previous configuration is one of the candidate configurations. In this step, a better configuration may not be found, in which case the output of this step will be the same as the previous configuration and the error value will remain unchanged.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

Table I. *Time complexity of brute-force, two-by-two search, and M-MS methods with respect to the size of input parameters, using big O.*

Inputs/methods*	No. of manipulator modules (Nmod)	No. of module configs (Ncon)	No. of pending modules (Npen)	No. of iterations (Nitr)
Brute-force	$\mathcal{O}(a^{Nmod})$ Exponential	$\mathcal{O}(Ncon^{b})$ Polynomial	–	–
Two-by-two	$\mathcal{O}(Nmod)$ Linear	$\mathcal{O}(Ncon^{2})$ Polynomial	–	$\mathcal{O}(Nitr)$ Linear
M-MS	$\mathcal{O}(Nmod)$ Linear	$\mathcal{O}(Ncon^{c})$ Polynomial	$\mathcal{O}(d^{Npen})$ Exponential	$\mathcal{O}(Nitr)$ Linear

*a, b, c, and d are some constants: Ncon, Nmod, Npen, and Ncon, respectively.

actuation[5]" (as is done in the two-by-two searching method), it will lead to slightly better results than the random configuration. But the "middle configuration" is an unrealistic configuration, and all these configurations must be replaced with real configurations in the iteration steps. As a result, the algorithm will be a little more complicated. For this reason, this improvement is omitted here, and a random configuration is used in Step 1.

The search space of the proposed method, which contains all the candidate configurations, is far smaller than that of the brute-force searching method. The search space in the brute-force searching method includes $Ncon^{Nmod}$ configurations. Now, if the number of iterations is Nitr, then the search space of the M-MS method will contain Nitr × $Ncon^{Npen}$ configurations. Therefore, the search space of the brute-force approach would grow exponentially as the number of modules increased, but in the multi-module method, the search space is not dependent on the number of manipulator modules (Nmod).

The number of cost evaluations in the M-MS method will be the same (Nitr × $Ncon^{Npen}$). The CPU time will be proportional to this value. Accordingly, the time complexity of brute-force, two-by-two search and M-MS methods with respect to the size of input parameters is presented in Table I using big O. As can be seen in Table I, the M-MS method has exponential time with respect to the size of pending modules. This means that with the increase in Npen, the CPU time grows exponentially in the M-MS method. Npen in the two-by-two search method equals two and is not changed. Nmod in brute force has an exponential effect, but in the two-by-two and M-MS methods, its effect is linear. Nmod in these two methods affects the time of evaluating the cost function because the transformation matrix of the modules must be multiplied together to calculate the transformation matrix of the manipulator.

To make the described method clearer, the related algorithm is presented below:

```
% M-MS algorithm:
Input( Nmod, Npen, Ncon )
    % Nmod is the number of modules in the manipulator.
    % Npen is the number of pending modules in each step of iteration.
    % Ncon is the number of configurations of the modules[6].
Input( { $T_1$, $T_2$, . . ., $T_{Ncon}$ }, $T_{tar}$ )
    % $T_i$ is the homogeneous transformation matrix of the module in its i'th configuration ( $i = 1$,
      2, . . ., Ncon ).
    % $T_{tar}$ represents the transformation matrix of the target- frame.
```

[5] Middle actuation, which leads to the middle configuration, is produced by actuating all actuators in an average of all discrete amounts. For example, for a binary cylinder, the middle actuation means that the cylinder is stretched to half its stroke.

[6] It is assumed here that all the modules of the manipulator are the same.

```
Input($\varepsilon$)
    % $\varepsilon$ is a threshold for allowable error.
For $i = 1$ : Nmod
$c_i$ = randi(Ncon) ;
        % $c_i$ represents the chosen configuration of the i'th module in this step of the solution. At first, it is chosen randomly, but it may be changed later in the iteration steps.
        % randi(n) is a function that returns a random integer on the interval 1 to n.
End
$[A_1, A_2, \ldots, A_{Nmod}] = [T_{c_1}, T_{c_2}, \ldots, T_{c_{Nmod}}]$ ;
        % $A_i$ is the homogeneous transformation matrix of the i'th module of the manipulator.
A_man = $A_1 A_2 \ldots A_{Nmod}$ ;
        % A_man is the transformation matrix of the manipulator, which represents the position and orientation of the end effector related to the base.
Error = distance( A_man, $T_{tar}$ ) ;
        % distance(A,B) is a function that calculates the distance between two frames A and B. The formula for this function is given in the appendix.
$n_{itr} = 0$ ;
        % $n_{itr}$ represents the number of iterations in this step of the solution.
While Error > $\varepsilon$
$n_{itr} = n_{itr} + 1$
    $[m_1, m_2, \ldots, m_{Npen}]$ = sort( randsample( Nmod, Npen ) ) ;
        % $m_i$ is an integer between 1 and Nmod, which indicates which module of the manipulator is considered the i'th pending module.
        % randsample(A,B) is a function that chooses B non-repetitive integers randomly on the interval 1 to A.
        % sort(V) is a function that sorts the vector of numbers V.
    For $k_1 = 1$ : Ncon
        For $k_2 = 1$ : Ncon
⋮
            For $k_{Npen} = 1$ : Ncon
$c_{m1} = k_1$ ;
$c_{m2} = k_2$ ;
⋮
$c_{mNpen} = k_{Npen}$ ;
```

```
            [ $A_1, A_2, \ldots, A_{Nmod}$ ] = [ $T_{c_1}, T_{c_2}, \ldots, T_{c_{Nmod}}$ ] ;
            A_man = $A_1\ A_2 \ldots A_{Nmod}$ ;
$E_{k_1 k_2 \cdots k_{Npen}}$ = distance( A_man, $T_{tar}$ ) ;
                % $E_{i\,j\ldots k}$ is the error related to the choice: the first pending module is in the i'th configu-
                  ration, the second one is in the j'th configuration and so on until the last one, which is
                  in the k'th configuration.
          End
⋮
        End
      End
      [ $c_{m_1}, c_{m_2}, \ldots, c_{m_{Npen}}$ ] = index( min { $E_{11\ldots1}, E_{11\ldots2}, \ldots, E_{Ncon\,Ncon\ldots Ncon}$ } ) ;
        % min { A, B, . . ., C } is a function that returns the minimum number in the set { A, B, . . ., C }.
        % index($A_{ij\ldots k}$) is a function that returns the index of $A_{ij\ldots k}$ as a vector of integers [ i, j, . . ., k ].
      Error = min { $E_{11\ldots1}, E_{11\ldots2}, \ldots, E_{Ncon\,Ncon\ldots Ncon}$ } ;
        % Error is the distance related to the $\mathbb{C} = [c_1, c_2, \ldots, c_{Nmod}]$.
End
Display( $c_1, c_2, \ldots, c_{Nmod}$ ) ;
    % The main output of the algorithm, which is the solution of the IK problem, is a set of integers,
      each of which is on the interval 1 to $N_{con}$ and represents the configuration of each module of the
      manipulator.
Nitr = $n_{itr}$ ;
    % $N_{itr}$ is the number of iterations occurred in the solution process.
Display(Nitr)
Display(Error)
    % Here Error represents the error of the final solution.
```

Figure 6 illustrates the flowchart of the M-MS method.

