Received 12 January 2023, accepted 22 February 2023, date of publication 27 February 2023, date of current version 6 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3249757

RESEARCH ARTICLE

Multi-Agent Action Graph Based Task Allocation and Path Planning Considering Changes in Environment

TAKUMA OKUBO[1], **AND MASAKI TAKAHASHI**[2], **(Member, IEEE)**
[1]Graduate School of Science and Technology, Keio University, Kohoku, Yokohama 223-8522, Japan
[2]Department of System Design Engineering, Faculty of Science and Technology, Keio University, Kohoku, Yokohama 223-8522, Japan

Corresponding author: Takuma Okubo (takuma.okubo.56@keio.jp)

This work was supported by the Core Research for Evolutional Science and Technology (CREST) of the Japan Science and Technology Agency (JST) under Grant JPMJCR19A1.

ABSTRACT Task allocation and path planning considering changes in the mobility of robots in the environment allows the robots to efficiently execute tasks with smaller travel times. A lunar base construction is one of the situations in which robots can more efficiently accomplish its goal by taking such environment changes into account when performing tasks. For the construction, we assumed that when a robot executes a task of building a road, the environment changes such that aisles that were unusable before the task become usable post execution. If such changes in environment are considered in advance, the robot can efficiently plan to wait until the environment changes and can move before executing the task. However, previous studies have not considered such changes, resulting in inefficient planning. To solve this problem, we developed a multi-agent action graph that consists of multiple layers and expresses the environment changes associated with task execution in terms of changes in these layers. In this graph, task allocation and path planning are formulated as a combinatorial optimization problem and are optimized using mixed-integer programming. Multi-agent action graphs and the proposed formulation enable efficient planning considering changes in the robots' mobility in advance. Through simulations, we confirmed that the proposed method completed the construction of the lunar base approximately 16.4% earlier than the conventional method, while consuming approximately 16.0% less total energy of the robots.

INDEX TERMS Task allocation, path planning, environment changes, multi-robot systems, robotic lunar surface operations.

I. INTRODUCTION

Task allocation [1] and path planning [2] are very important for multi-robots to perform tasks in a coordinated manner. Although there have been many studies on these planning problems in the past, it is difficult for these studies to plan for changes in the mobility of robots in the environment. By taking into account changes in the robot's mobility in advance, an approach that allows the robots to efficiently execute tasks with smaller travel times can be realized. There are situations in which robots can more efficiently accomplish its goal by taking such environment changes into account when performing tasks in the real world. In this study, we assume a situation in which robots' mobility in the environment may change: the construction of a lunar base by robots, which has been the focus of much attention in recent years.

Projects such as robotic lunar surface operations (RLSO) have attracted increasing attention toward the goal of building a human-capable lunar base and generating sufficient oxygen and hydrogen from the polar ice to operate it [3], [4]. The construction of a lunar base is necessary for long-term activities on the Moon; evidently, however, it is difficult for humans alone to construct a lunar base. Therefore, such plans typically envision the use of robots for construction. The construction of a lunar base using robots is divided into

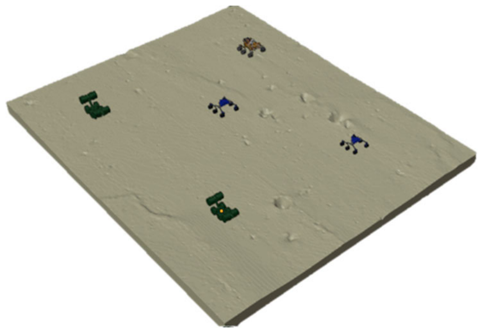
The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Alshabi.

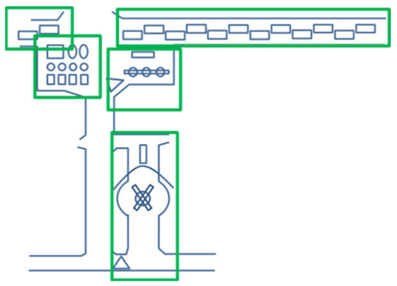
This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

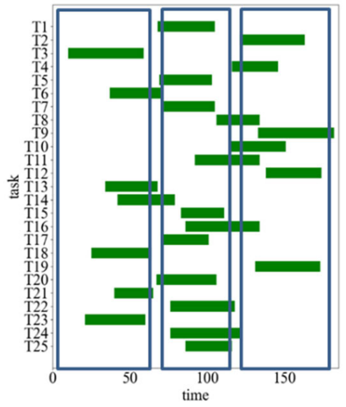
four major phases, as shown in Fig. 1. First, lunar surface conditions are surveyed robotically, as shown in Fig. 1(a). As in previous studies [5], [6], a geological survey of the lunar surface enables selection of the best site for the construction of the lunar base. Second, as shown in Fig. 1(b), the facility layout of the lunar base is determined based on the results of the lunar surface condition survey. To do so, it is necessary to realize a facility layout suitable for the mission, such as a plan to search for polar ice on the Moon's surface. Herein, optimization can generally be performed as a facility placement problem, which is a combinatorial optimization problem [7], [8]. Third, the schedule for base construction is determined, as shown in Fig. 1 (c). At this stage, it is necessary to consider the order in which tasks are to be executed to realize the base construction as well as the due dates of the tasks. Here, the specific steps, such as which cargo to transport and when to transport it, or when to construct a particular part of the facility, are not determined. Instead, only a rough schedule is determined, such as which facilities are to be constructed and when. Herein, optimization can generally be performed as a scheduling problem, which is also a combinatorial optimization problem. Many studies on scheduling such practical construction projects have been conducted so far [9], [10]. Finally, as shown in Fig. 1 (d), the robot constructs the lunar base based on the determined facility layout and schedule. In this phase, specific procedures are determined, such as which cargo is to be transported and when, and when specific parts of the facility are to be constructed. This study focused on this specific phase. An attempt to plan the construction of an entire facility layout may cause a combinatorial explosion, requiring a significant amount of computation power. In addition, major replanning may be necessary in the event of changes to the construction schedule. Therefore, in this stage, the facility layout determined in the second stage and the schedule determined in the third stage are divided into a certain number of sections and realized as shown in Figs. 1 (b) and (c).

In the construction of a lunar base using robots, the subject of this research, robots perform tasks such as carrying cargo, constructing facilities, and maintaining roads. Therefore, it is necessary to optimize task allocation, that is, which task is allocated to which robot. Path planning for each robot is also necessary to ensure route efficiency.

When robots are used to construct a lunar base, changes in the environment may occur as a result of the task execution, as shown in Fig. 2. In this study, we assumed that when the robot executes a task to build a road, the environment changes such that an aisle that was unusable before the task execution becomes usable. If the task allocation and path planning are optimized without considering such an environment change, the robot will execute the task in a roundabout manner, as shown in Fig. 2 (b). On the contrary, if such environment changes are taken into account in advance, the robot can wait until the environment changes and is ready to move, as shown in Fig. 2 (a), before making an efficient plan to execute the


(a) Geological survey of the lunar's surface


(b) Determination of facility layout for the lunar base



(c) Scheduling for lunar base construction

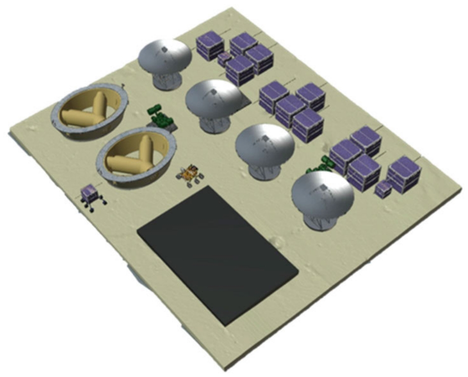

(d) Construction of the lunar base by robots

FIGURE 1. Lunar base construction process.

task. However, in conventional research, it is difficult to plan for such environment changes in advance. In fact, there are many current studies that optimize task allocation and path planning; however, because these methods do not consider changes in the environment, they either result in inefficient

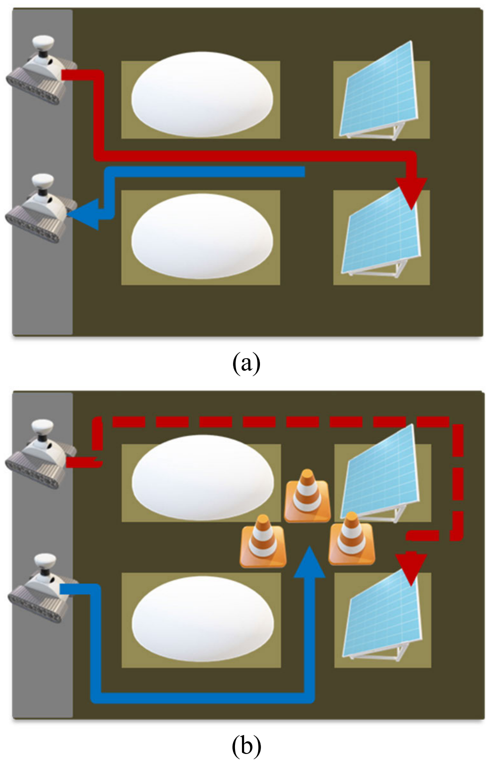


FIGURE 2. **Changes in the environment in which the robot can move as it performs the task.**

planning or can only plan to cope with changes by replanning after a change.

