IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 2, 2020, accepted June 29, 2020, date of publication July 2, 2020, date of current version July 15, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3006579*

# Multi-Objective Genetic Algorithm-Based Autonomous Path Planning for Hinged-Tetro Reconfigurable Tiling Robot

**KU PING CHENG[1], RAJESH ELARA MOHAN[1], NGUYEN HUU KHANH NHAN[2], AND ANH VU LE[1,2]**

[1]ROAR Laboratory, Engineering Product Development Pillar, Singapore University of Technology and Design, Singapore 487372
[2]Optoelectronics Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Nguyen Huu Khanh Nhan (nguyenhuukhanhnhan@tdtu.edu.vn)

This work was supported in part by the National Robotics Programme through its Robotics Enabling Capabilities and Technologies (Funding Agency) under Project 192 25 00051, in part by the National Robotics Programme through its Robot Domain Specific (Funding Agency) under Project 192 22 00058, and in part by the Agency for Science, Technology, and Research.

**ABSTRACT** Reconfigurable robots have received broad research interest due to the high dexterity they provide and the complex actions they could perform. Robots with reconfigurability are perfect candidates in tasks like exploration or rescue missions in environments with complicated obstacle layout or with dynamic obstacles. However, the automation of reconfigurable robots is more challenging than fix-shaped robots due to the increased possible combinations of robot actions and the navigation difficulty in obstacle-rich environments. This paper develops a systematic strategy to construct a model of hinged-Tetromino (hTetro) reconfigurable robot in the workspace and proposes a genetic algorithm-based method (hTetro-GA) to achieve path planning for hTetro robots. The proposed algorithm considers hTetro path planning as a multi-objective optimization problem and evaluates the performance of the outcome based on four customized fitness objective functions. In this work, the proposed hTetro-GA is tested in six virtual environments with various obstacle layouts and characteristics and with different population sizes. The algorithm generates Pareto-optimal solutions that achieve desire robot configurations in these settings, with O-shaped and I-shaped morphologies being the more efficient configurations selected from the genetic algorithm. The proposed algorithm is implemented and tested on real hTetro platform, and the framework of this work could be adopted to other robot platforms with multiple configurations to perform multi-objective based path planning.

**INDEX TERMS** Reconfigurable robot, tiling robotics, multi-objective path planning, genetic algorithm, NSGA-II.

## I. INTRODUCTION

Path Planning (PP) has been a fundamental field of study for autonomous mobile robots. For instance, autonomous underwater vehicles (AUVs) and autonomous surface vehicles (ASVs) enter dangerous waters to perform environmental monitoring or mapping [1], [2], and some autonomous surveillance robots are designed and deployed in military operations to perform investigation and rescue tasks in confined spaces or hard-to-reach areas. Due to the autonomous nature of the robots and the hazardous working environments they are deployed in, precise and adaptive path planning algorithms are crucial for them to operate appropriately.

The goal of mobile robot PP is to determine a collision-free path between the starting and goal points while optimizing the specific performance criterion. Some of the commonly adopted criteria include time consumption, energy consumption, and distance traveled [3].

In general, robot workspaces can be categorized into static environments and dynamic environments. [4] The two categories of methods can be further classified based on whether the robot possesses complete information regarding the surrounding environment [5]. A global path planning method studies the map in a fully observable environment and

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

generates a globally optimized path. The setup of a global PP algorithm allows the navigation path to be determined before the robot motion begins. On the other hand, a local PP is implemented in an unknown or partially known environment, where data feed from robot on-board sensors are required to construct environment maps and to direct the operations [6].

Reconfigurable robots are kinematic machines with variable morphologies, and the development of reconfigurability in robotics has received increasing attention since the 1980s [7]. Reconfigurable robot platforms can be categorized into two major classes [7] : intra-reconfigurable and inter-reconfigurable robots. An intra-reconfigurable robot changes its internal morphology without the requirement of external assembly or disassembly [8]. Examples include amphibious robots equipped with eccentric paddle mechanism (ePaddle) that enables versatile locomotion in amphibious environments [9] and pavement sweeping robot Panthera with the robot footprint can change the width to adapt with the working conditions [10], [11], Hornbill with reconfigurable manipulators [12]. An inter-reconfigurable robot consists of a congregation of homogeneous or heterogeneous modules and forms a variety of morphologies through assembly and disassembly process. Examples include chain-type systems like Tetriamond [13], CONRO [14], hTetro-Infi [15], hTrihex [16], Crystal [17], and M-Lattice [18]; as well as hybrid-type robots like M-TRAN [19].

Though a massive rise of reconfigurable robots is seen, autonomy systems developed for these platforms mostly emphasize autonomous motion control, and few explore the autonomous PP problem of reconfigurable robots. PP problem is modeled differently in reconfigurable robots and in fixed morphology robots. For fixed-morphology robots, exhaustive search such as Dijkstra and A* algorithms are commonly utilized to solve global PP problems; on the other hand, Ant Colony Optimization (ACO) [20], Particle Swarm Optimization (PSO) [21], Neural Network [22], Motion Planning [23], Path Tracking [24], Graph theory [25] and Genetic Algorithm (GA) [26] have been implemented to solve local PP problems. Among the existing PP methods, GA has shown its strength in convenient modeling, easy implementation, and practical problem solving [27] due to its flexibility to perform optimization without prior information [28], and its ability to explore the solution space [29], which hinges on the advantages of both deterministic and probabilistic schemes to improve solutions using operators like crossover and mutation [30]. Multiple modified genetic algorithms (MGAs) have been developed specifically for path planning tasks [31] and have been implemented on various autonomous robots that operate in environments with complicated terrains or with dynamic obstacles, like mobile manipulator robots [32] and unmanned aerial vehicles (UAVs) [33].

However, due to the intrinsic complexity of reconfigurable robots, autonomous motion planning between different configurations has been a difficult topic. PP problems for reconfigurable robots, which involve multiple configurations, are more challenging. With the increased degrees of freedom in these robots and the additional constraints due to different robot configurations, PP approaches designed for fixed-morphology robots mentioned above are no longer sufficient to determine optimal solutions. New or revised architecture-specific PP approaches have been designed to tackle PP problems base on the possible topology and the available motions. For instance, revised GAs with customized fitness functions are implemented to solve the PP problem of the lattice modules in M-Lattice robot [34]. To overcome stairs and obstacles, Kairo 3 robot makes use of extended RRT* algorithm [35] to autonomously calculate the actions required for the tasks [36]. Research has also been conducted to provide heuristic-based algorithms [37] and distributed planning algorithms [38] for lattice-type inter-reconfigurable robots that are less architecture-specific.

The hTetro robot platform, developed by Prabakaran *et al.* [39] is a chained-type inter-reconfigurable cleaning robot with seven potential configurations [40] which utilizes tileset theory to perform area coverage tasks with the awareness of energy consumption [41]–[43]. To carry out PP tasks on a hTetro platform, the algorithm should determine a valid path while taking several additional criteria into consideration including time consumption, path safety, and path smoothness. Time efficiency and safety consideration are generally required for real robot implementation [44], while path smoothness aims to improve robot service qualities for robots that could not easily perform jerk motion of state switching [45]. Therefore, a multi-objective evolutionary algorithm (MOEA) has been utilized in the proposed GA to approximate the Pareto optimal solution of any given environment settings for hTetro. Similar approaches that model multi-objective optimization problems (MOOP) for robot path planning tasks and attempt to solve them through evolutionary algorithms are shown in the works of [46] and [47].

The contribution of this paper is threefold. First, a new systematic approach path planning for tetromino-based reconfigurable robot using novel multi-objective genetic algorithm is proposed. The Pareto solution for the proposed optimization problem is found using modified Non-dominated sorting genetic algorithm-II (NSGA-II) [48]. Second, the proposed robot and workspace modeling techniques can be modified and implemented in other chain-type inter-reconfigurable robot platforms with multiple configurations. And third, with proper definitions of the fitness functions, the multi-objective optimization framework proposed in this work could be easily adapted to other robot architectures which also aim to achieve multiple goals during the path planning process.

The rest of the paper is organized as follows: Section II describes the reconfigurable robot platform and the workspace model. Section III provides a brief introduction to genetic algorithms. In Section IV, the proposed hTetro-GA is presented. Section V shows the simulations and results of the proposed algorithm. Finally, Section VII presents the conclusions along with a note of future developments.

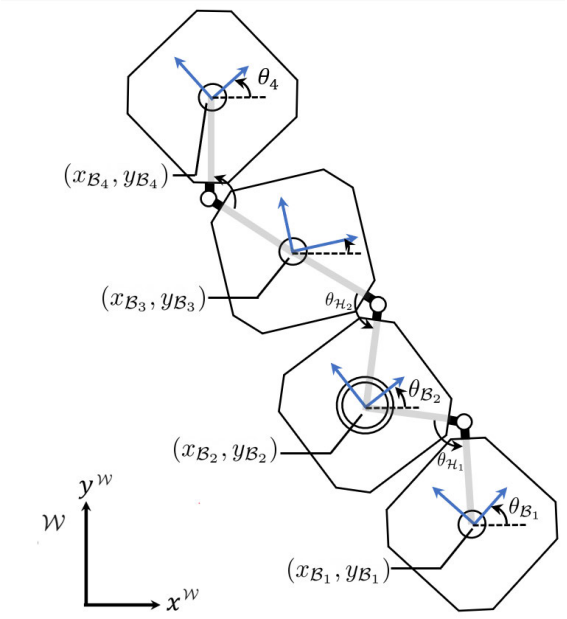



**FIGURE 1.** hTetro system model.

## II. HINGED-TETRO RECONFIGURABLE ROBOT

In this section, the different morphologies and actions available for the hTetro robot are introduced. The system model setup of the proposed path planning algorithm is also presented.

### A. hTetro HARDWARE ARCHITECTURE

hTetro is a chain-type inter-reconfigurable tiling robot as shown in Fig.1. The robot modules are referred to as "blocks" which share identical mechanical structures. The blocks could be disassembled freely, and additional blocks could be added to increase the degree of freedom of the entire architecture. In our work, the robot consists of four blocks connected by three hinges. The hinges allow the robot platform to perform shape-shifting and reassemble into multiple configurations. The top view graph of hTetro hardware components is shown in Fig.2. The perception component of hTetro is an RPLidar fixed on block 2 ($\mathcal{B}_2$). Each block is mounted with four geared 7.4V DC motors for balance locomotion. The servo motors mounted to the hinges could rotate the blocks clockwise or counter-clockwise to perform reconfiguration. The servo motors operate at 14.8V and with high torque of 77 kg.cm, which is enough to carry the robot block masses during transformation and to lock the position of the blocks during locomotion. Two servo motors are placed in block 2, and one sits in block 4. The Intel compute stick with ROS [49] based system installed controls all hTetro operations.

### B. hTetro ROBOT MODEL AND CONFIGURATION

In Fig.1, workspace $\mathcal{W} \subset \mathbb{R}^2$ is the 2-D Cartesian space where the hTetro robot navigates. The geometries of the four

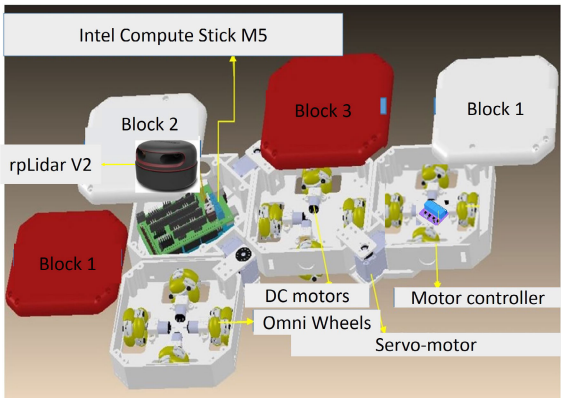



**FIGURE 2.** hTetro hardware components.

hTetro blocks are represented as $\mathcal{B}_n$ $(n = \{1, 2, 3, 4\})$, which are modelled as four squares of the width $d_{\mathcal{B}}$. The angle differences between the local frames of each block and the workspace frame are denoted as $\theta_{\mathcal{B}_n}$ $(n = \{1, 2, 3, 4\})$. The hinges are represented as $\mathcal{H}_n$ $(n = \{1, 2, 3\})$, and the hinge angles between two blocks are denoted as $\theta_{\mathcal{H}_n}$ $(n = \{1, 2, 3\})$, which follow the rotation constraints below:

$$\frac{\pi}{2} \leq \theta_{\mathcal{H}_1} = \frac{\pi}{2} + \theta_{\mathcal{B}_1} - \theta_{\mathcal{B}_2} \leq \frac{3\pi}{2}$$
$$\frac{\pi}{2} \leq \theta_{\mathcal{H}_2} = \frac{\pi}{2} + \theta_{\mathcal{B}_2} - \theta_{\mathcal{B}_3} \leq \frac{3\pi}{2}$$
$$\frac{\pi}{2} \leq \theta_{\mathcal{H}_3} = \frac{\pi}{2} + \theta_{\mathcal{B}_4} - \theta_{\mathcal{B}_3} \leq \frac{3\pi}{2}$$

The different combinations of hinge angles $\theta_{\mathcal{H}_n}$ $(n = \{1, 2, 3\})$ form shapes that simulate that of one-sided tetrominoes as shown in Fig. 3, which are defined as the 7 basic morphologies of an hTetro robot. The hinge angle combinations that form these morphologies are listed in Table 1.

