High-Dimensional Inverse Kinematics and Self-Reconfiguration Kinematic Control

Thomas Joseph Collins and Wei-Min Shen

Abstract This paper addresses two unique challenges for self-reconfigurable robots to perform dexterous locomotion and manipulation in difficult environments: high-dimensional inverse kinematics (HDIK) for > 100 degrees of freedom, and self-reconfiguration kinematic control (SRKC) where the workspace targets at which connectors are to meet for docking are not known a priori. These challenges go beyond the state-of-the-art because traditional manipulation techniques (e.g., Jacobian-based) may not be stable or scalable, and alternative approaches (e.g., genetic algorithms or neural networks) provide no guarantees of optimality or convergence. This paper proposes a new technique called Provably-convergent Swarm-based Inverse Kinematics (PSIK) that extends Branch and Bound Particle Swarm Optimization with a unique approach for dynamic target adaptation for self-reconfiguration. The PSIK algorithm can find globally optimal solutions for both HDIK and SRKC to any precision requirement (i.e., positive error tolerance) in finite or real-time for tree structures of self-reconfigurable robots. This algorithm is implemented and validated in high-fidelity, physics-based simulation using SuperBot as prototype modules. The results are very encouraging and provide feasible solutions for dextrous locomotion, manipulation, and self-reconfiguration.

1 Introduction

It has been a long-standing goal of self-reconfigurable robotics to create autonomous systems capable of dexterous locomotion and manipulation in difficult environments. Figure 1 shows an example application of self-assembly in space using self-reconfigurable robots. Such applications pose many technical challenges; this paper...
focuses on two that are unique for self-reconfigurable robots: high-dimensional inverse kinematics (HDIK) and self-reconfiguration kinematic control (SRKC).

The HDIK problem goes beyond the state-of-the-art because the degrees of freedom (DOF) are so high that traditional manipulation techniques – e.g., Jacobian-based methods, which have well-documented shortcomings such as numerical instabilities at singularities and poor scaling with the number of DOF – may become unstable or demand unrealistic computing time, making them poorly suited to trees of self-reconfigurable modules which often have many redundant DOF to provide the requisite manipulation dexterity. Alternative solution methods, e.g., numerical approximations, genetic algorithms, evolutionary algorithms, often work well in practice but provide no guarantees of global solution optimality, nor can they provably find solutions of sufficient quality (when solutions of a certain quality are known to exist). Such a guarantee is vital to ensuring safe (i.e., collision-free) operation.

The unique challenge of the SRKC problem is that a set of joint angles (joint configuration) must be found for all modules in the robotic system that precisely aligns two connectors for docking, but the target workspace poses for the connectors to meet are not known a priori and must be determined dynamically during reconfiguration. SRKC is crucial for robots that must change their configuration to achieve their tasks (e.g., an "octopus" robot changing the number/length of tentacles).

This paper presents an optimization algorithm for trees of self-reconfigurable robot modules that provably converges to globally optimal solutions to both problems (involving one or more end-effectors in the tree) in finite time given any positive error tolerance, called Provably-convergent Swarm-based Inverse Kinematics (PSIK). This PSIK algorithm extends the recent Branch and Bound Particle Swarm Optimization algorithm (BB-PSO) [28] with a unique approach to address the dynamic target problems of self-reconfiguration. This new algorithm requires only a forward kinematics model of the module(s) involved in the tree, making it applicable to any arbitrary tree structure of rigid body robotic modules. This algorithm is implemented with local message passing on a distributed system of SuperBot modules and validated in high-fidelity, real-time, physics-based simulation. Section II discusses related work. Section III provides background information on the branch and bound framework, Particle Swarm Optimization (PSO), and BB-PSO. Section IV details the proposed PSIK algorithm. Section V presents results with simulated SuperBot modules validating the algorithm and illustrating its usefulness in facilitating dextrous locomotion, manipulation, and self-reconfiguration.
2 Related Work