Therefore, the contribution of this study is that it achieves efficient task allocation and path planning by considering in advance the changes in environment associated with the execution of tasks by multiple robots. We developed a multi-agent action graph to optimize planning considering the changes in environment. A multi-agent action graph consists of multiple layers, in which environment changes associated with task execution are represented by changes in the layers. Specifically, the lower layer represents the environment state before task execution, while the upper layer represents the environment state after task execution. In this study, task allocation and path planning were formulated as a combinatorial optimization problem on the graph and optimized via mixed-integer programming. In the formulation, when one robot executes a task that causes an environment change, the other robots also transition to the post-change environment state. Additionally, to realize a task execution plan that assumes the construction of a lunar base, the types of tasks that can be executed by each robot and the execution time required to complete each task must be considered in the two plans. For cargo transport tasks, the number of packages that can be simultaneously transported by robots (capacity constraints) is considered. In addition, path planning must avoid collisions between robots. We developed an efficient approach to minimize the total travel time of the robot and, consequently, reduce the energy consumed by the robot on the Moon, while taking the above considerations into account in the task allocation and path planning processes. Therefore, we optimized the task allocation and path planning simultaneously to minimize the total travel time of the robot, instead of separately optimizing them, which is an approach adopted in many previous studies. As described above, the proposed method takes into account.

From the simulations, we confirmed that the proposed method achieves task allocation and path planning that takes into account changes in the robot's mobility in advance, which was difficult to achieve with previous methods. In fact, the proposed method significantly reduces the total travel time of robots and task completion time compared to the conventional state-of-the-art methods by taking environment changes into account in advance. On the other hand, we found that the computational performance of the proposed method needs to be improved.

The remainder of this paper is organized as follows: Section II describes previous related works; Section III describes the multi-agent action graph, the graphical representation of the proposed method; Section IV describes the formulation of the proposed method using multi-agent action graphs; Section V confirms the effectiveness of the proposed method through simulations; and Section VI concludes the paper.

II. RELATED WORKS

This section describes previous studies on task allocation and path planning for multiple robots. These can generally be solved as a combinatorial optimization problem.

A. STUDIES THAT CONSIDER ENVIRONMENT CHANGES

Although previous studies on task allocation and path planning that consider environment change into account exist, the environment changes addressed by them are defined differently. Park et al. [11] performed task allocation for a robot that collects garbage, treating changes in the amount of garbage as changes in the environment. They responded to this environment change by modifying the tasks already allocated to the robots. To reduce the need for coordination among robots when modifying task allocation, a game-theoretic approach was used, in which each robot independently selects its own task. As described earlier, this approach is easily adaptable to changes in the environment; however, it does not plan in advance for changes in the environment associated with task execution, as is the case in this study. Yu et al. [12] performed task allocation and path planning in an environment with unknown obstacles, dealing with environment changes that increase or decrease the number of targets. They adopted a flexible path-finding approach by dynamically reallocating tasks to reach multiple dynamically changing targets. Specifically, they proposed a method that reallocates tasks to each other when finding dynamically changing paths, because of the possibility of a particular target appearing or disappearing during a mobile mission. However, this study also did not consider the environment changes that accompany task execution.

Semiz et al. [13] proposed a multi-agent path search that treats environment changes as changes in the available nodes of the environment represented as a graph. The environment change treated here is the closest to the definition this study dealt with, in that the available nodes change. They also noted that there are no previous studies that deal with this type of environment change. They take the approach of avoiding the replanning of agents that are not affected by environment changes when they occur. However, it is difficult to achieve planning that considers environment changes in advance, as in our study, because the agents affected by the change are replanned. Kawasaki et al. [14] proposed a robot navigation system that considers the environment changes associated with task execution. Similar to our work, this work can plan in advance considering environment changes; however, because of the structure of the graph, it can only be applied to single-robot planning. To apply it to planning for multi-robots, it is necessary not only to propose a new graph but also to formulate it as a combinatorial optimization problem and plan efficiently.

As described above, there have been several studies on task allocation and path planning that consider environment changes in the past, but they have difficulty planning in advance to consider the changes. These approaches can only respond to environment changes through re-planning, so it is not possible to plan efficiently, where one robot waits for environment changes caused by another robot's task execution before executing the task.

B. STUDIES THAT DO NOT CONSIDER ENVIRONMENT CHANGES

Tai et al. [15] classified robot states into normal, delayed, and state in which the robot can recover from a delay, whereas Huang et al. [16] considered features such as the different types of tasks that can be performed for different types of robots. Xu et al. [17] aimed to keep the battery consumption of robots as small as possible, while Liu et al. [18] aimed to keep the battery consumption of robots as small as possible and to prevent overlapping paths between robots. Liu et al. [19] proposed a task allocation scheme that simultaneously considers time and capacity constraints with the aim to accommodate new tasks as they arise. Zhang et al. [20] proposed an approach that focuses not only on collisions between robots on a common cell but also on path conflicts between robots when they exchange positions in adjacent grid cells. This approach clearly prevented overlapping work paths between robots. Zhang et al. [21] provided four collision classifications and three solutions to achieve collision-free path planning among robots. In this study, collision avoidance was determined based on a rule basis. Lyu et al. [22] proposed an approach that aims to minimize the number of robots used, taking into account the transport time and avoidance of collisions between robots. Chen et al. [23] proposed an approach that simultaneously considers capacity constraints and collision avoidance. This study also utilized the cost of path planning after task allocation and path planning to again perform task allocation and path planning. As many of these conventional methods take the approach of optimizing task allocation and path planning separately, they may not be able to derive a globally optimal solution in some situations; or even if they can, it may take a long time to do so. In this study, we formulated a mixed-integer programming method that can derive a solution, optimize task allocation and path planning for each robot simultaneously in a single plan, and derive the globally optimal solution, thereby achieving an optimal solution with a smaller total robot travel time than current methods. While the above studies employ metaheuristics, such as genetic algorithms and annealing methods, there are also conventional studies that utilize mixed-integer programming. Poltena et al. [24] limited the number of robots allowed in the aisle and prohibited all possible collision types as a constraint. However, this study does not consider collisions outside the corridor. In addition, planning becomes inefficient because a large number of robots cannot enter the corridor. Zhonga et al. [25] used a method that limits the number of robots allowed in the same corridor. However, this study did not consider collisions between robots in the corridor, nor did it perform task allocation. Unlike these approaches, which utilize mixed-integer programming, we can achieve more rigorous collision avoidance because we consider collisions between robots on the same grid and collisions caused by passing each other between grids as constraints just like our previously proposed method [26]. Therefore, the total travel time of the robot can be reduced.

As described above, there have been several studies on task allocation and path planning that do not consider environment changes in the past. These studies are effective in other situations but become inefficient in situations where the robot's mobility changes.

III. MULTI-AGENT ACTION GRAPH

This section describes the multi-agent action graph. Unlike conventional graphical representations that can only represent a single environmental state, multi-agent action graphs can simultaneously represent multiple environmental states and transitions between them in a single graph. As explained previously, the proposed graph expresses changes in the mobility of robots in the environment and enables simultaneous optimization of task allocation and path planning considering the changes caused by multi-robot task execution.

A. GRID MAP

To graphically represent the environment, the environment in which the robot moves is first represented by a grid map. For example, the environment in Fig. 2 is represented by a grid map as shown in Fig. 3. In Fig. 3, the red grid represents the depot from which the robot departs. The blue grid represents the location of the lunar base facilities or the planned construction sites of the facilities. The white grid

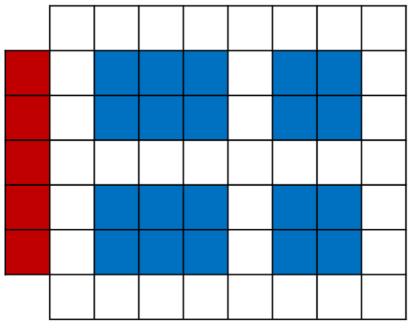


FIGURE 3. Environment represented by a grid map.

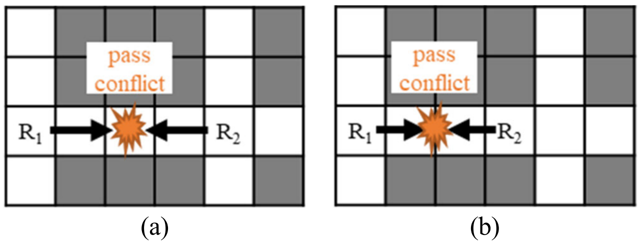


FIGURE 4. Conflict of robots.

represents the aisles along which the robot can move. The number of robots per grid is limited to one. Each robot can only move to the grid adjacent to the grid it is currently on or stay on the grid.

Utilizing the above grid map, two collisions are assumed in this study. The first is when multiple robots collide on a common grid, as shown in Fig. 4 (a). Since this study assumes that there can be only one robot for each grid, cases like Fig. 4 (a) are considered collisions. The second is a collision between the robots exchanging positions in adjacent grids, as shown in Fig. 4 (b). In this case, there is only one robot in each grid; however, in a real situation, robots can collide with each other. In this study, we developed an approach that assumes these two types of collisions and prohibits them as constraints.