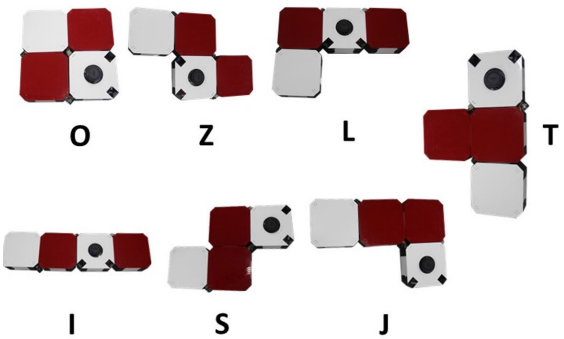



**FIGURE 3.** 7 basic morphologies of hTetro.

A total of six parameters are required to determine an hTetro configuration in $\mathcal{W}$, which is presented as follow:

*Definition 1 (Robot Configuration):* The configuration $\mathbf{q}$ of a hTetro robot is a six-element array

$$\mathbf{q} = [x_{\mathcal{B}_2}, y_{\mathcal{B}_2}, \theta_{\mathcal{B}_1}, \theta_{\mathcal{B}_2}, \theta_{\mathcal{B}_3}, \theta_{\mathcal{B}_4}]^T \quad (1)$$

**TABLE 1.** hTetro morphology table.

| Morphology | $[\theta_{\mathcal{H}_1}, \theta_{\mathcal{H}_2}, \theta_{\mathcal{H}_3}]$ | Morphology | $[\theta_{\mathcal{H}_1}, \theta_{\mathcal{H}_2}, \theta_{\mathcal{H}_3}]$ |
|---|---|---|---|
| I-shape | $[\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}]$ | O-shape | $[\frac{\pi}{2}, \frac{3\pi}{2}, \frac{\pi}{2}]$ |
| L-shape | $[\frac{3\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}]$ | J-shape | $[\frac{3\pi}{2}, \pi, \frac{3\pi}{2}]$ |
| Z-shape | $[\frac{3\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}]$ | S-shape | $[\frac{3\pi}{2}, \pi, \frac{\pi}{2}]$ |
| T-shape | $[\pi, \frac{3\pi}{2}, \frac{3\pi}{2}]$ | | |

where $x_{\mathcal{B}_2}$, $y_{\mathcal{B}_2}$ represents the $x$ and $y$ coordinates of the center of hTetro Block 2 ($\mathcal{B}_2$) as illustrated in Fig.1

### C. WORKSPACE MODEL

The workspace defined in this paper is modeled through the grid-based method. By performing a simple cellular decomposition technique with a fixed resolution, the workspace is separated into a collection of square-shaped grid cells of size $d_{\text{grid}}$. Based on the occupancy of obstacles in the workspace, each grid cell contains a variable that states whether any obstacle is in presence within the grid area [50]. In this paper, the grid width $d_{\text{grid}}$ is set to be identical to the hTetro block width $d_{\mathcal{B}}$ and the occupancy of obstacles in the workspace is defined as $\mathbf{W}_{\text{obs}}$.

### D. hTetro ROBOT MOTION AND NAVIGATION

Based on the architecture design of hTetro robot by Veerajagadheswar *et al.* [39], each hTetro block uses four omnidirectional wheels as its moving mechanism and utilizes hinge motors to shape-shift into different robot morphologies. This mechanical design enables an hTetro robot to perform three types of motion: translation ($\mathcal{T}$), rotation ($\mathcal{R}$), and shape-shift ($\mathcal{S}$).

The omnidirectional wheel mechanism allows the robot to perform an instant change of its moving direction. In this paper, a single translation motion command ($\mathcal{T}$) moves the hTetro robot in one of the four directions for a distance of $d_{\text{grid}}$. When the robot rotates, it rotates against the axis that passes through the center of $\mathcal{B}_2$ (denoted as $(x_{\mathcal{B}_2}, y_{\mathcal{B}_2})$ in Fig.1). In this paper, a single rotation motion command ($\mathcal{R}$) rotates the entire robot for 90° clockwise or counterclockwise. In the case of shape-shift ($\mathcal{S}$) motions, we assume that the desired shape to be $M$ and the initial hTetro block angles are $\theta^i_{\mathcal{B}_n}$. The ideal hinge angles for shape $M$ according to Table 1 ($\theta^M_{\mathcal{H}_n}$) will be utilized to determine the required heading angle change ($\triangle\theta_{\mathcal{B}_n}$) of each block during shape-shifting and is calculated as follow:

$$\triangle\theta_{\mathcal{B}_1} = \theta^i_{\mathcal{B}_1} - \theta^i_{\mathcal{B}_2} - \theta^M_{\mathcal{H}_1} + \frac{\pi}{2}$$
$$\triangle\theta_{\mathcal{B}_3} = \theta^i_{\mathcal{B}_3} - \theta^i_{\mathcal{B}_2} - \theta^M_{\mathcal{H}_2} + \frac{\pi}{2}$$
$$\triangle\theta_{\mathcal{B}_4} = \theta^i_{\mathcal{B}_4} - \theta^i_{\mathcal{B}_3} - \theta^M_{\mathcal{H}_3} + \frac{\pi}{2}$$

Table 2 introduces the motion commands (mc) for hTetro robots. The motion commands represent the encoding genes in the proposed hTetro-GA. The next robot configuration ($\mathbf{q}_{s+1}$) after a command is issued can be calculated simply by adding the change of robot configuration ($\triangle\mathbf{q}$) to current robot configuration ($\mathbf{q}_s$):

$$\mathbf{q}_{s+1} = \mathbf{q}_s + \triangle\mathbf{q} \tag{2}$$

**TABLE 2.** hTetro configuration motion command table.

| Actions | Motion command $\mathbf{mc}$ | Configuration change $\triangle\mathbf{q}$ |
|---|---|---|
| Translation ($\mathcal{T}$) | $\mathcal{T}_{x_+}$ (right) | $[d_{\text{grid}}\ 0\ 0\ 0\ 0\ 0]^T$ |
| | $\mathcal{T}_{x_-}$ (left) | $[-d_{\text{grid}}\ 0\ 0\ 0\ 0\ 0]^T$ |
| | $\mathcal{T}_{y_+}$ (forward) | $[0\ d_{\text{grid}}\ 0\ 0\ 0\ 0]^T$ |
| | $\mathcal{T}_{y_-}$ (backward) | $[0\ -d_{\text{grid}}\ 0\ 0\ 0\ 0]^T$ |
| Rotation ($\mathcal{R}$) | $\mathcal{R}_+$ (90° rotation) | $[0\ 0\ 0\ \pi/2\ 0\ 0]^T$ |
| | $\mathcal{R}_-$ (−90° rotation) | $[0\ 0\ 0\ -\pi/2\ 0\ 0]^T$ |
| Shapeshift ($\mathcal{S}$) | $\mathcal{S}_M$(shapeshift to $M$) | $[0\ 0\ \triangle\theta_{\mathcal{B}_1}\ 0\ \triangle\theta_{\mathcal{B}_3}\ \triangle\theta_{\mathcal{B}_4}]^T$ |

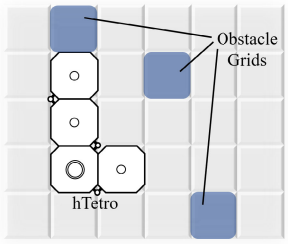
For robot platforms with fixed morphologies, route optimization usually focuses on minimizing the entire distance traveled. The path planning algorithms developed for these platforms attempt to search for the ideal trajectory with the shortest distance to navigate the robot from source to destination. Nevertheless, this is not the case for hTetro due to the three different motions it could potentially perform. Defining minimum distance traveled as the optimal goal for hTetro completely omits the cost of rotation motion and shape-shifting; therefore, an alternative optimization goal has to be defined, which is introduced later in section IV-B where the multi-objective evaluation technique is implemented.

### E. hTetro MOTION VALIDITY ANALYSIS

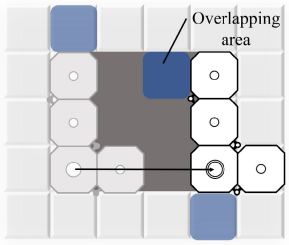
The simplicity to implement of approximate cellular decomposition [6] for the defined workspace has made the grid-based model to be one of the most popular path planning (PP) methods. A*, D*, and D* Lite based algorithms [51], [52] are commonly implemented to produce low-cost paths in minimum distance traversal problems; while wavefront PP [53], BSA [54], [55], and spiral-STC [56] algorithms aim to tackle coverage PP problems on grid-based models. These algorithms assume fixed-morphology robots with robot sizes smaller or equal to the grid size, so that the validity of the generated paths simply depends on whether these paths overlap with obstacle grids in the environment.

While designing grid-based PP for reconfigurable robots, however, determining the path validity becomes more complicated, which requires a full evaluation of the robot motions. In the case of hTetro, a crucial task is to ensure that the geometries of the four hTetro blocks do not intersect with any of the obstacle grids while the robot performs the three main motions: translation ($\mathcal{T}$), rotation ($\mathcal{R}$) and shape-shifting ($\mathcal{S}$). This validity check is the core constraint in the proposed hTetro-GA, and robot individuals that perform invalid motions during the navigation are likely to be rejected in the algorithm.

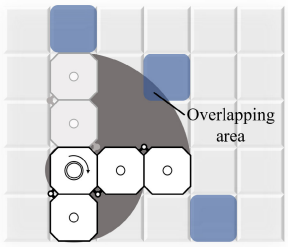
An example of robot motions and motion validity check is demonstrated in Fig.4. Fig.4a illustrates the starting position



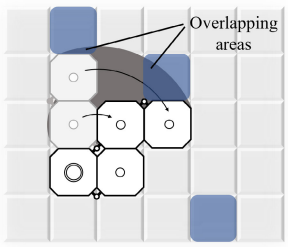

(a) hTetro starting position.




(b) Translation ($\mathcal{T}$) motion.




(c) Rotation ($\mathcal{R}$) motion.




(d) Shape-shift ($\mathcal{S}$) motion.

**FIGURE 4.** **Three hTetro motions and motion validity analysis.**

of an hTetro robot in the workspace with an L-shaped morphology. The obstacle grids are painted in blue in the presented scenario. Linear translation ($\mathcal{T}$) motion with command '$\mathcal{T}_{X_+}$' is performed in Fig. 4b, rotation ($\mathcal{R}$) motion with command '$\mathcal{R}_-$' is performed in Fig.4c, and shape-shifting ($\mathcal{S}$) with command '$\mathcal{S}_S$' is performed in Fig.4d. Areas swept by the four blocks are shaded. The motions described in these scenarios are invalid due to the overlapping areas between the swept areas and the obstacle grids. The validity check in each block is conducted through a point in the polygon (PIP) check [57], which allows an accurate estimation of the intermediate configurations of hTetro in continuous space when performing actions like shape-shifting.

## III. INTRODUCTION TO GENETIC ALGORITHM

Genetic algorithm (GA) is a universal searching and optimization algorithm introduced by John Holland based on the mechanics of Darwin's theory of evolution [58]. During the initialization of GA, a population of randomly generated individuals (chromosomes) is determined. The evaluation process in GA is then launched to calculate the corresponding fitness values for each individual. The selection criteria filter out individuals with weak performances. A new generation of the population is determined based on the encoded genes in the remaining individuals through biologic genetic operators such as mutation and crossover [29]. GA process creates offspring generation of populations that are more adapted to the environments and demonstrate better performance compared to their parents [59], [60].

Various GA algorithms have been developed to tackle PP problems considering GAs generally provide great potential and flexibility to solve combinatorial optimization problems [61]. Many of these algorithms modeled the environment utilizing cellular decomposition and grid-based methods. Y. Hu *et al.* implemented a knowledge-based GA with domain knowledge and small-scale local search in [26], which is capable of finding near-optimal robot path in both static and dynamic environments.

## IV. PROPOSED hTetro-GA

This paper expands on previous works of evolutionary algorithm based path planning described in Section III by modeling the environment as grid-based cells and proposes hTetro-GA, a global multi-objective genetic algorithm (MOGA) that solves the MOOP and provides the Pareto-optimal solution that navigates the hTetro robot to any desired destination. A list of terminologies frequently used in hTetro-GA is shown in Table 3 for reference.

Fig.5 illustrates the flowchart of the proposed hTetro-GA. hTetro-GA takes the workspace obstacle map ($\mathbf{W}_{\text{obs}}$), the roadmap ($\mathbf{Q}$), and several GA-related parameters as input and produces the ideal hTetro motion command sequence. A roadmap ($\mathbf{Q}$) is a series of predefined hTetro robot configurations ($\mathbf{q}$), which specifies the series of positions and morphologies that the robot should arrive at during the navigation process.