Traditional inverse kinematics methods, such as those based on the manipulator Jacobian, though widely and successfully used, are known to suffer from numerical issues around singularities and do not scale well with the number of DOF [5], making them impractical for HDIK. The limitations of Jacobian-based methods have led to a plethora of alternative solution strategies, including novel numerical techniques (e.g., [2]), those based on neural networks and fuzzy logic (e.g., [3, 17]), genetic algorithms (e.g., [20, 6]), probability theory (e.g., [7]). Heuristic optimization approaches such as those based on Particle Swarm Optimization (PSO) (e.g., [21, 22, 9]) and the Firefly Algorithm [23] have also been proposed but are primarily validated on low-DOF and planar manipulators and not designed to be used for general 6D (3 position DOF and 3 orientation DOF) HDIK problems. Several surveys of IK techniques exist, such as [1]. The vast majority of proposed methods are local solutions, meaning they do not provably converge to a solution and/or are susceptible to local minima. It is also unclear how existing approaches could be used to solve the SRKC problem, in which target poses are not known a priori. In [27], Shen et. al presented a distributed solution the inverse kinematics problem for docking in 3D (two position DOFs and one orientation DOF). However, this approach is not applicable to general 6D (3 position DOFs and 3 orientation DOFs) HDIK.

A number of modular and self-reconfigurable robot hardware systems have been developed, including [18, 28, 15, 8, 24]. In many of these systems, distributed algorithms have been developed for various tasks, including locomotion, manipulation, forward and inverse kinematics, and self-reconfiguration. These algorithms, in particular those related to inverse kinematics and SRKC, tend to be intimately tied to the hardware in question and not broadly applicable as they are primarily designed to validate the design of the hardware. Modular and self-reconfigurable manipulation has been looked at largely from a control perspective (e.g., [12]) and a hardware perspective (e.g., [32]). In [31, 30], distributed self-assembly using robotic manipulators made of lattice-based self-reconfigurable robots (Shady3D) and passive structures (e.g., beams) was demonstrated. In [4], the cooperative locomotion (on a discretized 2D grid) and manipulation (transport) of passive components by multiple self-reconfigurable serial robot manipulators was demonstrated, but SRKC was not considered. In [19], the autonomous locomotion, manipulation, and self-reconfiguration of 3D trusses was demonstrated. The results in [31, 30] and truss reconfiguration in [19] could be considered a form of SRKC for lattice-based robots, but it would not be applicable to robots without a lattice or truss. As will be shown, PSIK is capable of facilitating locomotion, manipulation, and self-reconfiguration in tree structures of self-reconfigurable robots on a continuous ground surface without a discretized grid or lattice. General and optimal solutions to planning self-reconfiguration for certain classes of self-reconfigurable robots have been developed (e.g., [14, 11]). Most plan only the connections and disconnections necessary to self-reconfigure, not the joint displacements necessary to line up connectors kinematically (as in SRKC). Those that do consider kinematics usually assume joints can take on only a few discretized angle values, greatly simplifying SRKC.
3 Algorithm Background

3.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [10] is a swarm-based optimization algorithm that has been shown to be effective in solving difficult optimization problems in many diverse domains. The basic idea is that a swarm of \( m \) particles, each with \( n \) dimensions, performs a randomized search in the space of possible \( n \)-dimensional solutions while communicating with other particles and maintaining its own history. Each particle \( i \) is a point \( x_i \) in the given search space \( S^m \subseteq \mathbb{R}^n \) with velocity \( v_i \) and has an associated error value equal to \( F(x_i) \), where \( F \) is the function to be minimized. Particles move around in this search space randomly but particle movement is biased toward a random weighted average of the direction to the best (lowest \( F(x_i) \)) position achieved by any particle in the swarm \( g \) (social component) and the direction to the best position achieved by each particle individually, \( p_i \) (history component). This focuses random searches on areas of the search space where a global optimum is expected to be. \( g \) and \( p_i \) are updated at each iteration. This randomized searching process continues until a certain fitness threshold \( h \) is reached by some particle in the swarm (i.e., a low enough value of \( F \) is found) or a maximum number of iterations \( N \) is performed. At termination, the global best position found (\( g \)) is returned.