B. GRAPHICAL REPRESENTATION

The proposed multi-agent action graph consists of multiple layers, where changes in the environment are represented by changes in the layers. For example, the environment change shown in Fig. 2 is represented by the multi-agent action graph shown in Fig. 5, based on the grid-mapped environment in Fig. 3. As shown in Fig. 5, a multi-agent action graph is a graph representation consisting of nodes and edges, where each layer represents a different environment state. The lower layer represents the environment state before the change, while the upper layer represents the environment state after the change, corresponding to Fig. 2 (b) and 2 (a), respectively.

The red nodes represent depots, blue nodes represent facilities, and black nodes represent aisles. The light-blue and yellow-green nodes indicate the presence of a task

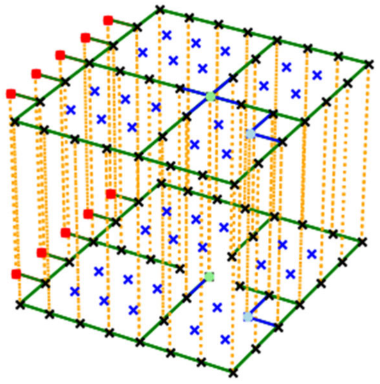


FIGURE 5. Environment represented by a multi-agent action graph.

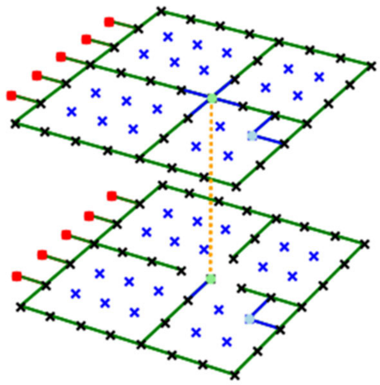


FIGURE 6. Environment represented by a multi-agent action graph with reduced number of edges.

location. Only one robot exists for each node. The edges drawn between nodes indicate that the robot can move. Each node in a layer has edges in the upper, lower, left, and right directions. However, the number of edges connecting to the depots and facilities is reduced, considering the environment shown in Fig. 3. In particular, the edges between the blue nodes are not drawn, considering the possibility that the robot cannot move within the facility. Fig. 5 shows that the edges that were not present in the lower layer are present in the upper layer, indicating that the mobility of the robot has changed. This graphical representation allows for the representation of the increase in the number of passable aisles after the road maintenance task is performed.

The orange edges drawn between the layers represent the robot moving to another layer when the environment changes. Fig. 6 shows only the orange edges between the nodes where the task causing the environment change is located. When a robot moves this edge on the graph, it means that an environment change occurred when the robot performed a task. When a robot performs a task that causes an environment change and transitions its environment state, other robots must also transition to the environment state after the change occurs. Therefore, the orange edge shown in Fig. 6 is insufficient, and many orange edges are drawn, as shown in

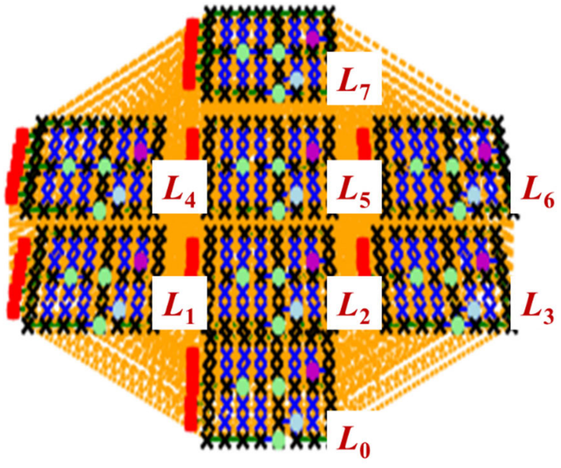


FIGURE 7. A multi-layer multi-agent action graph.

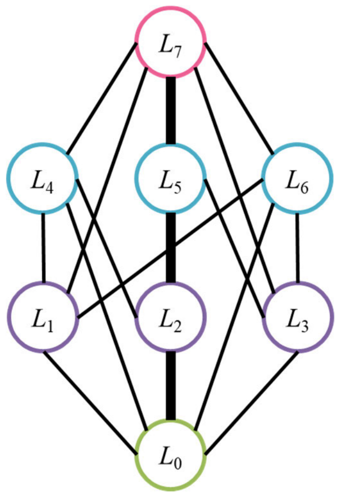


FIGURE 8. Connection of each layer of the multi-layer multi-agent action graph.

Fig. 5. By utilizing the graph shown in Fig. 5, robots can transition between environment states at any given position. In other words, when a robot performs a task that causes an environment change, all robots move from the layer before the change to the layer after the change using the orange edges shown in Fig. 5. However, orange edges are drawn only between nodes that have the same position in the grid map shown in Fig. 3 because they represent only transitions in the environment state and not positional movement. In addition, because all robots must exist in the same environment state, they always exist in the same layer of the graph.

C. MULTI-LAYER MULTI-AGENT ACTION GRAPH

If there are multiple tasks that cause changes in the environment, the number of layers must be multiplied. For example, if there are three tasks that cause environment changes, the environment is represented by a multi-layer multi-agent action graph consisting of eight environment states, as shown in Fig. 7. Assuming that the tasks that cause environment changes are a, b, and c, and that the layers are L_0–L_7 from the bottom, the tasks completed in each layer are as listed in Table 1. As shown in Table 1, the number of completed tasks in each layer is 0 for L_0, 1 for L_1–L_3, 2 for L_4–L_6, and 3 for L_7, and the layer height is determined by the number of completed tasks. The connection of each layer is shown in Fig. 8, with orange edges in Fig. 7. In Fig. 8, the bold lines indicate the overlapping lines. Note that the orange edges drawn between the layers represent only transitions in the environment state and not positional movements. In addition, the edges between the layers are drawn considering the case where multiple tasks that cause environment changes are executed simultaneously. For example, when tasks a, b, and c are executed in sequence, all the robots transition their environment states in the order of layers L_0, L_1, L_4, and L_7. However, when tasks a and b are executed simultaneously, followed by task c, all robots transition their environment state in the order of layers L_0, L_4, and L_7. Thus, the multi-layer multi-agent action graph can represent the changes in environment caused by the execution of multiple tasks by multiple robots.

TABLE 1. Completed tasks in each layer.

Layer	a	b	c
L_0	×	×	×
L_1	○	×	×
L_2	×	○	×
L_3	×	×	○
L_4	○	○	×
L_5	×	○	○
L_6	○	×	○
L_7	○	○	○

IV. FORMULATION FOR OPTIMIZATION

This section describes the formulation for simultaneous optimization of task allocation and path planning in a multi-agent action graph. The proposed formulation allows for optimization that takes into account the fact that multi-robot transitions between environmental states simultaneously. Unlike conventional formulations, which can only respond to environment changes by re-planning, this enables efficient planning considering changes in the robots' mobility in advance. Therefore, the proposed optimization achieves a multi-robot coordinated plan in which one robot waits for another robot to perform a task that causes a change and then performs the task with a smaller travel time.

As alluded previously, we used mixed integer programming for optimization and utilized Gurobi as our mathematical optimization solver. In mixed-integer programming, the optimization objectives and constraints of the optimization problem are first formulated. Then, the optimal solution of the formulated problem is derived by enumerating all solutions. However, when dealing with a problem with many constraints, as in this study, it is difficult to derive a solution through full enumeration because of the large

TABLE 2. **Nomenclature.**

Symbol	Definition
N	A set of all nodes of depots, facility construction site, and aisles shown in Fig. 5.
K	A set of discrete times at which robots can move through the environment.
$T_{tra,n}$	A set of transport tasks n in the same location in Fig. 3 that exist in separate layers in Fig. 5; part of the facility construction site nodes.
$T_{con,n}$	A set of construction tasks n in the same location in Fig. 3 that exist in separate layers in Fig. 5; part of the facility construction site nodes.
$T_{main,n}$	A set of maintenance tasks n in the same location in Fig. 3 that exist in separate layers in Fig. 5; part of the aisle nodes.
TN	A set of all transport task numbers in different locations in Fig. 3.
CN	A set of all construction task numbers in different locations in Fig. 3.
MN	A set of all maintenance task numbers in different locations in Fig. 3.
D_s	A set of all depot nodes in the lowest layer, which robots enter and leave.
D_e	A set of all depot nodes in the highest layer, which robots enter and leave.
R	A set of all robots that perform tasks.
R_{tra}	A set of all robots that perform transport tasks.
R_{con}	A set of all robots that perform construction tasks.
R_{main}	A set of all robots that perform maintenance tasks.
LN	A set of all layer numbers.
L_n	A set of all nodes of depots, facility construction site, and aisles in layer n.
J_i	A set of nodes that can be moved by the robot in one move, including node i at the current location; its neighbors above, below, right, and left in the same layer; and nodes that exist in the same location in Fig. 3 and in the other layers in Fig. 5.
TE_{tra}	The time required to perform the transport task.
TE_{con}	The time required to perform the construction task.
TE_{main}	The time required to perform the maintenance task.
c_{ij}^k	The time required for the robot to move between node i and node j at discrete time k.
h_i	Height of the layer where node i exists.
q_t	Capacity required for the agent to execute transport task t.
Q_r	Load capacity of robot r.
$x_{rij}^k \in \{0,1\}$	Binary variable indicating whether robot r moves between node i and node j at discrete time k.

number of case divisions. Therefore, in this study, the branch-and-bound method was used to reduce the computational complexity [27], [28].

