




A metaheuristic approach to optimal morphology in reconfigurable tiling robots

Manivannan Kalimuthu¹ · Thejus Pathmakumar¹ · Abdullah Aamir Hayat¹  · Mohan Rajesh Elara¹ · Kristin Lee Wood²

Received: 22 July 2022 / Accepted: 17 February 2023 / Published online: 10 April 2023
© The Author(s) 2023

Abstract

Reconfigurable robots are suitable for cleaning applications due to their high flexibility and ability to change shape according to environmental needs. However, continuous change in morphology is not an energy-efficient approach, with the limited battery capacity. This paper presents a metaheuristic-based framework to identify the optimal morphology of a reconfigurable robot, aiming to maximize the area coverage and minimize the energy consumption in the given map. The proposed approach exploits three different metaheuristic algorithms, namely, SMPSO, NSGA-II, and MACO, to generate the optimal morphology for every unique layout of a two-dimensional grid map by considering the path-length as the energy consumption. The novel feature of our approach is the implementation of the footprint-based Complete Coverage Path Planning (CCPP) adaptable for all possible configurations of reconfigurable robots. We demonstrate the proposed method in simulations and experiments using a Tetris-inspired robot with four blocks named *Smorphi*, which can reconfigure into an infinite number of configurations by varying its hinge angle. The optimum morphologies were identified for three settings, i.e., 2D indoor map with obstacles and free spaces. The optimum morphology is compared with the standard Tetris shapes in the simulation and the real-world experiment. The results show that the proposed framework efficiently produces non-dominated solutions for choosing the optimal energy-efficient morphologies.

Keywords Reconfigurable robots · Path planning · Area coverage · Metaheuristics algorithms · Design principles

Introduction

Cleaning is an essential function of daily routine, yet it is considered monotonous, dirty, and time-consuming. Moreover, humans prefer to prioritize other tasks over cleaning, which fosters the development of cleaning robots that can free up humans to do creative tasks. Several cleaning robots have been developed to tackle these problems, including Roomba, Xiaomi, Samsung, Dyson, etc. However, the existing robots' performance is limited by their morphology, primarily because they are not custom-made to work on the different layouts in the home. To address this limitation, reconfigurable robots are designed to change their morphology or shape to clean efficiently and access difficult-to-reach regions. When compared to fixed morphology robots, these reconfigurable robots provide better area coverage. However, at the same time, they consume more power to often change shapes to clean a given area, and they may not be able to function continuously for a long time since they draw power from finite source batteries. Hence, it will be useful to look

✉ Abdullah Aamir Hayat
abdullaamir@sutd.edu.sg; aamir_hayat@rediffmail.com

Manivannan Kalimuthu
manivannankalimuthu@mymail.sutd.edu.sg

Thejus Pathmakumar
thejus_pathmakumar@sutd.edu.sg

Mohan Rajesh Elara
rajeshelara@sutd.edu.sg

Kristin Lee Wood
kristin.wood@ucdenver.edu

¹ ROAR Lab, Engineering Product Development, Singapore University of Technology and Design, 8 Somapah Road, 487372 Singapore, Singapore

² College of Engineering, Design and Computing, University of Colorado Denver, 1200 Larimer St, Ste. 3034, Denver, CO 80204, USA

for a method to find an optimal shape of a reconfigurable robot that can be used during, say, a cleaning or maintenance task according to the given map of the deployment area.

Reconfigurable systems have a wide range of advantages in terms of multi-ability (in various configurations, the system can execute multiple functions at different times), evolution (the system's configuration can be altered by removing, substituting, and adding new elements), and survivability (the system can still work despite a potential failure of one or more components) [1]. Reconfigurable systems have been widely adopted in various applications, including reconfigurable manufacturing systems [2], field-programmable gate arrays [3], various software concepts [4], aerospace [5], etc. One of the main applications of reconfigurable systems is demonstrated in self-reconfigurable robots, which can modify their morphology manually or automatically depending on the situation or environment. The development of reconfigurable robotic systems has gained attention since 1980 [6]. Self-reconfigurable robots are broadly divided into two main categories: intra-reconfigurable and inter-reconfigurable [7]. Inter-reconfigurable robots change their function by adding a new module/system via assembly and disassembly. CONRO [8], Crystal [9], and M-Lattice [10] are a few examples. Intra-reconfigurable robots transform their configuration without the requirement for external attachment to enable a new function or improve an existing function. Examples include Panthera [11], a pavement-sweeping robot that can change its footprint by expanding and collapsing to clean the pavements while overcoming different obstacles, a ship maintenance robot with different end-effector to remove barnacles from the hull [12], and sTetro [13,14], a staircase cleaning robot. Therefore, the potential of morphology change in reconfigurable robots are beneficial for several applications.

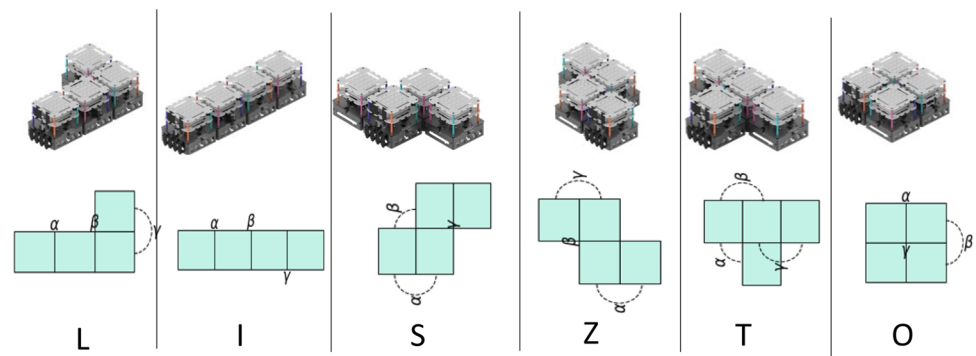
In recent years, there has been substantial progress in the mechanism design, control, perception, and autonomy in reconfigurable robots [15–20]. For example, the Tetris-inspired tiling robot reported in [21] can change into seven different configurations, namely, {I, J, L, O, S, T, Z} following the constraint of one-sided polyomino [22] to perform area coverage tasks [21,23,24]. Here the authors highlighted the advantage of reconfigurable robots over fixed morphology robots by demonstrating the higher area coverage task. Similarly, the concept proposed in [25] named pleomorphic h-Infi can change into an infinite number of configurations to perform cleaning tasks (as the constraint of a one-sided polyomino is not accounted). However, only a few authors have explored the energy optimization problem in reconfigurable robots. The energy consumption and time taken in self-reconfigurable robots are a crucial factor since these robots have a higher number of actuators than fixed morphology robots [26]. In [27], the author proposed a coverage path planning approach with the optimal energy consumption based on the Genetic Algorithm (GA) and the Ant Colony

Optimization (ACO) technique, and the reported work in [28] introduced a Reinforcement Learning (RL) based path planning approach suitable for producing Pareto plans, which allow the robot to cover a specified region on the map with the least amount of energy consumed. The suggested solution in [29] assures complete area coverage of the robot while decreasing the number of reconfigurations carried out, reducing time and energy consumption. However, to the best of the author's knowledge, none of the aforementioned works address the identification of a single optimal morphology to cover a given area resulting in minimizing energy consumption and maximizing the area coverage.

Path planning is an integral part of the self-reconfigurable robotic system [30]. Establishing a path which traverses across all points of a defined region in the map is considered as coverage path planning [31]. Classical exact cellular decomposition is one of the prominent CPP methods [32]. This method breaks down the coverable region (excluding obstacles) into a non-overlapping region called cells. Trapezoidal and boustrophedon are the two popular offline decomposition methods [33]. Grid-based methods, on the other hand, utilize a representation of the map in the form of a set of uniform grid cells. Each cell holds a value in this grid-based arrangement, whether it has an obstacle or free space. A grid map is simple to generate since it is generally modeled as an array, with each element containing occupancy information corresponding to the cell [34]. As a result, coverage algorithms most commonly use grid-based representations. However, grid-based algorithms are only suitable for indoor environments (relatively small maps) since they require accurate localization and suffer from the exponential growth of memory usage [31].

Wavefront algorithm [35] and spanning trees methods [36] is the widely adopted grid-based CPP approaches. Both methods can be used for offline and online approaches. Similarly, other methods include Graph-based coverage [37], landmark topological-based coverage [38], and 3D coverage [39,40], etc. CPP's main criteria include covering all points, reducing overlapping, continuous operation with no path repetition, avoiding obstacles, simple trajectories, and determining the optimal path. Similarly, optimal solutions for path planning and area coverage tasks can be identified using many approaches like Reinforcement Learning (RL), neural networks, metaheuristics algorithm, fuzzy logic, and linear optimization [41–44]. Among these, the metaheuristic-based algorithm has outperformed others due to its simplicity, flexibility, faster computation, and the ability to explore possible solutions without prior information [45]. Metaheuristics are designed to tackle combinatorial optimization problems, where classical methods are inefficient in producing a good solution [46]. Metaheuristic algorithms have recently been widely used in machine learning, computer vision, scheduling, robot motion planning, etc [47–49]. Further, Particle

Fig. 1 Set of Standard polyomino morphologies as reported in [21,26]



Swarm Optimization (PSO), Genetic Algorithm (GA), Simulated annealing (SA), Harmony search (HS), Ant Colony Optimization (ACO), and Grey Wolf Optimizer (GWO) are extensively used metaheuristics algorithms in mobile robots and UAVs for path planning and area coverage applications. [50–57].

This paper proposes a first-of-its-kind framework to identify an optimal morphology out of multiple morphologies taken by a Tetris-inspired reconfigurable robot for an area coverage of a given two-dimensional (2D) environmental map. The framework is devised to identify the best morphology accounting for maximizing task performance and minimizing energy consumption. The proposed framework uses metaheuristic algorithms to generate the optimal shape for the robot by treating it as a multi-objective optimization problem. The proposed framework is generic and can be applied to any class of tiling robot (polyominoes, heptiamonds, and hexiamonds [22]) that reconfigures in 2D space. However, this work demonstrates using a developed reconfigurable robot, named *Smorphi*, which can change its shape into infinite¹ configurations. On a larger scale, the proposed method allows end-users to customize the robot shape according to the environment where it has to be deployed. To the best of the author’s knowledge, an optimal shape generation framework for the reconfigurable robot which compromises the area coverage and energy consumption has not been addressed.

The objectives of the proposed work are:

1. Propose an Opt-Morph framework for identifying the optimal shape of a reconfigurable robot using a footprint-based complete coverage path planning approach.
2. Demonstrate the Opt-Morph framework in a reconfigurable tiling robot named *Smorphi* in three different simulated environments, using metaheuristics algorithms, namely, Speed-constrained Multi-objective Particle Swarm Optimization (SMPSO), Non-dominated Sort-

ing Genetic Algorithm-II (NSGA-II), and Multi-objective Ant Colony Optimization (MACO) algorithms.

