Expert Systems With Applications 201 (2022) 117060

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Metaheuristic based navigation of a reconfigurable robot through narrow spaces with shape changing ability

S.M. Bhagya P. Samarakoon *, M.A. Viraj J. Muthugala, Mohan Rajesh Elara

Engineering Product Development Pillar, Singapore University of Technology and Design, Singapore 487372, Singapore

ARTICLE INFO

Keywords:
Reconfigurable robotics
Grey Wolf Optimizer
Genetic Algorithm
Simulated Annealing
Navigation
Area coverage

ABSTRACT

Emerging technologies and robotics solutions for cleaning applications are rapidly progressing nowadays. Reconfigurable robots that can change shapes are introduced to overcome the limitations of fixed-shape robots. Reconfigurable robots should be capable of moving through narrow spaces with the help of continuous shape-changing to improve the coverage performance. In this regard, a reconfigurable robot requires perceiving the narrow spaces in between obstacles and generating the path to navigate. Most importantly, the robot should decide and change the shape configurations appropriately to move on that path without collisions. Furthermore, such a reconfiguration ability could be used for covering around obstacles. Therefore, this paper proposes a method to navigate a reconfigurable robot through narrow spaces or cover obstacles with the aid of continuous reconfiguration. In the first step, the method analyzes the obstacle clusters in a metric map to determine the appropriate paths to navigate. In the second step, a sequence of shapes for reconfiguration that allows the robot's movement on the path without collisions is determined. Metaheuristics, Grey Wolf Optimizer (GWO), Genetic Algorithm (GA), and Simulated Annealing (SA), are proposed to determine the shape sequence. Simulations have been conducted to validate the performance and behavior of the proposed method. The results confirm that the developed method is effective for navigating a reconfigurable robot through narrow spaces or covering obstacles with continuous shape-changing. According to the statistical outcomes, GWO requires less computational time compared to GA and SA.

1. Introduction

Rapid construction of buildings for residential, public, and office spaces can be seen nowadays (Thuesen & Koch-Ørvad, 2018). Frequent cleaning and maintenance are essential for these buildings to uphold the living quality. However, many countries suffer from the scarcity of laborers for cleaning and inspection tasks. Cleaning robots can be introduced to overcome these issues. Moreover, deploying robots for cleaning and disinfecting public places and quarantine facilities in a pandemic like COVID-19 can reduce the risk of spreading (Khan, Siddique, & Lee, 2020). Floor cleaning (Park & Lee, 2019), furniture cleaning (Elliott & Cakmak, 2018), façade cleaning (Muthugala, Vega-Heredia, Vengadesh, Sriharsha, & Elara, 2019), duct cleaning (Ito, Kawaguchi, Kamata, Yamada, & Nakamura, 2019), and wall cleaning (Muthugala, Vega-Heredia, Mohan, & Vishaal, 2020) are examples of building maintenance applications that utilize robots.

Among these cleaning applications, floor cleaning plays a major role. Commercially available fixed-shape robots intended for floor cleaning can be commonly found in many places (Kim et al., 2019; Romero-Martí, Núnez-Varela, Soubervielle-Montalvo, & Orozco-de-la Paz, 2016). Area coverage and energy usage of these robots have been extensively studied (Rosenstein, Chiappetta, Schnittman, & Pastore, 2016; Zheng et al., 2017). Navigation and path planning algorithms for fixed-shape cleaning robots have also been developed (Liu, Ma, & Huang, 2018; Zhao, Chen, Peter, & Wu, 2016). Although fixed-shape robots are used in public and residential premises, the productivity of the robots is limited due to the difficulty in navigating through narrow resulting a lower area coverage. Robot-inclusive spaces design guidelines have been defined in Elara, Rojas, and Chua (2014) to improve the area coverage and the coverage time for fixed-shape robots. In this work, commonly occupied obstacles in a typical domestic environment with default and robot-inclusive settings have been considered for the case study. Higher performance can be obtained in robot-inclusive space design than the default setting. However, it is not always convenient for users to rearrange their environment based on these guidelines. Moreover, a user expectation is to deploy a cleaning robot and obtain higher area coverage by the robot itself without

* Corresponding author.
E-mail addresses: bhagya_samarakoon@mymail.sutd.edu.sg (S.M.B.P. Samarakoon), viraj_jagathpriya@sutd.edu.sg (M.A.V.J. Muthugala), rajeshelara@sutd.edu.sg (M.R. Elara).

https://doi.org/10.1016/j.eswa.2022.117060
Received 17 August 2021; Received in revised form 14 December 2021; Accepted 28 March 2022
Available online 12 April 2022

additional infrastructure modifications. In addition to that, users expect the robot to perceive and adapt to a given environment and perform the cleaning.

Yim (2007) discussed capabilities, applications, and future challenges of reconfigurable robots. Based on these inspirations, floor cleaning robots with reconfigurable capabilities have been introduced to resolve the limitations of fixed-shape robots. A study has been conducted to evaluate the area coverage performance of a reconfigurable cleaning robot and a commercially available fixed-shape robot (Prabakaran, Elara, Pathmakumar, & Nansai, 2017). According to the outcomes of the study, the reconfigurable floor cleaning robot outperformed the fixed-shape robot in area coverage. Therefore, different types of reconfigurable robots have been introduced for floor cleaning applications. hTetro (Prabakaran et al., 2017), hTrihex (Apuroop, Le, Elara, & Sheu, 2021), and hTetrakis (Le, Nhan, & Mohan, 2020) can be considered as examples of state-of-the-art reconfigurable floor cleaning robots. Area coverage improvement and shortest path for navigation have been extensively investigated in these reconfigurable robots (Cheng, Mohan, Nhan, & Le, 2020; Lakshmanan et al., 2020; Le, Prabakaran, Sivanantham, & Mohan, 2018). Energy consumption and coverage are vital factors for a cleaning robot. In this regard tradeoff between area coverage and energy of a reconfigurable robot has also been studied in Muthugala, Samarakoon, and Elara (2020). Apart from shape-changing, changing the footprint size of a reconfigurable robot to improve the area coverage and coverage time have been studied by Samarakoon, Muthugala, Elara, et al. (2021). Even though the cited robots are capable of changing shapes, the shape-changing capability is limited to a few predefined shapes. Furthermore, the exact geometric shapes of obstacles are not considered during area coverage and navigation. To cope with these issues, the work (Samarakoon, Muthugala, & Elara, 2021; Samarakoon, Muthugala, Le, & Elara, 2020, 2021) provided solutions that the robot consider the exact geometrical shapes and cover obstacles reconfiguring beyond a set of predefined shapes. However, the scope of the cited work limited to covering an obstacle in a single step. Moreover, continuous navigation and reconfiguration required for moving through a narrow passage or covering an obstacle are not possible with the existing approaches.

Extensive application of metaheuristic optimization techniques could often be seen in robotics (Fausto, Reyna-Orta, Cuevas, Andrade, & Perez-Cisneros, 2020; Fong, Deb, & Chaudhary, 2015). Especially, metaheuristic optimization techniques are used in robot navigation and path planning. For example, a multi-objective grasshopper optimization algorithm for robot path planning in static environments has been proposed in Elmi and Efe (2018). The method can minimize the distance, time, and energy of a robot navigation task. The work (Jalali, Khosravi, Kebria, Hedjam, & Nahavandi, 2019) examined the use of six optimization techniques for autonomous wall-following by a robot. This work examined the multiverse optimizer, moth–flame optimization, particle swarm optimization, cuckoo search, Grey wolf Optimizer (GWO), and bat algorithm. The multiverse optimizer showed the best result for the wall following application among the six techniques. Obstacle present environments are common for robot navigation scenarios, and the work (Yang, Qi, Miao, Sun, & Li, 2018) introduced a collision-free path planner for complex maps using a double-layer ant colony optimization algorithm. Area coverage is an interesting subtopic under robotics path planning and navigation. Area coverage maximization and tradeoff of coverage parameters could be performed with the support of metaheuristic optimization techniques (Hameed, Bochtis, & Sørensen, 2013). The cited work proposed a genetic algorithm-based area coverage algorithm for an agricultural robot to achieve complete area coverage while avoiding obstacles. The use of metaheuristic optimization techniques for area coverage has also been explored for unmanned aerial vehicles (Rosalie, Brust, Danoy, Chaumette, & Bouvry, 2017).