Table 2 lists the variables and sets of variables used to express the objective function and constraints in the mathematical expressions. Using these variables, the objective function is formulated as (1) and constraints (2)–(21).

$$\text{minimize} \sum_{r\in R}\sum_{i\in N}\sum_{j\in J_i}\sum_{k\in K} c_{ij}^k x_{rij}^k \tag{1}$$

$$\text{subject to} \sum_{i\in N}\sum_{j\in J_i} x_{rij}^k = 1 \quad \forall r \in R,\ \forall k \in K \tag{2}$$

$$\sum_{j\in J_i} x_{rji}^k - \sum_{j\in J_i} x_{rij}^{k+1} = 0 \quad \forall r \in R,\ i \in N,\ k \in K(k \neq K_{end}) \tag{3}$$

$$\sum_{d_s\in D_s}\sum_{j\in J_{d_s}} x_{rd_sj}^{K_{start}} = 1 \quad \forall r \in R \tag{4}$$

$$\sum_{d_e\in D_e} x_{rd_ed_e}^{K_{end}} = 1 \quad \forall r \in R \tag{5}$$

$$\sum_{d_e\in D_e}\sum_{\substack{j\in J_{d_e};\\ j\neq d_e}}\sum_{k\in K} x_{rjd_e}^k = 1 \quad \forall r \in R \tag{6}$$

$$\sum_{r\in R}\sum_{j\in J_{d_s}} x_{rd_sj}^{K_{start}} \leq 1 \quad \forall d_s \in D_s \tag{7}$$

$$\sum_{r\in R}\sum_{\substack{j\in J_{d_e};\\ j\neq d_e}} x_{rjd_e}^{K_{end}} \leq 1 \quad \forall d_e \in D_e \tag{8}$$

$$\sum_{r\in R_{tra}}\sum_{t\in T_{tra,n}}\sum_{\substack{j\in J_t;\\ j\notin T_{tra,n}}}\sum_{k\in K} x_{rjt}^k = 1 \quad \forall n \in TN \tag{9}$$

$$\sum_{r\in R_{tra}}\sum_{t\in T_{tra,n}}\sum_{k\in K} x_{rtt}^k \geq TE_{tra} \quad \forall n \in TN \tag{10}$$

$$\sum_{r\in R_{con}}\sum_{t\in T_{con,n}}\sum_{\substack{j\in J_t;\\ j\notin T_{con,n}}}\sum_{k\in K} x_{rjt}^k = 1 \quad \forall n \in CN \tag{11}$$

$$\sum_{r\in R_{con}}\sum_{t\in T_{con,n}}\sum_{k\in K} x_{rtt}^k \geq TE_{con} \quad \forall n \in CN \tag{12}$$

$$\sum_{r\in R_{main}}\sum_{t\in T_{main,n}}\sum_{\substack{j\in J_t;\\ j\neq t}}\sum_{k\in K} x_{rjt}^k \geq 1 \quad \forall n \in MN \tag{13}$$

$$\sum_{r\in R_{main}}\sum_{t_s\in T_{main,n}}\sum_{\substack{t_e\in J_{t_s};\\ h_{t_e}>h_{t_s}}}\sum_{k\in K} x_{rt_st_e}^k \geq 1 \quad \forall n \in MN \tag{14}$$

$$\sum_{r\in R_{main}}\sum_{t\in T_{main,n}}\sum_{k\in K} x_{rtt}^k \geq TE_{main} \quad \forall n \in MN \tag{15}$$

$$\sum_{t\in T_{tra}}\sum_{\substack{j\in J_t;\\ j\neq t}}\sum_{k\in K} q_t x_{rjt}^k \leq Q_r \quad \forall r \in R_{tra} \tag{16}$$

$$\sum_{r\in R}\sum_{j\in J_i} x_{rij}^k \leq 1 \quad \forall i \in N,\ \forall k \in K \tag{17}$$

$$\sum_{r\in R} (x_{rij}^k + x_{rji}^k) \leq 1 \quad \forall i \in N,$$

$$\forall j \in J_i (i \neq j), \quad \forall k \in K \tag{18}$$

$$\sum_{i \in L_n} \sum_{\substack{j \in J_i; \\ j \notin L_n, \\ h_j > h_i}} \sum_{k \in K} x_{rij}^{k} \leq 1 \quad \forall r \in R, \forall n \in LN \tag{19}$$

$$\sum_{i \in L_n} \sum_{\substack{j \in J_i; \\ j \notin L_n, \\ h_j < h_i}} \sum_{k \in K} x_{rij}^{k} = 0 \quad \forall r \in R, \forall n \in LN \tag{20}$$

$$\sum_{i \in L_n} \sum_{\substack{j \in J_i; \\ j \notin L_n}} x_{r_1 ij}^{k} - \sum_{i \in L_n} \sum_{\substack{j \in J_i; \\ j \notin L_n}} x_{r_2 ij}^{k} = 0$$
$$\forall r_1, r_2 \in R (r_1 \neq r_2), \quad \forall k \in K, \forall n \in LN \tag{21}$$

Equation (1) is the objective function that minimizes the total travel time of all robots to achieve the most efficient approach while satisfying the constraints. Equations (2)–(21) are numerical representations of the constraints. Equation (2) is a constraint that states that each robot must be located at some node at all times. Equation (3) is a constraint that states that, at all times, each robot can only move to the current node, to the adjacent node on the same layer, or to a node representing the same position on another layer. Equations (4)–(8) are constraints for each robot to depart from and return to the depot. Equation (4) represents the constraint for each robot to depart from the depot. Equation (5) represents the constraint that each robot eventually returns to the depot in the top layer. Equation (6) represents the constraint that each robot can only enter the depot in the top layer once. Equation (7) represents the constraint that only one robot can depart from each depot in the lowest layer. Equation (8) represents the constraint that only one robot can return to each depot in the highest layer. Equations (9) and (10) represent the constraints on the baggage-transport task. Equation (9) represents the constraint that each transport task must be performed once by one of the transport robots. Equation (10) is the constraint that considers the execution time of the transportation task. The execution time of a task is the minimum time that a robot must stay at the node where the task exists to execute it. Equations (11) and (12) are the constraints on facility construction tasks. Equation (11) represents the constraint that each construction task must be executed once by one of the construction robots. Equation (12) is a constraint that considers the execution time of the construction task. Equations (13)–(15) represent the constraints on the road maintenance task. Equation (13) represents the constraint that each maintenance task must be performed once by one of the maintenance robots. Equation (14) represents the changes in environment that occur when each maintenance task is performed. Specifically, it states that the robot must move from the lower layer to the upper layer at least once between nodes representing two maintenance tasks that are in different layers in Fig. 5 and in the same position in Fig. 3. Equation (15) represents the constraint for considering the execution time of the maintenance task. Equation (16) represents the capacity constraint, which indicates that each transport robot can only simultaneously execute transport tasks that are less than or equal to its load capacity. Equations (17) and (18) are the constraints for avoiding collisions between robots. Equation (17) is the constraint that prohibits robots from colliding at the same node, as shown in Fig. 4(a). Equation (18) is the constraint that prohibits robots from colliding with each other by passing each other at adjacent nodes, as shown in Fig. 4 (b). Equations (19)–(21) are the constraints for all robots to simultaneously transition environment states when a task that causes a change in the environment is executed. Equation (19) is the constraint that each robot can move from one layer below to one above less than once. Equation (20) is the constraint that states that each robot cannot move from the upper layer to the lower layer. That is, the environment cannot return from the state after the maintenance task is executed to the state before the maintenance task is executed. Equation (21) is the constraint that all robots move to other layers simultaneously, and all robots are always in the same layer.

This formulation makes it possible to optimize at once which nodes of the multi-agent action graph each robot will move between at each discrete time. In other words, it is possible to plan at once which robot will perform which task, when, and along what path.

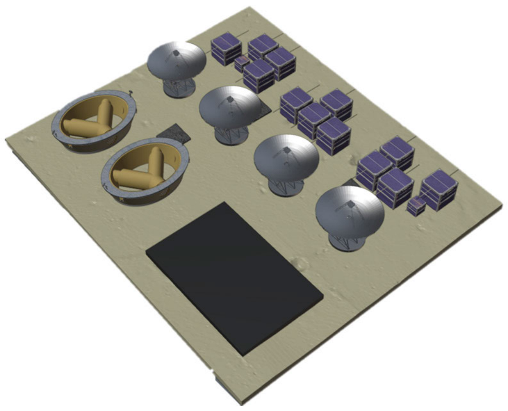


FIGURE 9. Simulation environment.

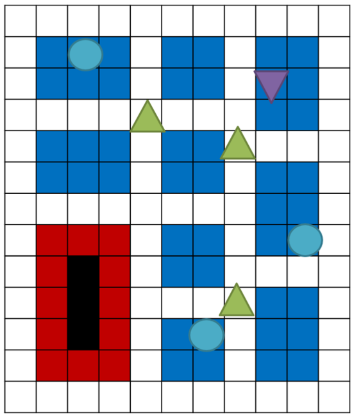


FIGURE 10. Simulation environment represented by a grid map.