3. Validation of the proposed Opt-Morph framework in the real-world settings using in-house developed *Smorphi* robot.

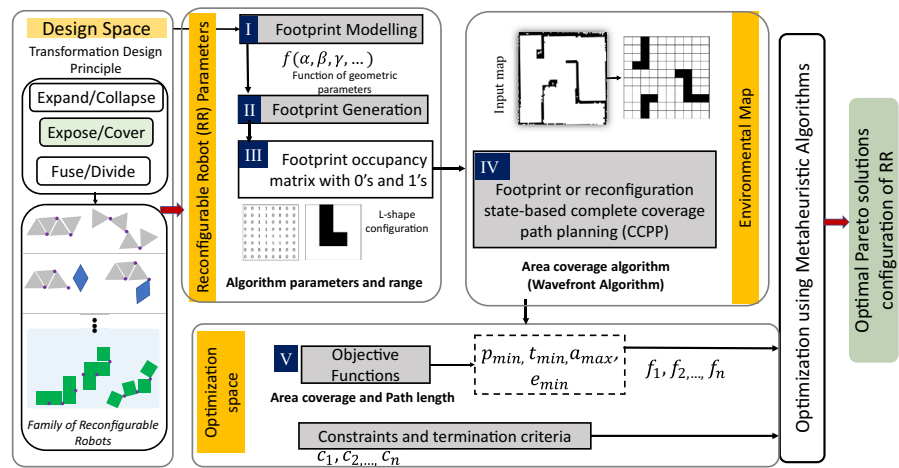
The rest of the article is organized as follows; The proposed framework is introduced in Sect. “Opt-Morph frameworksec:2”. Section “Design space” describes the mechanical design of the *Smorphi* robot along with the design principles and the kinematics of the robot. Section “Complete Area Coverage and Path Planning (CCPP) space” explains the implemented path planning and footprint-based area coverage. Section “inlinkOptimization spacesec:5” presents the optimization approach along with the objective functions. Results and discussion of the proposed framework, along with the experiments conducted, are presented in Sect. “Results and discussion”. Finally, Sect. “Conclusion” concludes the paper by summarizing the key findings and highlighting future work.

Opt-Morph framework

The proposed Opt-Morph framework for determining the optimal shape of a reconfigurable robot is illustrated in Fig. 2. The proposed framework is divided into three layers as shown in Fig. 2: I. Design space of a reconfigurable robot (here, dealt in Sect. “Design space”), II. Complete Area Coverage and Path Planning (CCPP) space (in Sect. “Complete Area Coverage and Path Planning (CCPP) space”), and III. Optimization space schemes (in Sect. “Optimization space”). In design space, to generalize the framework, we used a set of three transformational principles, namely expand/collapse, expose/cover, and fuse/divide, proposed in [58]. These three principles cover the entire spectrum of reconfigurable products or systems based on their means of reconfiguration. The transformation design principles and the design of the robot named *Smorphi* used to demonstrate the Opt-Morph framework is described in Sect. “Design space”.

¹ by not adhering to the constraints imposed as one-sided polyomino [22].

Fig. 2 Overview of the proposed Opt-Morph framework



The CCPP space focuses on the algorithms for coverage planning and computes the path-length and the percentage of area-covered by the robot based on the robot's footprint and map of the environment. The CCPP space follows the two-step approach to compute the percentage of area-covered and the path-length. First, the rectangular footprints (Fig. 2, 'I') corresponding to the robot shape generated by the design space is estimated. Then, an occupancy matrix for the individual shape is computed such that the occupancy grid's size will equal the dimensions of the rectangular footprint (Fig. 2, 'II'). The elements of the occupancy matrix are determined such that zero '0' represents the region unoccupied by the robot base and one '1' (Fig. 2, III) corresponds to the region occupied by the robot base. Then, with the footprint matrix and the 2D map as the input, the complete area coverage path planning is computed using the wavefront algorithm (Fig. 2, 'IV'). In the optimization space (Fig. 2, 'V'), for the given objective function and constraints, the algorithm search for the best optimal shape in the infinite pool of configuration by optimizing the path-length p_{min} , and the percentage of area-covered a_{max} (Fig. 2). This work uses metaheuristic-based search methods, namely, SMP SO, NSGA-II, and MACO, for the optimization. It is mainly because of the ability to produce an optimal solution with the least amount of information [59]. Finally, upon completing the optimization, the framework provides the non-dominated solutions for the given map. This work demonstrates the proposed framework using the reconfigurable tiling robot named *Smorphi*. A detailed discussion on the design of the *Smorphi* robot is discussed in the following section.

Design space

This section outlines the mechanical design process of the *Smorphi* robot along with the transformational design prin-

ciples and facilitators used along with the kinematics of the robot.

Transformational design principles

In this work, we exploited transformational design principles as an enabler for designing the reconfigurable robot named *Smorphi*. [58] author empirically analyzed all the reconfigurable products, resulting in three transformational principles: expand/collapse, expose/cover, fuse/divide, and twenty transformational facilitators. The transformational principle is a guideline, when embodied singly, creates transformation. Below is a brief description of the transformation principles.

- **Expand/collapse:** Changing the physical dimensions of an object to bring about an increase/decrease in an occupied volume primarily along an axis (1D), in a plane (2D), or 3D (three dimensions).
- **Expose/cover:** Concealing or revealing a new surface to alter functionality.
- **Fuse/divide:** Make a single functional device become two or more devices or vice versa where at least one of the multiple devices has a distinct functionality separate from the function of the single device.

A transformation facilitator is a design element that facilitates or assists reconfiguration or transitions. A facilitator alone will not result in reconfiguration; the facilitator must be used in conjunction with at least one principle. Please refer to [58] for a detailed description of transformational facilitators.

Our previous work extended the transformational design principles by adding a generator layer to design a reconfigurable pavement sweeping robot [11,60]. The generator is the mechanism construct that helps realize the transformational facilitator in the form of a mechanism, such as a linkage,

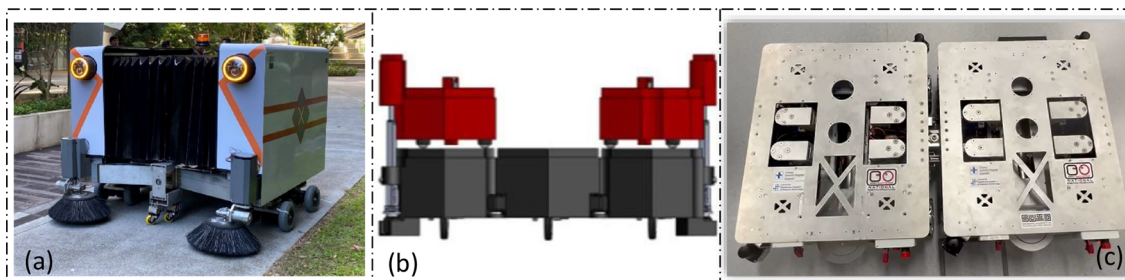


Fig. 3 Transformation principles; **a** Expand/Collapse, **b** Expose/Cover, and **c** Fuse/Divide

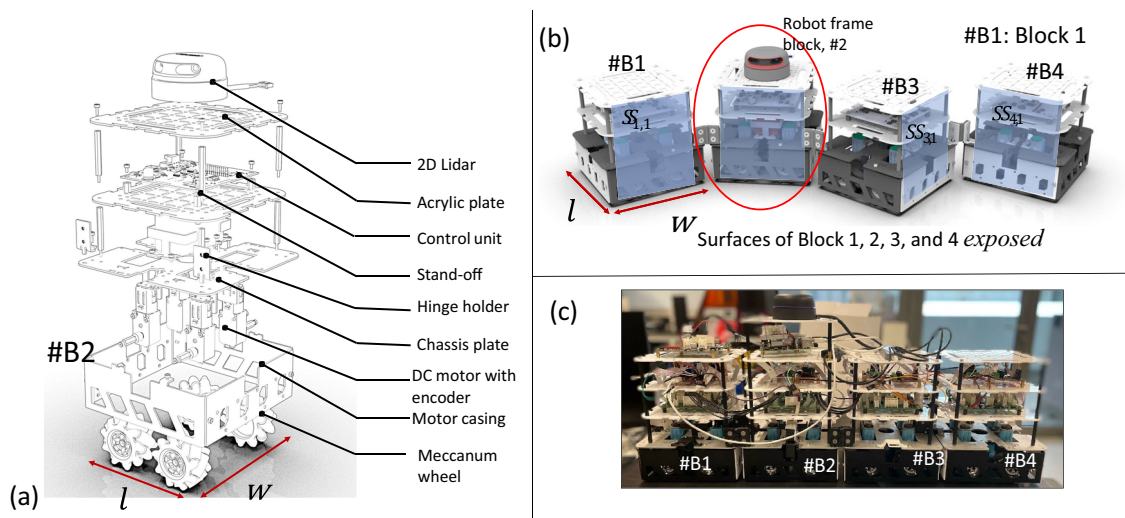


Fig. 4 Mechanical design of the Smorphi robot based on Expose/Cover transformation principle; **a** Exploded view for block 2 (#2), **b** 3D-Model of the robot, and **c** Fabricated *Smorphi* robot

gear, cam, belt, pulley, etc., that transforms the given input motion into the desired output motion by varying the joint configuration.

Figure 3 illustrates the examples of three transformational principles in reconfigurable robots. For example, in Fig. 3a pavement sweeping robot expands its body to clean the pavement and collapses its body to allow the pedestrians to pass [61]. The floor-cleaning robot is reconfigured into a wall-cleaning robot by revealing a new surface in Fig. 3b. The modular robot in Fig. 3c can allow multiple units to attach/detach to perform various logistics tasks. Deriving inspiration from the transformational principle, the *Smorphi* robot reconfiguration behavior is conceived with the aid of Expose/Cover principle. As the name implies, the *Smorphi* robot is designed to reveal/conceal the side surface of each block in order to reconfigure from one shape to another. The comprehensive mechanical design of the *Smorphi* robot is discussed in the following section.

Mechanical design

The detailed mechanical design of the *Smorphi* robot is shown in Fig. 4. The *Smorphi* robot is designed as an individual modular block that shares the common mechanical structure that can be connected manually through a hinge. The more number of the blocks increases the total degree of freedom of the robot. We have used four identical square blocks with a length of 10 cm² and the footprint of 100 cm² in the LLR (Left, Left, Right) hinge configuration. Each block of the designed robot is built in a four-layer structure. The locomotion units are mounted on the bottom plate, and low-level controllers, batteries, and other accessories are on the middle two plates. Perception and localization sensors are installed on the top plate to avoid obstruction while reconfiguring. All the plates are fabricated with the 0.2 cm thickness acrylic sheet except the chassis plate to reduce the overall weight. The aluminum standoff is used to adjust the height between different plates to keep it modular. In addition, the

² Though the SI unit of distance is meter (m), to avoid decimal places, we used centimeter (cm) as the unit of distance throughout the paper.

Table 1 Reconfigurable robot Smorphi specifications

Items	System specifications
Dimensions ($l \times w \times h$)	$20 \times 10 \times 45$ cm
Total weight	3 kg
Payload capacity	1 kg
Wheel diameter	6 cm
Ground clearance	1 cm
Maximum velocity of the robot	0.19 ms^{-1}
Maximum angular speed of the robot	1.25 rad s^{-1}
Locomotion power supply	11.1 V
Type of motor	Brushed DC

2D LiDAR is installed on top of the robot for mapping and localization, and the robot is controlled with the help of the Intel compute stick. The system specifications of the *Smorphi* robot is provided in the Table 1.