Navigating a reconfigurable robot through a narrow space or covering an obstacle with the aid of continuous shapeshifting is challenging. This paper proposes a novel method to navigate a reconfigurable robot with the aid of continuous shapeshifting ability. In addition to that, the method can be used to cover an obstacles while navigating around the obstacle. In the first stage of the method, an appropriate path to navigate through or cover an obstacle is determined by analyzing the metric map of the environment. In the next stage, the method generates a sequence of reconfigurations for the robot to move on that path without collision. Metaheuristic optimization algorithms are utilized in this regard. An overview of the proposed system is given in Section 2. Section 3 details the proposed navigation and reconfiguration strategy. Particulars on the validation of the proposed method are discussed in Section 4. Section 5 concludes the work.

2. Overview

2.1. Rationale behind the proposed method

Reconfigurable robots are used for different applications such as inspection, exploration, and search and rescue (Cramer et al., 2017; Dutta, Dasgupta, & Nelson, 2017; Kim, Alspach, & Yamane, 2017). The critical motivation for designing reconfigurable floor cleaning robots is to improve productivity. The ability to access narrow spaces using the shape-changing capability enables ways to improve area coverage which is a crucial performance indicator for a cleaning robot (Rhim, Ryu, Park, & Lee, 2007). In addition, the ability of reconfiguration helps to improve productivity in terms of energy and time. With the rapid growth of technology, different types of reconfigurable robots have been emerged to cope with diverse types of problems. However, most of the existing reconfigurable robots are designed to reconfigure into a few predefined shapes (less than ten shapes). Thus, navigating through a curved or cluttered narrow space is not possible for existing reconfigurable robots due to the consideration of limited shapes.

For example, consider the situation as depicted in Fig. 1(a), where a reconfigurable robot has to navigate through a narrow curved space. The robot should change its shape multiple times considering the environmental constraints while navigation to successfully move through it. An example scenario where a reconfigurable robot attempts to move around an obstacle for coverage is shown in Fig. 1(b). Similar to the scenario of navigation through a narrow space, the robot has to reconfigure multiple times while navigation to cover the given obstacle. The state-of-the-art reconfigurable robots consider only a limited number of shapes for the reconfiguration; the existing methods cannot accomplish the mentioned tasks. Samarakoon, Muthugala, and Elara (2021), Samarakoon, Muthugala, Le, and Elara (2020), Samarakoon et al. (2021) introduced coverage techniques that can be applicable for a reconfigurable robot to achieve beyond a set of predefined shapes. However, these methods are not designed for navigation purposes and are solely considered for one-time reconfiguration for area coverage when the robots encounter an obstacle.

2.2. System overview

The functional overview of the proposed system is shown in Fig. 2. A metric map of the surrounding environment is created based on the Lidar mounted in the robot. An occupancy grid map corresponding to the metric map is then generated considering a grid where the cell size is equivalent to the size of a one block of the robot. These steps related to map creations are performed by the Map Creator module. The corresponding maps are stored in the Map Database. The Obstacle Cluster Analyzer (OCA) is responsible for identifying the obstacle clusters and analyzing them. The OCA determines whether there are enough spaces to navigate through an obstacle clusters considering the exact geometric shapes retrieved from the metric map. Furthermore, single obstacles are labeled separately to cover around them. These data are fed to the Action Manger (AM) to take necessary actions. The AM is

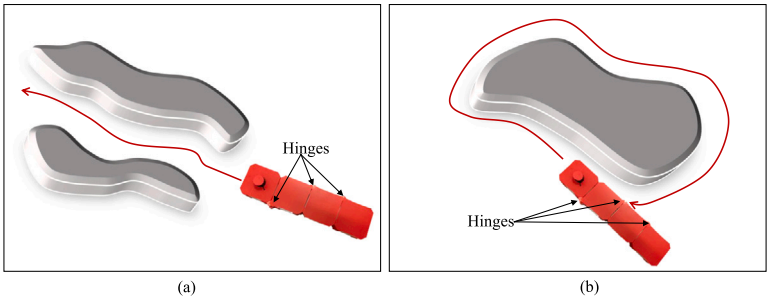

Fig. 1. (a) A scenario where a reconfigurable robot attempts to navigate through a narrow space. (b) The robot navigate around an obstacle for coverage.

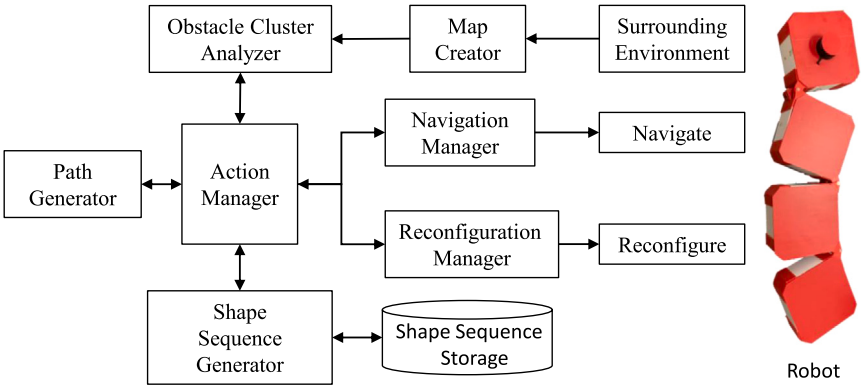

Fig. 2. System Overview.

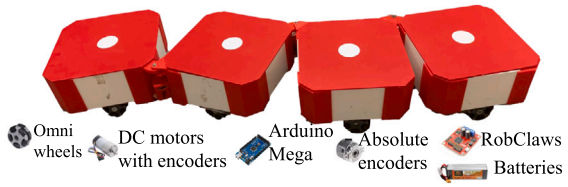

Fig. 3. hTetro-infi robot platform (Samarakoon, Muthugala, Abdulkader, et al., 2021).

responsible for overall coordination between the main modules of the system.

The Path Generator (PG) module is requested to generate a navigation path for the robot for moving through an obstacle cluster or covering a single obstacle identified by OCA. A path generated by the PG based on data of OCA is sent back to the AM. Then, the Shape Sequence Generator (SSG) is requested to determine a reconfiguration sequence for the robot that allows it to pass through an obstacle cluster of interest without collisions or cover a single obstacle of interest. In this regard, the SSG analyzes the path details and the map details considering a set of very closely apart waypoints in the path. For each key point, a set of hinge angles for the reconfiguration that assures the collision free movement of the robot along the path is determined using a metaheuristic optimization method. The determined sequences are stored in the Shape Sequence Storage. After completion of the shape sequence generation, the AM sends the data to the Navigation Manager (NM) and Reconfiguration Manager (RM) to make necessary navigation and reconfiguration of the robot to move on that path without collisions.