3. Numerical results

In this section, the method of calculating error is first described, then a 2D and a 3D case study are introduced, and finally, numerical results are presented in tables and diagrams and analyzed.

3.1. Definition of error

As discussed in Section 1, the workspace of a discrete manipulator is a finite set of discrete frames in space. If the target-frame is selected from among the frames in this set, the IK problem will have an exact solution; otherwise, even the best solution will not fit the target-frame. For the error to be real, all the problems discussed in this section will be of the first type, which is called "real problems." For this purpose, the target-frame is derived by solving a forward kinematic problem with a random configuration. The error is defined as the distance between the target-frame and the manipulator end-frame, related

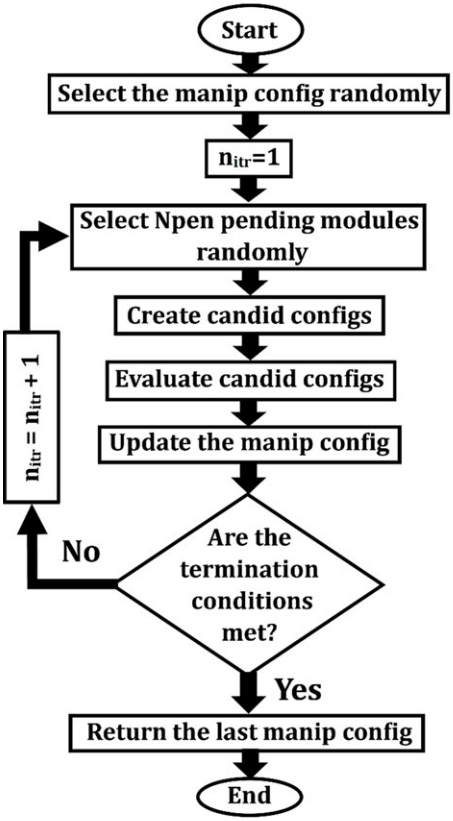

Start
Select the manip config randomly
n_{itr}=1
Select Npen pending modules randomly
Create candid configs
Evaluate candid configs
Update the manip config
Are the termination conditions met?
No
$n_{itr} = n_{itr} + 1$
Yes
Return the last manip config
End

Figure 6. *Flowchart of the M-MS method.*

to the configuration of the answer to the problem. The distance between the two frames is calculated using the definition given by Park [26] and described in Appendix A. In order to compare the error of various problems correctly, these errors should be dimensionless. To do this, the distance mentioned is divided by the shortest length of the manipulator. The length of the manipulator is the distance between the origin of the base-frame and the origin of the end-frame of the manipulator.

3.2. Introducing the case studies

3.2.1. 2D case study

A 20-module manipulator with similar VGT modules is considered the 2D case study. This manipulator is called the ''VGT manipulator.'' A VGT module introduced earlier in Section 1 and illustrated in Fig. 1. The lengths of the constant links (AB and CD in Fig. 1) are 1/20. As mentioned earlier, all actuators in this module are binary. The discrete amounts of actuation (which are the length of the links, AD, AC, and BC) are 1/20 and 1.5/20. Thus, the minimum and maximum lengths of the manipulator are $L_{min} = 1$ and $L_{max} = 1.5$, respectively. Readers are referred to Kim et al. [29] for the forward kinematics of this module.

3.2.2. 3D case study

A 20-module manipulator with similar 3-RPS modules is considered the 3D case study. This manipulator is called, in short, the ''3-RPS manipulator.'' A 3-RPS module, as shown in Fig. 7, is a spatial parallel robot with three binary prismatic actuators in its three legs, A_1B_1, A_2B_2, and A_3B_3. The baseplate ($A_1A_2A_3$) and the moving plate ($B_1B_2B_3$) are equilateral triangles. A_1, A_2, and A_3 are passive revolute

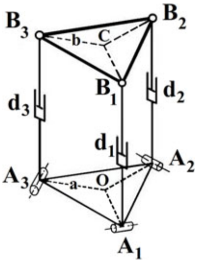


Figure 7. *A 3-RPS module.*

joints with rotation axes parallel to their opposite sides in triangle $A_1A_2A_3$. B_1, B_2, and B_3 are passive spherical joints. The distances between the center and corners of the two triangles are $a = b = 1/20$. The discrete amounts of actuation, which are the length of the legs, are 1/20 and 1.5/20. So, the minimum and maximum lengths of the manipulator are $L_{min} = 1$ and $L_{max} = 1.5$, respectively. Readers are referred to Kim et al. [29] for the forward kinematics of this module.

3.3. Numerical results

Figure 8a and b is presented to investigate the effect of the number of iterations (Nitr) on the error and the CPU time, respectively, for the 2D and 3D case studies. In both figures, the results for the 2-MS and 3-MS methods[7] are presented. The presented results are the average results of 100 real random problems. Of course, these 100 problems are considered common among all cases so that their results can be compared. All the calculations in this article were done with MATLAB software and an Intel 1.66 GHz processor.

According to these two figures, it is generally seen that the error decreases with the increase in Nitr, but the rate of this decrease decreases with the increase in Nitr until it remains constant after several iterations. On the other hand, the CPU time increases almost linearly with the increase in Nitr. The slope of this line is, as expected, higher in the 3-MS than in the 2-MS because, as mentioned in Section 3, the number of configurations examined in each step of the iteration in the three-module search is eight times that of the two-module search. In an equal Nitr, the error of the 3-MS is less than that of the 2-MS, but the CPU time is longer.