Smorphi has the ability to change into an infinite number of configurations by varying the three hinge angles. The *Smorphi* robot leverages on Mecanum wheel-based locomotion to navigate the environment, with each block equipped with four Mecanum wheels and motors. Since this locomotion type offers a holonomic drive where the total degree of freedom is equal to the controllable degree of freedom, the robot can move in all directions independently. Also, this locomotion type enables the robot to rotate with zero turning radius. The 7.4v DC motor with the encoder is used as a traction motor to drive the Mecanum wheel. As the Mecanum wheels drive at different speeds, the individual blocks rotate around the hinge axis to reconfigure. During this reconfiguration, one block should be kept stationary while the other three transform concerning that block. For *Smorphi* Block 2 (B2) is taken as the reference block and does not change its pose, while the rest of the block can reconfigure about its passive hinge.

Robot configurations, footprint, and kinematics

Figure 5 depicts the *Smorphi* robot's kinematic model in a 2D Cartesian coordinate system. The reconfigurable robot is represented as four identical squares of length $l = 2a$. The hinge angle α , β , and γ determine the robot's possible configuration space C_s and the respective footprint or workspace.

$$C_{s,fp} = f(\alpha, \beta, \gamma, l), \quad \text{for } 0 \leq \alpha \leq \pi, 0 \leq \beta \leq \pi, 0 \leq \gamma \leq \pi \quad (1)$$

where $C_{s,fp}$ is the footprint corresponding to the configuration space, and l is each block's geometric length and width and are constant. The possible shape from the infinite number of configurations that can be formed by changing the three

hinge angles is determined by Eq. 1. Each hinge angle can be varied between 0 and π , and the angles are 0 when all four blocks form a straight line or in I-configuration, as shown in Fig. 1. The kinematic of the individual block of *Smorphi* follows the holonomic locomotion supported by four Mecanum wheels as shown in Fig. 5a. The kinematics of a single block with four Mecanum wheels are given as:

$$\underbrace{\Phi_{m1}}_{4 \times 1} = \frac{1}{r_w} \underbrace{\mathbf{J}_{m1}}_{4 \times 3} \underbrace{\zeta_{m1}}_{3 \times 1} \quad (2)$$

where ϕ_{m1} is the vector of the wheel velocities of module 1 (m1), #1, i.e., $[\dot{\phi}_1 \ \dot{\phi}_2 \ \dot{\phi}_3 \ \dot{\phi}_4]^T$ and $\zeta_{m1} \equiv [\dot{x} \ \dot{y} \ \dot{\theta}]^T$ is the vector of robot velocity in the robot frame. To transfer this information to the world frame, the rotation matrix about Z-axis is included as $\mathbf{R}(\theta)\zeta_{m1}$. \mathbf{J}_{m1} is the Jacobian matrix for a single holonomic robot with four Mecanum wheels with elements as $[-1 \ 1 \ k_1; 1 \ 1 \ -k_1; -1 \ 1 \ -k_1; 1 \ 1 \ k_1]$ where the constant $k_1 = (l + w)$, i.e., basically the sum of length and breadth of each block dimension. Figure 5c shows the four block $m = 1, 2, \dots, 4$ and in total 16 Mecanum wheels. To establish the relation between the robot frame and the frame associated with individual blocks, the origin of the second block, i.e., #2 is selected as the robot frame with $O_R X_R Y_R$. Relative to this frame, the location of other individual blocks are assigned with the vector $\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4$ and can be geometrically found given the angle between the two consecutive blocks are known. Then the kinematic relation mapping the wheel velocity of a given block # n (here $n = 1, 2, 3$) in the robot frame is given by:

$$\Phi_n = \mathbf{J}_{m1} \frac{1}{r_w} \mathbf{R}_Z(\alpha_{n,n+1}) \mathbf{T}_d(\mathbf{d}_n) \zeta_R \quad (3)$$

where $\zeta_R = [\dot{x}_R \ \dot{y}_R \ \dot{\theta}_R]^T$ is the velocity vector of *Smorphi* in robot frame $X_R Y_R O_R$, i.e., placed on block #2. Also, $\mathbf{R}_Z, n(\alpha_{n,n+1})$ is the rotation about the Z-axes between the blocks, i.e., α, β, γ and \mathbf{T}_n is the translation matrix in 2D by the magnitudes of vector \mathbf{d} expressed as:

$$\mathbf{R}_z, n \equiv \begin{bmatrix} \cos \alpha_{n,n+1} & \sin \alpha_{n,n+1} & 0 \\ -\sin \alpha_{n,n+1} & \cos \alpha_{n,n+1} & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{T}_n \equiv \begin{bmatrix} 1 & 0 & -d_n \sin \psi_{n,n+1} \\ 0 & 1 & d_n \cos \psi_{n,n+1} \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Moreover, in the inertial frame $X_I O_I Y_I$ the inverse kinematics equation of the four-block system is derived as:

estimation based on the footprint. The coverage planning algorithm generates a global trajectory for the robot by avoiding collision with obstacles in a given map. The path planning algorithm takes in two inputs for the trajectory generation; the robot footprint and the 2D environment map.

Footprint generation

This section details the implementation of the footprint generation for a given configuration of the *Smorphi* robot.

The *Smorphi* robot's footprint is represented by four squares hinged at LLR corners. This makes the footprint to have a 2D geometry. Assuming block 2 (B2)'s pose is fixed, the remaining square blocks can be rotated about their hinge axes passing as shown in Fig. 5. The projection of vector \mathbf{a} onto the X - and Y - axes to determine the coordinates of the vertices, are given in the Eq. 6, and 7,

$$\mathbf{a}_x = [\mathbf{a}_1 \cdot \mathbf{x} \quad \mathbf{a}_2 \cdot \mathbf{x} \quad \dots \quad \mathbf{a}_n \cdot \mathbf{x}]^T \quad (6)$$

$$\mathbf{a}_y = [\mathbf{a}_1 \cdot \mathbf{y} \quad \mathbf{a}_2 \cdot \mathbf{y} \quad \dots \quad \mathbf{a}_n \cdot \mathbf{y}]^T \quad (7)$$

where \mathbf{a}_i are the position vector of the vertices of the polyomino-shaped *Smorphi* with total vertices of four blocks resulting in $n = 16$. Vector x and y are the unit vectors along X - and Y -axes, respectively. The dot product operator is used to calculate the magnitude of the projection of vector \mathbf{a}_1 on x as $\mathbf{a}_1 \cdot \mathbf{x}$.

Similarly, the dimension of the footprint bounding box, i.e. its length l and width w are determined by taking the difference between maximum and minimum values of the coordinate points along the X and Y axes, as shown in the Eqs. 8 and 9,

$$w_{fp} = \|\min(\mathbf{a}_x) - \max(\mathbf{a}_x)\| \quad (8)$$

$$w_{fp} = \|\min(\mathbf{a}_y) - \max(\mathbf{a}_y)\| \quad (9)$$

where \min and \max are functions for finding the minimum and maximum of a given set of numbers and $|\cdot|$ is an absolute value operator.

Coverage path planning algorithm

For the coverage planning, we adapted the wavefront algorithm [35]. The implementation of the grid-based wavefront path planning algorithm for complete area coverage is explained in Algorithm 1.

This Algorithm 1 works based on the breadth-first search method. As per the algorithm, the robot moves from the start point to the goal point while covering all the grids. Initially, all the grids are assigned with the 0 value except the grid corresponding to the obstacles, which are assigned with the -1 value. Then, the algorithm assigns the value from

Algorithm 1: Pseudocode for wavefront algorithm adapted for complete coverage planning

```

1 function NodeCost (x, y);
2 for Every node in the map: do
3   if Node == 0 and ≠ -1 : then
4     |  $x_{i+1} += 1$ ;
5     |  $x_{i-1} += 1$ ;
6     |  $y_{i+1} += 1$ ;
7     |  $y_{i-1} += 1$ ;
8   end
9 end
10 function GeneratePath (xi, yi);
11 Set (xi, yi) = 0;
12 if Unvisited neighbor with higher cost found: then
13   | Current node = Neighbor node;
14 else if No neighbor found: then
15   | Mark as visited then stop at the goal;
16 else if Neighbors is visited and goal is not reached: then
17   | Back track until finding the un-visited neighbor;
18 end

```

the goal point for all the grids with zero value in the order of increasing one until it fully covers the entire map. Each node assigns a cost to eight adjacent nodes in the existing wavefront algorithm while considering the robot's diagonal movement. However, since the *Smorphi* robot is assumed to move in only four directions, we modified the wavefront algorithm to assign costs to only four adjacent nodes (up, down, left, and right). Following, nodes have been assigned with a cost, the algorithm advances from the start node, choosing the neighbor with the highest cost until it reaches the goal node. Figure 6 shows the implemented coverage algorithm for the three different scenarios, the arrow mark on the Figure shows the direction of the path from the start node to the goal node.

Footprint-based path planning

After generating the footprint model, the input map is resized with respect to the width and length of the footprint size, as shown in the Eq. 10,

$$\left(x' = \frac{x}{w_{fp}}, y' = \frac{y}{l_{fp}} \right) \quad (10)$$

where x' and y' are the dimensions of scaled map, and w_{fp} and l_{fp} are the dimensions of footprint respectively. The scaled map, including the obstacle data, is passed to the path planner, which computes the cost and generates waypoints to cover the whole map. Following that, the robot moves from the starting position to the goal point by covering all the cells based on the waypoints received. After completing the coverage path planning with respect to the robot's trials, the area with zeroes is regarded as unvisited, while

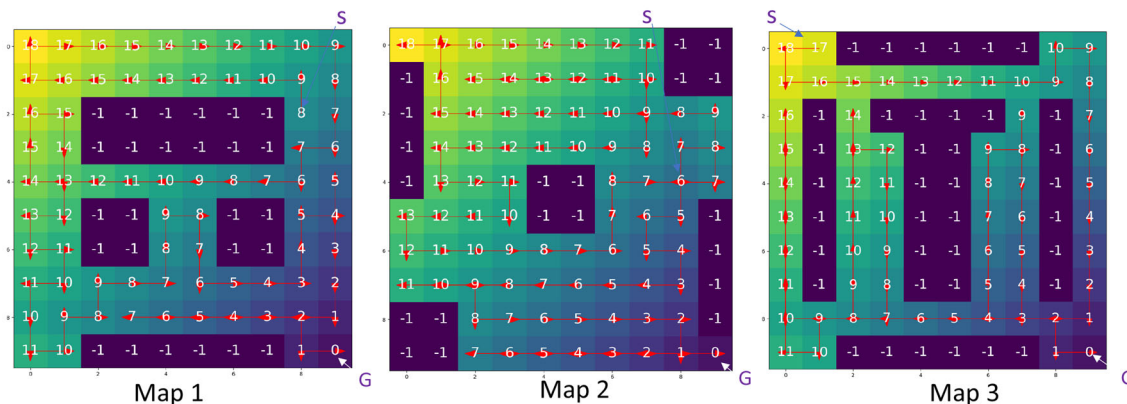


Fig. 6 Path planner in three distinct environments as Map-1, -2, and -3 (S: Start point, G:Goal point, -1: Obstacle, ≥ 0 : Free space and the value in each grid denotes the cost associated with it)

an area greater than zero is considered visited. The resulting footprint, path planning, and area coverage are shown in Fig. 7 for the values ($\alpha = 0$, $\beta = 90$, and $\gamma = 0$). Furthermore, the provided footprint generation approach avoids self-collision between individual blocks by determining the relative angle between each block. The pseudocode for the implemented footprint-based area coverage is shown in the Algorithm 2.