As shown in the flowchart, the hTetro-GA consists of three main loops: configuration loop, population loop, and child loop. Within a configuration loop, the hTetro robot navigates to the next configuration in the roadmap ($\mathbf{Q}$), and the loop terminates once the hTetro robot reaches the last configuration specified in $\mathbf{Q}$. In a configuration loop, GA is used to plan the path between different configurations. Once a new generation of the population is generated in the population loop, hTetro-GA operators and selection procedures are applied. The children ($p$) in the population perform simulated navigation in the child loop, in which the robot fitness values are evaluated to find the individual robot with the genes that result in the Pareto-optimal fitness values.

**TABLE 3. Terminologies in proposed hTetro-GA.**

| Terms | Type | Meaning |
|---|---|---|
| $\mathbf{q}$ | array | a predefined hTetro robot configuration ($\mathbf{q}$) array that represents the entire roadmap. |
| $\mathbf{W}_{\text{obs}}$ | array | a boolean array storing the occupancy of each grid in the workspace, its size equals to the number of grids in the workspace |
| $\mathbf{W}_{\mathbf{A}^*}$ | array | a float array that stores the $A^*$ distances between all configurations in the map to the next roadmap configuration, its size equals to number of grids in the workspace |
| $n_{\text{pop}}$ | var. | an integer variable that represents the total number of children in each population. |
| $\mathbf{P}^k$ | array | the $k$-th population in hTetro-GA, which is a $p$ array of size $n_{\text{pop}}$. |
| $\mathbf{P}^k_{\text{c}}$ | array | the $k$-th population generated through conservative GA operations, which is a $p$ array of size $n_{\text{pop}}$. |
| $\mathbf{P}^k_{\text{nc}}$ | array | the $k$-th population generated through non-conservative GA operations, which is a $p$ array of size $n_{\text{pop}}$. |
| $\mathbf{F}$ | array | the different fronts in non-dominated sorting process. $\mathbf{F}_k$ is a $p$ array that represents the $k$-th front. $\mathbf{F}_0$ is the leading front. |
| $\mathbf{mc}$ | set | a set of all possible hTetro motion commands(mc). $\mathbf{mc} = \{"\mathcal{T}_{x_+}","\mathcal{T}_{x_-}","\mathcal{T}_{y_+}","\mathcal{T}_{y_-}","\mathcal{R}_+","\mathcal{R}_-","\mathcal{S}_M"\}$ |
| $p$ | array | a string array that stores motion commands(mc) in a robot individual. $p_i$ represents the $i$-th motion command in $p$, where $p_i \in \mathbf{mc}$ |
| $l_p$ | var. | the length of the array $p$, which represents the total number of motion commands in a robot individual's gene |
| $l_{p,\text{max}}$ | const. | a predefined maximum starting length of a gene in robot individuals. |
| $\lambda_{\text{mc}}$ | var. | the starting probability of motion command mc in the gene, where mc $\in \mathbf{mc}$ |
| $\mathbf{sp}$ | set | the searching pattern for path safety fitness calculation. |
| $\lambda_{\mathbf{sp}}$ | const. | the searching pattern coefficient |
| $r_{\text{mu}}$ | var. | the mutation rate of each command in hTetro robot individuals ($p$) |
| $r_{\text{cr}}$ | var. | the crossover rate between two hTetro robot individuals ($p$) |
| $r_{\text{rm,t}}$ | var. | the removal rate of each translational command in hTetro robot individuals ($p$) |
| $r_{\text{rm,nt}}$ | var. | the removal rate of each non-translational command in hTetro robot individuals ($p$) |
| $r_{\text{rm,ra}}$ | var. | the rearrangement rate of each command in hTetro robot individuals ($p$) |

### A. INITIALIZATION OF hTetro-GA

In this paper, the configuration sequences in $\mathbf{Q}$ are denoted as $\mathbf{q}_s$, where $s$ represents the sequence number of the configuration. The next ideal configuration is represented as $\mathbf{q}_{s+1}$. In hTetro-GA, genetic algorithm is performed to determine the path between the two configurations – $\mathbf{q}_s$ and $\mathbf{q}_{s+1}$, and the parent population generates two types of offspring populations: (1) conservative GA operation population $\mathbf{P}^k_{\text{c}}$, and (2) non-conservative GA operation population $\mathbf{P}^k_{\text{nc}}$, where $k$ represents the $k$-th population in the algorithm. Details regarding $\mathbf{P}^k_{\text{c}}$ and $\mathbf{P}^k_{\text{nc}}$ will be introduced in Section IV-C.

During the population initialization process in the configuration loop, the first generation ($\mathbf{P}^0$) is generated, and the pseudo-code is shown in Algorithm 1. $\mathbf{P}^0$ consists of a total of $n_{\text{pop}}$ robot individuals. A robot individual is represented as $p$, which stores genetic information. The motion commands (mc) introduced in Table 2 are implemented to encode the chromosomes in the proposed hTetro-GA, and the total number of motion commands in robot individual $p$ is represented as $l_p$. During the initialization process of hTetro-GA, a predefined maximum length ($l_{p,\text{max}}$) determines the starting length of a gene in robot individuals. In Algorithm 1, the encoded motion command of each child in $\mathbf{P}^0$ are randomly determined based on the starting probability coefficient of each motion command ($\lambda_{\text{mc}}$). The determination process follows proportionate reproduction selection [62], where commands with a larger starting probability coefficient are more likely to be chosen.

As shown in Fig.5, once the starting population $\mathbf{P}^0$ is determined, the navigation process of $\mathbf{P}^0$ begins. In the child loop, the fitness of the path is calculated after each robot motion is performed, and the process repeats until the encoded genes in $p$ satisfy the child termination criterion. A child robot terminates when it fails path validity check mentioned in Section II-E, when it reaches the next configuration $\mathbf{q}_{s+1}$ in $\mathbf{Q}$, or when all motion commands in $p$ are fully executed.

Once all children in the population complete the navigation, the hTetro-GA operators and the hTetro-GA selection process produce a new generation of populations until the population satisfies any population termination criterion. The termination criteria include population reaching the predefined maximum allowed generations of populations ($n_{\text{pop,max}}$) and the convergence of the solution.

**Algorithm 1** Initialization Process of hTetro-GA

```
// Generation of random individuals in a population
Spawn P^0 with a total of n_pop children.
for all child p ∈ P^0 do
    t ← 0
    while t < t_max do
        rnd ← random(0, 1)
        for all mc ∈ mc do
            if rnd < λ_mc then
                p.Push(mc)
                t ← t + t_mc
                break
            end if
            rnd ← rnd − λ_mc
        end for
    end while
end for
```

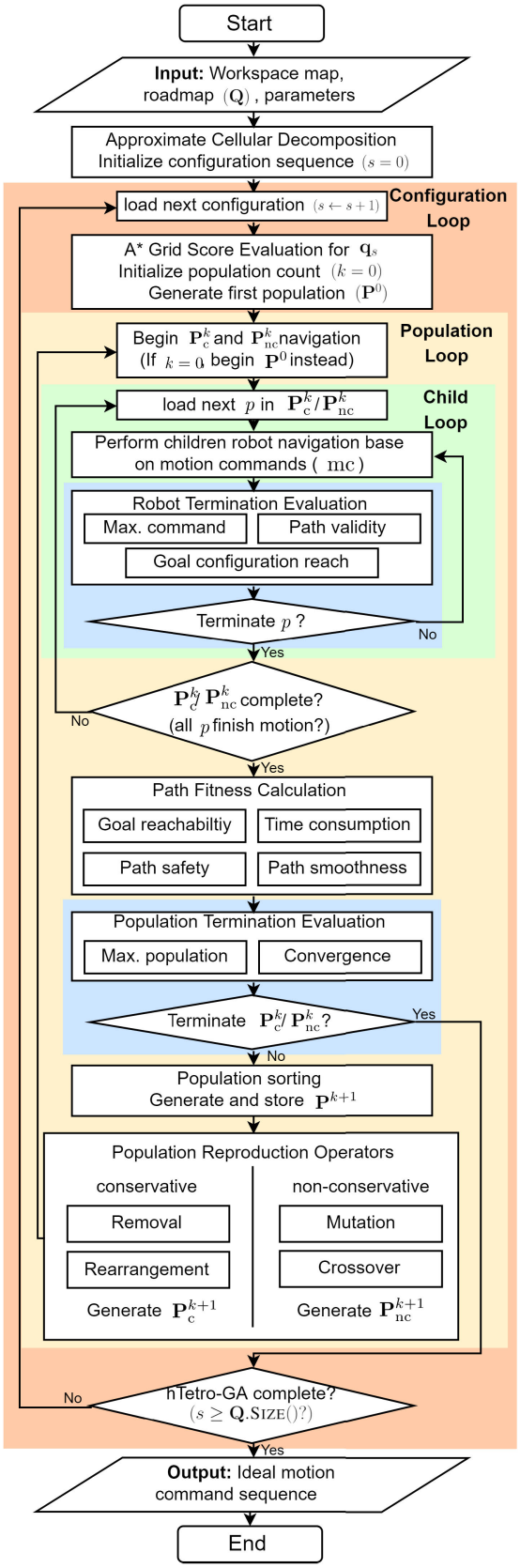



**FIGURE 5.** hTetro-GA flowchart.

### B. MAIN OBJECTIVES IN hTetro-GA OPTIMIZATION

The hTetro navigation problem described in this work is a multi-objective optimization problem (MOOP). In this paper, the ultimate goal within the GA operation in each configuration loop is to identify the best-performing motion command sequence that navigates the hTetro robot from $\mathbf{q}_s$ to $\mathbf{q}_{s+1}$. The MOOP evaluates the performance of the robot motion command using the four following criteria: **(1) goal reachability**, **(2) time consumption**, **(3) path smoothness**, and **(4) path safety**. Due to the command-based encoding genes in robot individuals ($p$), it is not guaranteed that all individuals in the population will arrive at the ideal configuration in the end. This limitation has introduced an additional constraint in our multi-objective optimization problem, and the solution should ensure that the first fitness value criterion – goal reachability – holds a higher priority compared to the three remaining criteria.

The four criteria selected for hTetro-GA are introduced and calculated as follow:

#### 1) GOAL REACHABILITY FITNESS ($f_{\text{gr}}$)

Goal reachability fitness is given the highest priority in the hTetro-GA path fitness evaluation process. The calculation of goal reachability fitness of robot individual $p$ is calculated using Equation 3.

$$f_{\text{gr}}(p) = \frac{1}{1 + \mathbf{W}_{A^*}(\text{Pos}(p_{l_p-1}))} \tag{3}$$

where:

$$\begin{aligned} \text{Pos}(p_i) &= \text{the position of block 2 } (x_{\mathcal{B}_2}, x_{\mathcal{B}_2}) \text{ after} \\ &\quad \text{executing } p_i \text{ (the } i\text{-th motion command} \\ &\quad \text{in individual } p) \\ \mathbf{W}_{A^*}(x, y) &= \text{the h-score in A* algorithm that estimates} \\ &\quad \text{the distance between position } (x, y) \text{ and the} \\ &\quad \text{position of the goal configuration [63]} \end{aligned}$$

In Equation 3, the function calculates the heuristic score of the position of block 2 after the final motion command is executed. An individual with an ending position near the goal position results in a higher $f_{\text{gr}}$ value. Once a robot individual reaches the next configuration ($\mathbf{q}_{s+1}$), the goal reachability fitness of the robot always yields a value of 1 since the distance between the robot and the goal configuration is 0. If the robot does not manage to reach the next configuration during the navigation process, the goal reachability fitness value will be less than 1.

#### 2) TIME CONSUMPTION FITNESS ($f_{\text{t}}$)

The time consumption fitness value of robot individual $p$ is calculated through Equation 4 as shown below.

$$f_{\text{t}}(p) = 1 - \frac{\sum_{i=1}^{l_p} t_{p_i}}{t_{\max}} \tag{4}$$

where:

$t_{p_i}$ = time consumption for hTetro motion command $p_i$ ($i$-th motion command in $p$), where $p_i \in \mathbf{mc}$

$\mathbf{mc} = \{\text{“}\mathcal{T}_{x_+}\text{”}, \text{“}\mathcal{T}_{x_-}\text{”}, \text{“}\mathcal{T}_{y_+}\text{”}, \text{“}\mathcal{T}_{y_-}\text{”}, \text{“}\mathcal{R}_+\text{”}, \text{“}\mathcal{R}_-\text{”}, \text{“}\mathcal{S}_M\text{”}\}$

In Equation 4, the denominator's value represents the total time consumption of the entire navigation process. A longer navigation process yields a smaller time consumption fitness value. It is worth noting that the calculation of the time consumption takes into the consideration of the fact that duration of hTetro rotation ($\mathcal{R}$) and shape-shifting ($\mathcal{S}$) may not be identical to the translation ($\mathcal{T}$) motion: the appearing frequencies of each motion command in the gene are multiplied by the time consumption of the corresponding motion based on real-world measurement values.