TABLE 3. Execution time for each task.

Task	Execution time
Transport	3
Construction	6
Maintenance	5

V. SIMULATION

In this section, we describe the simulations performed to verify the effectiveness of the proposed method. By utilizing multi-agent action graphs, we confirmed that the proposed approach is more efficient than conventional methods taking into account the changes in environment associated with task execution.

A. SIMULATION 1

We first confirm that the proposed method is more effective than conventional methods in a simulation environment where changes may occur with task execution.

1) PURPOSE

In this simulation, the following two points were confirmed.

First, the proposed method realizes task allocation and path planning that considers the changes in environment associated with task execution in advance. Specifically, we confirmed that the proposed method realizes a plan that considers the fact that corridors that were impassable before the execution of a road maintenance task will become passable after the execution of the task. Second, the use of the proposed multi-agent action graph contributes to efficient planning. Specifically, as shown in Fig. 2 (a), we confirmed that the total travel time of the robot is kept small by planning environment changes in advance.

2) METHOD

This simulation was performed using the environment shown in Fig. 9, which assumes a part of a lunar base, referring to the lunar base envisioned in the literature [4]. To realize the construction of the lunar base, the robot was responsible for transporting cargo, building facilities, and maintaining roads. The simulation environment is represented by a grid map, as shown in Fig. 10. The position of each task is indicated by a light-blue circle for the transportation task, a purple inverted triangle for the construction task, and a yellow-green triangle for the maintenance task. To perform these tasks, we assumed a situation in which two transportation robots, one construction robot, and two maintenance robots are used; we also considered the types of tasks that can be performed by each robot. As a capacity constraint, we assumed that a maximum of two transportation tasks can be simultaneously executed. The execution time for each task is shown in Table 3 and is expressed based on the time it takes the robot to move one grid, where 1 is the time required to move. The simulation was repeated 10 times under these conditions. Because the position of each task was set randomly, the positions of the tasks shown in Fig. 10 also changed randomly.

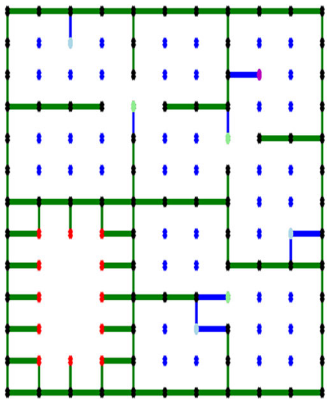


FIGURE 11. Simulation environment represented by a conventional graphical representation.

We used two comparison methods, comparison methods A and B. Both methods adopt an approach that simultaneously optimizes task allocation and path planning by utilizing conventional graphical representations, as shown in Fig. 11. Comparison method A is based on reference [13] and reproduces a state-of-the-art method that accounts for changes in the environment, which we have updated to fit the simulation environment. Therefore, it cannot plan the environment changes associated with task execution in advance but can respond by re-planning as changes occur. Specifically, when a change occurs, the graphical representation shown in Fig. 11 is updated to match the environment, and the robot's position at the time of the change is used as its initial position for optimization once again. Comparison method B is based on reference [26] and reproduces a state-of-the-art method that does not account for changes in the environment, which we have updated to fit the simulation environment. Therefore, it cannot respond the environment changes associated with task execution. In both methods, other constraints, such as the types of tasks that can be executed by the robots, task execution time, capacity constraints, and avoidance of collisions between robots, are considered in the same way as in the proposed method, and optimization is performed using mixed-integer programming. The comparison of these methods allows us to confirm whether the proposed optimization using multi-agent action graphs is more effective than other state-of-the-art methods in a simulation environment where changes in the mobility of robots may occur with task execution.

Two performance metrics were established to determine the efficiency of task allocation and path planning. The first is the task completion time. This is the difference between the time when the first robot leaves the depot and the time when the last robot returns to the depot. By comparing the task completion times, it is possible to evaluate how quickly the construction of the lunar base can be completed. The second factor is the total travel time

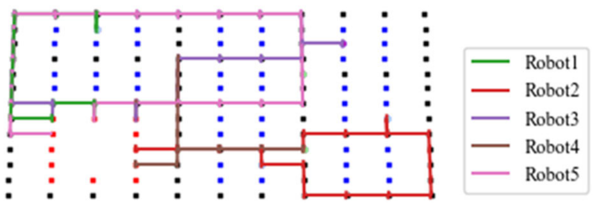


FIGURE 12. Results of the comparison method A.

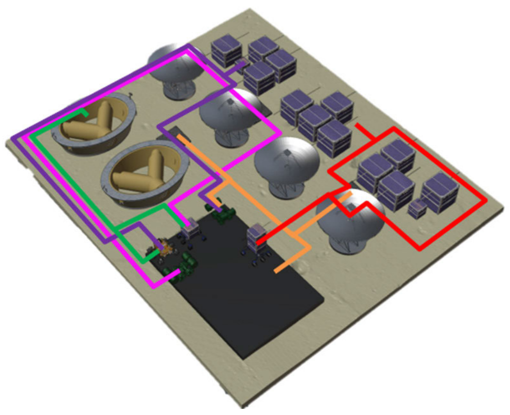


FIGURE 13. Results of the comparison method A.

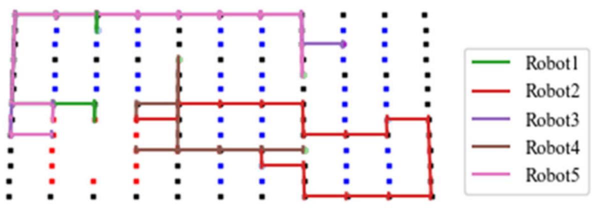


FIGURE 14. Results of the comparison method B.

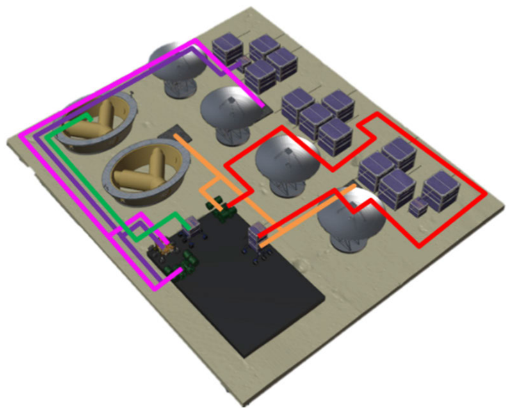


FIGURE 15. Results of the comparison method B.

of the robots. This is the sum of the time spent by each robot from when it leaves the depot to when it returns. By comparing the total travel times of the robots, we can evaluate the amount of energy they consumed. The efficiency of the approach is determined by comparing the results of 10 simulations performed by the comparison method A, the comparison method B, and the proposed method using these indicators.

3) RESULT

Among the 10 simulation results, Figs. 12 and 13 show an example of the results of the comparison method A, Figs. 14 and 15 show an example of the results of the comparison method B, and Figs. 16 and 17 show an example of the results of the proposed method. From Figs. 12 and 13, we confirmed that the comparison method A responds to the fact that the number of passages available for traffic increases but does not consider it before task executions. From Figs. 14 and 15, it can be seen that the comparison method B does not consider the transition of the environment state and fails to consider the fact that the number of passages available for traffic increases even after the execution of the maintenance task. However, Fig. 16 shows that the proposed method plans for the transition of the environment from a state in which none of the tasks that cause environment changes are executed to a state in which two tasks are executed, and then to a state in which three tasks are executed. Fig. 17 also shows that the proposed method considers the fact that a corridor that was impassable before the execution of the maintenance task becomes passable after the execution of the maintenance task. From the above, we confirmed that the proposed method realizes task allocation and path planning that considers the changes in environment associated with task execution in advance, which is the first item to be verified.

Next, Table 4 shows the comparison results of the three methods in terms of two indices: task completion time and total travel time of the robot. The results are the averages of 10 simulation runs, rounded to the first decimal place. Table 4 shows that the proposed method reduces the task completion time by approximately 16.4% and the total travel time of the robots by approximately 16.0% compared with the comparison method A, and reduces the task completion time by approximately 26.8% and the total travel time of the robots by approximately 22.2% compared with the comparison method B. These results confirm the second validation item; that is, the use of multi-agent action graphs contributes to the realization of efficient planning.

4) DISCUSSION

The simulation results confirmed that the use of multi-agent action graphs enables a more efficient approach than other state-of-the-art methods, considering the environment changes associated with task execution in advance. In particular, the results of the evaluation of the two indices, the task completion time and total travel time of the robots, indicate that the planning is significantly more efficient than the conventional methods. In fact, comparing Figs. 18, 19 and 20, which represent the results shown in Figs. 13, 15 and 17 in multiple steps, it can be seen that each method responds differently to changes in the environment. The comparison

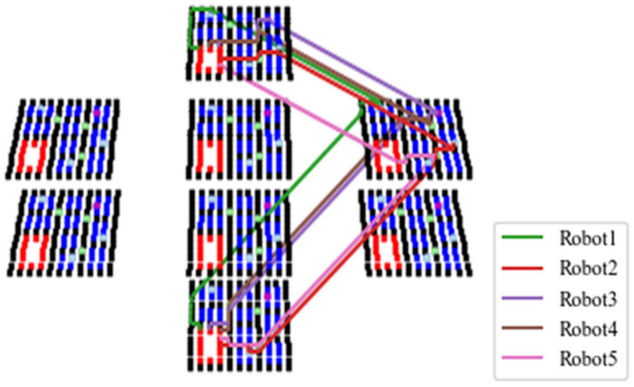


FIGURE 16. Results of the proposed method.