The numerical results presented in Fig. 8 showed that increasing the number of pending modules (Npen) reduces the error, but at the same time, the CPU time increases. Therefore, choosing Npen will be a challenge that should be considered.

Figure 9-a and b is presented for 2D and 3D case studies, respectively, to investigate the effect of increasing Npen on the efficiency of the solution method. For this purpose, the performance of 1-MS, 2-MS, 3-MS, and 4-MS methods has been compared. In each of the mentioned methods, the problem has been solved for various amounts of Nitr, and in each case, the CPU time and error have been evaluated. Then these points are connected by a curve in the CPU time error diagram. The presented results are the average results of 100 real random problems. Of course, these 100 problems are considered common among all cases so that their results can be compared.

According to these two figures, in general, it is not possible to say which method is better than the others. Rather, in each time interval, the best method may be different. In short times (less than 0.01 s), the error increases with the increase of Npen, but in longer times, this advantage is gradually reversed, for example, in times of more than 10 s, in both figures, the lowest error belongs to 4-MS, followed by 3-MS, 2-MS, and 1-MS methods, respectively. Therefore, in general, it is not possible to decide on the amount of Npen, and the maximum acceptable CPU time for the problem should be determined. A CPU time below 1 s is usually considered acceptable for online problems. In this case, it can be seen in both

[7] 2-MS and 3-MS methods are M-MS methods in which Npen is 2 and 3, respectively.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

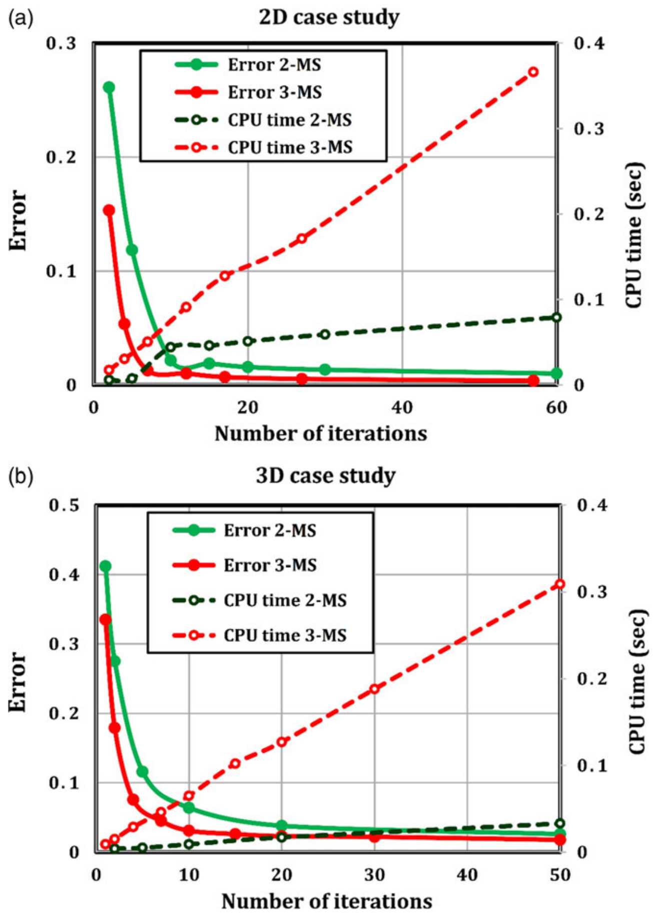


Figure 8. *Examining the effect of the number of iterations on error and CPU time of the 2D case study (a) and the 3D case study (b) in two 2-MS and 3-MS methods.*

figures that the 2-MS and 3-MS methods are superior to the 1-MS and 2-MS methods. In this case, the competition between the 2-MS and 3-MS methods is close, but the 3-MS is a little better. Note that both the horizontal and vertical axes are logarithmic.

Figure 10 is presented to validate the efficiency of the proposed algorithm. In this figure, the 2-MS and 3-MS methods are compared with nine meta-heuristic algorithms: the SA algorithm, GA, PSO, ant colony optimization (ACO), gray wolf optimizer (GWO), stochastic fractal search (SFS), whale optimization algorithm (WOA), Giza pyramid construction (GPC), and flying fox optimization (FFO) algorithm, all for the 2D case study (Fig. 10-a) and the 3D case study (Fig. 10-b).

In all methods, a common random problem has been solved one hundred times, and the average values of the results are presented. For the SA, GA, and PSO algorithms, the special functions for them, which are available in MATLAB, have been used. For the ACO, GWO, SFS, WOA, GPC, and FFO, the prepared codes in Refs [40–45] are used, respectively.

In the SA algorithm, for both 2D and 3D case studies, the initial temperature is 100 and the re-annealing interval is 100. By changing the maximum iteration value, different points of the corresponding curve are obtained in the figure. In GA, for both 2D and 3D case studies, the population size is 200, the elite count is 2, and the crossover fraction is 0.8. By changing the value of generation, different points of the corresponding curve are created in the figure. In PSO, for both 2D and 3D case studies, the value of particle swarm is 100, and different points of the curve are obtained from the change in the max-iterations value.

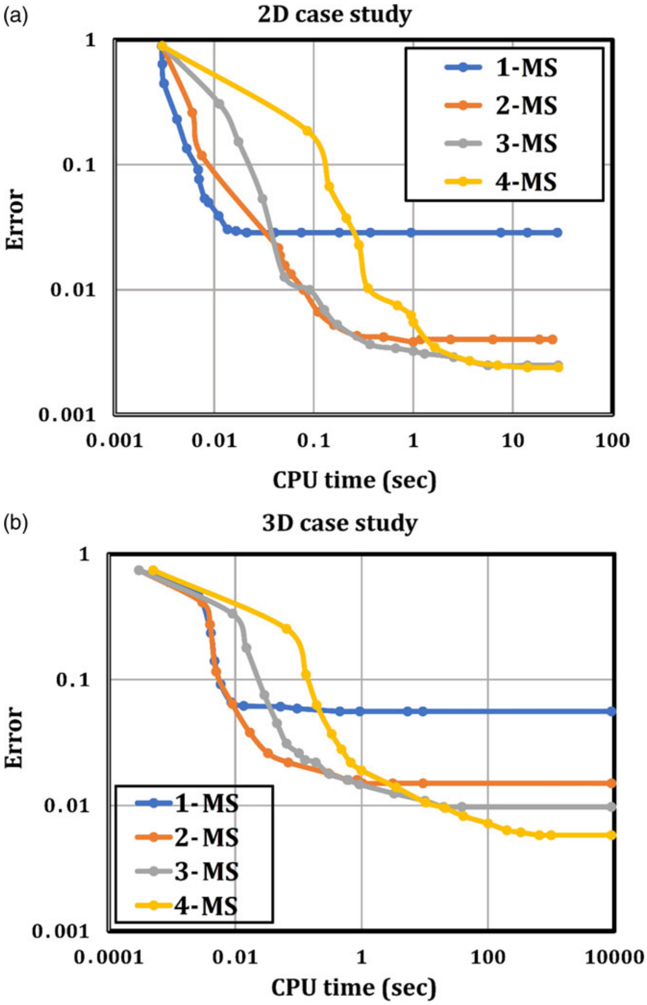


Figure 9. *Comparison of the performance of 1-MS, 2-MS, 3-MS, and 4-MS methods in terms of error and CPU time for the 2D case study (a) and the 3D case study (b).*