#### 3) PATH SMOOTHNESS FITNESS ($f_{\text{sm}}$)

The path smoothness fitness value of robot individual $p$ is calculated using Equation 5.

$$f_{\text{sm}}(p) = 1 - \frac{\sum_{i=1}^{l_p-1} dif(p_i, p_{i+1})}{l_p - 1} \tag{5}$$

where:

$$dif(p_i, p_{i+1}) = \begin{cases} 1, & \text{if } p_i \neq p_{i+1} \\ 0, & \text{if } p_i = p_{i+1} \end{cases} \quad (p_i, p_{i+1} \in \mathbf{mc})$$

In Equation 5, a smaller difference between neighbor motion commands contributes to a higher path smooth fitness value. This definition encourages path with high consistency in the motion commands.

#### 4) PATH SAFETY FITNESS ($f_{\text{sf}}$)

The path safety fitness describes the security of the overall robot path throughout the navigation, and its value of robot individual $p$ is calculated through Equation 6.

$$f_{\text{sf}}(p) = 1 - \frac{d_{\text{grid}}}{l_p \cdot \lambda_{\text{sp}}} \sum_{i=1}^{l_p} \sum_{n=1}^{4} \sum_{(x,y)_{\mathbf{sp}} \in \mathbf{sp}} \frac{\mathbf{W}_{\text{obs}}(P(p,i,n) + (x,y)_{\mathbf{sp}})}{||(x,y)_{\mathbf{sp}}||} \tag{6}$$

where:

$$\mathbf{W}_{\text{obs}}(x,y) = \begin{cases} 1, & \text{if } (x,y) \text{ is within an obstacle grid} \\ 0, & \text{otherwise} \end{cases}$$

In Equation 6, the robot sums up the occupancy information of the grids within a certain searching range throughout the navigation. The searching pattern profile is represented as **sp** with a searching pattern specific coefficient ($\lambda_{\text{sp}}$), and $(x, y)_{\mathbf{sp}}$ is a vector representation of a grid within the search pattern with respect to the origin, which is the center of block 2 in our case. In this paper, the searching patterns are circles with predefined radii and center at each block in the hTetro robot. With Equation 6, obstacles that are presented within the searching pattern reduce the overall path safety fitness ($f_{\text{sf}}$), and the robot individual is less desired during the optimization process. In Fig.6, a searching radius of $2 \cdot d_{\text{grid}}$ is implemented, and an obstacle grid may result in a decrease of the $f_{\text{sf}}$ value several times if it locates within the radii of multiple circles. This implementation ensures that the robot can navigate safely in any shape configuration by keeping a certain distance with the environment obstacles.

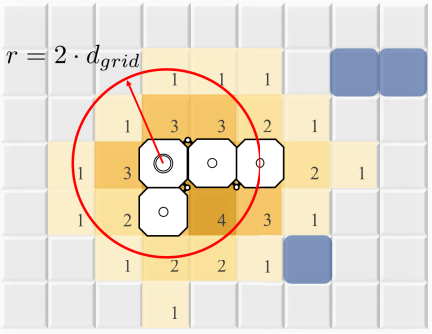



**FIGURE 6.** hTetro robot path safety fitness evaluation. The values in each grid represents the number of searching pattern circles the grid is within.

### C. hTetro-GA MULTI-OBJECTIVE TECHNIQUES AND POPULATION SORTING

Techniques that attempt to solve MOOPs have been vastly studied in the past two decades due to the presence of various objectives in recent research optimization problems. MOOPs are intractable optimization problems with the conflicting nature among the optimization parameters [64]. With the fitness values defined in Section IV-B, we can mathematically formulate the MOOP for hTetro robot navigation as follow:

$$\begin{aligned} &\text{minimize } F(p) = (-f_{\text{gr}}(p), -f_{\text{t}}(p), -f_{\text{sm}}(p), -f_{\text{sf}}(p))^T \\ &\text{subject to } p \in \Omega \end{aligned}$$

where:

$\Omega$ = the decision set which includes all feasible solutions $p$

Since an ideal path that optimizes all four fitness value functions does not exist in most case scenarios [65], the realization of Pareto optimality in the solutions has become our main focus. Details regarding the Pareto optimality is introduced in Stadler's work [66]. And the main goal of our MOOP is to find a Pareto optimal set, which consists entirely of Pareto optimal solutions $p^* \in \Omega$.

Multi-objective evolutionary algorithms (MOEAs) are capable of approximating the Pareto optimal sets within a single run [65] of the evolutionary algorithms. Non-dominated sorting genetic algorithm-II (NSGA-II), a classical MOEA technique, has been proven to provide results with better quality in solving MOOPs [67]. This paper implements the NSGA-II technique to approach our hTetro navigation MOOP due to the following advantages NSGA-II possess [48]:

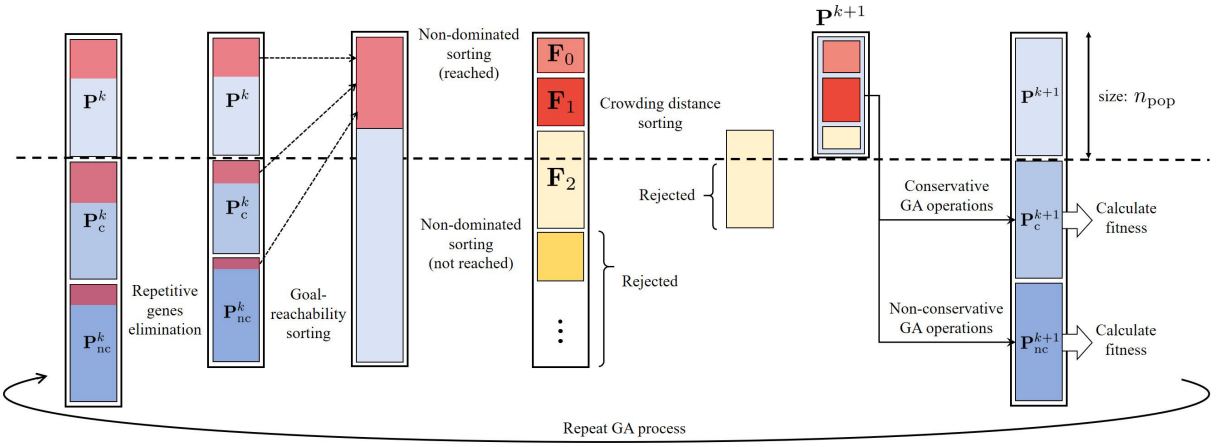



**FIGURE 7.** hTetro-GA with implementation of NSGA-II technique.

1) NSGA-II approach reduces the time complexity of the presented MOOP from $O(4n^3_{pop})$ to $O(4n^2_{pop})$.
2) The elitism approach in NSGA-II preserves the entire gene information in the process so that once a robot successfully reaches the goal, it is guaranteed that at least one of the individuals in the next generation population will reach the destination as well.
3) The density estimation and crowding distance calculation introduced in NSGA-II efficiently preserves the diversity among the population members, which is crucial in our presented MOOP so that different routes to the destination can be explored.

The implementation of NSGA-II techniques in our proposed hTetro-GA is shown in Fig.7. Several modifications are applied to the proposed algorithm to increase the converging speed and to improve the quality of the results. The procedure of the proposed hTetro-GA sorting and selection technique are listed as follow:

1) The algorithm takes in three groups of population, namely $\mathbf{P}^k$, $\mathbf{P}^k_c$, and $\mathbf{P}^k_{nc}$ for analysis.
2) The algorithm eliminates extra copies of robot individuals in the population that carry genetic information identical to existing individuals. This is to prevent identical well-performing individuals from doubling everytime hTetro-GA executes and swarming the fronts in future generation populations.
3) Perform **goal reachability sort**, which sorts the entire population into two sections. The section with robot individuals that reaches the goal ($f_{gr}(p) = 1$) is shaded in red in to Fig.7.
4) Perform **fast non-dominated sorting** independently in both sections in the previous step. The fast non-dominated sorting algorithm implemented is identical to the approach in the original NSGA-II paper [48]. Concatenate the results and generate a sequence of fronts $\mathbf{F}$. Since the goal reachability fitness has a higher priority compared to the other three criteria, robot individuals with smaller goal reachability fitness values ($f_{gr}$) are strictly Pareto-dominated by those with higher values.
5) Push the fronts to the next population $\mathbf{P}^{k+1}$. If the total individuals of all the fronts are larger than $n_{pop}$, perform **crowding distance sort** in the last front, which sorts the individuals in ascending orders. The calculation of crowding distance follows the formula introduced in the NSGA-II paper [48]. The well-performing individuals with small crowding distances in the last front are then pushed to $\mathbf{P}^{k+1}$ until all population slots are filled in.
6) Perform conservative GA operation and non-conservative GA operation on $\mathbf{P}^{k+1}$, which generates two extra population groups – $\mathbf{P}^{k+1}_c$ and $\mathbf{P}^{k+1}_{nc}$, respectively. Perform hTetro navigation and fitness analysis on the generated individuals and repeat the entire process.

### D. hTetro-GA REPRODUCTION OPERATORS

Apart from the reproduction operators that are implemented in classical GAs, such as genetic mutation and genetic crossover, this paper introduces two extra operators for the motion command-based hTetro robot genes, namely rearrangement, and removal.

Constantly improving the performance of hTetro robot individuals ($p$) across the population is crucial in GAs, and with the definition of the hTetro robot genetic model, the introduction of the three extra operators speeds up the algorithm and improves the performance of the results. In this paper, the genetic operations are separated into two categories: (1) **conservative GA operations** and (2) **non-conservative GA operations**. Conservative GA operations are operations that ensure that the $(x, y)$ portion of the vector sum of configuration changes ($\sum \triangle \mathbf{q}$) remains the same, which results in offspring robots with the same ending position as their parent robots. Conservative GA operations excel at exploring various possibilities to arrive at a certain destination, which is useful when $\mathbf{q}_{s+1}$ has already been explored

and reached. Once a robot reaches $\mathbf{q}_{s+1}$, conservative GA operations will improve the genes while keeping the goal reachability of the individual intact. Non-conservative GA operations, on the other hand, includes the operations that do not preserve the same ending position or morphology of the parent robots, which is crucial during the early phase of hTetro-GA when all robots are still searching for a valid path to the next configuration.

Non-conservative GA operations implemented in this paper include mutation and crossover, while conservative GA operations include removal and rearrangement. The operations are defined as follow:

### 1) MUTATION OPERATOR

The mutation operator in hTetro-GA is the classic GA mutation operator, which modifies a single motion command (mc) within the chromosome to a random motion command within the motion command set (*mc*). Since the motion commands in *mc* includes all three types of motions, a single mutation in a gene may alter a translation motion command into a shape-shift or a rotation command. Therefore, the navigation result of the mutated gene may end up with a different position or morphology compared to the parent gene. The mutation rate of each motion command is denoted as $r_{\text{mu}}$.

Fig.8a shows the mechanism of the mutation operator in hTetro-GA. The mutation operator helps increase the gene diversity and is the primary operator that contributes to the gene evolution during the early exploration phase of the algorithm when the goal remains undiscovered by any of the robot individuals. Mutating a translation motion command ($\mathcal{T}$) to rotation ($\mathcal{R}$) or shape-shifting ($\mathcal{S}$) is crucial when the robot tries to steer through narrow spaces or attempts to avoid dynamic obstacles.

### 2) CROSSOVER OPERATOR

The hTetro-GA crossover method implemented is a single-point crossover operator, and the offspring individual possesses the genetic information from both parents combined. The mechanism of the crossover operator is illustrated in Fig.8b. In hTetro-GA, crossover operation occurs with a rate of $r_{\text{cr}}$ on each robot individual.

To increase the likelihood of improvements in the offspring hTetro robot individuals, deterministic tournament selection with a tournament size of 4 is chosen to decide the two best parents for crossover operations. The individuals in the tournaments are compared based on their rankings determined previously in the NSGA-II procedure, and if the individuals locate in the same front, the crowding distances are compared instead.

### 3) REMOVAL OPERATOR

The hTetro-GA removal operator is introduced to eliminate motion commands within the robot individuals. Since the sorting procedure takes the time consumption fitness ($f_{\text{t}}$) into consideration, the individual that navigates to the goal with minimum time spent is always preferred. The removal operator continuously simplifies the motion commands in the individuals across each generation of populations, which gradually speeds up the computational process in each population loop as the chromosomes' sizes shrink after each generation of populations. The removal operator is a conservative GA operator, so the removed motion commands do not alter the final position or morphology of the hTetro robot.