Algorithm 2: Footprint based area coverage

```

1 function FootpringGeneration ( $\alpha, \beta, \gamma$ );
2 for Every  $\alpha, \beta, \gamma$ : do
3   if the blocks are collided: then
4      $R_a$  = Calculate relative angle();
5     Update  $\alpha, \beta, \gamma$  based on the  $R_a$ ;
6   else
7     return  $\alpha, \beta, \gamma$ ;
8
9 end
10 for Every shape in the possible configurations: do
11   Calculate the coordinates // Equation 3 & 4;
12   Calculate the footprint // Equation 5 & 6;
13 end
14 while State = True do
15    $r_m$  = resize the initial map (footprint) // Equation 7 ;
16   Path Planner ( $r_m$ ) // ;
17   Compute cost();
18   Generate way points();
19   Translate way point to initial map();
20   MoveRobot();
21    $f_1$  = Compute Path-Length();
22    $f_2$  = Compute Area-Coverage();
23   State  $\leftarrow$  False;
24 end

```

Optimization space

This section discusses the modeling of the problem, optimization objectives, and optimization algorithms’ implementation in detail. The optimal shape-finding problem defined in this work is modeled as a Multi-Objective Optimization Problem (MOOP). For each iteration, the algorithm calculates the path-length and the percentage of area covered in a given map. The MOOP algorithms evaluate the possible shapes with respect to the objective function resulting in Pareto optimal solutions. In this MOOP, the path length is assumed to be proportional to the energy consumption. The overall environment is considered a $m \times n$ matrix map, with each cell representing the following information: free space, visited region, obstacle region, and overlap region.

$$C_t = \min \left(f_1(\alpha, \beta, \gamma)_{(\alpha, \beta, \gamma)=180^\circ} \right)_{(\alpha, \beta, \gamma)=0^\circ} + \max \left(f_2(\alpha, \beta, \gamma)_{(\alpha, \beta, \gamma)=180^\circ} \right) \tag{11}$$

The total cost function C_t for the defined problem is stated in the Eq. 11. The functions f_1 and f_2 calculate the path-length P_l and the percentage of area-covered A_c , respectively. This problem is modeled as a min-max optimization problem, as the algorithm should aim to minimize the path length and maximize the percentage of area-covered.

Metaheuristic algorithms for optimization

Metaheuristic is an iterative solution-finding process inspired by nature that guides a subordinate heuristic by incorporating various techniques for exploring the solution space. Metaheuristics algorithm works based on two principles; choosing the best solution and randomization. The best solution ensures optimal convergence, and randomization is used

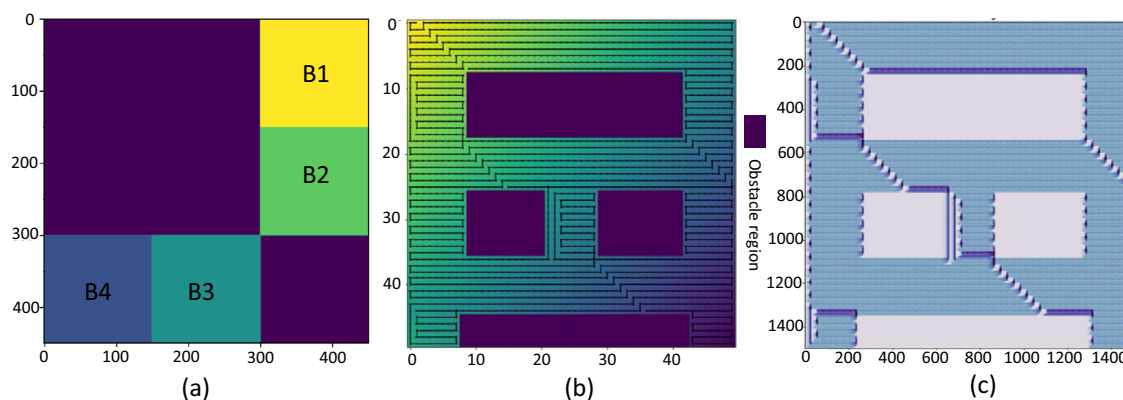


Fig. 7 **a** Generated footprint for the shape ($\alpha = 0$, $\beta = 90$, and $\gamma = 0$) (matrix of zeroes and ones), **b** Path planner, and **c** Complete area coverage

to avoid the solutions being trapped in the local optima. In this work, we exploited three different metaheuristic algorithms to optimize the objective function, namely, Speed-constrained Multi-objective Particle Swarm Optimisation (SMPSO), Non-Dominated Sorting Genetic Algorithm-II (NSGA-II), and Multi-objective Ant Colony Optimization (MACO). Using metaheuristic approaches a group of “non-inferior” solutions can be identified that define a limit in the goal space where no objective can be improved without losing at least one of the others is known as a Pareto optimum solution. The working principle and the implementation of those algorithms are discussed in the following sections.

Speed-constrained Multi-objective Particle Swarm Optimisation (SMPSO)

Particle swarm optimization is the bio-inspired metaheuristic optimization algorithm introduced by Eberhart and Kennedy in 1995 [62]. It is categorized as a swarm-based optimization approach influenced by bird flocking. In the Particle Swarm Optimization method, the particle is referred to as each possible solution, and the swarm is the population of solutions in the objective space.

In PSO, each particle is randomly distributed in the state space with the initial particle (\vec{p}_0) and an initial velocity (\vec{v}_0). For every iteration, all the individual particles record their best solution \vec{p}_{best} . As the particle starts to move, the new velocity \vec{v}_{n+1} , and the particle’s next solution \vec{p}_{n+1} is updated as per the Equations 12, 13, and 14 until the maximum number of generation reaches [63].

$$\vec{x}_i(q) = \vec{x}_i(q-1) + \vec{v}_i(q) \quad (12)$$

$$\vec{v}_i(q) = \omega \cdot \vec{v}_i(q-1) + C_1 \cdot r_1 (\vec{x}_{pi} - \vec{x}_i) + C_2 \cdot r_2 (\vec{x}_{gi} - \vec{x}_i) \quad (13)$$

$$\vec{p}_{n+1} = \vec{p}_n + \vec{v}_{n+1} \quad (14)$$

The notations used in the above Equations are adapted from [63]. Speed Constrained Multi-Objective Optimization (SMPSO) is an improved version of the PSO algorithm [63], which facilitates the quicker exploration of the Pareto front and a more uniform examination of the search region.

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \quad (15)$$

$$v_{i,j}(t) = \begin{cases} \delta_j & \text{if } v_{i,j}(t) > \delta_j \\ -\delta_j & \text{if } v_{i,j}(t) \leq -\delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \quad (16)$$

As the name suggests, the SMPSO algorithm constrains the motion of the particles to avoid swarm explosion; swarm explosion makes the algorithm inefficient by skipping the non-visited region. This approach updates the new velocity of the particles from PSO by multiplying it with the constriction coefficient. The Equation for the constriction coefficient and dimensional speed range [63] is mentioned in Eqs. 15, and 16. Also, the pseudocode for the implemented SMPSO is shown in Algorithm 3.

Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

Genetic algorithm (GA) is a search heuristic influenced by Charles Darwin’s theory of biological evolution, and it is classified as an evolution-inspired metaheuristic optimization technique. NSGA-II is an improved version of the genetic algorithm proposed by Deb et al. [64]. It is the most prevalent form of GA, mainly used for multi-objective optimization. The NSGA-II algorithm is built on four operators: selection, crossover, mutation, and crowding distance. The selection operator is responsible for choosing solutions from a pool of options by eliminating the incompetent ones. The crossover operator creates a new copy by combining the good substrings from parent populations. By doing a local search,

Algorithm 3: SMPSO algorithm for identifying Opt-Morph considering Path-length and Area-Coverage

```

1 Input = [α, β, γ];
2 function InitializeSwarm (p0, v0, n);
3 InitializeLeadersArchive();
4 i = 0;
5 // n - number of maximum iterations;
6 while i < n do
7   ComputeVelocity // Equation 10 - 13 ;
8   UpdatePos() // Equation 9 ;
9   PerfofmMutation() ;
10  Evaluation() ;
11  UpdateLeaders() ;
12  UpdateArchive() ;
13  DetermineQuality() ;
14  i ++ ;
15 end
16 returnLeadersArchive();
17 Output =
  [Opt(α, β, γ (Path length(f1), Area covered(f2))];
    
```

the mutation operator enhances the solution. Finally, the crowding distance operator helps maintain diversity among all the populations by supporting the distant solutions. The crowding distance [65] is defined as,

$$C_d = \sum_{j=0}^{N_{obj}} \frac{F_j^{d+1} - F_j^{d-1}}{(F_j^d)_{max} - (F_j^d)_{min}} \tag{17}$$

The pseudocode for the implemented NSGA-II is shown in Algorithm 4. For the detailed scheme of NSGA-II, please refer to the article [64].

Algorithm 4: NSGA-II for identifying Opt-Morph

```

1 Input = [α, β, γ];
2 InitializePopulationPo ;
3 GenerateRandomPopulationSize - N';
4 EvaluateObjectiveFunctions;
5 RankPopulationBasedOnParetoFront;
6 GenerateChildPopulationselection, crossover, mutation;
7 i = 0;
8 while i < size - N' do
9   for every parent and child in the population do
10    AssignRank;
11    Generate set of non - dominated solutions;
12    Determine Cd();
13    loop until N';
14    Choose points with higher Cd;
15    Create(Binary tournament selection, recombined and mutation) i ++ ;
16  end
17 end
18 Output =
  [Opt(α, β, γ (Path - length(f1), Area - covered(f2))];
    
```

Algorithm 5: MACO for identifying Opt-Morph

```

1 Input = [α, β, γ];
2 InitializePheromone τ0, N';
3 i = 0;
4 while i < size - N' do
5   for each colony do
6     for each ant do
7       ConstructSolution(Probabilisticscheme) //
        Equation 16;
8     end
9   end
10  for each solution in the node do
11    PerformLocalDaemonSearch;
12  end
13  UpdatePheromoneStructure //Equation 17;
14  i++;
15 end
16 Output =
  [Opt(α, β, γ (Path - length(f1), Area - covered(f2))];
    
```

Multi-objective Ant Colony Optimization (MACO)

Ant Colony Optimization (ACO) is a swarm-based meta-heuristic method developed by Marco Dorigo in 1996 [66, 67], inspired by the behavior of real ants and their pheromone characteristics. It is mainly used for solving complex combinatorial optimization problems.

In ACO, all the pheromone values are initially assigned with a value of τ_0 , and the algorithm starts with constructing a set of possible solutions. The algorithm chooses a feasible solution during each construction and adds it to the partial solution. The feasible solution selection is based on the probability scheme described in Eq. 18. Further, the local demon search is performed to improve the solutions constructed by the ant. Finally, the pheromone values are increased with respect to the components found with the high-quality solution as described in the Eq. 19,

$$p(c_i^j | s_p) = \frac{\tau_{ij}^\alpha \cdot [\eta(c_i^j)]^\beta}{\sum_{c_i^l \in N(s_p)} \tau_{ij}^\alpha \cdot [\eta(c_i^l)]^\beta}, \quad \forall c_i^j \in N(s_p) \tag{18}$$

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \tag{19}$$

The pseudocode for the implemented multi-objective ACO is detailed in the Algorithm 5. For the detailed scheme of MACO, please refer to the article [68,69].

In the next section, the results obtained from the meta-heuristic algorithms discussed here are highlighted and discussed.