2.3. Robot platform

The robot platform (Samarakoon, Muthugala, Abdulkader, et al., 2021) consists of four square-shaped blocks connected serially through three hinges, as shown in Fig. 3. The size of each block is 25 cm x 25 cm x 13 cm (length x width x height). Each block consists of a four-wheel omni drive powered by 6 V DC motors. The omni wheel diameter is 56 mm. The nominal speed of the drive motors is 56 rpm and the motors are equipped with incremental encoders of 8256 counts per revolution. The robot can reconfigure into different shapes by the relative motion of these blocks around the passive hinges. Absolute encoders with 1° resolution are comprised in each hinge to measure the hinge angles. The permittable hinge range is from 0° to 180°. The hinges can be reconfigured with an accuracy of 3°. PID controllers are used for controlling the hinges. The low-level modeling and control of the robot platform are detailed in Samarakoon, Muthugala, Abdulkader, et al. (2021). An Arduino mega controller performs low-level controlling functionalities of the robot, such as kinematic processing for reconfiguration and locomotion. The hinge encoders and microcontroller are connected through a Transistor–Transistor Logic (TTL) serial communication link. Roboclaw 2 × 7.5 A dual channel motor controllers are used to control the drive motors. The RoboClaw 2 × 7 A is a motor controller designed to control two brushed DC motors at 7.5 A continuous with up to 15 A peak per channel. The motor drivers themselves handle the encoder reading of drive motors. Each block of the robot is fixed with two motor drives for controlling the four DC motors. The communication between the motor drivers and the microcontroller is also performed through TTL serial communication protocol. High-level processing tasks such as mapping and path planning for area coverage are performed in a compute stick installed in the robot. The computer stick has an Intel Core m5-6Y57 Processor with 4 GB memory, and has enough computational resources for performing computationally intensive activities such as executing optimization algorithms. A 2D Lidar is fixed to the robot for perceiving the environment. This Lidar can perform range measurements with an angular resolution of 0.225° and range resolution of 0.5 mm. The accuracy of the readings is 1% of the distance measurement. The robot is powered by four 7.4 V 5500 mAh Li-Po batteries placed one in each block. The battery supply is directly connected to the power input of

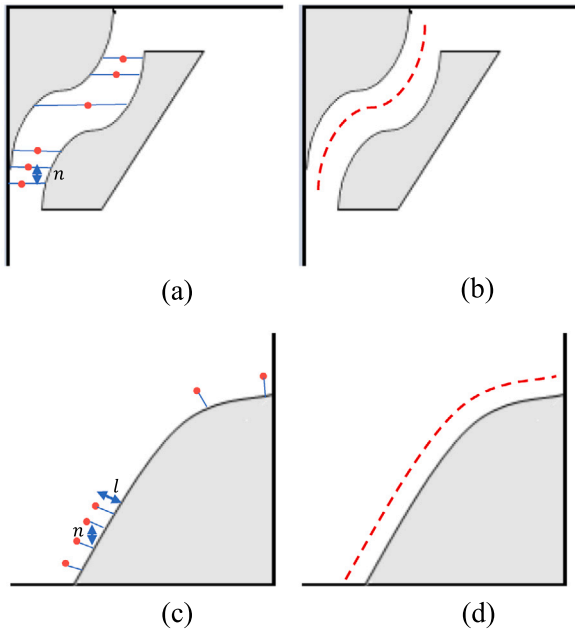

Fig. 4. (a) and (b) represent the robot navigation path generation. (c) and (d) show the steps for path generation for obstacle covering while navigating.

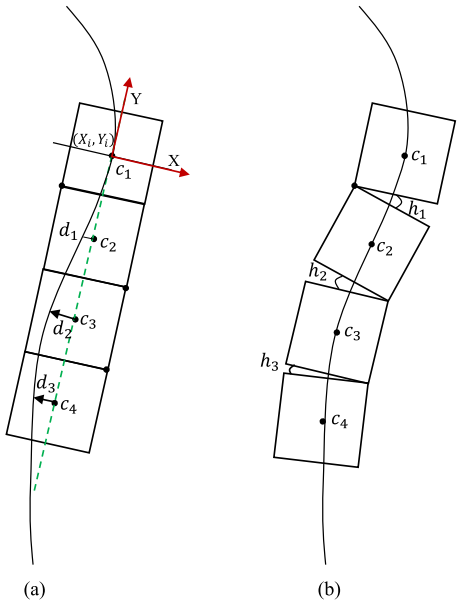

Fig. 5. Determination of the appropriate hinge angles to make the robot on the path. (a): Measurements used for determining the hinge angles used for the reconfiguration. (b): The robot is on the path after the reconfiguration.

the motor drivers responsible for controlling the drive motors. A 5 V DC step-down voltage regulator (rated 5 A) is used for converting the batter voltage to the 5 V DC supply required for the microcontroller, compute stick, and Lidar.

3. Navigation strategy based on reconfigurations

3.1. Determining the path to follow

The robot perceives the environment from Lidar information and generates a metric map. The occupancy grid map of the environment is generated from the metric map. Then, the robot separately analyzes the obstacle clusters and determines the accessibility considering the robot size form factor. If the obstacle clusters are accessible, where the narrow space between two obstacles is less than the robot's minimum width, the robot determines a path to navigate through it. An example accessible narrow space is depicted in Fig. 4(a). The robot finds the midpoints between the boundaries of the narrow space considering a small interval (denoted as n). Then, the successive midpoints are connected to obtain the path to navigate (see Fig. 4(b)). A scenario where the robot navigates for covering an obstacle is shown in Fig. 4(c). A set of points where a point is generated by the perpendicular distance l away from the obstacle outline is considered in this event. The points are generated with an interval n along the surface of the object. Then the points are connected for the navigation path, which is shown in Fig. 4(d).

3.2. Generating reconfiguration sequence

After determining the path to navigate, the robot has to determine a sequence of shape reconfiguration that allows the robot to move on the navigation path without colliding with obstacles. The method proposed in this regard is given Algorithm 1 considering the scenario shown in Fig. 5 where the robot is on a previously determined path segment. Point 'P' is considered the current point on the line where the robot is positioned (c_1 is on P). Tangent to the navigation path at the point P is determined to decide the robot's heading. It is assumed that the Y-axis of the robot should be aligned with the tangent. Thus, the heading of the robot, θ_{robot} is $\angle tangent - \frac{\pi}{2}$. When the robot is on the point P with the Y-axis aligned to the tangent is shown in Fig. 5(a). Maintaining the robot's shape in line with the path is crucial for properly navigating through the narrow space or covering an obstacle. In other words, the centers of the robot (i.e., c_1, c_2, c_3, c_4) should be on the navigation path or near to the path that could replicate the line (similar to 5(b)).

In this regard, the perpendicular distance from each center of the robot to the navigation path (denoted as d_i for $i = 1, 2, 3$) is computed when the robot is aligned on the tangent. If the robots' centers can produce offsets similar to the computed distances $d_i, i = 1, 2, 3$ through reconfiguration, the robot can align its shape on the navigation path. Therefore, the hinge angles (denoted as h_i for $i = 1, 2, 3$) that make the robot on the navigation path as shown in Fig. 5(b) should be determined.

Suppose, the offsets of the centers along the perpendicular to tangent as d_i' for $i = 1, 2, 3$ when the robot reconfigured to h_i for $i = 1, 2, 3$ as shown in Fig. 6. The relationship between the hinge angles and the corresponding positioning of the centers, c_2, c_3, and c_4 with respect to the robots frame, ${}^1c_i$ can be obtained using the kinematic relationships given in (1) and (2), where t_{X_i} and t_{Y_i} are the translations of the $(i+1)$th

Algorithm 1: Generating the reconfiguration sequence

input : *navigation path, metric map*
output: $X_{robot}, Y_{robot}, \theta_{robot}, h_1, h_2, h_3$
P = initial position;
while *(! path ends)* **do**
  $X_{robot} = P_x$;
  $Y_{robot} = P_Y$;
  find tangent at P;
  $\theta_{robot} = \angle tangent$ - $\pi/2$;
  Calculate d_1, d_2, d_3;
  $[h1, h2, h3]$ = optimization algorithm(d_1, d_2, d_3, obstacle cluster);
  store$[X_{robot}, Y_{robot}, \theta_{robot}, h_1, h_2, h_3]$;
  $P = next\ point$;
end

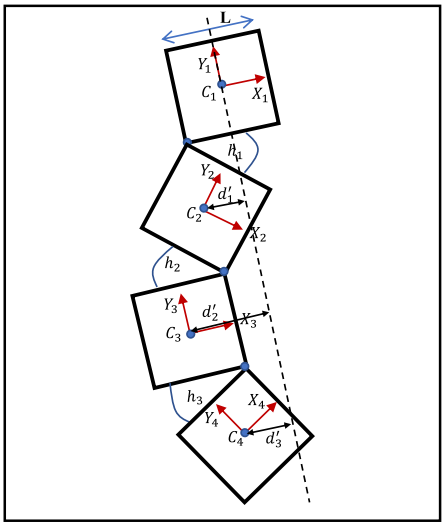

Fig. 6. Kinematic frame assignment of the robot.