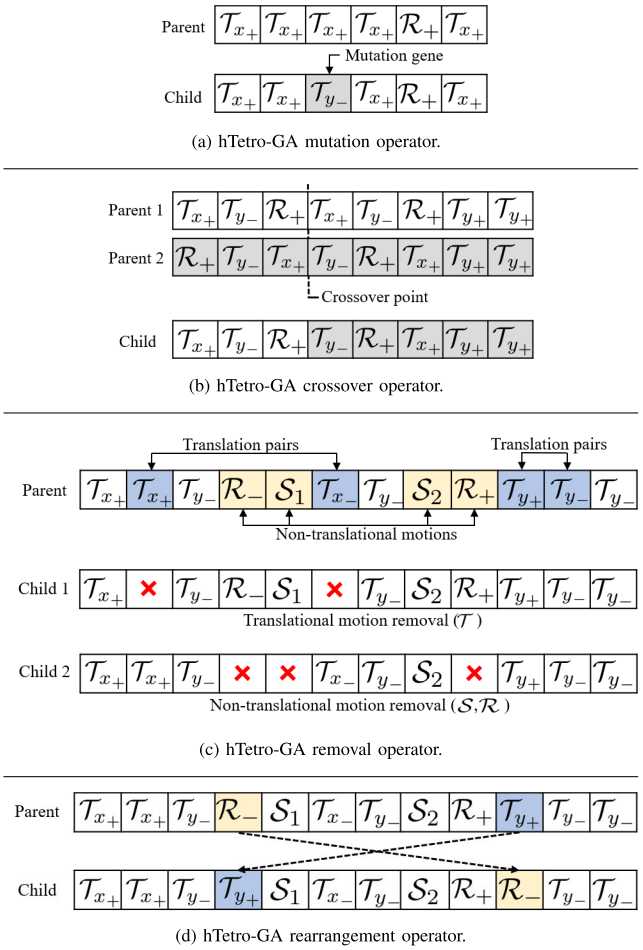



**FIGURE 8.** hTetro-GA reproduction operators.

Removal operations on non-translational motions such as rotation motions ($\mathcal{R}$) and shape-shift motions ($\mathcal{S}$) are applied arbitrarily. Each single point of non-translational command has a removal rate of $r_{\text{rm,nt}}$ in the gene. On the other hand, instead of single-point removals, removal operations on translational motions are executed in pairs to preserve the fidelity of path destination. The two translation pairs in this paper are: ("$\mathcal{T}_{x_+}$", "$\mathcal{T}_{x_-}$") and ("$\mathcal{T}_{y_+}$", "$\mathcal{T}_{y_-}$"). Each translation motion command has a removal rate of $r_{\text{rm,t}}$. While executing removal operations on translational motions, the algorithm randomly searches two translation motion commands within the chromosome. If the two commands form a translation pair, the operator removes both simultaneously. Fig.8c illustrates the possible outcomes of removal operations for both translational motion commands and non-translational motion commands, where the red "X" signs represent removed genes.

#### 4) REARRANGEMENT OPERATOR

hTetro rearrangement operator performs swapping between two random motion commands within a chromosome. Rearrangement is a conservative GA operation since all motion commands are preserved during the process. Fig.8d demonstrates the process of rearrangement operator. This operator works well during the later phase of the GA process when the goal has been explored, as it excels at exploring the different combinations of the motion commands that result in maximum path smoothness fitness ($f_{\text{sm}}$) value and maximum path safety fitness ($f_{\text{sf}}$) value. Each motion command has a rearrangement rate of $r_{\text{ra}}$.

## V. hTetro-GA SIMULATION

The evaluation of the proposed hTetro-GA is conducted through simulations in virtual environments. This section introduces the setup of the virtual environments and the parameters being evaluated. The section then analyzes the results and discusses the proper starting parameters for the proposed algorithm.

### A. VIRUTAL ENVIRONMENT SETUP

In this paper, virtual environments are defined by obstacle maps ($\mathbf{W}_{\text{obs}}$) and roadmaps ($\mathbf{Q}$). To comprehensively evaluate the proposed algorithm, virtual environments with different obstacles in terms of size, position, and mobility (static and dynamic) are built.

The environments presented in this study are all in the size of $24 \cdot d_{\text{grid}} \times 24 \cdot d_{\text{grid}}$ and are illustrated in Fig.9, which include H-shaped obstacles (Fig.9b), random obstacles (Fig.9a), spiral obstacles (Fig.9c), 3-slit obstacles (Fig.9d), perpendicular dynamic obstacles (Fig.9e), and parallel dynamic obstacles (Fig.9f) environments.

In these environments, the cellular decomposition method is implemented for all static and dynamic obstacles so that they are considered as grid-shaped obstacles within our model. During the robot termination evaluation process, the path viability is evaluated continuously to check whether any of the hTetro blocks collide with the obstacles. The dynamic obstacles in Fig.9e and Fig.9f perform simple patrolling motions on a predefined route. The navigation speed of the dynamic obstacles is set to $d_{\text{grid}}$ per time instance. Multiple roadmaps that specify robot configurations throughout the navigation are defined and are illustrated in the figures. All roadmaps in this study begin and end with configurations of O-shaped morphology. In the perpendicular dynamic obstacles environment (Fig.9e), four configurations are specified in the roadmap instead of just the starting and goal configuration, so the configuration loop in the hTetro-GA process (Fig.5) is performed three times.

### B. hTetro-GA SIMULATION PARAMETERS

Most evolutionary algorithms include multiple parameters such as population size ($n_{\text{pop}}$), mutation, and crossover rates. The best values for these parameters are usually problem-specific. In the hTetro-GA performance simulation, population size is an independent variable, while other parameters are considered as control variables. The parameters are introduced as follow:

#### 1) POPULATION SIZE ($n_{\text{pop}}$)

Population size is a parameter that significantly affects the performance of an evolutionary algorithm. An algorithm with a larger population size explores the solution space more completely, which yields better solutions in most cases, but it also requires more computational resources. In this simulation, population size of $n_{\text{pop}} = 25, 50, 100$ are analyzed. The algorithm's efficiency is evaluated by tracking the population number and time consumption when the robot first reaches the next configuration and when the population loop terminates. The effectiveness of the algorithm is determined through the fitness values of the final hTetro motion command sequence $\hat{\mathbf{m}}_c$. In this simulation, hTetro-GA is executed 50 times for each population size in each virtual environment.

#### 2) REPRODUCTION OPERATOR PARAMETERS

The parameters used in reproduction operators affect the gene diversity between the parent and child populations. The parameters introduced in section Section IV-D include $r_{\text{mu}}$, $r_{\text{cr}}$, $r_{\text{rm,t}}$, $r_{\text{rm,nt}}$, and $r_{\text{ra}}$.

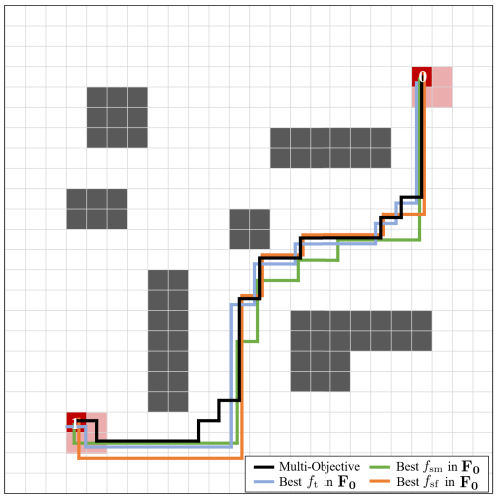
In the non-conservative operation process of hTetro-GA, the mutation rate $r_{\text{mu}}$ is set to $\frac{1}{l_{p,\max}}$, while the crossover rate $r_{\text{cr}}$ is set to 0.5 throughout the navigation process. The selection of $r_{\text{mu}}$ results that approximately 1 mutation occurs in each individual in the next population group ($\mathbf{P}_{\text{nc}}^{k+1}$) during the early phase of navigation when the goal configuration has not been reached. This mutation rate produces fewer mutations in the later phase of navigation so that other GA operators are weighted more while refining the path information in the genes. On the other hand, setting $r_{\text{cr}}$ to a constant 0.5 represents that around half of $\mathbf{P}_{\text{nc}}^{k+1}$ is generated from crossover operations, allowing fast solution exploration of the algorithm.

In conservative operation process, the translational motion command removal rate $r_{\text{rm,t}}$, non-translational motion command removal rate $r_{\text{rm,nt}}$, and the rearrangement rate $r_{\text{ra}}$ are all set to $\frac{1}{l_p}$. This value will result in one single removal operation to occur in each individual on average, which provides sufficient gene variety in the next conservative population group ($\mathbf{P}_{\text{c}}^{k+1}$) to continue the GA process.

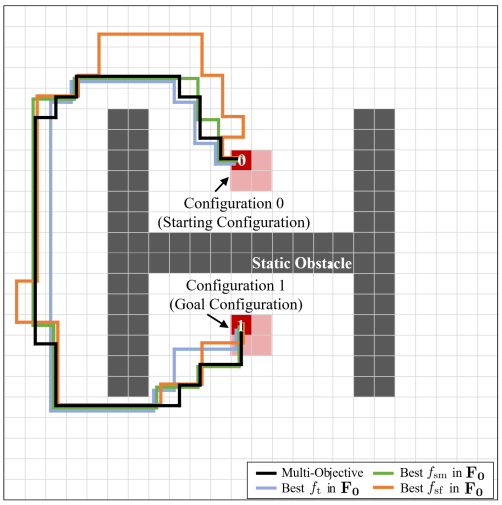
#### 3) hTetro-GA SPECIFIC PARAMETERS

Several problem-specific parameters are introduced during the initialization of the hTetro-GA process in this simulation.

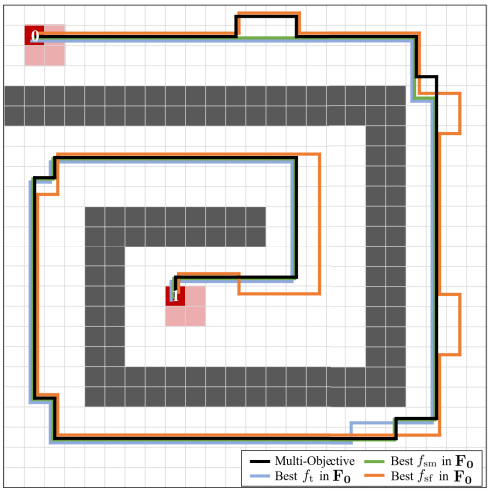
The starting probability coefficient of each motion command ($\lambda_{\text{mc}}$) determines the appearance rate of each motion command in the starting gene. The coefficients for the translational motions are set to 0.25, and the coefficients for rotation and shape-shifting are set to 0. This setup speeds up the searching process of the goal configuration as rotations and



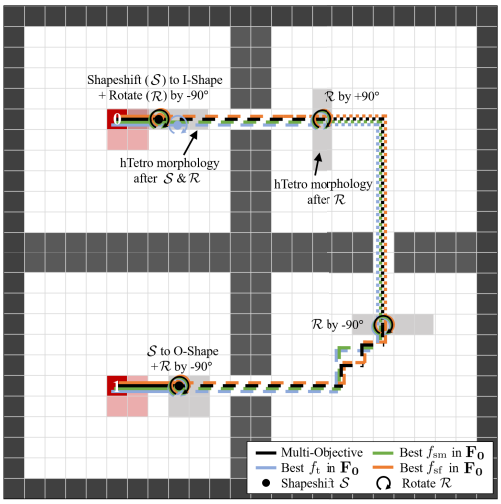

(a) Random Obstacles Environment



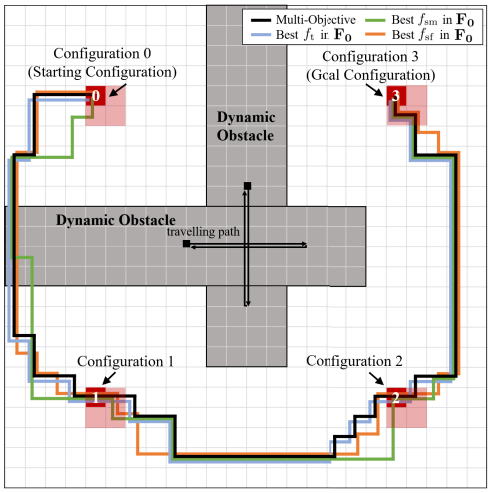

(b) H-shaped Obstacle Environment



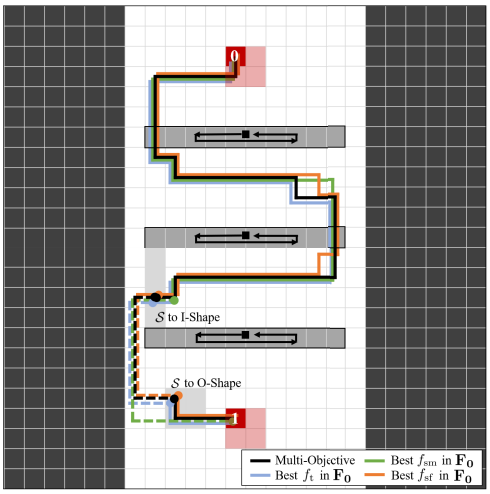

(c) Spiral Obstacle Environment




(d) 3-Slit Obstacle Environment




(e) Perpendicular Dynamic Obstacles Environment




(f) Parallel Dynamic Obstacles Environment

**FIGURE 9.** Virtual environment constructed for hTetro-GA analysis and ideal path results based on best time consumption fitness ($f_c$), best path smoothness fitness ($f_{sm}$), best path safety fitness ($f_{sf}$), and best overall fitness within the first front ($\mathbf{F_0}$) of the final population in hTetro-GA.