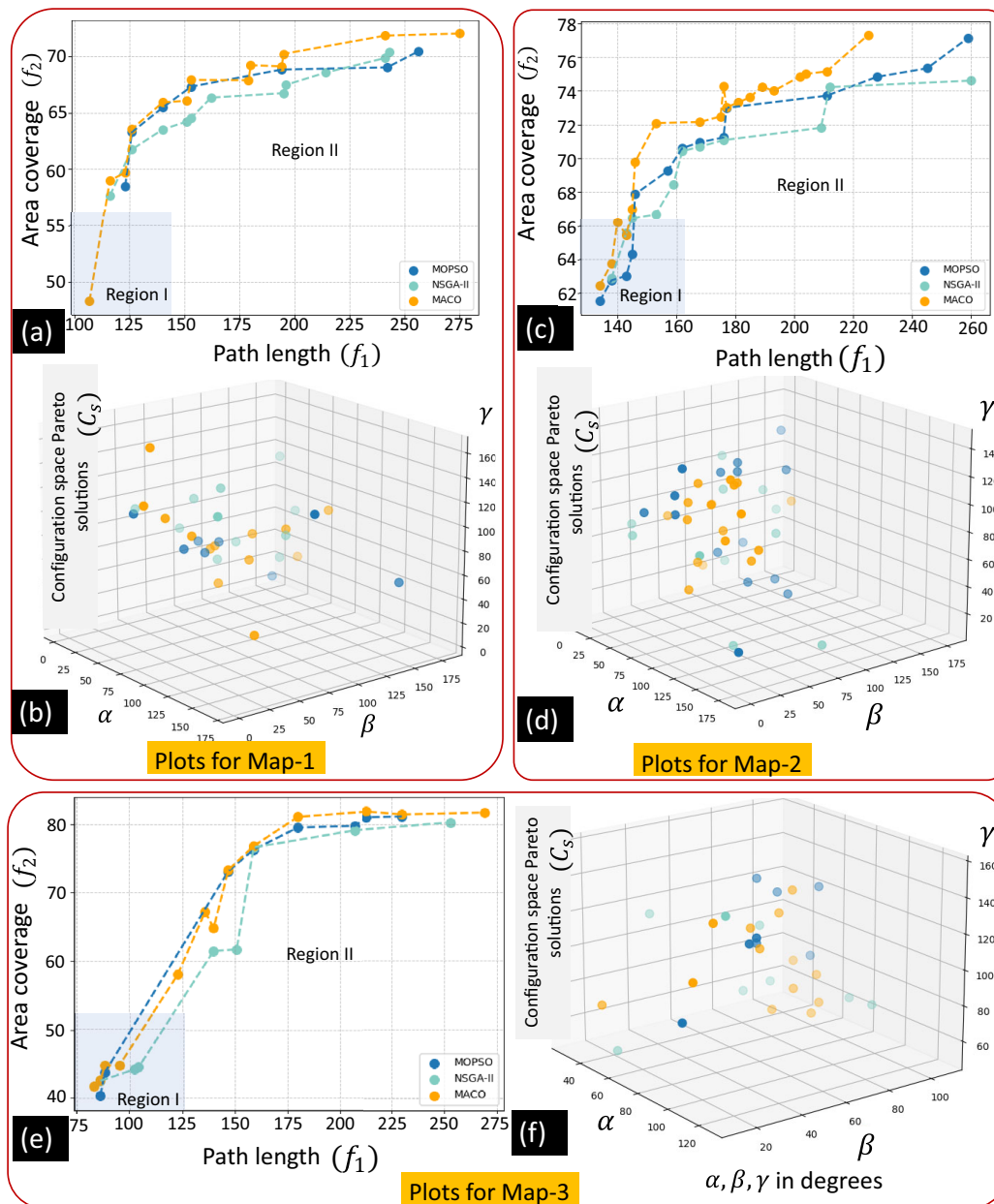


Fig. 8 Simulated Pareto front solutions using MOPSO, NSGA-II, and MACO in three different environments, i.e., Map-1, -2, and -3

Results and discussion

The simulation results and validation of the defined problem using the three metaheuristic multi-objective optimization methods are discussed in this section. The simulation was performed on three different maps, each with a size of 500×500 cm. Each map is unique in terms of the obstacle shape, position, and orientation. In this approach, we chose a set of parameters to provide a fair comparison between the three methods. SMPSO method uses a particle size of 100. Similarly, NSGA-II and MACO use an internal population size of 100; and the number of generations is set to 100 for

all approaches. We applied polynomial mutation and simulated binary crossover (SBX) as operators for mutation and crossover in the NSGA-II algorithm, using binary tournament selection for the parents. We discuss the optimal shapes obtained using the three algorithms next.

Effect of the heuristic approach used for identifying Opt-Morph

Figure 8a–f depict the set of non-dominated solutions and the design space for the SMPSO, NSGA-II, and MACO algorithms in Maps-1, -2, and -3. The α , β , and γ in the design

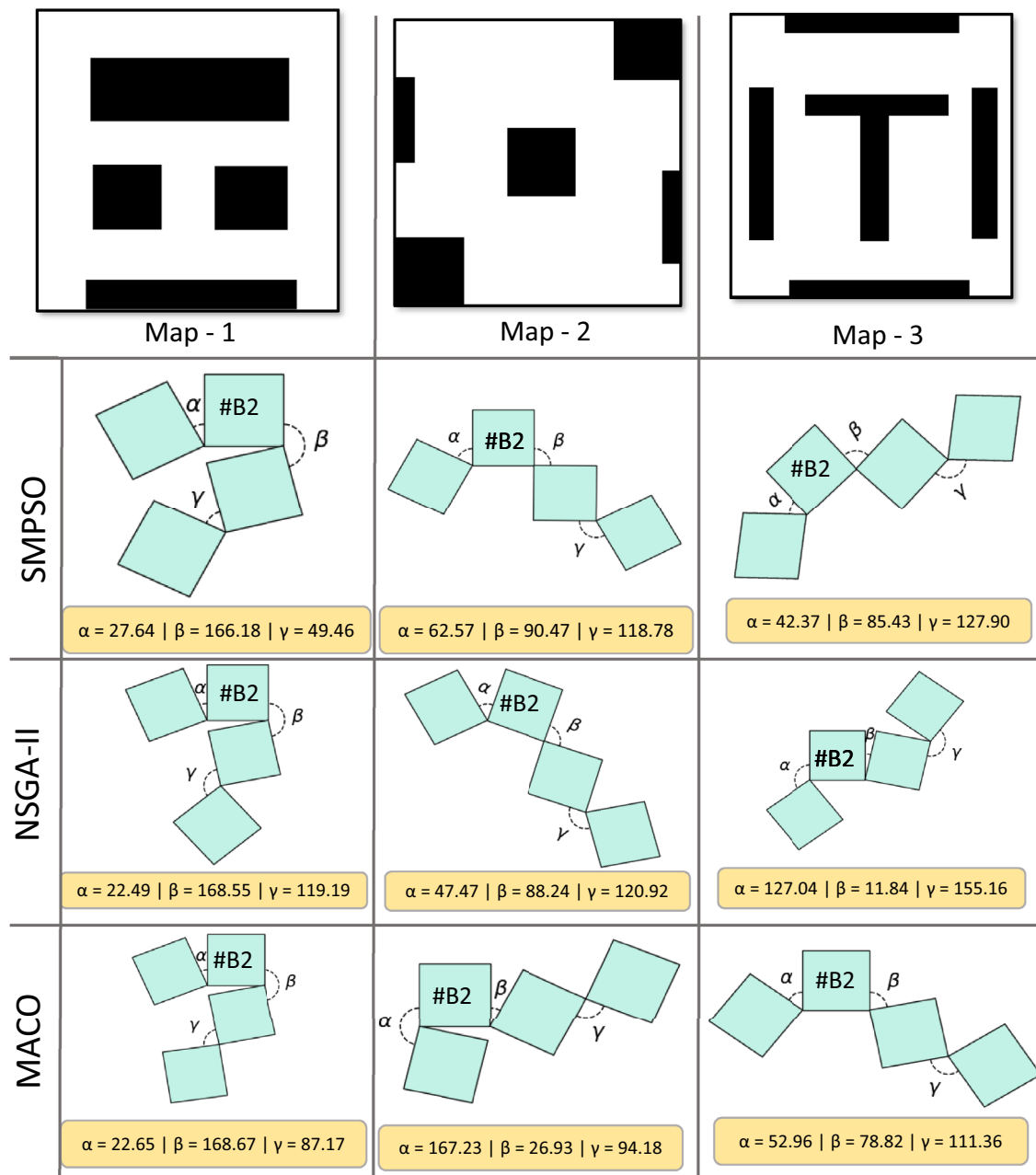


Fig. 9 Generated optimal morphologies for *Smorphi* out of multiple configurations for three different settings, Map-1, -2, and -3

space represent the three hinge angle of the Smorphi robot. In the objective space, f_1 and f_2 denote the path length and the percentage of area covered, respectively. Similarly, Fig. 9 illustrates some of the Smorphi robot’s optimal morphology Table: Please specify the significance of the symbol [bold] reflected inside Table [2, 3] by providing a description in the form of a table footnote. Otherwise, kindly amend if deemed necessary.taken from the computed Pareto front.

From the outcome of the heuristic approaches, it was observed that the non-dominated solutions are mainly pop-

ulated in the region-II with path-length (f_1) values between 125 and 275 cm and the percentage of area-covered (f_2) values between 50 and 78, respectively, as shown in the Fig. 8a, c, e. The NSGA-II algorithm generated fewer non-dominated solutions than the SMPSO and MACO in maps -1, -2, and -3, as shown in Fig. 8a, c, e. Region-I and region-II in the Pareto front mainly distinguishes the morphologies based on task efficiency. Over here, we differentiated region-1 and 2 in the Pareto front by considering cleaning as the main task the robot has to perform. The morphology in the region-1

Table 2 Comparison on the f_1 and f_2 based on the footprint size ($C_{s,fp}$) in C_s and PF C_s for Map-1

N	$C_s(\alpha, \beta, \gamma)$ (degree)	w_{fp} (cm)	l_{fp} (cm)	$C_{s,fp}$ (cm) ²	f_1 (cm)	f_2 (%)	PF $C_s(\alpha, \beta, \gamma)$ (degree)	w_{fp} (cm)	l_{fp} (cm)	$C_{s,fp}$ (cm) ²	f_1 (cm)	f_2 (%)
1	(100, 100, 100)	37	31	1147	157	63.81	(146.98, 166.17, 49.46) ¹	25	31	775	256	70.43
2	(58, 120, 100)	36	36	1296	126	60.16	(74.28, 41.24, 82.81) ¹	50	19	950	194	68.83
3	(40, 122, 42)	32	35	1120	173	64.04	(23.06, 40.29, 100.92) ²	50	16	800	243	70.4008
4	(40, 40, 42)	48	18	864	201	64.73	(22.49, 168.54, 119.18) ²	25	35	875	214	68.57
5	(0, 68, 0)	36	28	1008	180	61.34	(61.03, 36.23, 105.15) ³	50	20	1000	195	70.23
6	(25, 92, 0)	33	30	990	177	61.66	(114.28, 67.96, 18.24) ³	38	29	1102	180	69.22

C_s : is the configuration space defined by angles α, β, γ in degrees (refer Fig. 5)