In ACO, for both 2D and 3D case studies, the number of iterations is 50, and different points of the curve are obtained from the change in the number of ants. In GWO, for both 2D and 3D case studies, the number of search agents is 50, and different points of the curve are obtained from the change in the max-iterations value. In SFS, for both 2D and 3D case studies, the number of start points (population size) is 10, the maximum number of diffusion (number of neighbors) is 10, and different points of the curve are obtained from the change in the maximum generation value.

In WOA, for both 2D and 3D case studies, the number of search agents (population size) is 100, and different points of the curve are obtained from the change in the maximum iterations value. In GPC, for both 2D and 3D case studies, maximum number of iterations (days of work) is 50, the value of substitution probability is 0.1, and different points of the curve are obtained from the change in the number of workers (population size). In FFO, for both 2D and 3D case studies, different points of the curve are obtained from the change in the maximum number of evaluating the cost function.[8]

[8] The population size in FFO is not an input variable. It is evaluated by the algorithm based on the number of variables and equals 25 for both 2D and 3D case studies.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

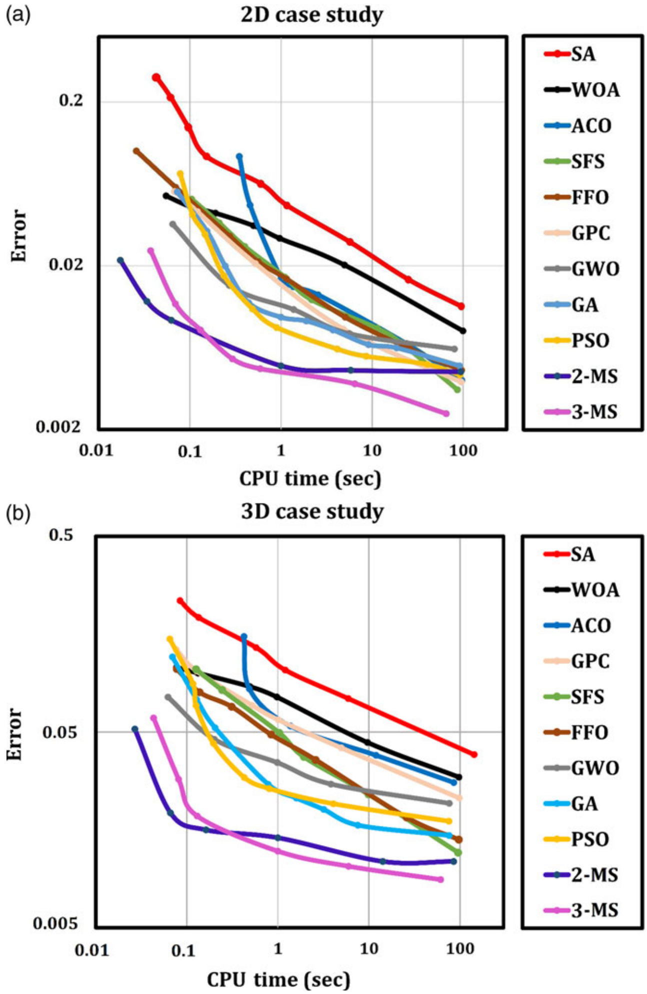
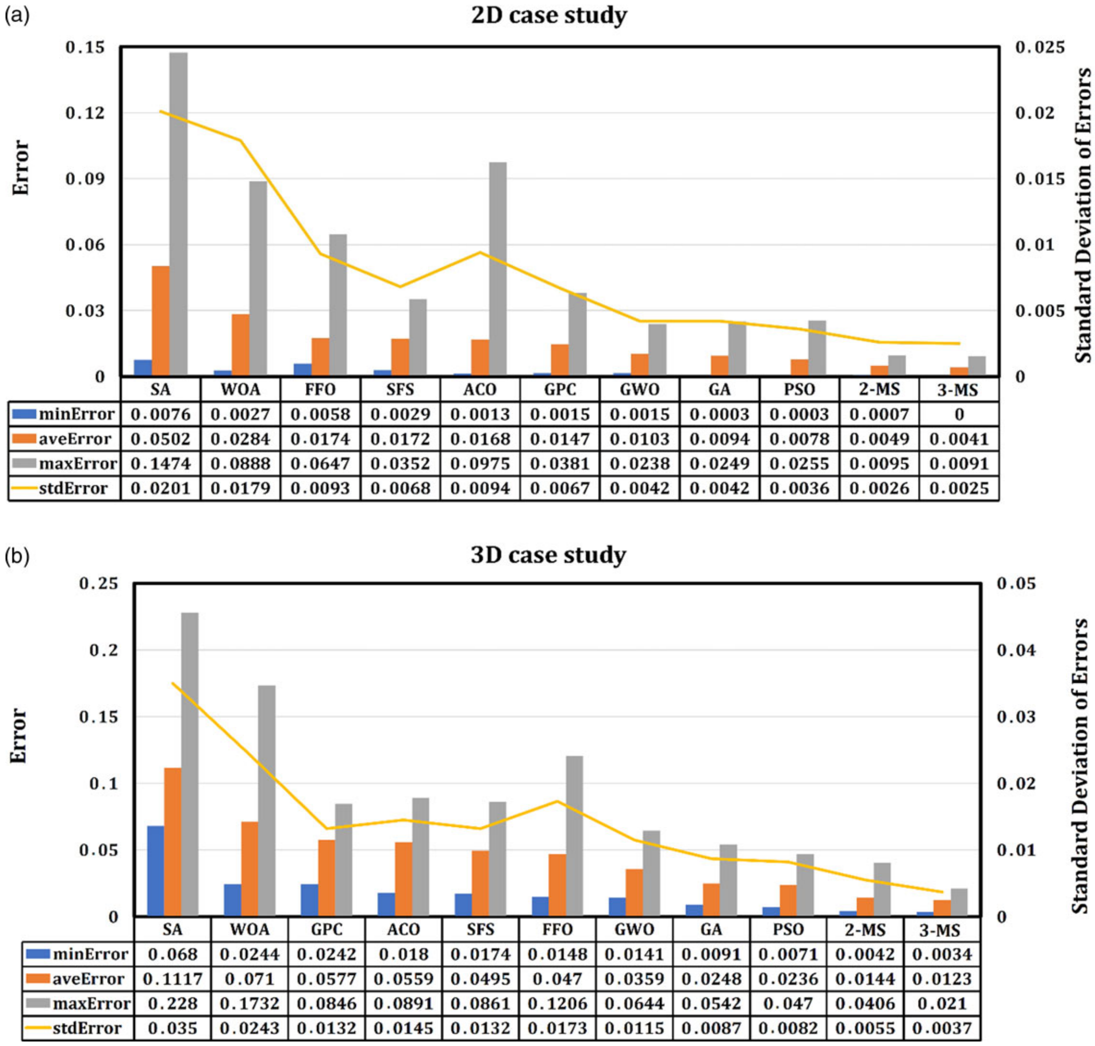