3.2 Branch and Bound Framework

The branch and bound framework refers to a class of global optimization algorithms in which a finite search space is recursively and exhaustively searched until a global optimum (minimum, in this case) of some function is found. Branch and bound algorithms work by partitioning a search space recursively according to some branching rule. At each step, a list of currently active partition elements (those portions of the search space that may still contain a solution) is kept. Also at each step, a search strategy determines which partition element to further refine (i.e., further partition), leading to more and more active partition elements in this list as the algorithm iterates. A bounding rule keeps track of, for each active partition element, an upper (worst known solution error) and lower (best possible solution error) bound (\( \alpha_i \) and \( \beta_i \), respectively) on possible solutions in that partition element. After each branching and bounding operation, the overall upper and lower bounds for the entire search space across all active partition elements \( \alpha = \min \alpha_i \) and \( \beta = \min \beta_i \) are updated. When \( \alpha - \beta = 0 \), the global minimum has been found. For more details on the algorithm, please see [29, 13]. Note that to ensure convergence in a finite amount of time, an error tolerance \( \varepsilon = \alpha - \beta \in \mathbb{R}^+ \) must be selected and the global minimum of \( F \), which is, of course, the best choice for \( \beta_i \) at each partition element, must be known [29] (which, as is shown, is the case for the problems considered).
3.3 Branch and Bound Particle Swarm Optimization

Simply put, Branch and Bound Particle Swarm Optimization (BB-PSO) [29] is an embedding of PSO within the branch and bound framework. Assuming the global minimum of the function \( F(S^n) \) to be minimized is known (where \( S^n \) is a hypercube in \( R^n \) such as, in this work, the continuous space of joint angles subject to joint limits), PSO is used as a metaheuristic to estimate the upper bound \( \alpha_i \) of each partition. The only change required to the PSO algorithm is that each swarm must search only in the bounds of the partition element hypercube in which it spawned. The known global minimum value of \( F \) is the \( \beta_i \) for each partition. The convergence of the BB-PSO algorithm to the global minimum of \( F \) and the convergence in a finite amount of time given positive error tolerance \( \varepsilon \) is theoretically proved in [29].

4 The PSIK Algorithm for HDIK and SRKC

The BB-PSO algorithm facilitates the desired global convergence and optimality claims, but its application to the HDIK and SRKC problems is not straightforward. Previous uses of other PSO variants as IK solvers in the literature focused primarily on planar, low-DOF manipulation problems, and did not consider collision and self-collision avoidance. The PSIK algorithm extends these previous works with general-purpose objective functions for 6D (3 position and 3 orientation DOF) HDIK and SRKC problems applicable to any modular or self-reconfigurable robotic tree with known forward kinematics. Though PSIK makes use of the BB-PSO framework to minimize these objective functions, they are also applicable to traditional PSO and other heuristic optimization methods.

4.1 High Dimensional Inverse Kinematics as Optimization

Consider workspace goal pose \( T \). Assume that a forward kinematics model capable of giving the poses of any connectors in the self-reconfigurable robot tree for any joint configuration \( q \) is given, \( W(q) = K(q) \) which is easily expressed in closed form for any tree of rigid body robot modules using homogeneous transformation matrices. \( W(q) \) is the workspace pose of the connector being used as an end-effector, which is a function of joint angles \( q \). Let \( C(q) \) be a collision function which returns 0 if a set of joint angles is collision-free and self-collision-free and 1 otherwise (e.g., computed using geometric overlap queries between the robotic system and known environmental obstacles). Then, the HDIK problem can be solved by minimizing:

\[
F(q) = a_p P_{error}(q) + a_o O_{error}(q) + a_c C(q)
\]
In the above equation $P_{error}$ is the Euclidean position distance between $T$ and $W(q)$, while $O_{error}$ is some measure of orientation error between $T$ and $W(q)$. $\alpha_p$ and $\alpha_o$ are optional constants (set to 1, by default) weighing the differing importance of $P_{error}$ and $O_{error}$ (as they are measured on different scales). $\alpha_c$ is a large positive constant penalizing collisions and ensuring only collision-free configurations meet the required error tolerance. There are a number of ways to measure $O_{error}$, but, for this work, the magnitude of difference in Roll-Pitch-Yaw Euler Angles (in degrees) between $T$ and $W(q)$ is minimized. Eq. 1 can be generalized to the case of multiple end-effectors by summing up the $P_{error}$'s, $O_{error}$'s and collision errors for each end effector and minimizing one large sum. Multi-objective optimization methods could be used instead, but they are left for future work. Solving the HDIK problem is then reduced to finding a joint configuration $q$ minimizing Eq. 1.