$C_{s,fp}$: Footprint area by the configuration space, PF C_s : Pareto front configuration space

¹ SMPSO:

² NSGA-II:

³ MACO

Table 3 Dimensions, i.e., length l_{fp} and width w_{fp} of the configuration space footprint $C_{s,fp}$ (Fig. 5 of the tiling shapes I, J, O, S, T, Z

N	$C_s(\alpha, \beta, \gamma)$ (degree)	w_{fp} (cm)	l_{fp} (cm)	$C_{s,fp}$ (cm) ²	f_1 (cm)	f_2 (%)
1	(0, 0, 0) \equiv I	40	10	400	414	65.92
2	(0, 0, 180) \equiv J	30	20	600	303	66.84
3	(0, 180, 0) \equiv O	20	20	400	452	70.88
4	(180, 90, 0) \equiv S	20	30	600	309	65.97
5	(90, 180, 180) \equiv T	20	30	600	309	66.07
6	(180, 0, 180) \equiv Z	20	30	600	309	66.53

is considered less suitable for cleaning applications as they can only cover less percentage of the area in a given map as shown in Fig. 8a, c, e. In contrast, the morphology produced in region-2 is more suitable for cleaning application, as they have a higher percentage of area-covered (f_1) with the optimal path length (f_1) compared to the morphology produced in region-I. Table 2 highlights some of the values from region-II obtained for Map-1 with the magnitude of the configuration space (α, β, γ) and the objectives f_1 and f_2 . Besides, the finding indicates that the results produced by the MACO algorithm are superior to SMPSO and NSGA-II for generating optimal morphology, which compromises the path-length and the percentage of area covered. Overall, all three algorithms converged as per the objective function for generating optimal energy-efficient morphology. Similarly, in the configuration space highlighted in Fig. 8b, d, f, $\alpha, \beta,$ and γ values corresponding to the f_1 and f_2 are randomly distributed in the range of [0,180]. However, these results mentioned are for 100 populations and 100 generations and may vary with the different number of populations and generations.

Effects of morphologies

In addition, to validate the simulated findings, we compared the Pareto front to the widely adopted standard polyomino configurations in tiling robots as reported in [21,23]. Our simulated finding suggests that these standard polyomino configurations are considered to be sub-optimal compared to the shapes generated from our algorithm in terms of the path length and the percentage of area-covered as shown in Table 2, and 3.

For example, in Table 3, row 3, the standard polyomino configuration ‘O’ with $(\alpha, \beta, \gamma) = (0, 180, 0)$ covered 70.88% of the area in the map-1 with a path-length of 452 cm. Even though the shape has a higher percentage of area covered, the path length required to cover that given map is also higher, implying the energy consumption is also higher. On the other hand, the shape generated by the MACO algorithm with the PF $C_s(\alpha, \beta, \gamma = 114.28, 67.96, 18.24)$ (Table 2, row 6); path-length ($f_1 = 180$ cm) and percentage of area-covered ($f_2 = 69.22\%$) is produced in the optimal range as per the objective function. Hence this shape is considered optimal morphology for Map-1 compared to the standard polyomino configuration ‘O’. Further, this reveals the usefulness of infinite configurations in area coverage problems.

Further, to validate our simulated findings, we compared the Pareto front solutions with the random configuration chosen from the Smorphi robot. Table 4 shows the path length and the area covered for the set of random shapes. The results indicate that the random shapes taken from the configuration space are also considered sub-optimal compared to the morphology generated in the Pareto front (Table 2). For example, the random configuration with the $C_s(\alpha, \beta, \gamma = 16.26, 1.37, 22.60)$ (Table 4, row 5) covered 69.41% of the area(f_2) with the path-length of 282 cm(f_2), whereas the configuration produced by the MACO algorithm PF $C_s(\alpha, \beta, \gamma = 114.28, 67.96, 18.24)$ (Table 2, row 6) covered the area of 69.22% (f_2) with the much lesser path

Table 4 Dimensions, i.e., length l_{fp} and width w_{fp} of the random configuration space footprint $C_{s,fp}$

N	Random $C_s(\alpha, \beta, \gamma)$ (degree)	w_{fp} (cm)	l_{fp} (cm)	$C_{s,fp}$ (cm) ²	f_1 (cm)	f_2 (%)
1	(134.13, 156.07, 2.50)	28	27	756	205	58.65
2	(42.82, 49.74, 44.34)	48	22	1056	170	56.74
3	(115.31, 42.75, 130.63)	43	23	989	191	61.94
4	(122.01, 94.04, 113.86)	40	30	1200	138	58.31
5	(16.26, 1.37, 22.60)	45	16	720	282	69.41
6	(109.20, 20.02, 15.29)	42	22	924	187	59.59

length ($f_1 = 180$ cm) compared to the random configuration. Similarly, in Table 4 the random configuration with the $C_s(\alpha, \beta, \gamma = 115.31, 42.75, 130.63)$ (Table 4, row 3) has a path-length ($f_1 - 191$ cm), and the percentage of area covered as ($f_2 - 61.94\%$). Even though this shape has a shorter path length (f_1), the percentage of area covered (f_2) is lower when compared to the morphology produced in the Pareto front $C_s(\alpha, \beta, \gamma = 61.03, 36.23, 105.15)$ (Table 2, row 5), which covers significantly more area than the random configuration ($f_2 = 68.33\%$) for nearly the same path-length ($f_1 - 195$ cm), making it a more optimal configuration for area coverage tasks with lesser energy consumption. These results substantiate the multi-objective optimisation algorithm's usefulness in identifying the optimal morphology.

Effects of the footprint size

From the simulation, we identified that the footprint size is inversely proportional to the path length; as the footprint size is larger, the path length is shorter. This makes the shape with a larger footprint a suitable candidate for the optimal morphology, and the morphology with a lesser footprint size (I, J, O, S, T, Z) is the least suitable candidate for the optimal morphology, as shown in the Tables 3, and 2. However, as per our objective function, the morphology with the shorter path length alone cannot be optimal. Since the problem is defined in such a way that the optimal shape generated is the trade-off between path length and percentage of area covered. On the other hand, the area coverage solely relies on the footprint occupancy matrix and the path generated rather than the footprint size.

As shown in Table 2, shapes from the Pareto front are verified with the random values chosen from the configuration space (C_s). The results show that even the larger footprint size alone cannot be an optimal morphology. For example, in Table 2, row 2, the random value chosen from the $C_s(\alpha, \beta, \gamma = 58, 120, 100)$ has a lesser path length ($f_1 = 126$) with lower energy consumption; however, for that shape, the percentage of area covered ($f_2 = 60.16\%$) is also lesser, making it as a sub-optimal morphology. Sim-

ilarly, in the case of the shape generated from the PF $C_s(\alpha, \beta, \gamma = 74.28, 41.24, 82.81)$. Table 2, row 2) has the optimal path-length ($f_1 = 194$) and the percentage of area-covered ($f_2 = 68.83\%$), making it an optimal morphology for map-1. Hence, the metaheuristic algorithms were useful in identifying the optimal value that compromises the percentage of area covered and path-length. Further, this example justifies the use case of these meta-heuristic algorithms for this combinatorial problem.

Heuristic algorithms and computational time

Figure 10 summarizes the computational time taken³ by the SMPSO, NSGA-II and MACO algorithms to complete the 100 generations in three different simulated environments. Among these three methods, the SMPSO algorithm has less computational cost than the other two algorithms; this is mainly because SMPSO uses mathematical operators instead of evolutionary operators for generating solutions. Similarly, in Map-2, the NSGA-II algorithm converges faster than the MACO and SMPSO algorithms. Further, from Fig. 10, we can observe that the MACO algorithm slightly took more computational cost than the other two algorithms in the three different maps. In addition, we observed that there are only slight changes in the computational cost between different environments even though the obstacle complexity in every map is different; this is mainly due to the size of the map being kept constant for the three simulations. However, on the other hand, heuristic-based optimization algorithms are stochastic in nature; we cannot conclude one type of optimization algorithm will always produce the best solution with less computational cost; it varies with respect to the environmental uncertainty, design parameters, nature of the algorithm, type of problem, computational power, etc.

Experiments

This section details the experiments conducted to validate the simulation study using the real *Smorphi* robot. The experiments were carried out in a $3\text{ m} \times 3\text{ m}$ lab setup designed to resemble the simulated environment. Occupancy grids such as warning signs and cardboards are considered obstacles, and the remaining space in the bounding box is considered free space for the robot to navigate. The real experiments aim to show the optimal morphology generated through the simulation is energy efficient to cover the area while maximizing the area coverage and minimizing the path length. We conducted four set of experiments, including shapes from standard polyomino sets, SMPSO, NSGA-II, and MACO. For all the experiments, the robot is teleoperated with the

³ PC specifications used during optimization: Intel Core i7 4-Core (11th Gen), 2.8 GHz, 512GB SSD and 16GB RAM.

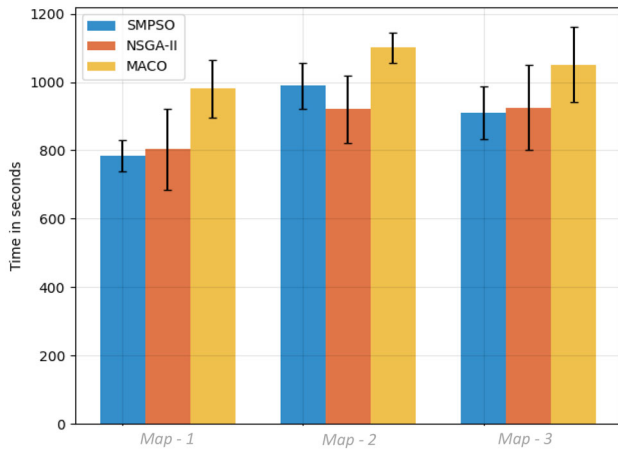


Fig. 10 Computational time taken by the optimization algorithm in the three different environments ran on a given personal computing device

same speed of 0.1 m/s. For every shape, the path length (f_1) and the percentage of area covered (f_2) are tracked using the overhead camera mounted on the ceiling. Besides, during the experiment, the angle between the hinges is locked using an acrylic mechanical stopper to restrict the hinges during navigation.