frame's origin with respect to ith along X_i and Y_i axes, respectively. Here, ${}^{i}c_i$ denotes the position of the center c_i with respect to the ith frame. Then, d_i' can be obtained as in (3) considering the translation of c_i along X_1 axis.

$$ {}^{1}c_i = \prod_{j=1}^{i-1} {}_{j+1}^{j}T \; {}^{i}c_i \quad \text{for } i = 2, 3, 4 \tag{1} $$

$$ {}_{i+1}^{i}T = \begin{bmatrix} \cos h_i & -\sin h_i & t_{X_i} \\ \sin h_i & \cos h_i & t_{Y_i} \\ 0 & 0 & 1 \end{bmatrix} \tag{2} $$

$$ d_i' = ({}^{1}c_i)_{X_1} \quad \text{for } i = 2, 3, 4 \tag{3} $$

If the robot's shape is perfectly aligned on the navigation path, it should satisfy the condition given in (4). However, finding the hinge angles that satisfy this condition is challenging since only an approximated solution exists in most cases. Furthermore, the solution should assure that there is no collision of the robot with the obstacles in this situation. Metaheuristic optimization techniques have proven to be effective in solving this sort of problem. Therefore, metaheuristic optimization techniques are used to find the best hinge angles that satisfy the condition (4) while ensuring no collision. Grey Wolf Optimizer (GWO), Genetic Algorithm (GA), and Simulated Annealing (SA) have been proposed in this regard where the optimization goal is to minimize the cost function given in (5). Here, Q is a positive integer used to penalize the situations of collisions. The proposed optimization techniques are detailed in Section 3.3

$$ d_i = d_i' \quad \text{for } i = 1, 2, 3 \tag{4} $$

$$ f(h_1, h_2, h_3) = \begin{cases} \sum_{i=1}^{3} (d_i' - d_i)^2, & \text{If no collision} \\ Q, & \text{Otherwise} \end{cases} \tag{5} $$

$$ \text{where, } 0 < h_i < 180^\circ \quad \text{for } i = 1, 2, 3 $$

Therefore, after calculating d_1, d_2, d_3, and d_4, the algorithm calls the optimization method that return the hinge angles. After determining the hinge angles, and the robot's positions and heading that align the robot in the navigation path are stored in the shape sequence storage memory. After that, the next position on the navigation path is selected as the current position, P, and the steps are iterated until the path ends.

3.3. Metaheuristic optimization techniques

Metaheuristic optimization techniques are increasingly used in many engineering and science applications (Cintrano, Chicano, & Alba, 2020; Gupta et al., 2021). The reasons behind the popularity of the techniques are local optima avoidance, derivation-free mechanism, flexibility, and simplicity (Mirjalili, Mirjalili, & Lewis, 2014). According to the No Free Lunch (NFL) theorem (Wolpert & Macready, 1997), it has been proven that there is no specific metaheuristic optimization technique best suited for all the optimization problems. The performance of the particular metaheuristic optimizer could change from one problem to another problem. Therefore, three metaheuristic optimization techniques with different characteristics were investigated for the proposed method. Grey Wolf Optimizer (GWO) is mainly chosen for the problem solver and Genetic Algorithm (GA) and Simulated Annealing (SA) are used to compare the performance. GWO is chosen since it is a recently introduced method and has shown efficient performance compared to other techniques in a vast range of engineering applications. GA has been selected for the comparison since it has been a well-known and established metaheuristics for many years and is widely used for solving problems in many application domains. Both GWO and GA are population-based metaheuristics, and it would be better for the comparison to include a non-population-based metaheuristic. Therefore, SA, a well-established non-population-based metaheuristic for searching global optimum, has also been selected for comparison.

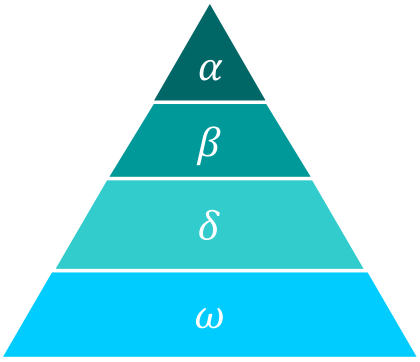

Fig. 7. The social hierarchy of a grey wolf group.

3.3.1. Grey Wolf Optimizer (GWO)

Grey Wolf Optimizer (GWO) was introduced in 2014 by Mirjalili et al. (2014). GWO has outperformed compared to several techniques such as particle swarm optimization, gravitational search algorithm, and differential evolution in many engineering problems. GWO is a swarm intelligence technique that mimicks the leadership hierarchy of grey wolves for hunting. These predators are at the top of the food chain and they choose to live in a group of 5–12 on average. The hierarchy of a grey wolf group is depicted in Fig. 7. The alpha (α) can be considered as the leader which makes decisions related to hunting and other typical activities. Alpha is not the strongest wolf in the pack but the perfect one to manage the group. The other wolves follow the commands given by alpha. Beta (β) represents the second level of the hierarchy that supports alpha in decision making and it acts as a communicator in between other wolves and alpha. Delta (δ) wolves are below alpha and beta but control the omega (ω) wolves. Caretakers, elders, scouts, and hunters fit into this category. The last category belongs to omega wolves. All other categories are dominant to omega and play the role of scapegoat. Although omega does not seem to be important to the group but the losing of omega makes internal fights and problems.

Grey wolf hunting is mainly done as, tracking, chasing, and approaching a prey. Then wolves persuade, encircle, and harass the prey until the prey stops moving. Finally, the wolves attack the prey. In GWO, the hunting techniques and social hierarchy of grey wolves have been mathematically modeled. The current best fitting solution is considered as alpha (α), second and third best solutions are considered

Algorithm 2: GWO for Determining the Shape to Reconfigure

input : $d_1, d_2, d_3, Obstacle\_cluster$
output: h_1, h_2, h_3
initialization;
Generate initial grey wolf population;
Initialize a, A, and C;
X_α = the best search agent;
X_β = the second search agent;
X_δ = the third search agent;
while *(! Stop condition)* **do**
  for *Each search agent* **do**
    Update the current position of the search agent using (12);
  end
  Update a, A and C;
  Calculate the fitness of all search agents using (5);
  Update X_α, X_β, X_δ;
  $t = t + 1$;
end
Return best individuals (X_α);

as beta (β) and delta (δ) respectively. Omega (ω) is assigned for the rest of the candidate solutions. Grey wolves encircling behavior is mathematically modeled by (6) and (7), where $\vec{A}$ and $\vec{C}$ symbolized for coefficient vectors. Position vectors of the grey wolf indicate from $\vec{X}_p$ and $\vec{X}$. The current iteration is indicated by t.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{6}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{7}$$

The vectors $\vec{C}$ and $\vec{A}$ are calculated from (8) and (9). When the number of iterations increases, the components of $\vec{a}$ linearly decrease from 2 to 0. Further, r_1 and r_2 are random vectors belong to [0,1]

$$\vec{C} = 2 \cdot \vec{r_1} \tag{8}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r_2} - \vec{a} \tag{9}$$

Grey wolves are capable of identifying the location of the prey, and they are able to surround the prey. Alpha is considered the leader, and it guides the hunting. The hunting behavior is mathematically simulated by taking alpha as the best candidate solution, and alpha, beta, and delta have the perception of the prey's location. The three best solutions are given for alpha, beta, and omega respectively, and the remaining best solutions are assigned for omega. Eqs. (10), (11), and (12) are responsible for locating the prey for hunting. The main steps of the GWO are given in Algorithm 2. This process is iterated until a stopping condition is met. This optimization task is a low-dimensional problem since it involves determining three variables (i.e., the hinge angles) that minimize the cost function. For low-dimensional problems, much of the existing work utilized small population sizes. Therefore, the population size was determined by trial and error, considering small population sizes between 20–60 in steps of 10. A population size of 30 was adequate for GWO to achieve adequate performance in minimizing the cost function. Further increase in population size did not show a considerable performance improvement. Therefore, the number of grey wolves is chosen as 30 for this problem. A maximum number of 6000 function evaluations were assigned for the algorithm. This value was also chosen based on trial and error. A function tolerance of 1e-6 or 6000 function evaluations are defined as the stopping conditions.