4.2 Self-Reconfiguration Kinematic Control as Optimization

The SRKC problem seems entirely different from the HDIK problem upon initial inspection, as the pose at which the connectors are to meet is not given as input. Naive approaches such as fixing one end-effector in space and using this pose as the target pose may result in unsolvable problems. The key insight is that, rather than searching for a target pose and optimizing connectors toward that pose, PSIK searches directly for joint configurations of the tree in which the chosen connectors are aligned with one another, maximizing the number of potential solutions.

In Eq. 1, $P_{error}$ and $O_{error}$ are error terms relative to a fixed target $T$. In the SRKC problem, there is no fixed target $T$. Rather, the joint configuration $q$ minimizing the error between the workspace poses of two chosen connectors in the tree $\{W_1(q), W_2(q)\} = K(q)$ (returned by the forward kinematics model) must be found. Note that a fixed transformation must usually be applied to one of the connectors such that minimizing the error between them results in aligning them appropriately for docking. By redefining $P_{error}$ to be the Euclidean position distance between $W_1(q)$ and $W_2(q)$ and redefining $O_{error}$ to be the magnitude of difference in Roll-Pitch-Yaw Euler Angles (in degrees) between $W_1(q)$ and $W_2(q)$, Eq. 1 can again be minimized over joint configuration $q$ to solve the SRKC problem. It is important to note that the function $F$ in Eq. 1 – whether for the HDIK problem or SRKC problem and regardless of the number of end-effectors – has a known theoretical lower bound minimum value of 0 for solvable problems, making the theoretical proofs of convergence to the global optimum in finite time for any error tolerance given in [29] applicable to this work.

As illustrated in Algorithm 1, by minimizing Eq. 1 using PSIK, one arrives at a globally convergent and optimal HDIK and SRKC solution. At each step, a randomly chosen partition element is refined into two new partition elements by cutting the partition with an $n - 1$ dimensional hyperplane at the midpoint of a randomly chosen dimension (Line 14, `BranchPartition`). Any active partition is, by construction, a hypercube in $\mathbb{R}^n$ and can clearly be further refined in this manner. Partition el-
Algorithm 1: The PSIK Algorithm

Input:
\( \varepsilon \): error threshold
\( F \): objective/error function (Eq. 1)
\( S^n \): initial search hypercube with bounds
\( M_{\text{Parts}} \): maximum allowable partitions

Output: \( g \): best particle found

1. Function \( \text{PSIK()} \)
2. \( \beta := 0; \)
3. \( \alpha := \text{MAX \_FLOAT}; \)
4. \( M := \{ S^n \}; \quad // \text{active partitions} \)
5. \( g := \text{RandSolution}(); \)
6. while \( \alpha - \beta \geq \varepsilon \) and \( M.\text{size()} < M_{\text{Parts}} \) do
7. foreach active partition \( M_i \) do
8. \( \{ \text{bestPos}, \alpha_i \} := \text{PSOBound}(\varepsilon, F, M_i); \quad // \text{PSO using Eq. 1} \)
9. if \( \alpha_i < \alpha \) then
10. \( g := \text{bestPos}; \)
11. \( \alpha := \alpha_i; \)
12. \( i := \text{RandomPartition}(); \)
13. Remove partition \( M_i \) from list \( M; \)
14. \( \{ M_{1}, M_{2} \} := \text{BranchPartition}(i); \)
15. Replace \( M_i \) with \( M_{1} \) and \( M_{2} \) in \( M; \)
16. return \( g; \)

Elements are selected randomly for further refinement with equal probability given to all partition elements (Line 12, \( \text{RandomPartition}() \)). The \( \text{PSOBound}() \) function in Line 8 is simply the use of traditional PSO to minimize objective function \( F \) (Eq. 1) in the bounds of the current partition element. This returns both a best solution position (joint configuration) and best solution fitness (\( \text{bestPos} \) and \( \alpha_i \), respectively). For the problems considered here, the authors suggest that for PSO smaller swarms of particles (e.g., 30-50) should be used with small maximum iteration counts (e.g., 50-100) to allow more branching to occur, forcing PSO out of local minima. The ranges given are based on extensive experimentation done by the authors.