Figures 11, 12, 13, and 14 depict actual experiments performed using four different shapes. To fairly compare the three optimization algorithms, among the multiple morphologies generated from the Pareto front, we chose the morphology with the highest area coverage and lesser path length for the real-world experiment from the respective algorithm. The path-length (f_1) and percentage of area-covered (f_2) for each of the four shapes are shown in Table 5. The results indicate that the forms taken from the Pareto front outperformed the standard polyomino configuration. Furthermore, the shape generated from the MACO algorithm has a higher percentage of area covered with lesser path-length

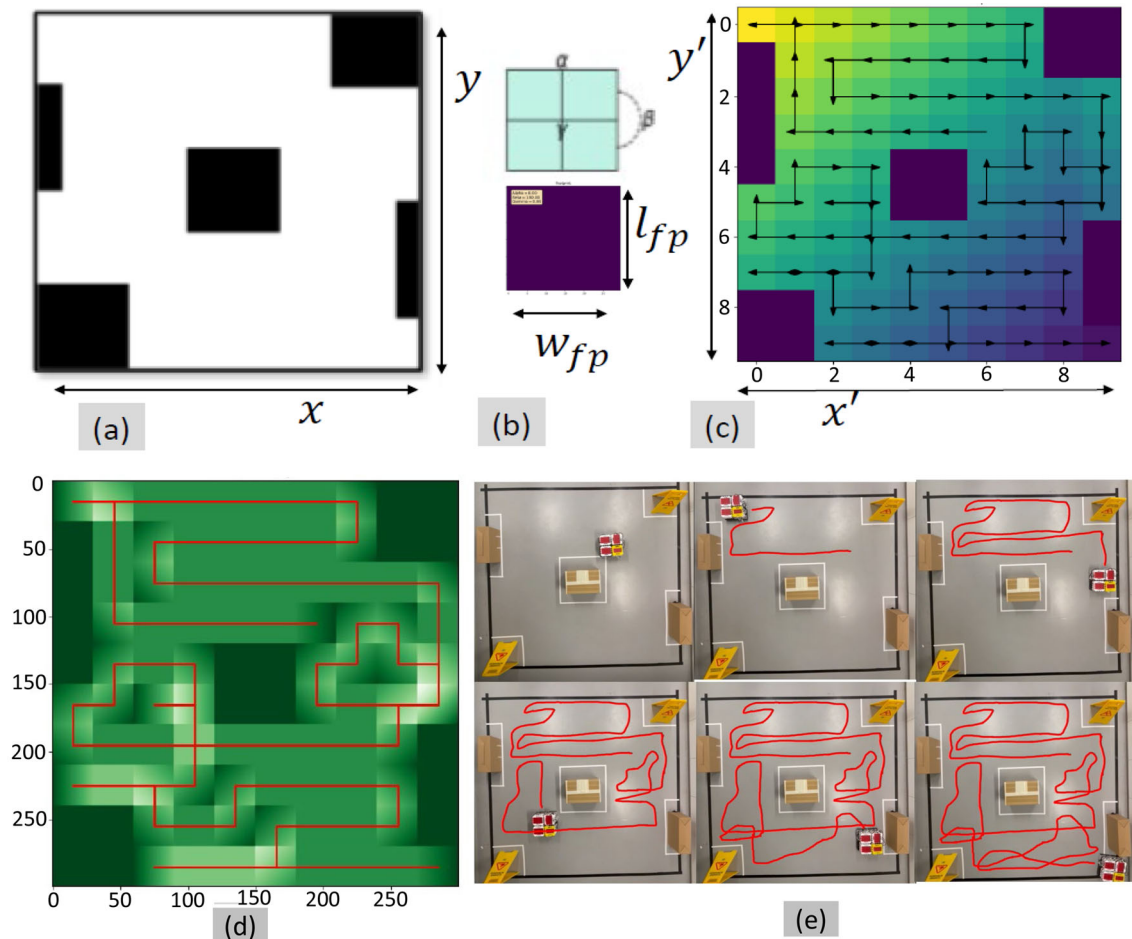


Fig. 11 Standard morphology as O-shape with **a** Map-2 with obstacles, **b** Footprint of the O-shape, **c** Path-planner output with start (S) and goal (G) points indicated, **d** Area-coverage with the selected morphology,

and **e** The experimental overlay map with the path length of 92 cm and the percentage of area-coverage as 79.2%

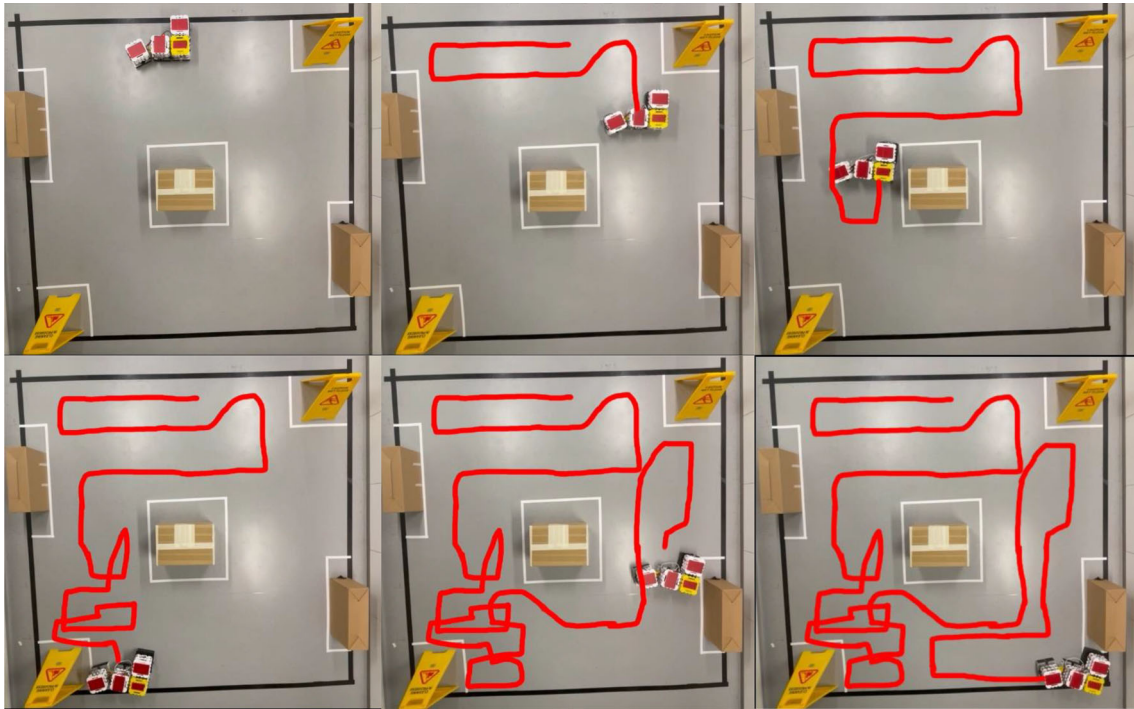


Fig. 12 Real-world experiment using the optimal morphology (refer Table 5) generated from the SMPSO algorithm

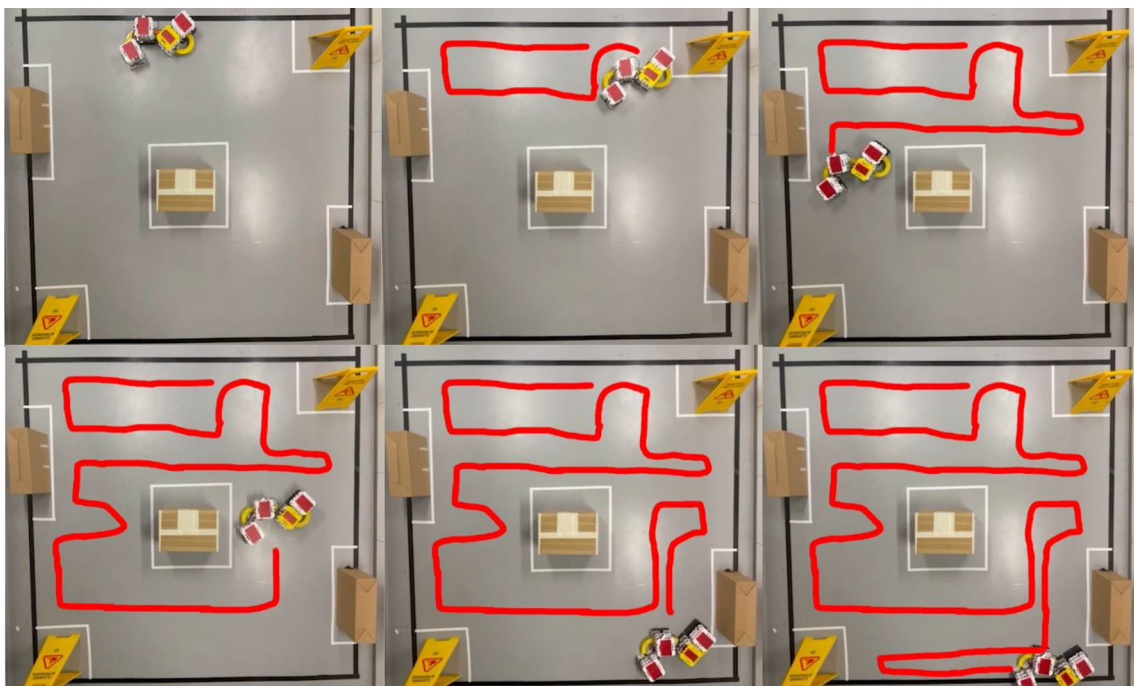


Fig. 13 Experimental results using the optimal morphology (refer Table 5) generated from the NSGA-II algorithm



Fig. 14 Real-world experiment using the optimal morphology (refer Table 5) generated from the MACO algorithm

Table 5 Actual and simulated path-length(f_1) and the percentage of area-covered(f_2) for four different morphologies

C_s (α, β, γ) Algorithm (degree)	Path-length (cm)		Area-covered (%)	
	Simulation	Actual	Simulation	Actual
(0,180,0) \equiv O, Standard shape	92	92	81	79.2
(28.19, 0,180) SMPSO	44	44	79.2	74.87
(117.52, 41.28, 177.37) NSGA-II	34	34	77.9	74.52
(122.62, 54.94, 179.26) MACO	37	37	80.5	76.2

compared to the other two algorithms. Besides, as mentioned in Table 5 the difference in the area coverage between the actual and simulation is mainly due to the wheel slippage and the manual error during the teleoperation. When the robot is commanded to move in a straight line, the wheel slippage causes the robot to drift from the global path causing it to leave some area uncovered. In conclusion, the experiment results satisfy our simulation finding for generating optimal energy-efficient morphology.

Overall, the suggested framework can determine the optimal morphology by making trade-offs between the path length and the percentage of area covered using the metaheuristic algorithm. Moreover, this work revealed how each possible shape in a reconfigurable robot could be exploited for area coverage applications.

Conclusion

Given the current limitations of the existing approach for reconfigurable robots in performing efficient cleaning with optimal energy consumption, we propose a framework for generating an optimal morphology of a reconfigurable robot. The reconfigurable robot considered in this work consists of four blocks and is referred to as *Smorphi*. The robot can take several morphologies apart from the set of standard polyomino shapes, i.e., I, J, L, O, S, T, Z. The objective was to maximize the area coverage and minimize energy consumption. The proposed framework is validated through simulation and real-world experiments using three different metaheuristic algorithms: SMPSO, NSGA-II, and MACO. Moreover, the proposed method can be extended for different classes of reconfigurable robots to select an optimal configuration depending on the given application such as cleaning, surveillance or reconnaissance task, among others.

Furthermore, insight into the optimal morphologies is gained through the simulation and experimental results. The

results show that the each approach finds a near-optimal energy-efficient morphology for the given map. Besides, the shape generated from the MACO approach is superior to the other two algorithms, which can be considered a suitable algorithm for generating the optimal-energy efficient morphology. Overall, the proposed method opens a new avenue for further research on identifying Opt-Morph for different classes of reconfigurable robots. The future work includes a reinforcement learning-based approach to determine the optimal shape in reconfigurable robots and developing a platform-agnostic framework for optimal shape generation in a different class of reconfigurable robots, developing a suitable control strategy, and accounting for the rotational and diagonal motion of the reconfigurable robot in the global path planner.

Author Contributions Manivannan Kalimuthu conceptualized, designed, acquired/analyzed/interpreted data, created codes and drafted the manuscript as part of the reported work. Thejus Pathakumar and Abdullah Aamir Hayat designed, interpreted data, created codes and drafted the manuscript as part of the reported work. Mohan Rajesh Elara and Kris L. Wood supervised the work and reviewed the article.