$$\vec{D_\alpha} = |\vec{C_1} \cdot \vec{X_\alpha} - \vec{X}|, \vec{D_\beta} = |\vec{C_2} \cdot \vec{X_\beta} - \vec{X}|, \quad \vec{D_\delta} = |\vec{C_3} \cdot \vec{X_\delta} - \vec{X}| \tag{10}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1} \cdot (\vec{D_\alpha}), \vec{X_2} = \vec{X_\beta} - \vec{A_2} \cdot (\vec{D_\beta}), \quad \vec{X_3} = \vec{X_\delta} - \vec{A_3} \cdot (\vec{D_\delta}) \tag{11}$$

Algorithm 3: GA for Determining the Shape to Reconfigure

input : $d_1, d_2, d_3, Obstacle\_cluster$
output: h_1, h_2, h_3
initialization;
Generate initial population;
while *(! Stop condition)* **do**
  Evaluate the fitness based on (5);
  Select best fits;
  crossover;
  Mutation;
end
Return best individuals;

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{12}$$

3.3.2. Genetic Algorithm (GA)

A Genetic Algorithm (GA) (Kramer, 2017) is a metaheuristic optimization technique inspired by biological evolution. This method is a population-based approach where a set of solutions is evaluated in each algorithm iteration. The main steps of the GA are given in Algorithm 3. At the start, a set of uniformly distributed random solutions within the search space is initialized. This set is known as the initial population. Here, the size of the initial population is set to 50 by trail and error considering a population variation in between 20–60 similar to the case of GWO since the problem is low dimensional (three variables). The corresponding cost of each solution in the current population is then evaluated using the cost function given. The resulting raw cost values are scaled based on their rank to remove the spread of the raw scores before selecting the best fits. Next, the best-fit solutions among the population members are selected using the stochastic uniform selection approach. In this selection approach, the population members are represented on a line where the line length is proportional to the scaled costs. Next, the selection of parents is performed considering equivalent step sampling. Then, the crossover operation in which the genes of two parents are crossover to generate offspring is applied to the selected parents to generate a fraction of the new population. The crossover fraction is configured to 0.8. Another set of offspring is created by making random changes to a single parent known as mutation. In addition to that, a few members of the current population are directly passed to the next population. These members are known as elite members. The elite count is considered as 0.05 of the total population. This process is repeated until a stopping criterion (reaching a function tolerance of 1e-6 or 6000 function evaluations are used) is met. After reaching a stopping criterion, the best individual is selected as the solution for the optimization.

3.3.3. Simulated Annealing (SA)

Simulated Annealing (SA) (Ingber, 1996) is a metaheuristic optimization technique inspired by the physics of the annealing process of metals where heating and controlled cooling are done to improve the properties of the material. SA is a single-point-based algorithm where only one potential solution is generated at an instance. Thus, the evaluation of the cost function is computationally efficient in single-point-based methods compared to population-based methods since the cost function has to be evaluated only for a single solution. However, SA requires a good initial solution for the proper convergence to the global optimum.

The flow of the SA is given in Algorithm 4. An initial solution has to be defined by the user before the starting of the iterative process. At the beginning of the iterative process, the method generates a random trial solution around the current solution considering a probability distribution with a scale depending on the current temperature (Initially

Algorithm 4: SA for Determining the Shape to Reconfigure

input : $d_1, d_2, d_3, Obstacle\_cluster$
output: h_1, h_2, h_3
initialization;
$k = 0$;
Initial solution;
Initial temperature;
Current Solution = Initial solution;
Current Temperature = Initial temperature;
Calculate cost current solution based on (5);
while *(! Stop condition)* **do**
  Generate New Trial Solution;
  Calculate cost new solution based on (5);
  Δ = cost new solution - cost current solution;
  if $\Delta < 0$ **then**
    Current Solution = New Solution;
    cost current solution = cost new solution;
  else
    $\rho = \frac{1}{1+\exp(\frac{\Delta}{\text{Current Temperature}})}$;
    if $\rho >$ *random[0,1]* **then**
      Current Solution = New Solution;
    end
  end
  Current Temperature = Initial temperature*0.95^k;
  if *K reaches Reannealing number* **then**
    k = Reannealing Function();
  end
  $k++$;
end
Return Current Solution;

temperature is defined as 100). Then the cost of the new solution is evaluated considering the cost function for the optimization given in (5). If the cost of this trial solution is less than the cost of the current solution, the new solution is considered the current solution. Otherwise, the new solution is accepted at random, considering the acceptance probability, ρ. The current temperature is systematically lowered with iterations to reduce the distance between the current and the next trial solutions. After the annealing parameter, k reaches the reannealing number (100 here), reannealing is triggered. The reannealing process raises the temperature by setting the annealing parameter k to a lower value than the iteration number. This iterative process is repeated until a stopping criterion is met. Reaching a function tolerance of 1e-6 or reaching the number of evaluations of the cost function to 6000 are considered the stopping conditions. The current solution at the reach of the stopping condition is considered as the final solution of the algorithm.

The situation of all the hinges in [0°,0°,0°] position is considered as the initial solution of SA for finding the hinge angles for the first point in the path (i.e., P = initial position). The previously found solution is used as the initial solution when finding the hinge angles for the successive points (i.e., initial solution for SA at P = best solution for SA at $P-1$). This consideration would help to result in a minor reconfiguration between successive steps.

4. Results and discussion

4.1. Experimental setup

Simulations have been conducted using MATLAB for evaluating the performance and the behavior of the proposed method. In this regard, eight test cases of navigating through narrow spaces or covering obstacles have been considered. The scope of the evaluations was limited to the scenarios of navigating through or continuous covering of obstacles clusters, where the coverage of non-obstacle clusters (free areas) was not considered. This decision was made to highlight only the contribution of this work since many existing algorithms are available for covering the non-obstacle clusters. The obtained results for the test cases are given in Fig. 8, where the robot is traced to visualize the robot's movement. The images were created, downsampling the robot's movement by 10 to improve the clarity of the visualization. The corresponding simulation videos are given as supplementary material to complement the visualization. The performance of the proposed three optimization techniques in terms of final cumulative costs of best solutions and the total computational time is given in Table 1 for comparison. It should be noted that the simulations were carried out on a laptop with 16 GB memory and the Intel Core i7-9750H processor. Parallel processing was not used in this regard.

4.2. Results

Case 'a' consists of a situation where the robot has to navigate through a narrow space with the aid of continuous reconfiguration ability. Without reconfiguration, it cannot go through it. This scenario was from the obstacle cluster 'a' of the occupancy grid map shown in Fig. 9. The proposed method could successfully navigate the robot through the narrow space without any collisions, as shown in Fig. 8(a) in the events of GWO, GA, and SA. The algorithm has considered 44 points in determining the sequence of reconfigurations. The total cost of best solutions in the events of GWO, GA, and SA is 409.6, 425.5, and 411.0, respectively. The variations of the total cost between the optimization techniques were minor. However, a notable difference between the computational time of the techniques could be observed where the computational time of GWO was the minimum with 69.6 s. The highest computational time was observed from GA (209.7 s), while the computational time of SA was 128.7 s. These observations suggested that the GWO was computationally efficient compared to GA and SA while maintaining the same level of performance in this case.

The scenario of the robot moving around obstacle 'b' in the map shown in Fig. 9 for coverage was considered for test case 'b'. In all the events of the three considered optimization techniques, the robot was capable of navigating around the obstacle while reconfiguring for the coverage. Similar to the previous case, only a slight variation of total cost could be observed across the three optimization techniques. However, the computational time had a considerable variation where GWO took the least computational time (130.5 s) while GA took the highest computational time (437.4 s), suggesting GWO is computationally effective compared to GA and SA in this case also.