5 Validation and Experiments

5.1 Traditional PSO for HDIK using PSIK Objective Functions

Though traditional PSO [10] has been used as an IK solver before in the literature, its use has been limited primarily to low-DOF or planar robot manipulators. The authors first conducted a suite of tests on simulated hyper-redundant serial manipulators made of SuperBot modules (Figure 2) aimed at determining how well PSO solved general 6D (position and orientation) HDIK problems when minimiz-
ing PSIK objective function Eq. 1 as the number of DOF of the manipulator increased. The results are presented in Figure 3. 200 runs were done per manipulator size with particle swarms ranging in size from 200-350 particles and maximum iteration counts ranging from 1500-3000 (depending on manipulator size). All solutions were collision-free and self-collision-free. When applied to tree structures of modules, similar results were achieved. Though PSO with the PSIK objective function consistently found high-quality solutions, it occasionally failed to converge within the given tolerance, and the resulting solution could be quite poor.

Fig. 2 SuperBot ([24]) manipulators ranging from 30 to 180 DOF (10-60 SuperBot modules) were used to test PSO as an HDIK solver with PSIK objective functions. Illustrated here are a 30 DOF manipulator (left), a 90 DOF manipulator (middle), and a 180 DOF manipulator (right).

<table>
<thead>
<tr>
<th>DOF (n)</th>
<th>Avg. Runtime (seconds)</th>
<th>Avg. posError</th>
<th>Avg. orientError</th>
<th>Avg. Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1.57</td>
<td>0.00046</td>
<td>0.00314</td>
<td>275.84</td>
</tr>
<tr>
<td>60</td>
<td>3.36</td>
<td>0.00034</td>
<td>0.00260</td>
<td>383.165</td>
</tr>
<tr>
<td>90</td>
<td>7.46</td>
<td>0.00036</td>
<td>0.00210</td>
<td>533.055</td>
</tr>
<tr>
<td>120</td>
<td>15.47</td>
<td>0.00034</td>
<td>0.00554</td>
<td>697.945</td>
</tr>
<tr>
<td>150</td>
<td>22.11</td>
<td>0.00036</td>
<td>0.00515</td>
<td>764.065</td>
</tr>
<tr>
<td>180</td>
<td>37.03</td>
<td>0.00032</td>
<td>0.00237</td>
<td>947.255</td>
</tr>
</tbody>
</table>

Fig. 3 Traditional PSO with PSIK objective functions for HDIK results. posError and orientError are $P_{error}$ and $O_{error}$, respectively, as defined in Equation 1. Avg. Iterations is the average number of PSO iterations performed before the fitness threshold of $h = 0.001$ was met.

5.2 PSIK as an HDIK and SRKC solver

Encouraged by PSO’s performance using PSIK objective functions, but hoping to rectify its failure to converge to globally optimal solutions, we next evaluated the full PSIK algorithm as an HDIK and SRKC solver. The configurations tested (with end-effector connectors highlighted yellow) are visualized in Figure 4.

For clarity of presentation, the test cases are divided into the following categories (applicable tree configurations from Figure 4 are given in parentheses):

1. Category I: Solve 6D (position and orientation) HDIK for all end-effectors given target poses for each (Applies to (i) - (v)).
2. Category II: Solve 6D (position and orientation) HDIK for one randomly chosen end-effector given a target pose while others avoid collision (Applies to (iv), (v)).
3. Category III: Solve 3D (position only) HDIK for all end-effectors given target positions for each (Applies to (i) - (v)).