**TABLE 4.** hTetro-GA efficiency and final fitness performance table.

| Environment | Population size | Reach population | Terminate population | Reach time(s) | Terminate time(s) | Goal reachability $f_{gr}$ | Time consumption $f_t$ | Path smoothness $f_{sm}$ | Path safety $f_{sf}$ |
|---|---|---|---|---|---|---|---|---|---|
| Random | $n_{pop} = 25$ | 2.88± 0.49 | 36.84± 8.16 | **1.78**± 0.19 | **10.79**± 4.47 | 1.000± 0.000 | 0.556± 0.992 | 0.368± 0.106 | 0.283± 0.043 |
| | $n_{pop} = 50$ | 2.36± 0.39 | 22.48± 11.04 | 3.98± 0.65 | 23.85± 6.92 | 1.000± 0.000 | 0.580± 0.113 | 0.389± 0.084 | 0.291± 0.059 |
| | $n_{pop} = 100$ | **1.84**± 0.27 | **21.84**± 5.79 | 6.47± 0.87 | 45.18± 16.56 | 1.000± 0.000 | **0.633**± 0.014 | **0.451**± 0.079 | **0.305**± 0.034 |
| Spiral | $n_{pop} = 25$ | 104.52± 20.05 | 119.32± 19.09 | **84.68**± 33.53 | **94.56**± 32.78 | 1.000± 0.000 | 0.546± 0.073 | **0.601**± 0.111 | 0.208± 0.020 |
| | $n_{pop} = 50$ | 54.68± 14.82 | 67.68± 14.77 | 91.67± 22.29 | 120.32± 18.72 | 1.000± 0.000 | **0.593**± 0.083 | 0.574± 0.116 | 0.194± 0.015 |
| | $n_{pop} = 100$ | **43.24**± 8.10 | **57.68**± 12.02 | 141.15± 13.10 | 154.23± 16.18 | 1.000± 0.000 | 0.588± 0.092 | 0.587± 0.111 | **0.223**± 0.023 |
| H-shape | $n_{pop} = 25$ | 12.65± 4.67 | 34.80± 7.35 | **6.13**± 1.75 | **14.36**± 3.28 | 1.000± 0.000 | 0.496± 0.046 | 0.463± 0.093 | 0.326± 0.089 |
| | $n_{pop} = 50$ | 6.05± 1.17 | 33.45± 10.74 | 6.62± 1.18 | 23.20± 6.25 | 1.000± 0.000 | 0.493± 0.075 | 0.481± 0.106 | 0.328± 0.068 |
| | $n_{pop} = 100$ | **5.61**± 0.84 | **26.05**± 5.91 | 9.73± 1.89 | 34.92± 7.65 | 1.000± 0.000 | **0.513**± 0.061 | **0.498**± 0.067 | **0.331**± 0.092 |
| 3-Slit | $n_{pop} = 25$ | 374.02± 35.17 | 393.80± 34.10 | **55.57**± 11.14 | **58.49**± 1.07 | 1.000± 0.000 | 0.444± 0.027 | 0.450± 0.056 | 0.234± 0.020 |
| | $n_{pop} = 50$ | 210.60± 27.56 | 252.42± 35.43 | 56.44± 6.32 | 66.14± 8.52 | 1.000± 0.000 | 0.442± 0.031 | 0.462± 0.070 | 0.242± 0.012 |
| | $n_{pop} = 100$ | **181.20**± 20.73 | **201.6**± 18.73 | 90.67± 23.77 | 100.30± 24.74 | 1.000± 0.000 | **0.480**± 0.024 | **0.472**± 0.057 | **0.247**± 0.010 |
| Perpendicular Dynamic | $n_{pop} = 25$ | 1.42± 0.49 | 24.6± 4.85 | **23.08**± 3.59 | **28.75**± 3.50 | 1.000± 0.000 | 0.704± 0.034 | 0.441± 0.096 | 0.387± 0.100 |
| | $n_{pop} = 50$ | 1.12± 0.324 | 19.20± 3.09 | 26.93± 4.64 | 33.05± 5.45 | 1.000± 0.000 | **0.726**± 0.110 | 0.459± 0.150 | 0.604± 0.266 |
| | $n_{pop} = 100$ | **1.04**± 0.20 | **19.04**± 2.95 | 51.50± 12.52 | 64.32± 11.72 | 1.000± 0.000 | 0.710± 0.131 | **0.460**± 0.153 | **0.610**± 0.246 |
| Parallel Dynamic | $n_{pop} = 25$ | 10.24± 4.08 | 26.6± 4.72 | **6.59**± 2.14 | **14.58**± 2.41 | 1.000± 0.000 | 0.495± 0.090 | 0.464± 0.101 | 0.167± 0.016 |
| | $n_{pop} = 50$ | 6.66± 3.74 | **23.56**± 3.48 | 6.92± 1.36 | 19.88± 2.99 | 1.000± 0.000 | 0.531± 0.037 | **0.494**± 0.068 | 0.167± 0.018 |
| | $n_{pop} = 100$ | **4.82**± 0.96 | 24.00± 4.96 | 9.92± 1.49 | 34.82± 7.40 | 1.000± 0.000 | **0.537**± 0.078 | 0.491± 0.081 | **0.171**± 0.016 |

Trial number = 50. Data are recorded in a mean ± standard deviation format. Bold values represent the population size that outperforms the others in the category.

shape-shift are often unnecessary during the beginning of the navigation with no obstacle encountered.

The predefined gene maximum length ($l_{p,\max}$) depends on the size and the complexity of the environment. In the simulation, $l_{p,\max} = 200$ is chosen for spiral obstacle environment, and $l_{p,\max} = 100$ is selected for other environments. The maximum population number $n_{\text{pop,max}}$ is set to 1000 in this simulation to ensure that the algorithm has enough time to determine the best motion command sequence.

In Equation 6 where path safety fitness is calculated, a custom searching pattern parameter *sp* is defined. This paper implemented a circle shaped searching pattern with $2 \cdot d_{\text{grid}}$ as mentioned previously. The searching pattern coefficient is set to $6 + 2\sqrt{2}$ in this setting.

## VI. RESULTS AND DISCUSSION

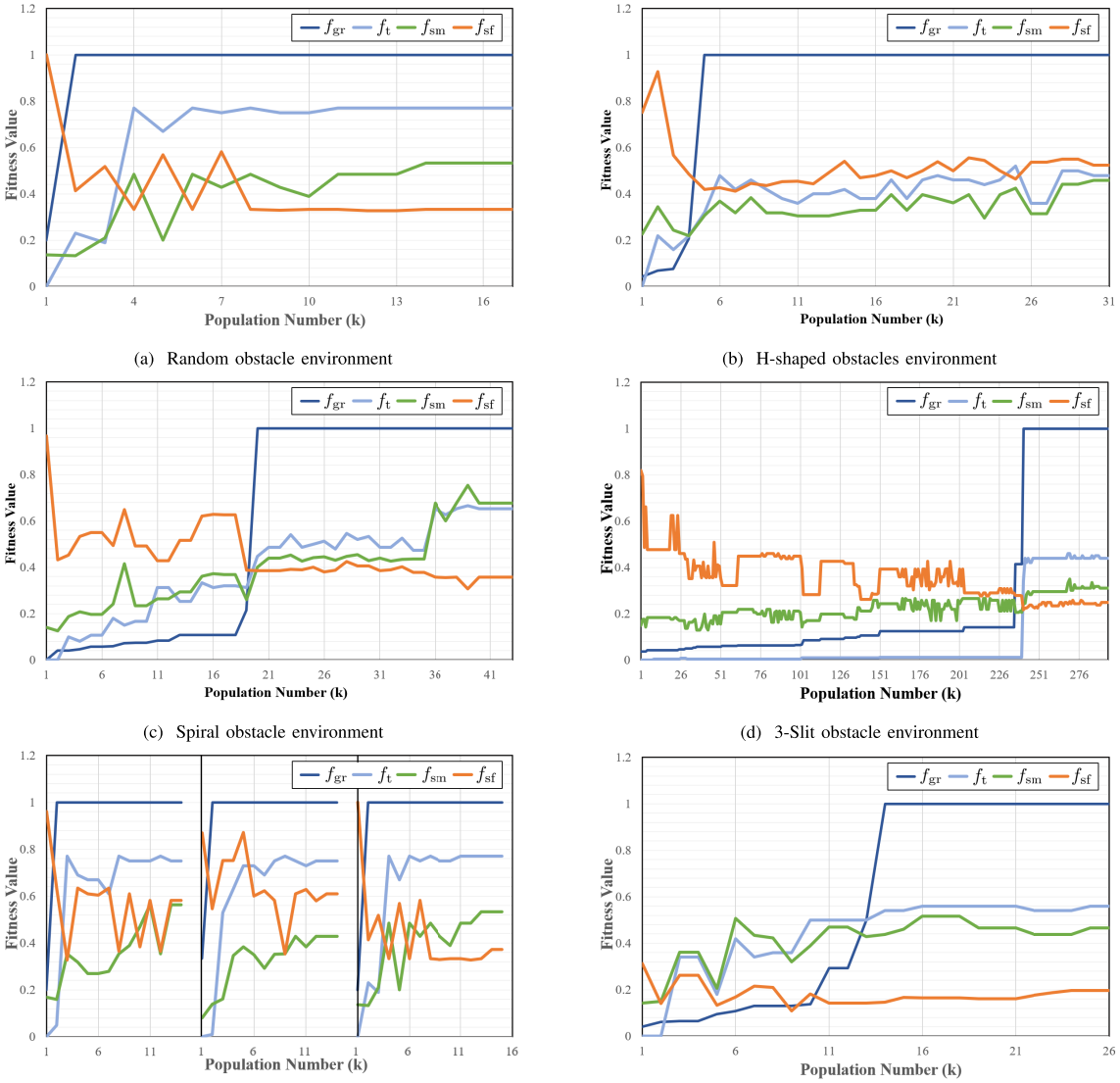
The performance of the proposed hTetro-GA is recorded in Table 4. In this table, the average number of populations required for the algorithm to find a motion command sequence that first reaches the next configuration and to reach convergence is shown. The average time consumption of the two instances is recorded as well. The output solution is selected by choosing the individual with the smallest crowding distance in the globally optimal Pareto-optimal set ($\mathbf{F}_0$). The fitness values of the final output multi-objective based motion command sequence are shown in the table, which includes the goal reachability fitness ($f_{\text{gr}}$), time consumption fitness ($f_{\text{t}}$), path smoothness fitness ($f_{\text{sm}}$), and path safety fitness ($f_{\text{sf}}$). To better visualize the output of hTetro-GA, the solutions are illustrated in all environments in Fig.9.

### A. NAVIGATION STRATEGY OF hTetro-GA

The navigation results of proposed hTetro-GA illustrated in Fig.9 demonstrate several essential features of our algorithm. It is shown in the H-shaped and spiral obstacle environments (Fig.9b, 9c) that the algorithm is capable of determining feasible paths in maps with complicated obstacle setup instead of merely performing a greedy search of the shortest distance to the next configuration.

With the starting and goal configuration both having the identical morphology – O-shaped morphology, it is shown in all virtual environments that the algorithm prioritizes generating motion command sequences without any shape-shift or rotations. This is because shape-shift or rotations result in lower time consumption fitness value, and the robot individual is more likely to be rejected during the non-dominated sort in hTetro-GA. However, due to the design of 3-slit and parallel obstacle environments (Fig. 9d, 9f), hTetro individuals with only translational motion commands fail to navigate successfully to the destination with only O-shaped morphology. It is demonstrated in both scenarios that the algorithm determines the positions where valid shape-shift and rotation actions can be conducted and use transformed configuration to navigate through narrow obstacles. It is also observed that in the multi-objective based solution in both environments, the transformations occur when the robot drives near the obstacle and attempts to transform back to the O-shaped morphology before it arrives at the goal configuration, implying that the preferred morphology of the current hTetro-GA setup is O-shaped morphology.