In all the test cases, the proposed method was successful in either navigating through narrow spaces or covering an obstacle with the aid of continuous reconfigurations. The three optimization techniques considered for the work were found to be effective in this regard. The mean total cost of the optimization techniques was 2316.2 (SD = 2574.1), 2312.2 (SD = 2579.7), and 2292.6 (SD = 2568.9) for GWO, GA, and SA, respectively. A significant difference between the total cost could not be detected from the one-way ANOVA test conducted on costs across the three techniques ($F_{2,21} = 0$, $P = 0.9998$). Thus, the performance of the three techniques does not have a significant difference in finding the best solutions. On the other hand, a significant difference between the computational time between the optimization technique could be detected from the one-way ANOVA test conducted for the computational time ($F_{2,21} = 26.86$, $P = 0.0001$). The lowest mean computational time was recorded from GWO (M = 94.2 s, SD = 27.6), while the highest mean was observed from GA (M =290.8 s, SD = 83.2). SA took a mean computational time of 162.2 s (SD = 34.9). According to overall results, it can be concluded that the considered three optimization methods have similar performance in navigating and covering obstacles by appropriately reconfiguring; however, GWO is computationally efficient compared to GA and SA in the process.

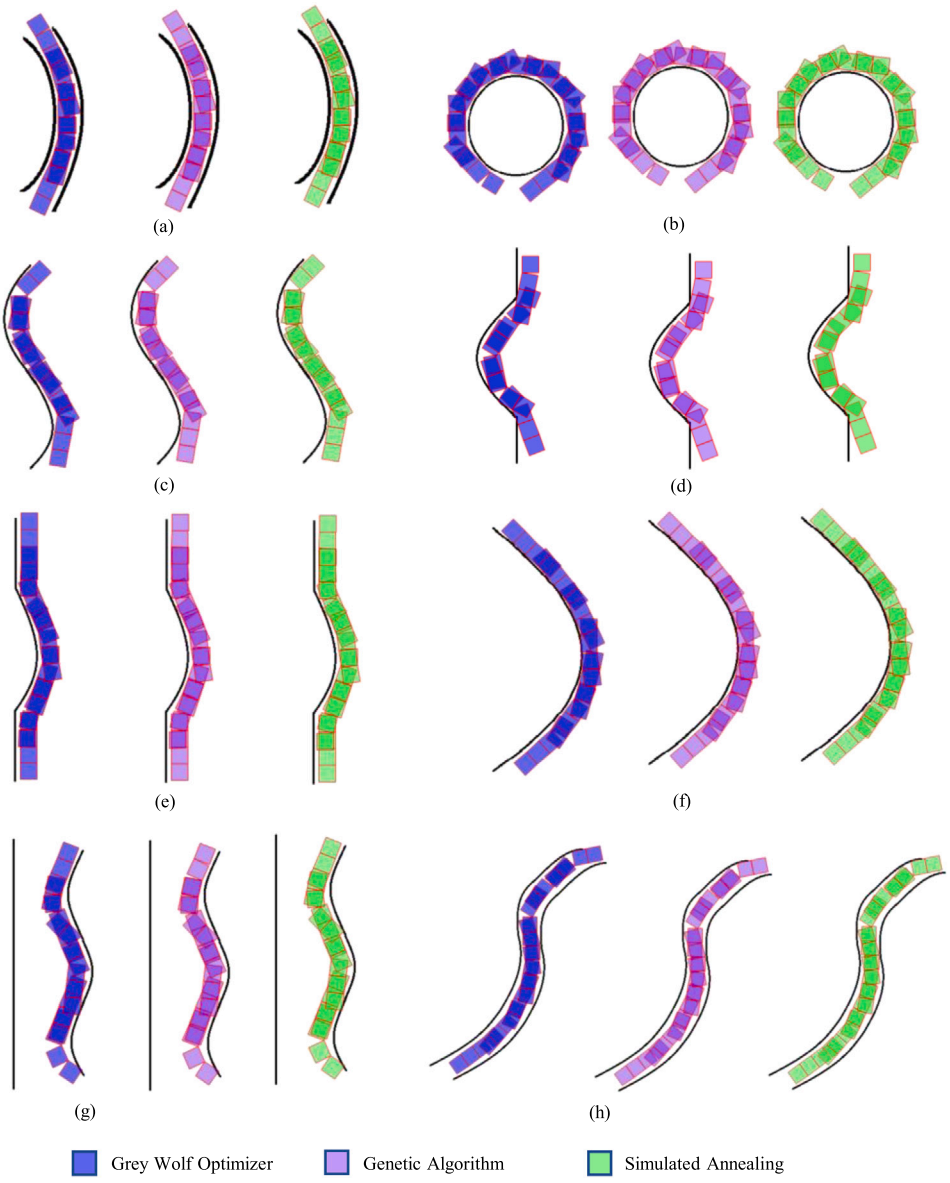

Fig. 8. Simulation results for the three optimization techniques. It should be notated that the robot was traced at a very lower rate (down sampled by 10) compared to the actual number of points considered for the computation to maintain the readability of the images.

4.3. Discussion

According to the experimental outcomes, the proposed method can navigate a reconfigurable robot through a narrow space or cover around an obstacle with the aid of continuous reconfiguration. Narrow spaces where a robot cannot navigate without this sort of continuous reconfiguration could often be observed in environments cluttered with obstacles such as furniture and pillars, which is essential for a robot intended for area coverage. Furthermore, the ability to continuously move around an obstacle for covering is desired for a robot intended for area coverage applications. Nevertheless, state-of-the-art methods cannot navigate a reconfigurable robot through narrow spaces or cover obstacles with the aid of continuous reconfigurations. In Particular, most of the state-of-the-art methods of intra-reconfigurable robots are limited to consideration of a small number of predefined shapes for the reconfiguration (e.g., Prabakaran et al., 2017: 7 shapes, Apuroop et al., 2021: 2 shapes, and Le et al., 2020: 3 shapes). On the other hand, the existing methods that consider beyond a small number of predefined shapes (e.g., Samarakoon, Muthugala, & Elara, 2021; Samarakoon, Muthugala, Le, & Elara, 2020; Samarakoon et al., 2021) for the reconfiguration are limited to one-step reconfiguration for accomplishing a task and do not consider the continuous reconfiguration along with navigation. Therefore, the method proposed in this paper contributes to improving the state of the art of reconfigurable robots. In particular, the findings of the work would be useful for the development of intra-reconfigurable robots intended for area coverage applications such as floor cleaning.

The method proposed in this work has been presented and validated considering hTetro, which is an intra-reconfigurable robot with four square shape blocks. Intra-reconfigurable robots designed for area coverage applications with different configurations such as different block shapes (polyforms) and the number of blocks (e.g., hTrihex (Apuroop et al., 2021): 3 hexagon-shaped blocks and hTetrakis (Le et al., 2020): 3 triangular-shaped blocks) could be seen in the literature. hTetro has been considered for this work since it is one of the versatile designs mostly used in the literature compared to other intra-reconfigurable robots with different polyforms and block numbers. Even though the method has been formulated based on hTetro, the method could easily be adapted to any other intra-reconfigurable robot. The modification

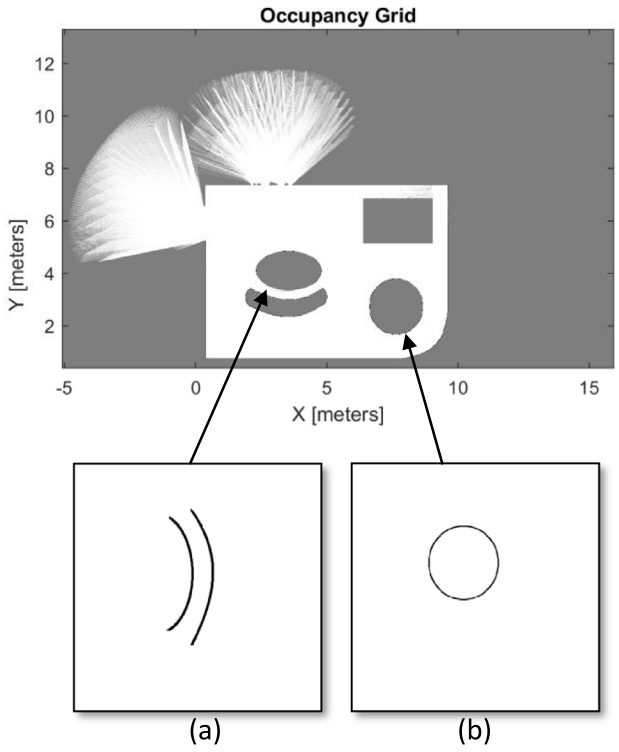

Fig. 9. Full occupancy grid map corresponding to test case 'a' and 'b'.