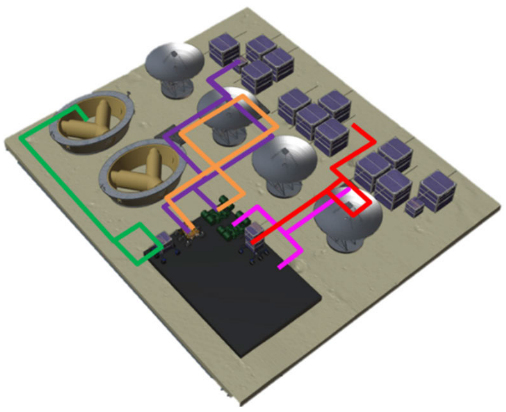


FIGURE 17. Results of the proposed method.

method A responds to changes by re-generating paths after changes have occurred, and the comparison method B generates paths regardless of the changes. The proposed method considers the changes in advance and generates paths that allow each robot to wait for environment changes and perform its tasks on efficient paths. In this example, the comparison method A required 42 task completion time and 142 total travel time of the robots, the comparison method B required 44 task completion time and 174 total travel time of the robots, and the proposed method required 32 task completion time and 118 total travel time of the robots. In other words, the proposed method reduced the task completion time by approximately 23.8% and the total travel of the robots by approximately 16.9% compared to the comparison method A, and reduced the task completion time by approximately 27.3% and the total travel of the robots by approximately 32.2% compared to the comparison method B.

B. SIMULATION 2

Computational performance is also important in order to implement the proposed method in practice. Therefore, we secondly investigate the computational performance of the proposed method.

TABLE 4. Comparison of efficiency.

Indicator	Task completion time	Total travel time of the robots
Comparison method A	37.9	126.0
Comparison method B	43.3	136.0
Proposed method	31.7	105.8

1) PURPOSE

In this simulation, we examined the computational time required to optimize task allocation and path planning using multi-agent action graphs. In particular, we investigated the relationship between the size of the environment in which the optimization is performed and computation time. By examining this relationship, we can obtain an index for determining the size of one compartment when dividing the environment into compartments, as shown in Fig. 1 (b), when actually constructing the lunar base.

2) METHOD

Three transportation tasks, one construction task, and three maintenance tasks were executed by two transportation robots, one construction robot, and two maintenance robots, respectively. To investigate the relationship between the number of nodes and computation time required for optimization, the number of nodes in the environment was varied from 376 to 1504. Because the number of nodes indicates the size of the environment, the relationship between the size of the environment and computation time can be determined by examining the relationship between the number of nodes and computation time. To accurately examine the relationship between the number of nodes and computation time, the type and number of tasks and the type and number of robots are fixed for all environments. In other words, because the number of maintenance tasks is fixed at three, the proposed method performs optimization by utilizing an 8-layer multi-agent action graph. The simulation was performed on a PC with Intel (R) Core (TM) i7-10700F CPU @2.90 GHz, with 32.0 GB RAM, and Gurobi 9.5.0 was utilized as the mathematical optimization solver.

3) RESULT

The relationship between the number of nodes and computation time is shown in Fig. 21. The figure shows that the computation time required for optimization increases as the number of nodes increases. In particular, if the number of nodes increases excessively, the computation time increases extremely. In fact, the computation time for the case with 1504 nodes was approximately 8.8 times longer than that for the case with 1120 nodes. This result confirms that there is no simple proportional relationship between the number of nodes and computation time.

4) DISCUSSION

To determine the size of the environment to be divided into compartments, as shown in Fig. 1 (b), it is necessary to investigate the relationship between the size of the environment

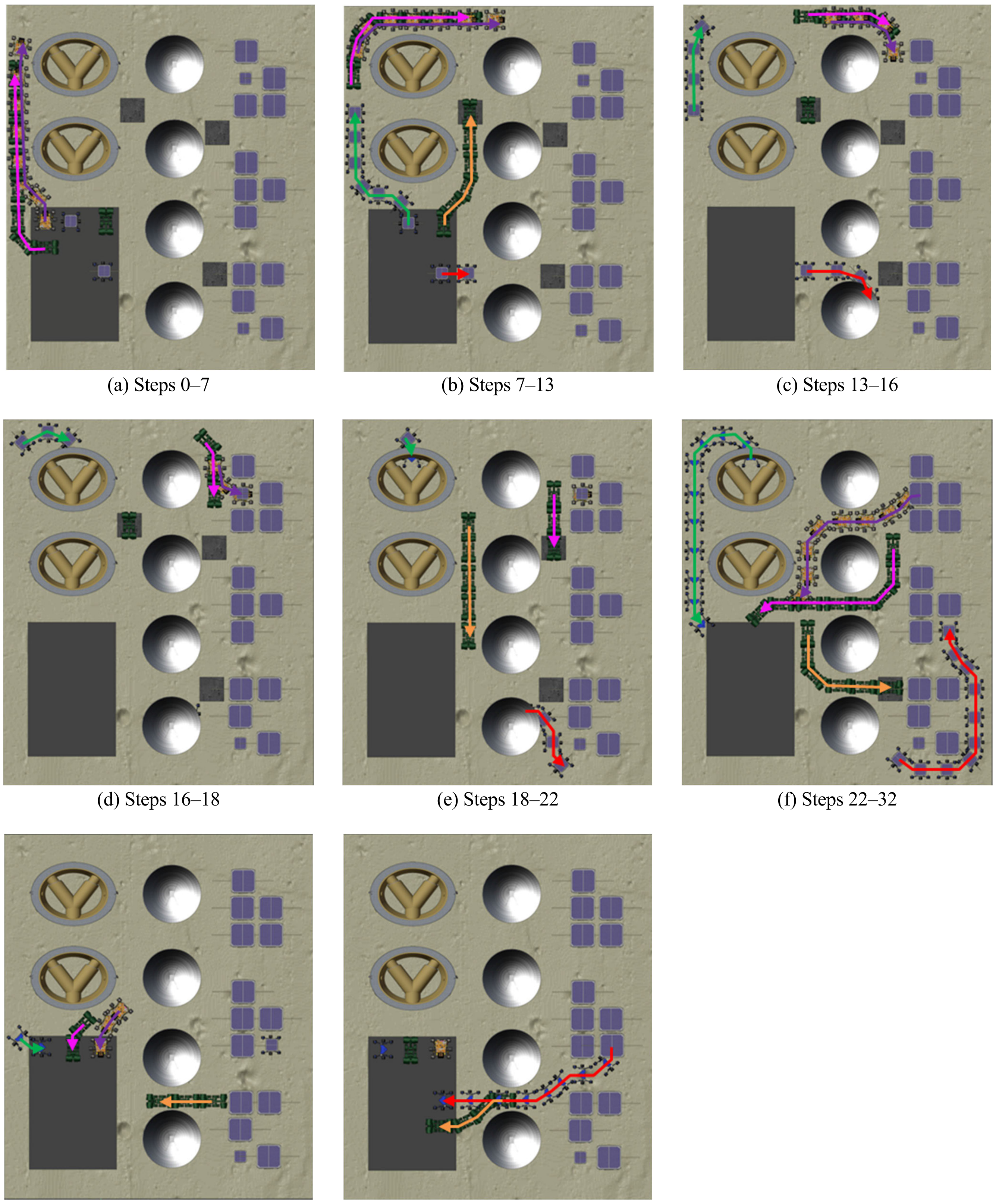
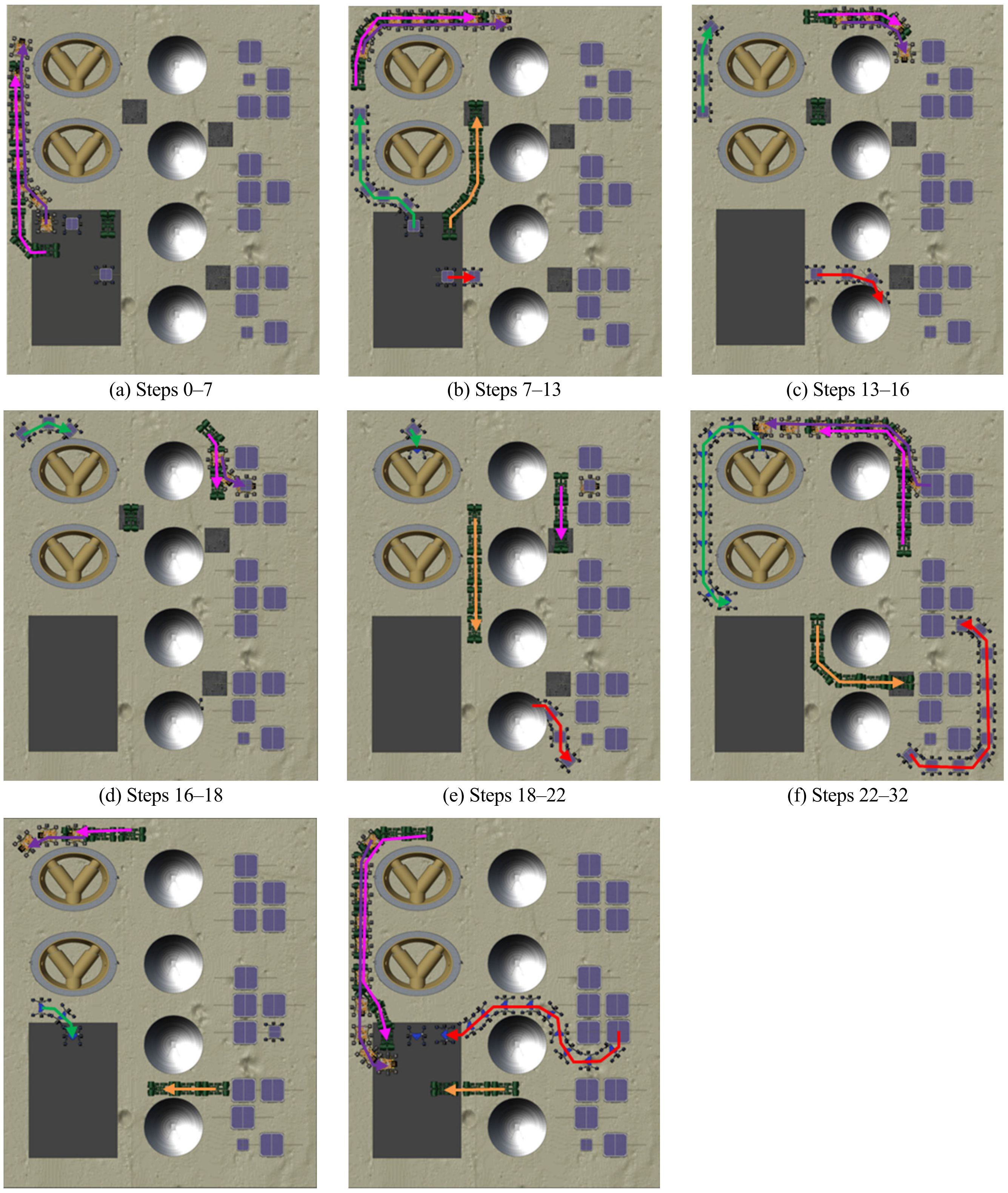