Even though a total of 7 morphologies are available for the hTetro robot navigation, the two most used morphologies are O-shaped and I-shaped morphologies. According to the experiments conducted, the best navigation strategy hTetro-GA suggests is to utilize O-shaped morphology to travel in open spaces, which maximizes the path safety fitness value, while utilizing I-shaped morphology to travel through obstacles when the algorithm fails to find a path to the goal configuration with only translation motion commands. The other 5 morphologies are not as competitive against the O-shaped and I-shaped morphologies in terms of the maximization of path safety and the capability to reach goal configurations. Even though these morphologies may show up during the early phase of the navigation, as the MOGA gradually optimizes the solution set, these individuals fall




(a) Random obstacle environment

(b) H-shaped obstacles environment

(c) Spiral obstacle environment

(d) 3-Slit obstacle environment

(e) Perpendicular dynamic obstacle environment

(f) Parallel dynamic obstacle environment

**FIGURE 10.** Fitness values of the hTetro-GA multi-objective based best performing individual in each generation of population. Fig. 10e illustrates the navigation outcome from three configuration loops in hTetro-GA.

into non-leading fronts during non-dominated sorting and are eventually rejected. Note that the definition of fitness values influences the choice of configurations in the solutions, so the algorithm may yield solutions with more than 2 configurations combined if fitness values are defined differently. Since the proposed MOOP solution selects a Pareto-optimal solution, it is not guaranteed that it will find the shortest path as it may evaluate other criteria more (like path safety). Paths only indicate the trajectory of Block 2 in Fig. 9b, Fig. 9c and the hTetro robot moves in O-configuration, the paths seem to be longer to prevent the other blocks from getting too close to the left side of the H, and corner obstacle (which reduces the path safety fitness).

### B. MULTI-OBJECTIVE PATH PLANNING PERFORMANCE

Through the implementation of the NSGA-II technique, the proposed hTetro-GA demonstrates a strong capability to determine the globally Pareto-optimal set in the search space. The implementation of elitism ensured that the genetic information from robot individuals that successfully navigate to the goal configurations would be carried to the future generation of populations.

In Fig.9, the multi-objective based output solutions are marked as a black-colored route in all environments. Three additional solutions within $\mathbf{F}_0$ are shown, which represents the motion command sequences with best $f_t$, $f_{sm}$, and $f_{sf}$. Note that the best sequence with $f_{gr}$ is not listed as all individuals in $\mathbf{F}_0$ successfully reached the target configurations in $\mathbf{Q}$ and possess goal reachability fitness values of 1. It is observed that the four routes all take similar approaches to navigate to the goal configurations. As shown in Fig.9b, solutions with the highest $f_{sf}$ might perform additional motions in order to increase the safety; while solutions that possess the highest $f_{sm}$ may compromise on the safety fitness

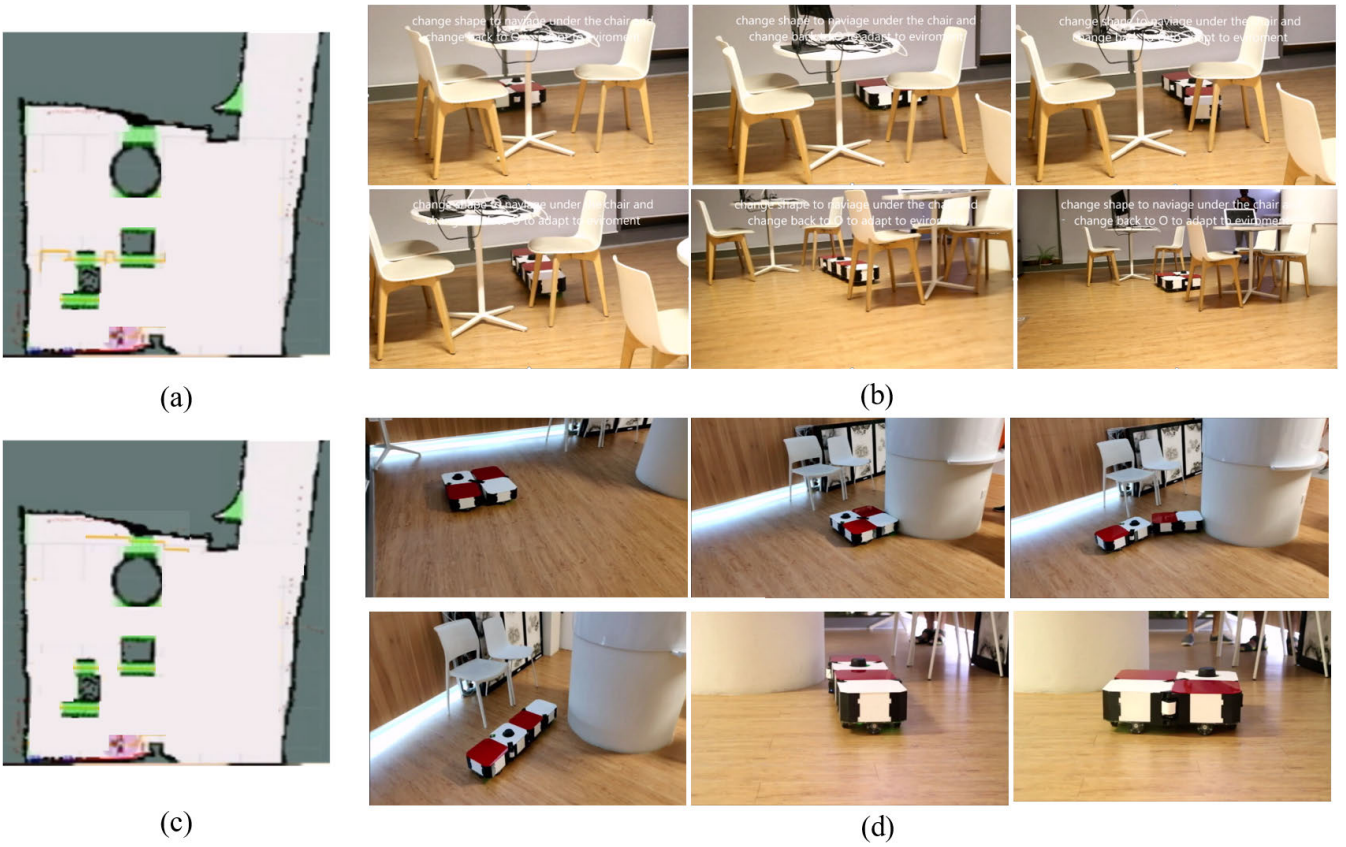



**FIGURE 11.** Implementation of path planning on the real robot platform. (a), (c) built map and generated path plans. (b),(d) the sequences of robot navigation to go from source to destination with shape shifting between O to I.

$f_{sm}$ values (Fig.9e, Fig.9f). The multi-objective based solutions, which consist of individuals with the maximum sum of the four fitness values within $\mathbf{F}^0$ demonstrate consistent performance throughout the entire hTetro-GA navigation compared to solutions which focus only on a single fitness value.

In Fig.10, the four fitness values in each generation of populations in the multi-objective based solutions are plotted. The implementation of the NSGA-II technique allows the proposed algorithm to find a balanced solution between the four fitness values. As shown in the figures, the goal reachability fitness ($f_{gr}$) value reaches 1 once the algorithm finds a valid motion command sequence to navigate to the goal configuration. After the goal configuration is found, the algorithm will aim to improve path safety, path smoothness, and time consumption fitness values until the solution converges. It is worth noting that the path safety fitness ($f_{sf}$) values are considerably high at the beginning of the navigation. This occurs when the first few generations of populations fail to reach the goal configurations and only navigate in safe grids that are distant to the obstacles in the environment. Even though $f_{sf}$ has a trend towards declining during the navigation process, the non-dominated sorting process will attempt to maximize the path safety of the solutions once the goal configurations are reached.

### C. EFFECT OF POPULATION SIZE ON hTetro-GA PERFORMANCE

In Table 4, it is demonstrated that differences in $n_{pop}$ greatly influences the number of the population when the first $p$ reaches the goal configuration and when the process is terminated. The final output multi-objective based solutions in the three $n_{pop}$ scenarios all demonstrate consistent capability to navigate to the goal configuration without failing ($f_{gr} = 1$). As recorded in Table 4, when the population size increases, the values of $f_t, f_{sm}$ and $f_{sf}$ all increase accordingly. This has shown that initializing the algorithm with a larger population size results in a fast exploration of the space, and the algorithm is capable of finding a valid solution early, with the solution found being refined with higher fitness values. Despite the early termination of the population, the average time consumption for each operation still increases significantly when $n_{pop}$ rises due to the extra computation load in the navigation and selection process. In the presented virtual environment, a starting $n_{pop}$ of 50 has shown to be effective as it provides near-optimal outcome within a short time period; however, as the environment size increases, a larger $n_{pop}$ may be required during the initialization process to speed up the early exploration process in the algorithm.

### D. DEPLOYMENT OF PATH PLANNING ON hTetro

The generated path by proposed path planning method is implemented to the real hTetro robot platform. The real experiment aims to show that the robot platform can follow the found optimal path, change to required shapes to overcome the narrow space while avoiding obstacle safety.The room is modeled as a grid-based environment in ROS, and occupied grids such chairs, tables, and walls are considered as obstacle grids. Grid width ($d_{\text{grid}}$) and hTetro block width ($d_{\mathcal{B}}$) are both set to 25cm in this implementation. Hector SLAM [68] is used for map construction and robot localization. Fig.11 shows the hardware implementation of hTetro-GA. In Fig.11a,b, the path that connects start and goal configuration which are defined near the obstacles is generated with the proposed algorithm based on a provided map, and in Fig.11b,d, the shape-shift and translation motion are both demonstrated by the robot to successfully navigate through the obstacles. During navigation, additional robot pose check has been implemented into the workflow: the robot location and orientation are monitored and adjusted in real-time to ensure that the robot does not deviate from the ideal trajectory, which may result in collisions with obstacles in the environment. It is also observed that since robot shape-shift motions do not always sweep out the same area modeled in Fig.4 during motion validity analysis, providing a smaller clearance for robot shape-shifting could result in collisions in real-world scenarios. The validity analysis in hTetro-GA workflow (Fig.5) has been adjusted to assume a larger sweeping radius of 35cm from hTetro blocks to mitigate this issue. We are also in the state of developing the hTetro so that it works autonomously in wider testbed environments with complex obstacle settings. Once the stable platform has been constructed, GA algorithms with different parameter settings will be evaluated.

## VII. CONCLUSION

This paper presents a novel algorithm, hTetro-GA, which is a global path planning algorithm for reconfigurable robots. The proposed algorithm focuses on multi-objective optimization and attempts to find the solution with Pareto-optimal goal reachability, time consumption, path safety, and path smoothness through genetic algorithms.

In this paper, the model of the hTetro robot and the grid-based workspace are first constructed. The paper then introduces the robot configuration and motion validity, which are both crucial discussion topics in systems that involve reconfigurable robots. The workflow of hTetro-GA is next organized into three loops where genetic algorithms are executed to navigate between different configurations. Four fitness objective functions are then introduced to evaluate the performance of the robot individuals from different perspectives. In order to solve the MOOP in this study, the NSGA-II technique is implemented to determine the Pareto-optimal robot individual with the best performing motion command sequence in the generation. Novel genetic algorithm operators are introduced in this paper due to the self-reconfigurable nature of hTetro to generate a wide variety of individuals in the genetic pool, which helps the algorithm to find Pareto-optimal paths during navigation.

The proposed hTetro-GA has shown the strong capability of determining feasible motion command sequences to the goal configurations in various environments with different settings. It manages to handle dynamic obstacles in given environments, which makes the real-world implementation of the algorithm in known environments feasible. The feature of identifying navigation paths based on a roadmap with multiple designated configurations is useful in multiple scenarios. During an exploration or a rescue task, especially in hazardous environments, the proposed algorithm can be used to generate paths for reconfigurable robots to safely navigate to multiple destinations within the area to perform specific actions.

Future research areas are as follow: **(1) Fine-tuning of hTetro-GA parameters:** Multiple parameters are involved in the presented hTetro-GA, and the algorithm performance of different $n_{\text{pop}}$ values is analyzed in this paper. We would like to expand the work and analyze the performance when parameters like searching pattern (**sp**), mutation rate ($r_{\text{mu}}$), the virtual environment size are modified. **(2) Increased complexity setup:** The proposed algorithm would be revised by allowing hTetro block disassembly during each navigation to improve the algorithm efficiency by increasing the degree of freedom. Architectures with 5 or more hTetro blocks should be explored for further performance comparison. **(3) Optimization of hTetro-GA workflow:** The proposed algorithm is a global PP algorithm, and we would like it to work better in local PP problems by making modifications like storing unknown obstacles encountered in $\mathbf{W}_{\text{obs}}$ and performing re-calculation of hTetro-GA when the obstacle map is updated to reroute the path. **(4) Implementation of other multi-objective optimization techniques:** Due to the fact that the traditional PP algorithms such as A*, D*, and artificial potential field could not be directly implemented on reconfigurable robots, the comparative analysis between multiple algorithms could not be presented in this work. Implementing other multi-objective algorithms like MOPSO [69] and $\epsilon$-constraint method [70] for hTetro in the future will allow us to compare their performance with the proposed NSGA-II based technique.

## REFERENCES

[1] J. Tisdale, Z. Kim, and J. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robot. Autom. Mag.*, vol. 16, no. 2, pp. 35–42, Jun. 2009.
[2] Y. Liu and R. Bucknall, "Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment," *Ocean Eng.*, vol. 97, pp. 126–144, Mar. 2015.
[3] P. Raja, "Optimal path planning of mobile robots: A review," *Int. J. Phys. Sci.*, vol. 7, no. 9, pp. 1314–1320, Feb. 2012.
[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2011.
[5] H. Miao and Y.-C. Tian, "Dynamic robot path planning using an enhanced simulated annealing approach," *Appl. Math. Comput.*, vol. 222, pp. 420–437, Oct. 2013.