Table 1
Results of the test cases: Variation of total cost and computational time.

Case	Optimization technique	Number of points	Total cost	Time (s)
a	GWO	44	409.6	69.6
	GA	44	425.5	209.7
	SA	44	411.0	128.7
b	GWO	87	8295.1	130.5
	GA	87	8350.2	437.4
	SA	87	8297.5	204.6
c	GWO	44	1420.9	65.9
	GA	44	1436.2	205.9
	SA	44	1413.8	119.9
d	GWO	47	3438.6	74.4
	GA	47	3290.8	228.6
	SA	47	3290.2	129.2
e	GWO	58	1697.8	90.9
	GA	58	1700.3	288.9
	SA	58	1695.8	171.7
f	GWO	61	948.0	96.3
	GA	61	945.9	301.8
	SA	61	934.5	182.8
g	GWO	51	1094.1	85.6
	GA	51	1098.9	269.6
	SA	51	1068.9	152.6
h	GWO	71	1225.5	140.4
	GA	71	1250.1	384.4
	SA	71	1229.2	208.3

of the robot kinematic model is sufficient in this regard. Hence, the adaptability to other reconfigurable robots is straightforward.

The maximum turn of a curved that the robot can successfully manage is an important performance factor. The robot was moved around curved obstacles with different turn angles to identify this angle. The curves considered in this regard had turn angles 40°, 80°, 100°, 110°, and 120°. The robot was not effective in the case of the 120° turned curve, where the 110° turned curve was found to be the maximum curve that the robot could successfully manage. Therefore, the maximum turn of a curve should be less than or equal to 110° for ascertaining the performance.

It is considered that the robot has a complete map of the environment where the robot is deployed. The OCA examines the map and identifies the clusters that the robot can access before the execution of the robot navigation. Therefore, the robot only attempts to move through narrow spaces accessible for the robot. In this regard, it is assumed that the robot would not encounter dynamic changes in the environment. This assumption limits the workability of the robot in dynamic environments where the positions of obstacles may be changed. The expansion of the proposed method to work on dynamic environments is proposed for future work.

Simulations have been performed to validate the performance and behavior of the proposed method. These simulations have been conducted considering the actual parameters of the robot and environments. The position accuracy of the robot was assumed as 1 cm within the frame of an obstacle cluster during the continuous reconfiguration and navigation. Even though the considered Lidar has a higher resolution of 0.5 mm, a lesser resolution (1 cm) for the simulation has been considered to accommodate possible inaccuracies caused by the noises during the operation. Therefore, the outcomes of the simulations would not be heavily deviated from the real-world robot deployments and ascertain the real-world applicability. Furthermore, this paper is the first step of a novel research direction that considers continuous reconfiguration of a robot while navigation. Therefore, the implication of this work is vital for the development of the state of the art of reconfigurable robotics even though real-world experiments have not been conducted. At the current stage, even though the hardware platform (presented in Section 2.3) is available, the low-level controlling functionalities of the robot are limited to decoupled reconfiguration and navigation. In other words, the low-level controlling functionalities of the robot proposed in Samarakoon, Muthugala, Abdulkader, et al. (2021) require modifications for facilitating concurrent navigation and reconfiguration of the robot. The modification required for the low-level controlling functions for achieving continuous reconfiguration while navigation is currently performed. Implementation of the proposed method on the hardware platform presented in Section 2.3 and experimenting with the proposed method considering real-world deployments are the immediate next steps of the work.

The parameters of the optimization techniques could be fine-tuned to improve the performance of the system. For example, only a linear change of the parameter $\vec{a}$ of GWO is considered, and a faster convergence would be observed if a nonlinear change of the parameters was considered. This sort of fine-tuning of the optimization techniques is proposed for future work. According to the considered navigation strategy, the cost function used in this work is the sole objective function that satisfies the required conditions for navigating through narrow spaces or covering obstacles with continuous reconfiguration while ensuring collision avoidance. However, for future work, it would be interesting to modify the navigation strategy and formulate an objective function that improves the performance of the robot. Furthermore, other metaheuristic optimization techniques such as differential evolution could be used to achieve the robot's required behavior. Therefore, exploring other metaheuristic optimization techniques for solving this particular problem is another potential future work.

5. Conclusions

Robots are used to aid the cleaning of buildings, such as floor cleaning. Area coverage is one of the foremost demanding performance features of floor cleaning robots. Reconfigurable robots have been introduced to mitigate the area coverage limitations of the fixed-shape robots. Robots operated on environments cluttered with obstacle with various shapes such as furniture often needs to move through narrow spaces. The ability of continuous reconfiguration could help in this

regard. Therefore, a novel method that enables a reconfigurable robot to move through narrow spaces or cover around obstacles with the aid of continuous reconfiguration has been proposed in this paper.

The proposed method can decide a navigate path by analyzing the obstacle clusters in a metric map. The method then determines a sequence of shape reconfiguration that allows the robot to navigate on the path without collision. Metaheuristic optimization methods, Grey Wolf Optimizer (GWO), Genetic Algorithm (GA), and Simulated Annealing (SA), have been proposed for determining the sequence of shape reconfiguration.

Simulations have been conducted considering scenarios where a cleaning robot could often encounter in typical indoor environments. The results verify that the proposed method can effectively navigate a reconfigurable robot through narrow spaces or around obstacles for coverage with the aid of the reconfiguration. Furthermore, all the three investigated optimization methods were found to be effective in the determination of the reconfiguration sequence. In contrast, a significantly lower computational time was demanded by GWO compared to the other methods. Therefore, GWO is computationally efficient in this application.

The ability to navigate through narrow spaces and continuous covering around obstacles is crucial for the productivity of a reconfigurable robot intended for area coverage applications. Therefore, the findings of this work would be of great interest for improving the productivity of robots demanding coverage performance.

CRediT authorship contribution statement

S.M. Bhagya P. Samarakoon: Conceptualization, Formal analysis, Methodology, Data curation, Software, Visualization, Validation, Writing – original draft. **M.A. Viraj J. Muthugala:** Conceptualization, Formal analysis, Validation, Writing – review & editing. **Mohan Rajesh Elara:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the National Robotics Programme under its Robotics Enabling Capabilities and Technologies, Singapore (Funding Agency Project No. 192 25 00051), National Robotics Programme under its Robot Domain Specific, Singapore (Funding Agency Project No. 192 22 00058), National Robotics Programme under its Robotics Domain Specific, Singapore (Funding Agency Project No. 192 22 00108), and administered by the Agency for Science, Technology and Research, Singapore.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2022.117060.

References

Apuroop, K. G. S., Le, A. V., Elara, M. R., & Sheu, B. J. (2021). Reinforcement learning-Based Complete Area coverage path planning for a modified htrihex robot. *Sensors*, *21*(4), 1067.

Cheng, K. P., Mohan, R. E., Nhan, N. H. K., & Le, A. V. (2020). Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot. *IEEE Access*, *8*, 121267–121284.

Cintrano, C., Chicano, F., & Alba, E. (2020). Using metaheuristics for the location of bicycle stations. *Expert Systems with Applications*, *161*, Article 113684.

Cramer, N., Tebyani, M., Stone, K., Cellucci, D., Cheung, K. C., Swei, S., et al. (2017). Design and testing of FERVOR: Flexible and reconfigurable voxel-based robot. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2730–2735). IEEE.

Dutta, A., Dasgupta, P., & Nelson, C. (2017). Adaptive locomotion learning in modular self-reconfigurable robots: A game theoretic approach. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3556–3561). IEEE.