(a) Steps 0–7 (b) Steps 7–13 (c) Steps 13–16
(d) Steps 16–18 (e) Steps 18–22 (f) Steps 22–32
(g) Steps 32–34 (h) Steps 34–42

FIGURE 18. Results of the comparison method A.


(a) Steps 0–7 (b) Steps 7–13 (c) Steps 13–16

(d) Steps 16–18 (e) Steps 18–22 (f) Steps 22–32

(g) Steps 32–34 (h) Steps 34–44

FIGURE 19. Results of the comparison method B.

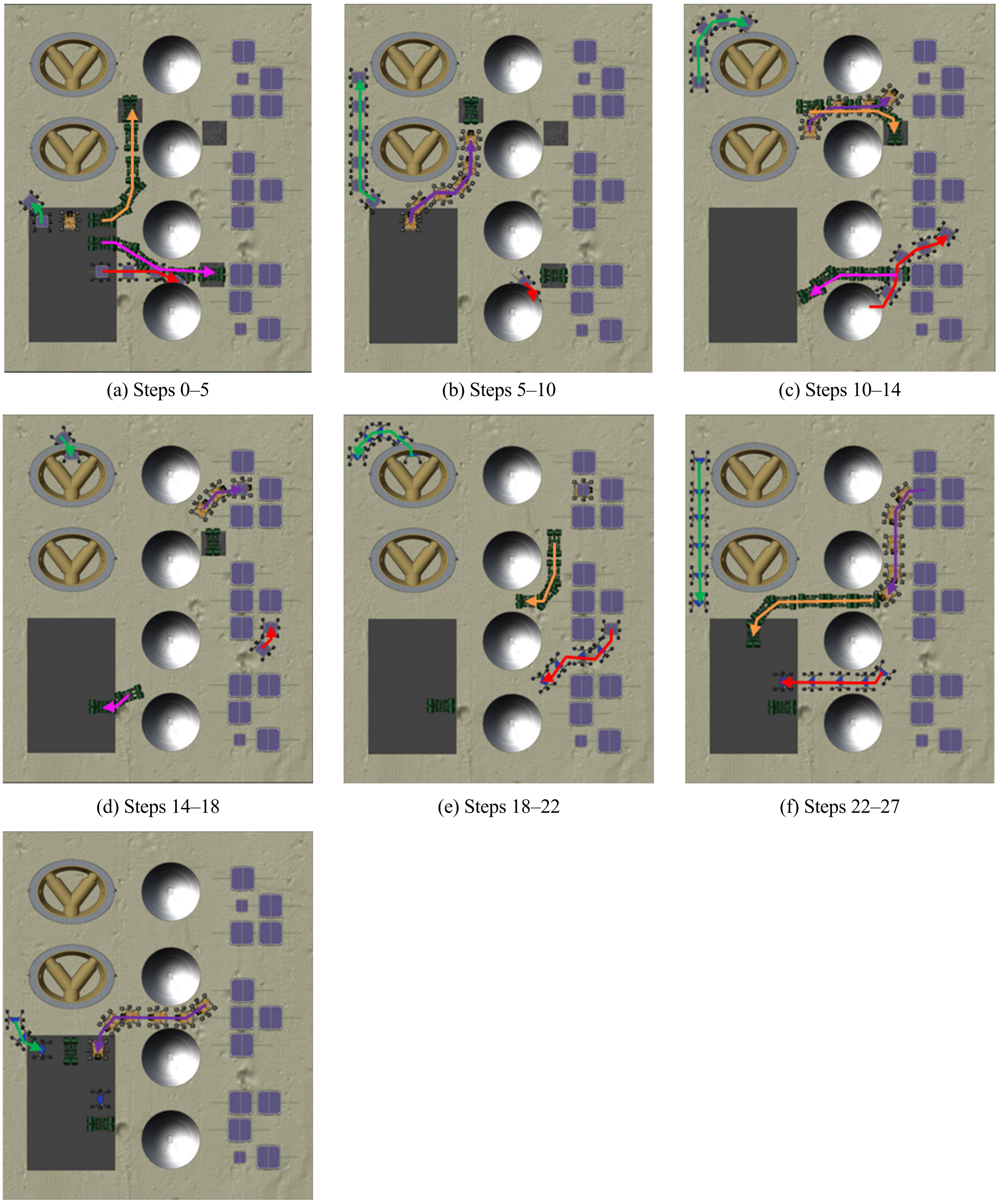

(a) Steps 0–5
(b) Steps 5–10
(c) Steps 10–14
(d) Steps 14–18
(e) Steps 18–22
(f) Steps 22–27
(g) Steps 27–32

FIGURE 20. **Results of the proposed method.**

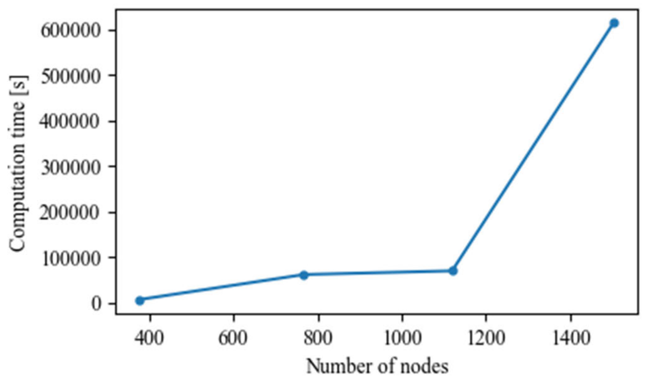


FIGURE 21. Relationship between the number of nodes and computation time.

and computation time, as shown in this simulation. As shown in the results, the larger the size of a compartment, the larger is the size of the environment that can be optimized; however, the computation time increases. Therefore, when applying the proposed method to an actual system, the designer must first determine the number of nodes that can be optimized within the desired computation time. Then, the designer determines the size of the environment for a compartment based on the number of nodes. The size of one node is the size of one robot. It should also be noted that increasing the size of the environment increases not only the computation time but also the size of the environment to be replanned when modifications to the plan are necessary. For example, in the case shown in Fig. 21, it is better to select an environment size of 1120 nodes, which is the number of nodes before the computation explodes. Of course, better performance can be obtained if optimization is performed using a CPU with superior processing speed, so it is also necessary to select the performance of the CPU according to the desired computation time. In particular, it is possible to use CPUs with higher computing performance if the assumption is made that calculations can be performed on the ground in advance. The results for the computing machines used in this study are for reference only.