Figure 10. *Comparing the performance of 2-MS and 3-MS methods with nine meta-heuristic search algorithms: SA, WOA, ACO, GPC, SFS, FFO, GWO, GA, and PSO, for the 2D case study (a) and the 3D case study (b).*

The remaining parameters are set to default values in all meta-heuristic algorithms mentioned above. Readers can find more information on the nine mentioned meta-heuristic algorithms in references [46–54].

According to the figure, it can be said that in both 2D and 3D case studies, the 2-MS and 3-MS methods are superior to the nine meta-heuristic search methods. In the comparison between the two methods, 2-MS and 3-MS, as it was said about Fig. 9, it is not possible to comment in general, but it depends on the CPU time interval. In both 2D and 3D case studies, in CPU times less than 0.1 s, 2-MS is superior, but in CPU times close to 1 s and more, 3-MS performs better. It is not possible to form a general opinion about the nine meta-heuristic search algorithms, and the situation is different at different CPU time intervals. For example, for a CPU time of 1 s in the 2D case study, the following meta-heuristic algorithms work better in order: PSO, GA, GWO, GPC, ACO, SFS, FFO, WOA, and SA. In the 3D case study, the order is slightly different: PSO, GA, GWO, FFO, SFS, ACO, GPC, WOA, and SA. In both 2D and 3D case studies, it can be said that in the comparison between the nine meta-heuristic algorithms, algorithms PSO, GA, and GWO have the best performance and algorithms SA and WOA have the worst performance. Note that both the horizontal and vertical axes are logarithmic.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press



	SA	WOA	FFO	SFS	ACO	GPC	GWO	GA	PSO	2-MS	3-MS
minError	0.0076	0.0027	0.0058	0.0029	0.0013	0.0015	0.0015	0.0003	0.0003	0.0007	0
aveError	0.0502	0.0284	0.0174	0.0172	0.0168	0.0147	0.0103	0.0094	0.0078	0.0049	0.0041
maxError	0.1474	0.0888	0.0647	0.0352	0.0975	0.0381	0.0238	0.0249	0.0255	0.0095	0.0091
stdError	0.0201	0.0179	0.0093	0.0068	0.0094	0.0067	0.0042	0.0042	0.0036	0.0026	0.0025

	SA	WOA	GPC	ACO	SFS	FFO	GWO	GA	PSO	2-MS	3-MS
minError	0.068	0.0244	0.0242	0.018	0.0174	0.0148	0.0141	0.0091	0.0071	0.0042	0.0034
aveError	0.1117	0.071	0.0577	0.0559	0.0495	0.047	0.0359	0.0248	0.0236	0.0144	0.0123
maxError	0.228	0.1732	0.0846	0.0891	0.0861	0.1206	0.0644	0.0542	0.047	0.0406	0.021
stdError	0.035	0.0243	0.0132	0.0145	0.0132	0.0173	0.0115	0.0087	0.0082	0.0055	0.0037

Figure 11. *Comparison of the error of 2-MS and 3-MS methods with nine meta-heuristic search algorithms: SA, WOA, ACO, GPC, SFS, FFO, GWO, GA, and PSO, all in 1 s CPU time, for two case studies: 2D (a) and 3D (b).*

The proposed method and the nine presented meta-heuristic search algorithms are all nondeterministic search algorithms, which give different results from different runs. For this reason, in the previous diagrams, the problems were solved 100 times, and the average values of the results were presented.

An algorithm will be more robust if its answer changes less during different executions for the same input. To validate the robustness of the proposed method, a common random real problem has been solved one hundred times in both 2D and 3D case studies in 1 s, and the standard deviation of the errors of the 2-MS and 3-MS methods has been compared with the nine meta-heuristic algorithms in Fig. 11. Figure 11-a and b is related to the 2D and 3D case studies, respectively.

The minimum, average, maximum, and standard deviation (std) of the errors have been obtained for each method. The smaller the standard deviation of the error, the more robust the method will be. The standard deviation of the error is represented by an orange curve. Based on this, the order of robustness of the algorithms in the 2D case study, from more to less, is as follows: 3-MS, 2-MS, PSO, GA, GWO, GPC, SFS, FFO, ACO, WOA, and SA. The 3D case study follows the same order, but the ACO and