4. Category IV: Solve 3D (position only) HDIK on one end-effector and 6D (position and orientation) HDIK on the other given target poses (Applies to (iv)).

5. Category V: Solve SRKC problem to reconfigure from (iii) to (iv).

6. Category VI: Solve SRKC problem to reconfigure configuration (v) by randomly selecting two of the three end-effectors to connect.

7. Category VII: Solve SRKC problem to reconfigure from (iv) to (iii).

Figure 6 tabulates the results. Each row is a configuration/test category pair. For each such pair, the algorithm was run 100 times. $\varepsilon$ represents the error tolerance given to the program. The column $\varepsilon$-Success is the percentage of the 100 test cases in which a solution of acceptable quality was found within a fixed time limit (200 seconds). Based on experiments the authors have performed with new versions of the SINGO [26] connector of real-world SuperBot modules, $\varepsilon$ values between 0.004 and 0.006 result in consistent successful grasping/docking, which is why most test cases use $\varepsilon$ values in this range. The cases in which $\varepsilon$ is much greater correspond to cases where PSIK has difficulty converging to such small $\varepsilon$ values (primarily when the position and orientation of multiple end-effectors was to be simultaneously solved, which makes sense given that they are difficult multi-objective optimization problems). Figure 5 shows sample numerical runs in which it is observed that the error monotonically decreases as a function of the number of active partitions, validating that the spawning of new partitions forces PSO out of local minima. In each partition, 20 particles were used with a maximum iteration count of 50.
5.2.1 Locomotion, Manipulation and Self-Reconfiguration Results

**Fig. 7** Top: A 6-module, 18-DOF SuperBot tree reconfiguring into a snake. Bottom: A 6-module, 18-DOF SuperBot snake manipulator reconfiguring into a tree.

**Fig. 8** Top left to bottom right: a demonstration of a 6-module, 18-DOF SuperBot tree performing a pick, transport, and place task.

Figure 7 demonstrates self-reconfiguration of a 6-module 18-DOF SuperBot tree into a long snake and vice versa. Figure 8 demonstrates a 6-module, 18-DOF SuperBot tree locomoting to, picking up, transporting, and placing the six red cylindrical objects in their respective goal areas (blue) in a zero-gravity environment. The modules’ connectors could dock to the ground plane (at any continuous position) and objects directly. Figure 8 is a novel demonstration of non-discretized locomotion with manipulation and transportation never before performed by a tree structure of modular robots (even in simulation) without a grid or lattice. A single point of contact for manipulation and locomotion was assumed to be sufficient, with the motors of the module connected to the ground plane powerful enough to support the structure’s weight. PSIK, which was used by an elected leader module to solve for each of the over 40 foot placements, object pick ups, and object placements required in Figure 8 and to solve the SRKC problems in Figure 7, was implemented on this distributed set of SuperBot modules using local message passing. The leader module (which can change dynamically) acted as the kinematic base of the system, using a message passing BFS procedure to discover the kinematic structure of the tree, and used RRT-connect [16] to plan collision-free paths between computed joint configurations. The error tolerance for each solution was between 0.004 and 0.006 and valid solutions were found at every step (within 10-15 seconds on average).
6 Discussion and Future Work

This paper proposed a general, globally convergent, globally optimal algorithm for solving the HDIK and SRKC problems in finite time given any positive error tolerance applicable to self-reconfigurable robot trees called Provably-convergent Swarm-based Inverse Kinematics (PSIK). Physics-based simulation results validated the method and illustrated its ability to produce globally optimal solutions quickly enough to facilitate dextrous locomotion, manipulation, and self-reconfiguration. PSIK represents a key stepping stone in searching for a fully distributed general solution for the HDIK and SRKC problems. The current method assumes that the required information is distributed among modules and collected via local message passing. It is not fully distributed as a certain module has indeed collected a global picture of the kinematic tree. Fortunately, this leader module is not fixed and can be arbitrarily elected in real time (easily switching to other modules if damaged). To make the method fully distributed, our next step is to combine dynamic leader-selection (e.g., [25]) with the information collection of kinematic structure so that leader election is fully distributed and non-deterministic.

References