[6] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1, pp. 113–126, Oct. 2001, doi: 10.1023/A:1016639210559.

[7] V. Kee, N. Rojas, M. R. Elara, and R. Sosa, "Hinged-tetro: A self-reconfigurable module for nested reconfiguration," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Jul. 2014, pp. 1539–1546.

[8] N. Tan, N. Rojas, R. E. Mohan, V. Kee, and R. Sosa, "Nested reconfigurable robots: Theory, design, and realization," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 7, p. 110, Jul. 2015.

[9] Y. Sun and S. Ma, "ePaddle mechanism: Towards the development of a versatile amphibious locomotion mechanism," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 5035–5040.

[10] A. V. Le, A. A. Hayat, M. R. Elara, N. H. K. Nhan, and K. Prathap, "Reconfigurable pavement sweeping robot and pedestrian cohabitant framework by vision techniques," *IEEE Access*, vol. 7, pp. 159402–159414, 2019.

[11] L. Yi, A. V. Le, A. A. Hayat, C. S. C. S. Borusu, R. E. Mohan, N. H. K. Nhan, and P. Kandasamy, "Reconfiguration during locomotion by pavement sweeping robot with feedback control from vision system," *IEEE Access*, vol. 8, pp. 113355–113370, 2020.

[12] M. A. V. J. Muthugala, A. V. Le, E. S. Cruz, M. R. Elara, P. Veerajagadheswar, and M. Kumar, "A self-organizing fuzzy logic classifier for benchmarking robot-aided blasting of ship hulls," *Sensors*, vol. 20, no. 11, p. 3215, Jun. 2020.

[13] A. V. Le, N. H. K. Nhan, and R. E. Mohan, "Evolutionary algorithm-based complete coverage path planning for tetriamond tiling robots," *Sensors*, vol. 20, no. 2, p. 445, Jan. 2020.

[14] W.-M. Shen, B. Salemi, and P. Will, "Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 700–712, Oct. 2002.

[15] S. M. B. P. Samarakoon, M. A. V. J. Muthugala, A. V. Le, and M. R. Elara, "hTetro-Infi: A reconfigurable floor cleaning robot with infinite morphologies," *IEEE Access*, vol. 8, pp. 69816–69828, 2020.

[16] A. V. Le, R. Parween, R. E. Mohan, N. H. K. Nhan, and R. E. Abdulkader, "Optimization complete area coverage by reconfigurable hTrihex tiling robot," *Sensors*, vol. 20, no. 11, p. 3170, Jun. 2020.

[17] Z. Butler, R. Fitch, and D. Rus, "Distributed control for unit-compressible robots: Goal-recognition, locomotion, and splitting," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 418–430, Dec. 2002.

[18] S. Tian, Z. Yang, Z. Fu, and H. Zheng, "A self-reconfigurable robot M-lattice," in *Intelligent Autonomous Systems* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, 2017, pp. 563–571.

[19] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 431–441, Dec. 2002.

[20] M. A. P. García, O. Montiel, O. Castillo, and R. Sepúlveda, "Optimal path planning for autonomous mobile robot navigation using ant colony optimization and a fuzzy cost function evaluation," in *Analysis and Design of Intelligent Systems Using Soft Computing Techniques* (Advances in Soft Computing). Berlin, Germany: Springer, 2007, pp. 790–798.

[21] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2004, pp. 2473–2478.

[22] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Autom. Construct.*, vol. 112, Apr. 2020, Art. no. 103078.

[23] P. Veerajagadheswar, K. Ping-Cheng, M. R. Elara, A. V. Le, and M. Iwase, "Motion planner for a tetris-inspired reconfigurable floor cleaning robot," *Int. J. Adv. Robot. Syst.*, vol. 17, no. 2, pp. 1–27, 2020.

[24] Y. Shi, M. R. Elara, A. V. Le, V. Prabakaran, and K. L. Wood, "Path tracking control of self-reconfigurable robot hTetro with four differential drive units," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 3998–4005, Jul. 2020.

[25] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots," *IEEE Access*, vol. 7, pp. 94642–94657, 2019.

[26] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 5, Apr./May 2004, pp. 4350–4355.

[27] O. Kramer, *Genetic Algorithm Essentials*. Cham, Switzerland: Springer, 2017.

[28] F. C. J. Allaire, M. Tarbouchi, G. Labonté, and G. Fusina, "FPGA implementation of genetic algorithm for UAV real-time path planning," *J. Intell. Robot. Syst.*, vol. 54, nos. 1–3, pp. 495–510, Mar. 2009.

[29] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm," *Comput. Electr. Eng.*, vol. 38, no. 6, pp. 1564–1572, Nov. 2012.

[30] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Y. Leong, "Crossover and mutation operators of genetic algorithms," *Int. J. Mach. Learn. Comput.*, vol. 7, no. 1, pp. 9–12, Feb. 2017.

[31] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *J. Comput. Sci.*, vol. 25, pp. 339–350, Mar. 2018.

[32] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, and B. Bouzouia, "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control," *Robot. Auton. Syst.*, vol. 89, pp. 95–109, Mar. 2017.

[33] V. Roberge, M. Tarbouchi, and G. Labonte, "Fast genetic algorithm path planner for fixed-wing military UAV using GPU," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 5, pp. 2105–2117, Oct. 2018.

[34] E. Guan, Z. Fu, W. Yan, D. Jiang, and Y. Zhao, "Self-reconfiguration path planning design for M-lattice robot based on genetic algorithm," in *Intelligent Robotics and Applications* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2011, pp. 505–514.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[36] L. Pfotzer, M. Staehler, A. Hermann, A. Roennau, and R. Dillmann, "KAIRO 3: Moving over stairs & unknown obstacles with reconfigurable snake-like robots," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2015, pp. 1–6.

[37] B. Ramalingam, A. Lakshmanan, M. Ilyas, A. Le, and M. Elara, "Cascaded machine-learning technique for debris classification in floor-cleaning robot application," *Appl. Sci.*, vol. 8, no. 12, p. 2649, Dec. 2018.

[38] Z. Butler, K. Kotay, D. Rus, and K. Tomita, "Generic decentralized control for a class of self-reconfigurable robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 809–816.

[39] V. Prabakaran, M. R. Elara, T. Pathmakumar, and S. Nansai, "hTetro: A tetris inspired shape shifting floor cleaning robot," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 6105–6112.

[40] A. Le, V. Prabakaran, V. Sivanantham, and R. Mohan, "Modified A-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors*, vol. 18, no. 8, p. 2585, Aug. 2018.

[41] A. Manimuthu, A. V. Le, R. E. Mohan, P. Veerajagadeshwar, N. H. K. Nhan, and K. P. Cheng, "Energy consumption estimation model for complete coverage of a tetromino inspired reconfigurable surface tiling robot," *Energies*, vol. 12, no. 12, p. 2257, Jun. 2019.

[42] A. Le, P.-C. Ku, T. T. Tun, N. H. K. Nhan, Y. Shi, and R. Mohan, "Realization energy optimization of complete path planning in differential drive based self-reconfigurable floor cleaning robot," *Energies*, vol. 12, no. 6, p. 1136, Mar. 2019.

[43] A. Le, M. Arunmozhi, P. Veerajagadheswar, P.-C. Ku, T. H. Minh, V. Sivanantham, and R. Mohan, "Complete path planning for a tetris-inspired self-reconfigurable robot by the genetic algorithm of the traveling salesman problem," *Electronics*, vol. 7, no. 12, p. 344, Nov. 2018.

[44] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3D path planning algorithms," *J. Control Sci. Eng.*, vol. 2016, pp. 1–22, Mar. 2016.

[45] L. Xu, D. Wang, B. Song, and M. Cao, "Global smooth path planning for mobile robots based on continuous bezier curve," in *Proc. Chin. Automat. Congr. (CAC)*, Oct. 2017, pp. 2081–2085.

[46] B. K. Oleiwi, R. Al-Jarrah, H. Roth, and B. I. Kazem, "Multi objective optimization of trajectory planning of non-holonomic mobile robot in dynamic environment using enhanced ga by fuzzy motion control and A*," in *Neural Networks and Artificial Intelligence* (Communications in Computer and Information Science). 2014, pp. 34–49.

[47] S. Thabit and A. Mohades, "Multi-robot path planning based on multi-objective particle swarm optimization," *IEEE Access*, vol. 7, pp. 2138–2147, 2019.

[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[49] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009, vol. 3, no. 3, p. 5.

[50] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013.
[51] P. Yap, "Grid-based path-finding," in *Advances in Artificial Intelligence* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2002, pp. 44–55.
[52] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Robotics Research* (Springer Tracts in Advanced Robotics). Berlin, Germany: Springer, 2007, pp. 239–253.
[53] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, 1993, pp. 533–538.
[54] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A complete coverage algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 2005, pp. 2040–2044.
[55] Y.-H. Choi, T.-K. Lee, S.-H. Baek, and S.-Y. Oh, "Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 5788–5793.
[56] Y. Gabriely and E. Rimon, "Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 954–960.
[57] P. S. Heckbert, *Graphic Gems IV*. Washington, DC, USA: American Psychological Association, 1994.
[58] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 2010.
[59] A. M. Aibinu, H. B. Salau, N. A. Rahman, M. N. Nwohu, and C. M. Akachukwu, "A novel clustering based genetic algorithm for route optimization," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 4, pp. 2022–2034, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2215098616300738
[60] G. A. Jayalakshmi, "Genetic algorithm based automation methods for route optimization problems," in *Automation*, F. Kongoli, Ed. London, U.K.: IntechOpen, Jul. 2012.
[61] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robot. Auton. Syst.*, vol. 86, pp. 13–28, Dec. 2016.
[62] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Found. Genet. Algorithms*, vol. 1, pp. 69–93, 1991.
[63] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
[64] Y. Yusoff, M. S. Ngadiman, and A. M. Zain, "Overview of NSGA-II for optimizing machining process parameters," *Procedia Eng.*, vol. 15, pp. 3978–3983, Jan. 2011.
[65] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, Mar. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650211000058
[66] W. Stadler, "A survey of multicriteria optimization or the vector maximum problem, part I: 1776–1960," *J. Optim. Theory Appl.*, vol. 29, no. 1, pp. 1–52, Sep. 1979.
[67] W. Zhang, W. Xu, and M. Gen, "Multi-objective evolutionary algorithm with strong convergence of multi-area for assembly line balancing problem with worker capability," *Procedia Comput. Sci.*, vol. 20, pp. 83–89, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050913010430
[68] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 2013, pp. 624–631.
[69] C. A. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, 2002, pp. 1051–1056.
[70] G. Mavrotas, "Effective implementation of the $\varepsilon$-constraint method in multi-objective mathematical programming problems," *Appl. Math. Comput.*, vol. 213, no. 2, pp. 455–465, 2009.

**KU PING CHENG** received the B.Sc. degree in computer engineering from the Singapore University of Technology and Design, in 2017. He is currently working on autonomous robotics as a Research Officer with the Engineering Product Development Pillar, Singapore University of Technology and Design. His research interests include robotics and automation, intelligent robots, control systems, and computer vision.

**RAJESH ELARA MOHAN** received the B.E. degree from Bharathiar University, India, in 2003, and the M.Sc. and Ph.D. degrees from Nanyang Technological University, in 2005 and 2012, respectively. He is currently an Assistant Professor with the Engineering Product Development Pillar, Singapore University of Technology and Design. He is also a Visiting Faculty Member with the International Design Institute, Zhejiang University, China. He has published over 80 articles in leading journals, books, and conferences. His research interests include robotics with an emphasis on self-reconfigurable platforms as well as research problems related to robot ergonomics and autonomous systems. He was a recipient of the Tan Kah Kee Young Inventors' Award, in 2010, the ASEE Best of Design in Engineering Award, in 2012, and the SG Mark Design Award, in 2016 and 2017.

**NGUYEN HUU KHANH NHAN** defended his Ph.D. thesis at the Institute of Research and Experiments for Electrical and Electronic Equipment, Moscow, Russia. He is currently working as a Lecturer with the Faculty of Electrical and Electronic Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam. His research interests include VLSI, MEMS and LED driver chips, robotics vision, robot navigation, 3D video processing.

**ANH VU LE** received the B.S. degree in electronics and telecommunications from the Ha Noi University of Technology, Vietnam, in 2007, and the Ph.D. degree in electronics and electrical from Dongguk University, South Korea, in 2015. He is currently with the Optoelectronics Research Group, Faculty of Electrical and Electronics Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam. He is also working as a Postdoctoral Research Fellow with the ROAR Laboratory, Singapore University of Technology and Design. His current research interests include robotics vision, robot navigation, human detection, action recognition, feature matching, and 3D video processing.