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

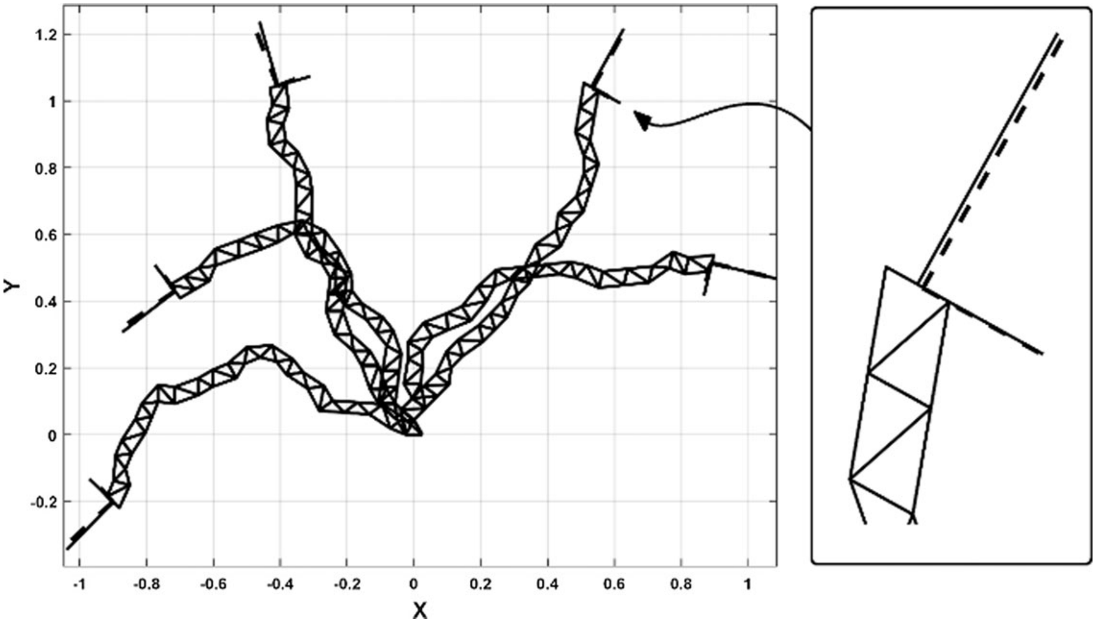


Figure 12. *The solution configurations of five real random IK problems for the 2D case study, which were solved using the 3-MS method with 50 iterations. The end frames are represented by solid lines, while the target frames are represented by dashed lines.*

FFO are swapped out. The order of the bar graphs is based on the average error. In both 2D and 3D case studies, methods 3-MS and 2-MS are more robust than the nine meta-heuristic algorithms. In the comparison between nine meta-heuristic algorithms in both 2D and 3D case studies, algorithms PSO, GA, and GWO are the most robust, and algorithms SA and WOA are the least robust (similar to the average error).

Figures 12 and 13 are presented in order to provide the reader with a clear understanding of the proposed method's error rate. Each of these two figures shows the solution configuration for five real random problems. Figures 12 and 13 are related to 2D and 3D case studies, respectively. The method used to solve the problem in these two figures is the 3-MS method with 50 iterations. In these figures, the target frames are shown with the dashed lines, and the end frames are shown with the solid lines. The average error in the cases shown in Fig. 12 is 0.0055, and the average CPU time is 0.425 s. The average error in the cases shown in Fig. 13 is 0.0143, and the average CPU time is 0.375 s.

4. Conclusions

In this article, the M-MS method is presented to solve the IK problem of DAHRMs. Numerical tests were done on two 2D and 3D case studies. Numerical results indicated that increasing the number of pending modules would lead to a decrease in errors but an increase in CPU time. Therefore, it will be a challenge to pick the best option for the number of pending modules. The answer to this question is dependent on the maximum acceptable CPU time, as revealed by numerical investigations in both 2D and 3D case studies. For example, for a CPU time of less than 1 s in both case studies, the 3-MS will be the best choice. To validate the effectiveness of the proposed method, it was compared with nine meta-heuristic search algorithms: SA, GA, PSO, ACO, GWO, SFS, WOA, GPC, and FFO, in terms of CPU time and error. The 2-MS and 3-MS methods were superior to other methods in both 2D and 3D case studies. By examining the error's standard deviation in both case studies, it was determined that the 2-MS and 3-MS methods were more robust than the others.

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

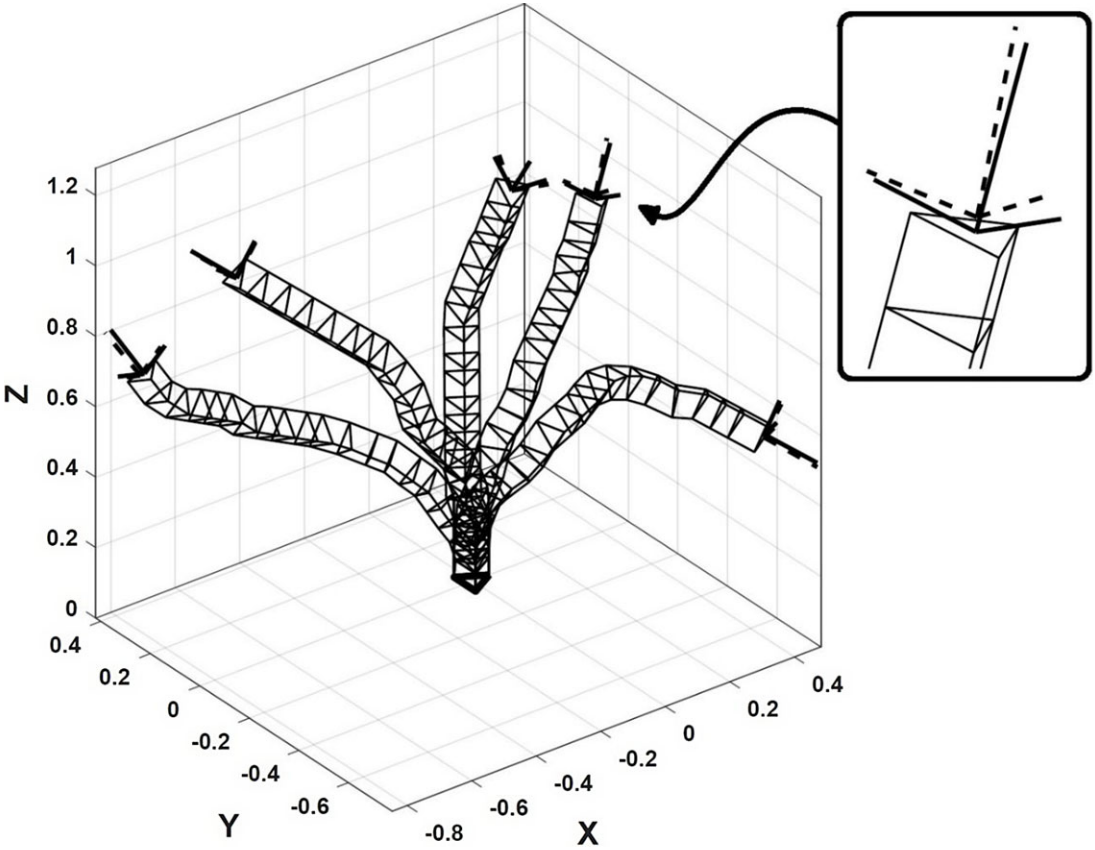


Figure 13. *The solution configurations of five real random IK problems for the 3D case study, which were solved using the 3-MS method with 50 iterations. The end frames are represented by solid lines, while the target frames are represented by dashed lines.*

Author contributions. All work related to this article was done by Alireza Motahari.