Elara, M. R., Rojas, N., & Chua, A. (2014). Design principles for robot inclusive spaces: A case study with roomba. In *2014 IEEE international conference on robotics and automation* (pp. 5593–5599). IEEE.

Elliott, S., & Cakmak, M. (2018). Robotic cleaning through dirt rearrangement planning with learned transition models. In *2018 IEEE international conference on robotics and automation* (pp. 1623–1630). IEEE.

Elmi, Z., & Efe, M. O. (2018). Multi-objective grasshopper optimization algorithm for robot path planning in static environments. In *2018 IEEE international conference on industrial technology* (pp. 244–249). IEEE.

Fausto, F., Reyna-Orta, A., Cuevas, E., Andrade, A. G., & Perez-Cisneros, M. (2020). From ants to whales: metaheuristics for all tastes. *Artificial Intelligence Review*, *53*(1), 753–810.

Fong, S., Deb, S., & Chaudhary, A. (2015). A review of metaheuristics in robotics. *Computers and Electrical Engineering*, *43*, 278–291.

Gupta, S., Abderazek, H., Yıldız, B. S., Yildiz, A. R., Mirjalili, S., & Sait, S. M. (2021). Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. *Expert Systems with Applications*, Article 115351.

Hameed, I. A., Bochtis, D., & Sørensen, C. A. (2013). An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas. *International Journal of Advanced Robotic Systems*, *10*(5), 231.

Ingber, L. (1996). Adaptive simulated annealing (ASA): Lessons learned. In *Control and cybernetics*.

Ito, F., Kawaguchi, T., Kamata, M., Yamada, Y., & Nakamura, T. (2019). Proposal of a peristaltic motion type duct cleaning robot for traveling in a flexible pipe. In *2019 IEEE/RSJ international conference on intelligent robots and systems* (pp. 6614–6621). IEEE.

Jalali, S. M. J., Khosravi, A., Kebria, P. M., Hedjam, R., & Nahavandi, S. (2019). Autonomous robot navigation system using the evolutionary multi-verse optimizer algorithm. In *2019 IEEE international conference on systems, man and cybernetics* (pp. 1221–1226). IEEE.

Khan, Z. H., Siddique, A., & Lee, C. W. (2020). Robotics utilization for healthcare digitization in global COVID-19 management. *International Journal of Environmental Research and Public Health*, *17*(11), 3819.

Kim, J., Alspach, A., & Yamane, K. (2017). Snapbot: a reconfigurable legged robot. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5861–5867). IEEE.

Kim, J., Mishra, A. K., Limosani, R., Scafuro, M., Cauli, N., Santos-Victor, J., et al. (2019). Control strategies for cleaning robots in domestic applications: A comprehensive review. *International Journal of Advanced Robotic Systems*, *16*(4), Article 1729881419857432.

Kramer, O. (2017). Genetic algorithms. In *Genetic algorithm essentials* (pp. 11–19). Springer.

Lakshmanan, A. K., Mohan, R. E., Ramalingam, B., Le, A. V., Veerajagadeshwar, P., Tiwari, K., et al. (2020). Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot. *Automation in Construction*, *112*, Article 103078.

Le, A. V., Nhan, N. H. K., & Mohan, R. E. (2020). Evolutionary algorithm-based complete coverage path planning for tetriamond tiling robots. *Sensors*, *20*(2), 445.

Le, A. V., Prabakaran, V., Sivanantham, V., & Mohan, R. E. (2018). Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor. *Sensors*, *18*(8), 2585.

Liu, H., Ma, J., & Huang, W. (2018). Sensor-based complete coverage path planning in dynamic environment for cleaning robot. *CAAI Transactions on Intelligence Technology*, *3*(1), 65–72.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61.

Muthugala, M. A. V. J., Samarakoon, S. M. B. P., & Elara, M. R. (2020). Tradeoff between area coverage and energy usage of a self-reconfigurable floor cleaning robot based on user preference. *IEEE Access*, *8*, 76267–76275.

Muthugala, M. A. V. J., Vega-Heredia, M., Mohan, R. E., & Vishaal, S. R. (2020). Design and control of a wall cleaning robot with Adhesion-awareness. *Symmetry*, *12*(1), 122.

Muthugala, M. A. V. J., Vega-Heredia, M., Vengadesh, A., Sriharsha, G., & Elara, M. R. (2019). Design of an adhesion-aware façade cleaning robot. In *2019 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1441–1447). IEEE.

Park, H.-Y., & Lee, J.-M. (2019). Development of a floor-mopping robot. In *2019 19th international conference on control, automation and systems* (pp. 678–680). IEEE.

Prabakaran, V., Elara, M. R., Pathmakumar, T., & Nansai, S. (2017). Htetro: A tetris inspired shape shifting floor cleaning robot. In *2017 IEEE international conference on robotics and automation* (pp. 6105–6112). IEEE.

Rhim, S., Ryu, J.-C., Park, K.-H., & Lee, S.-G. (2007). Performance evaluation criteria for autonomous cleaning robots. In *2007 international symposium on computational intelligence in robotics and automation* (pp. 167–172). IEEE.

Romero-Martí, D. P., Núnez-Varela, J. I., Soubervielle-Montalvo, C., & Orozco-de-la Paz, A. (2016). Navigation and path planning using reinforcement learning for a roomba robot. In *2016 XVIII congreso Mexicano de robotica* (pp. 1–5). IEEE.

Rosalie, M., Brust, M. R., Danoy, G., Chaumette, S., & Bouvry, P. (2017). Coverage optimization with connectivity preservation for uav swarms applying chaotic dynamics. In *2017 IEEE international conference on autonomic computing* (pp. 113–118). IEEE.

Rosenstein, M. T., Chiappetta, M., Schnittman, M., & Pastore, A. (2016) Autonomous coverage robot. Google Patents. US Patent 9, 408, 515.

Samarakoon, S. M. B. P., Muthugala, M. A. V. J., Abdulkader, R. E., Si, S. W., Tun, T. T., & Elara, M. R. (2021). Modelling and control of a reconfigurable robot for achieving reconfiguration and locomotion with different shapes. *Sensors*, *21*(16), 5362.

Samarakoon, S. M. B. P., Muthugala, M. A. V. J., & Elara, M. R. (2021). Toward obstacle-specific morphology for a reconfigurable tiling robot. *Journal of Ambient Intelligence and Humanized Computing*, 1–13.

Samarakoon, S. M. B. P., Muthugala, M. A. V. J., Elara, M. R., et al. (2021). Toward pleomorphic reconfigurable robots for optimum coverage. *Complexity*, *2021*.

Samarakoon, S. M. B. P., Muthugala, M. A. V. J., Le, A. V., & Elara, M. R. (2020). Htetro-infi: A reconfigurable floor cleaning robot with infinite morphologies. *IEEE Access*, *8*, 69816–69828.

Samarakoon, S. M. B. P., Muthugala, M. A. V. J., Le, A. V., & Elara, M. R. (2021). Toward complete area coverage of a reconfigurable tiling robot by following obstacle shape. *Complex & Intelligent Systems*, *7*(2), 741–751.

Thuesen, C., & Koch-Ørvad, N. (2018). Construction transformation. In *Sustain conference 2018: creating technology for a sustainable society* (pp. A–1). Technical University of Denmark.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Yang, H., Qi, J., Miao, Y., Sun, H., & Li, J. (2018). A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization. *IEEE Transactions on Industrial Electronics*, *66*(11), 8557–8566.

Yim, M. (2007). Modular self-reconfigurable robot systems: Challenges and opportunities for the future. *The IEEE Robotics & Automation Magazine*, *10*, 2–11.

Zhao, Z., Chen, W., Peter, C. C., & Wu, X. (2016). A novel navigation system for indoor cleaning robot. In *2016 IEEE international conference on robotics and biomimetics* (pp. 2159–2164). IEEE.

Zheng, K., Chen, G., Cui, G., Chen, Y., Wu, F., & Chen, X. (2017). Performance metrics for coverage of cleaning robots with MoCap system. In *International conference on intelligent robotics and applications* (pp. 267–274). Springer.