Funding This research is supported by the National Robotics Program under its Robotics Enabling Capabilities and Technologies (Funding Agency Project No. 1922500051), National Robotics Program under its Robot Domain Specific (Funding Agency Project No. 192 22 00058), National Robotics Program under its Robotics Domain Specific (Funding Agency Project No. 1922200108), and administered by the Agency for Science, Technology and Research. We wish to acknowledge the support of the SUTD DesignZ Center, the Comcast Media and Technology Center (CU Denver), and the College of Engineering, Design, and Computing (CU Denver)."

Data availability The data used during the current study are available from the first author and corresponding author upon reasonable request.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest

Ethics approval Not applicable.

Consent to participate: Not applicable.

Consent for publication: Yes

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copy-

right holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Ferguson S, Siddiqi A, Lewis K, de Weck OL (2007) Flexible and reconfigurable systems. *Nomenclat Rev* 48078:249–263
2. Koren Y et al (1999) Reconfigurable manufacturing systems. *CIRP Ann* 48(2):527–540
3. Lysaght P, Stockwood J (1996) A simulation tool for dynamically reconfigurable field programmable gate arrays. *IEEE Trans Very Large Scale Integrat (VLSI) Syst* 4(3):381–390
4. Compton K, Hauck S (2002) Reconfigurable computing: a survey of systems and software. *ACM Comput Surv (csuR)* 34(2):171–210
5. Costantine J, Tawk Y, Barbin SE, Christodoulou CG (2015) Reconfigurable antennas: design and applications. *Proc IEEE* 103(3):424–437
6. Fukuda T, Nakagawa S (1987) A dynamically reconfigurable robotic system (concept of a system and optimal configurations) *SPIE* 856:588–595
7. Tan N, Hayat AA, Elara MR, Wood KL (2020) A framework for taxonomy and evaluation of self-reconfigurable robotic systems. *IEEE Access* 8:13969–13986
8. Castano A, Behar A, Will PM (2002) The conro modules for reconfigurable robots. *IEEE/ASME Trans Mech* 7(4):403–409
9. Rus D, Vona M (2000) A basis for self-reconfiguring robots using crystal modules. *IEEE* 3:2194–2202
10. Østergaard EH, Kassow K, Beck R, Lund HH (2006) Design of the atron lattice-based self-reconfigurable robot. *Autono Robots* 21(2):165–183
11. Hayat AA, Yi L, Kalimuthu M, Elara M, Wood KL (2022) Reconfigurable robotic system design with application to cleaning and maintenance. *J Mech Design* 144(6):063305
12. Pathmakumar T, Sivanantham V, Anantha Padmanabha SG, Elara MR, Tun TT (2021) Towards an optimal footprint based area coverage strategy for a false-ceiling inspection robot. *Sensors* 21(15):5168
13. Ilyas M, Yuyao S, Mohan RE, Devarassu M, Kalimuthu M (2018) Design of stetro: a modular, reconfigurable, and autonomous staircase cleaning robot. *J Sens* 2018
14. Yuyao S, Elara MR, Kalimuthu M, Devarassu M (IEEE, 2018) stetro: a modular reconfigurable cleaning robot 1–8
15. Tun TT, Elara MR, Kalimuthu M, Vengadesh A (2018) Glass facade cleaning robot with passive suction cups and self-locking trapezoidal lead screw drive. *Automat Construct* 96:180–188
16. Ramalingam B et al (2021) Stetro-deep learning powered staircase cleaning and maintenance reconfigurable robot. *Sensors* 21(18):6279
17. Le AV, Veerajagadheswar P, Thiha Kyaw P, Elara MR, Nhan NHK (2021) Coverage path planning using reinforcement learning-based tsp for htetran-a polyabolo-inspired self-reconfigurable tiling robot. *Sensors* 21(8):2577
18. Samarakoon SBP et al (2021) Modelling and control of a reconfigurable robot for achieving reconfiguration and locomotion with different shapes. *Sensors* 21(16):5362
19. Sinha A, Tan N, Mohan RE (2014) Terrain perception for a reconfigurable biomimetic robot using monocular vision. *Robot Biomimet* 1(1):1–11
20. Rubenstein M, Payne K, Will P, Shen W-M (2004) Docking among independent and autonomous conro self-reconfigurable robots 3:2877–2882
21. Prabakaran V, Elara MR, Pathmakumar T, Nansai S (2017) htetro: A tetris inspired shape shifting floor cleaning robot 6105–6112

22. Golomb SW (1996) Polyominoes: puzzles, patterns, problems, and packings Vol. 111
23. Prabakaran V, Elara MR, Pathmakumar T, Nansai S (2018) Floor cleaning robot with reconfigurable mechanism. *Autom Construct* 91:155–165
24. Cheng KP, Mohan RE, Nhan NHK, Le AV (2020) Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot. *IEEE Access* 8:121267–121284
25. Samarakoon SBP, Muthugala MVJ, Le AV, Elara MR (2020) hTetro-infi: a reconfigurable floor cleaning robot with infinite morphologies. *IEEE Access* 8:69816–69828
26. Hayat AA, Karthikeyan P, Vega-Heredia M, Elara MR (2019) Modeling and assessing of self-reconfigurable cleaning robot htetro based on energy consumption. *Energies* 12(21):4112
27. Le AV et al (2019) Realization energy optimization of complete path planning in differential drive based self-reconfigurable floor cleaning robot. *Energies* 12(6):1136
28. Le AV et al (2020) Reinforcement learning-based energy-aware area coverage for reconfigurable hrombo tiling robot. *IEEE Access* 8:209750–209761
29. Cheng KP, Mohan RE, Nhan NHK, Le AV (2019) Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots. *IEEE Access* 7:94642–94657
30. Le AV, Prabakaran V, Sivanantham V, Mohan RE (2018) Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor. *Sensors* 18(8):2585
31. Galceran E, Carreras M (2013) A survey on coverage path planning for robotics. *Robot Auton Syst* 61(12):1258–1276
32. Lumelsky VJ, Mukhopadhyay S, Sun K (1990) Dynamic path planning in sensor-based terrain acquisition. *IEEE Trans Robot Automat* 6(4):462–472
33. Choset H, Pignon P (1998) Coverage path planning: the boustrophedon cellular decomposition 203–209
34. Moravec H, Elfes A (1985) High resolution maps from wide angle sonar 2:116–121
35. Zelinsky A, Jarvis RA, Byrne J, Yuta S et al (1993) Planning paths of complete coverage of an unstructured environment by a mobile robot 13:533–538
36. Gabriely Y, Rimon E (2002) Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot 1:954–960
37. Xu L (2011) Graph planning for environmental coverage. Carnegie Mellon University
38. Wong SC, MacDonald BA (2003) A topological coverage algorithm for mobile robots 2:1685–1690
39. Atkar PN, Choset H, Rizzi AA, Acar EU (2001) Exact cellular decomposition of closed orientable surfaces embedded in/spl rfr//sup 3 1:699–704
40. Cheng P, Keller J, Kumar V (2008) Time-optimal uav trajectory planning for 3d urban structure coverage 2750–2757
41. Bello I, Pham H, Le QV, Norouzi M, Bengio S (2016) Neural combinatorial optimization with reinforcement learning. [arXiv:1611.09940](https://arxiv.org/abs/1611.09940)
42. Tagliarini GA, Christ JF, Page EW (1991) Optimization using neural networks. *IEEE Trans Comput* 40(12):1347–1358
43. Muthugala MVJ, Samarakoon SBP, Mohan Rayguru M, Ramalingam B, Elara MR (2020) Wall-following behavior for a disinfection robot using type 1 and type 2 fuzzy logic systems. *Sensors* 20(16):4445
44. Savinell JM, Palsson BO (1992) Network analysis of intermediary metabolism using linear optimization. i. development of mathematical formalism. *J Theor Biol* 154(4):421–454
45. Ting T, Yang X-S, Cheng S, Huang K (2015) Hybrid metaheuristic algorithms: past, present, and future. *Recent Adv Swarm Intell Evolut Comput* 71–83
46. Osman IH, Kelly JP (1996) Meta-heuristics: an overview. *Meta-heuristics* 1–21
47. Yu Y et al (2021) Adsorption control of a pipeline robot based on improved PSO algorithm. *Complex Intell Syst* 7(4):1797–1803
48. Geng N, Chen Z, Nguyen QA, Gong D (2021) Particle swarm optimization algorithm for the optimization of rescue task allocation with uncertain time constraints. *Complex Intell Syst* 7(2):873–890
49. Mo Y, You X, Liu S (2022) Multi-colony ant optimization with dynamic collaborative mechanism and cooperative game. *Complex Intell Syst* 1–18
50. Masehian E, Sedighzadeh D (2010) A multi-objective PSO-based algorithm for robot path planning 465–470
51. Wang Y et al (2019) Reconnaissance mission conducted by uav swarms based on distributed PSO path planning algorithms. *IEEE Access* 7:105086–105099
52. Hu Y, Yang SX (2004) A knowledge based genetic algorithm for path planning of a mobile robot 5:4350–4355
53. Albina K, Lee SG (2019) Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms. *IEEE Access* 7:14246–14255
54. Valente J, Del Cerro J, Barrientos A, Sanz D (2013) Aerial coverage optimization in precision agriculture management: a musical harmony inspired approach. *Comput Electron Agric* 99:153–159
55. Tsuzuki MdSG, de Castro Martins T, Takase FK (2006) Robot path planning using simulated annealing. *IFAC Proc* 39(3):175–180
56. Chibin Z, Xingsong W, Yong D (2008) Complete coverage path planning based on ant colony algorithm 357–361
57. Pazooki M, Mazinan A (2018) Hybrid fuzzy-based sliding-mode control approach, optimized by genetic algorithm for quadrotor unmanned aerial vehicles. *Complex Intell Syst* 4(2):79–93
58. Singh V et al (2009) Innovations in design through transformation: a fundamental study of transformation principles. *J Mech Design* 131(8)
59. Vikhar PA (2016) Evolutionary algorithms: a critical review and its future prospects 261–265
60. Kalimuthu M, Hayat A, Elara M, Wood K (2021) Transformation design principles as enablers for designing reconfigurable robots 85420:V006T06A008
61. Ramalingam B et al (2021) Deep learning based pavement inspection using self-reconfigurable robot. *Sensors* 21(8):2595
62. Eberhart R, Kennedy J (1942) Particle swarm optimization 1948
63. Nebro AJ et al (2009) SMPSO: a new PSO-based metaheuristic for multi-objective optimization, 66–73
64. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evolut Comput* 6(2):182–197
65. Wang X, Hirsch C, Kang S, Lacor C (2011) Multi-objective optimization of turbomachinery using improved nsga-ii and approximation model. *Comput Methods Appl Mech Eng* 200(9–12):883–895
66. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans Evolut Comput* 1(1):53–66
67. Dorigo M, Di Caro G (1999) Ant colony optimization: a new metaheuristic 2:1470–1477
68. Biscani F, Izzo D (2020) A parallel global multiobjective framework for optimization: pagmo. *J Open Sour Softw* 5(53):2338
69. Acciarini G, Izzo D, Mooij E (2020) Mhaco: a multi-objective hypervolume-based ant colony optimizer for space trajectory optimization, 1–8

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.