Financial support. This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

Competing interests. The authors declare no competing interests exist.

Ethical approval. Not applicable.

References

[1] D. L. Pieper, The Kinematics of Manipulators under Computer Control, Ph.D. Dissertation (Stanford University, 1968).

[2] G. S. Chirikjian, “A Binary Paradigm for Robotic Manipulators,” **In:** *IEEE Int. Conf. on Robotics and Automation, San Diego* (1994) pp. 3063–3070.

[3] Y. Chi, Y. Tang, H. Liu and J. Yin, “Leveraging monostable and bistable pre-curved bilayer actuators for high-performance multitask soft robots,” *Adv. Mater. Technol.* **5**(9), 2000370 (2020).

[4] Y. Lin, C. Zhang, W. Tang, Z. Jiao, J. Wang, W. Wang, Y. Zhong, P. Zhu, Y. Hu, H. Yang and J. Zou, “A bioinspired stress-response strategy for high-speed soft grippers,” *Adv. Sci.* **8**(21), 2102539 (2021).

[5] X. Wang, A. Khara and C. Chen, “A soft pneumatic bistable reinforced actuator bioinspired by Venus Flytrap with enhanced grasping capability,” *Bioinspir. Biomim.* **15**(5), 056017 (2020).

[6] M. Mungekar, L. Ma, W. Yan, V. Kackar, S. Shokrzadeh and M. K. Jawed, “Design of bistable soft deployable structures via a Kirigami-inspired planar fabrication approach,” *Adv. Mater. Technol.* **8**(16), 2300088 (2023).

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press

[7] X. Wang, H. Zhou, H. Kang, W. Au and C. Chen, "Bio-inspired soft bistable actuator with dual actuations," *Smart Mater. Struct.* **30**(12), 125001 (2021).

[8] J. McWilliams, Y. Yuan, J. Friedman and C. Sung. Push-on push-off: A compliant bistable gripper with mechanical sensing and actuation. *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, IEEE (2021) pp. 622–629.

[9] Y. Tang, Y. Chi, J. Sun, T.-H. Huang, O. H. Maghsoudi, A. Spence, J. Zhao, H. Su and J. Yin, "Leveraging elastic instabilities for amplified performance: Spine-inspired high-speed and high-force soft robots," *Sci. Adv.* **6**(19), eaaz6912 (2020).

[10] R. Masana, S. Khazaaleh, H. Alhussein, R. S. Crespo and M. F. Daqaq, "An origami-inspired dynamically actuated binary switch," *Appl. Phys. Lett.* **117**(8), 081901 (2020).

[11] G. Miron, A. Girard, J.-S. Plante and M. Lepage, "Design and manufacturing of embedded air-muscles for a magnetic resonance imaging compatible prostate cancer binary manipulator," *J. Mech. Design* **135**(1), 011003 (2013).

[12] E. Tzorakoleftherakis, A. Mavrommati and A. Tzes, "Design and implementation of a binary redundant manipulator with cascaded modules," *J. Mech. Robot.* **8**(1), 011002 (2016).

[13] S. Tappe, P. Boyraz, H. Korz and T. Ortmaier, "Design, Production and Integration of a Shape Sensing Robotic Sleeve for a Hyper-Redundant, Binary Actuated Robot," *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE (2018) pp. 298–303.

[14] L. Cappello, M. Xiloyannis, B. K. Dinh, A. Pirrera, F. Mattioni and L. Masia, "Multistable series elastic actuators: Design and control," *Robot. Auton. Syst.* **118**, 167–178 (2019).

[15] D. Dragone, L. Randazzini, A. Capace, F. Nesci, C. Cosentino, F. Amato, E. De Momi, R. Colao, L. Masia and A. Merola, "Design, Computational Modelling and Experimental Characterization of Bistable Hybrid Soft Actuators for a Controllable-Compliance Joint of an Exoskeleton Rehabilitation Robot," **In:** *Actuators*. vol. **11** (MDPI, 2022) pp. 32.

[16] S. J. Lee, A. M. Bilton and S. Dubowsky, "On the kinematics of solar mirrors using massively parallel binary actuation," *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, **44106** (2010) pp. 1177–1186.

[17] S. M. Rudolph, M. W. Nurnberger, H. F. Alqadah and J. P. Bobak, "Ultra-Low-Loss, Binary-State Elements for a Mechanically Actuated Reconfigurable Reflectarray," *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, IEEE (2019) pp. 1951–1952,

[18] Q. Ze, S. Wu, J. Dai, S. Leanza, G. Ikeda, P. C. Yang, G. Iaccarino and R. R. Zhao, "Spinning-enabled wireless amphibious origami millirobot," *Nat. Commun.* **13**(1), 3118 (2022).

[19] A. Mohand-Ousaid, I. Bouhadda, G. Bourbon, P. L. Moal, Y. Haddab and P. Lutz, "Compact digital microrobot based on multistable modules," *IEEE Robot. Autom. Lett.* **6**(2), 1926–1933 (2021).

[20] H. Hussein, I. Bouhadda, A. Mohand-Ousaid, G. Bourbon, P. L. Moal, Y. Haddab and P. Lutz, "Design and fabrication of novel discrete actuators for microrobotic tasks," *Sens. Actuat. A: Phys.* **271**, 373–382 (2018).

[21] L. Zhou and H. Xie, "A Novel Out-of-Plane Electrothermal Bistable Microactuator," *2019 20th International Conference on Solid-State Sensors, Actuators and Microsystems & Eurosensors XXXIII (TRANSDUCERS & EUROSENSORS XXXIII)*, IEEE (2019) pp. 1953–1956.

[22] D. K. Patel, X. Huang, Y. Luo, M. K. J. Mrunmayi Mungekar, L. Yao and C. Majidi, "Highly dynamic bistable soft actuator for reconfigurable multimodal soft robots," *Adv. Mater. Technol.* **8**(2), 2201259 (2023).