The results also show that when planning in a large-scale environment or when the number of layers in the multi-agent action graph increases as the number of tasks causing the environment changes increase, the number of nodes in the graph that the proposed method handles in optimization increases, which may cause computational performance problems. Therefore, it is necessary to propose an optimization method that improves computational performance with as little loss of solution accuracy as possible. Specifically, the first step is to improve the solution derivation process of mixed-integer programming. In recent years, methods that dramatically improve the computational performance of mixed-integer programming by convexifying the model or using the decomposition adjustment method instead of the branch-and-bound method have been proposed [29], [30]. These studies do not degrade the accuracy of the solutions derived using mixed-integer programming. It is also possible to apply optimization methods other than mixed-integer programming. As described in Section II, there are several cases where metaheuristics such as genetic algorithms have been applied in conventional research. We will consider applying these methods in the future in this study as well. However, to apply metaheuristics, it is necessary to develop a neighborhood search method and a solution-updating strategy suitable for optimization using multi-agent action graphs. In recent years, there has also been increasing interest in using deep reinforcement learning as an optimization method [31]. Deep reinforcement learning is expected to derive solutions with higher accuracy than metaheuristics. However, there are few examples of its application to constrained real-world problems; thus, the method needs to be developed to be applied to this research. As described above, we will apply various methods in the future to investigate the most suitable optimization method for this study.

VI. CONCLUSION

Herein, we proposed a task allocation and path planning method using a multi-agent action graph to consider changes in the mobility of robots in the environment. Conventional methods can respond to changes in the number of tasks through replanning; however, they cannot plan in advance considering the changes in environment upon a task's execution. Our proposed optimization method overcomes this challenge using a multi-agent action graph, thereby reducing a robot's overall energy consumption. Multi-agent action graphs and the proposed formulation enable efficient planning considering changes in the robots' mobility in advance. Simulations confirmed that the proposed method achieves significantly more efficient planning than conventional methods for two indices: task completion time and total travel time of robots. This indicates that the proposed method achieves more efficient planning, with completion of the lunar base construction earlier and with less energy consumed by the robots. However, the results of the proposed method also show that planning in a large-scale environment cause computational performance problems. Therefore, future prospects include developing an optimization method suitable for utilizing multi-agent action graphs to realize planning in large-scale environments that require higher computational performance.

REFERENCES

[1] M. N. Janardhanan, "Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems," *Ind. Robot, Int. J. Robot. Res. Appl.*, vol. 47, no. 6, pp. 929–942, Sep. 2020.
[2] S. Lin, A. Liu, J. Wang, and X. Kong, "A review of path-planning approaches for multiple mobile robots," *Machines*, vol. 10, no. 9, pp. 1–27, 2022.
[3] A. Austin, B. Sherwood, J. Elliott, A. Colaprete, K. Zacny, P. Metzger, M. Sims, H. Schmitt, S. Magnus, T. Fong, M. Smith, R. P. Casillas, A. Scott Howe, G. Voecks, M. Vaquero, and V. Vendiola, "Robotic lunar surface operations 2," *Acta Astronautica*, vol. 176, pp. 424–437, Jan. 2020.
[4] B. Sherwood, "Principles for a practical moon base," *Acta Astronautica*, vol. 160, pp. 116–124, Jul. 2019.

[5] M. Y. Marov and E. N. Slyuta, "Early steps toward the lunar base deployment: Some prospects," *Acta Astronautica*, vol. 181, pp. 28–39, Apr. 2021.

[6] A. Fursova and E. Nikolaev, "Thermal deformation analysis of a 3D printed Kingdon ion trap for the moon environment," *Adv. Space Res.*, vol. 70, no. 1, pp. 211–222, Jul. 2022.

[7] D. C. Turkoglu and M. E. Genevois, "A comparative survey of service facility location problems," *Ann. Oper. Res.*, vol. 292, no. 1, pp. 399–468, Sep. 2020.

[8] S. T. W. Mara, R. J. Kuo, and A. M. S. Asih, "Location-routing problem: A classification of recent research," *Int. Trans. Oper. Res.*, vol. 28, no. 6, pp. 2941–2983, Nov. 2021.

[9] M. Tomczak and P. Jaśkowski, "Scheduling repetitive construction projects: Structured literature review," *J. Civil Eng. Manage.*, vol. 28, no. 6, pp. 422–442, May 2022.

[10] J. Rosłon, M. Książek-Nowak, and P. Nowak, "Schedules optimization with the use of value engineering and NPV maximization," *Sustainability*, vol. 12, no. 18, p. 7454, Sep. 2020.

[11] S. Park, Y. D. Zhong, and N. E. Leonard, "Multi-robot task allocation games in dynamically changing environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 8678–8684.

[12] W.-Y. Yu, X.-Q. Huang, H.-Y. Luo, V.-W. Soo, and Y.-L. Lee, "Auction-based consensus of autonomous vehicles for multi-target dynamic task allocation and path planning in an unknown obstacle environment," *Appl. Sci.*, vol. 11, no. vol. 11, p. 5057, 2021.

[13] F. Semiz and F. Polat, "Incremental multi-agent path finding," *Future Gener. Comput. Syst.*, vol. 116, pp. 220–233, Mar. 2021.

[14] Y. Kawasaki, S. Mochizuki, and M. Takahashi, "ASTRON: Action-based spatio-temporal robot navigation," *IEEE Access*, vol. 9, pp. 141709–141724, 2021.

[15] R. Tai, J. Wang, and W. Chen, "A prioritized planning algorithm of trajectory coordination based on time Windows for multiple AGVs with delay disturbance," *Assem. Autom.*, vol. 39, no. 5, pp. 753–768, Nov. 2019.

[16] Y. Huang, Y. Zhang, and H. Xiao, "Multi-robot system task allocation mechanism for smart factory," in *Proc. IEEE 8th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, May 2019, pp. 587–591.

[17] W. Xu, S. Guo, X. Li, C. Guo, R. Wu, and Z. Peng, "A dynamic scheduling method for logistics tasks oriented to intelligent manufacturing workshop," *Math. Problems Eng.*, vol. 2019, pp. 1–18, Apr. 2019.

[18] Y. Liu, S. Ji, Z. Su, and D. Guo, "Multi-objective AGV scheduling in an automatic sorting system of an unmanned (intelligent) warehouse by using two adaptive genetic algorithms and a multi-adaptive genetic algorithm," *PLoS ONE*, vol. 14, no. 12, pp. 1–21, 2019.

[19] Z. Liu, H. Wang, W. Chen, J. Yu, and J. Chen, "An incidental delivery based method for resolving multirobot pairwised transportation problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1852–1866, Jul. 2016.

[20] H. Zhang, H. Luo, Z. Wang, Y. Liu, and Y. Liu, "Multi-robot cooperative task allocation with definite path-conflict-free handling," *IEEE Access*, vol. 7, pp. 138495–138511, 2019.

[21] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26022–26035, 2018.

[22] X. Lyu, Y. Song, C. He, Q. Lei, and W. Guo, "Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems," *IEEE Access*, vol. 7, pp. 74909–74924, 2019.

[23] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5816–5823, Jul. 2021.

[24] L. Poltena and S. Emde, "Scheduling automated guided vehicles in very narrow aisle warehouses," *Omega*, vol. 99, pp. 1–20, Mar. 2021.

[25] M. Zhonga, Y. Yangb, Y. Dessoukyc, and O. Postolache, "Multi-AGV scheduling for conflict-free path planning in automated container terminals," *Comput. Ind. Eng.*, vol. 142, pp. 1–11, Apr. 2020.

[26] T. Okubo and M. Takahashi, "Simultaneous optimization of task allocation and path planning using mixed-integer programming for time and capacity constrained multi-agent pickup and delivery," in *Proc. 22nd Int. Conf. Control, Autom. Syst. (ICCAS)*, Busan, South Korea, Nov. 2022, pp. 1088–1093.

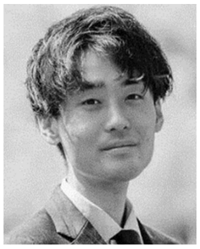
[27] T. Fujieda, "Branch- and cut algorithm for mixed integer programming," *J. Soc. Instrum. Control Eng.*, vol. 42, no. 9, pp. 770–775, 2003.

[28] M. Kubo, "Introduction to mathematical optimization," in *New Mathematical Optimization: Solving with Python and Gurobi*. Tokyo, Japan: Kindaikagakusya, 2012, pp. 1–40.

[29] M. Nozarian, A. H. Nikoofard, and A. Fereidunian, "Efficient MILP formulations for AC optimal power flow to reduce computational effort," *Int. Trans. Electr. Energy Syst.*, vol. 30, no. 8, Aug. 2020, Art. no. e12434.

[30] A.-B. Liu, P. B. Luh, M. A. Bragin, and B. Yan, "Ordinal-optimization concept enabled decomposition and coordination of mixed-integer linear programming problems," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5051–5058, Oct. 2020.

[31] Q. Wang and C. Tang, "Deep reinforcement learning for transportation network combinatorial optimization: A survey," *Knowl.-Based Syst.*, vol. 233, Dec. 2021, Art. no. 107526.



TAKUMA OKUBO received the B.S. degree from the Department of System Design Engineering, Keio University, Yokohama, Japan, in 2021, where he is currently pursuing the degree. His primary research interests include combinational optimization, task allocation, and path planning.

MASAKI TAKAHASHI (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in engineering from Keio University, Yokohama, Japan, in 2000, 2002, and 2004, respectively. In 2004, he was a Research Assistant with the 21st Century COE Program. From 2005 to 2008, he was a Research Assistant with the Department of System Design Engineering, Keio University, where he became an Associate Professor in 2009 and a Professor in 2019. His primary research interests include human–robot interaction, motion and vibration control, and sensor fusion.