[23] Y. Chi, Y. Li, Y. Zhao, Y. Hong, Y. Tang and J. Yin, "Bistable and multistable actuators for soft robots: Structures, materials, and functionalities," *Adv. Mater.* **34**(19), 2110384 (2022).

[24] L. S. Novelino, Q. Ze, S. Wu, G. H. Paulino and R. Zhao, "Untethered control of functional origami microrobots with distributed actuation," *Proc. Natl. Acad. Sci.* **117**(39), 24096–24101 (2020).

[25] H. Son, Y. Park, Y. Na and C. K. Yoon, "4D multiscale origami soft robots: A review," *Polymers-BASEL* **14**(19), 4235 (2022).

[26] I. R. Ebert-Uphoff. *On the Development of Discretely-Actuated Hybrid-Serial-Parallel Manipulators. Order No. 9821118* (The Johns Hopkins University, Maryland, 1998).

[27] J. Suthakorn. *Paradigms for Service Robotics*, Order No. 3080775 (The Johns Hopkins University, Maryland, 2003).

[28] V. A. Sujan, M. D. Lichter and S. Dubowsky, "Lightweight Hyper-Redundant Binary Elements for Planetary Exploration Robots," *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*. vol. **2**, IEEE, (2001) pp. 1273–1278.

[29] Y. Y. Kim, G.-W. Jang and S. J. Nam, "Inverse kinematics of binary manipulators by using the continuous-variable-based optimization method," *IEEE Trans. Robot.* **22**(1), 33–42 (2006).

[30] G. S. Chirikjian and D. S. Lees, "Inverse kinematics of binary manipulators with applications to service robotics," *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, IEEE, **3**, (1995) pp. 65–71.

[31] I. Ebert-uphoff and G. S. Chirikjian, "Efficient workspace generation for binary manipulators with many actuators," *J. Robot. Syst.* **12**(6), 383–400 (1995).

[32] I. Ebert-Uphoff and G. S. Chirikjian, "Inverse Kinematics of Discretely Actuated Hyper-Redundant Manipulators Using Workspace Densities," *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, **1**. (1996) pp. 139–145.

[33] J. Suthakorn and G. S. Chirikjian, "A new inverse kinematics algorithm for binary manipulators with many actuators," *Adv. Robot.* **15**(2), 225–244 (2001).

[34] S. dos Santos, M. A. M. Felipe and V. A. Sujan, "Inverse kinematics of a binary flexible manipulator using genetic algorithms," *18th International Congress of Mechanical Engineering*, November 6-11, 2005, ABCM, Ouro Preto, MG, 2005.

[35] A. Motahari, H. Zohoor and M. H. Korayem, "A new inverse kinematic algorithm for discretely actuated hyper-redundant manipulators," *Lat. Am. Appl. Res.* **43**(2), 161–168 (2013).
[36] A. Motahari, H. Zohoor and M. H. Korayem, "Discrete kinematic synthesis of discretely actuated hyper-redundant manipulators," *Robotica* **31**(7), 1073–1084 (2013).
[37] A. Motahari, H. Zohoor and M. H. Korayem, "A new obstacle avoidance method for discretely actuated hyper-redundant manipulators," *Sci. Iran.* **19**(4), 1081–1091 (2012).
[38] A. Motahari, H. Zohoor and M. H. Korayem, "A new motion planning method for discretely actuated hyper-redundant manipulators," *Robotica* **35**(1), 101–118 (2017).
[39] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *J. Mech. Des.* **117**, 1995, 48–54.
[40] M. Heris (2015). Ant Colony Optimization (ACO). https://www.mathworks.com/matlabcentral/fileexchange/52859-ant-colony-optimization-aco.
[41] S. Mirjalili (2018). Grey Wolf Optimizer (GWO), *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo
[42] H. Salimi (2014). Stochastic Fractal Search (SFS), *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/47565-stochastic-fractal-search-sfs.
[43] S. Mirjalili (2018). The whale optimization algorithm, *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/55667-the-whale-optimization-algorithm
[44] S. Harifi (2020). Giza Pyramids Construction (GPC) algorithm, *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/80467-giza-pyramids-construction-gpc-algorithm
[45] K. Zervoudakis (2023). Flying Foxes Optimization (Fuzzy self-tuning optimizer), *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/104810-flying-foxes-optimization-fuzzy-self-tuning-optimizer
[46] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, "Optimization by simulated annealing," *Science* **220**(4598), 671–680 (1983).
[47] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (MIT Press, Cambridge, 1992).
[48] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE, **4**, (1995) pp. 1942–1948.
[49] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Magaz.* **1**(4), 28–39 (2006).
[50] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.* **69**, 46–61 (2014).
[51] H. Salimi, "Stochastic fractal search: A powerful metaheuristic algorithm," *Knowl.-Based Syst.* **75**, 1–18 (2015).
[52] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.* **95**, 51–67 (2016).
[53] S. Harifi, J. Mohammadzadeh, M. Khalilian and S. Ebrahimnejad, "Giza Pyramids Construction: An ancient-inspired metaheuristic algorithm for optimization," *Evol. Intell.* **14**(4), 1743–1761 (2021).
[54] K. Zervoudakis and S. Tsafarakis, "A global optimizer inspired from the survival strategies of flying foxes," *Eng. Comput.* **39**, 1–34 (2022).

Appendix

Each frame can be defined by a homogeneous transformation matrix (g), which can be expressed as follows:

$$g = \begin{bmatrix} R & b \\ 0^T & 1 \end{bmatrix} \in SE(N) \tag{A1}$$

where $R \in SO(N)$ is the rotation matrix and $b \in R^N$ is the position vector. N for 2D cases is 2, and for 3D cases it is 3. The distance between two frames, which are illustrated by g_1 and g_2, can be expressed as follows:

$$D(g_1, g_2) = \sqrt{\|b_1 - b_2\|^2 + L^2 \left\|\log R_1{}^T R_2\right\|^2} \tag{A2}$$

where $\|\cdot\|$ is the Euclidean norm. L is a parameter mainly introduced to match the units of the squared terms. In this article, the value of $L = 0.1$ was used. Readers are referred to Park's paper [39] for more information.

Cite this article: A. Motahari (2024). "Solving the inverse kinematics problem of discretely actuated hyper-redundant manipulators using the multi-module searching method", Robotica **42**, 911–930. https://doi.org/10.1017/S0263574723001893

https://doi.org/10.1017/S0263574723001893 Published online by Cambridge University Press