International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **10**, October **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 4167–4179

ON COMPLETENESS OF SCALABLE PATH PLANNING FOR MULTIPLE ROBOTS USING GRAPH WITH CYCLES

Hyun-Wook Jo[1], Joo-Ho Park[2], Gwang Tae Kim[1] and Jong-Tae Lim[1,*]

[1]Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
373-1, Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea
*Corresponding author: jtlim@ee.kaist.ac.kr

[2]Electrical Integration and Test Team
Satellite Test Division
Korea Aerospace Research Institute
169-84, Gwahak-ro, Yuseong-gu, Daejeon 305-806, Korea

Received October 2012; revised February 2013

Abstract. *This paper presents a problem of finding collision-free paths for multiple robots in the complex environment. Many existing results for the problem cannot guarantee to find a solution or to reduce the computational complexity as the number of robots increases. In this paper, using a graph with cycles, we show that multiple robots can determine collision-free paths. Specifically, we propose an algorithm which guarantees completeness and scalability even though the empty space is small due to a crowd of robots.*

Keywords: Completeness, Scalability, Path planning, Graph with cycles

1. **Introduction.** As addressed in [20], path planning is important in many real-life problems such as robotics, military applications, logistics, and commercial games. Especially, as the use of multiple robots increases recently due to many potential benefits, path planning for multiple robots has been researched as a critical issue [12]. The objective of the path planning is to move multiple robots from their start positions to goal positions without any collision. Approaches for the path planning of multiple robots are largely classified into two types of methods such as decoupled methods and coupled methods.

Decoupled methods find a path for each robot independently from other robots, and then the paths of all robots are coordinated using various techniques [2, 3, 5, 7, 15-18] to avoid collision. In the velocity tuning technique developed in [7], the velocity of each robot is adapted to avoid collision with other robots. In the prioritized planning technique developed in [2, 3], priority is given to each robot and by moving robots in order of priority, collision is avoided. Although the computational complexity of the above methods is low, decoupled methods cannot guarantee completeness, where completeness means that if there is a collision-free path, finding the path is guaranteed [14]. On the other hand, coupled methods find a path for each robot by considering all robots' current configuration and goal configuration concurrently [8, 9, 13]. With the search algorithms like A^* algorithm in [18], coupled methods can achieve completeness. However, since the computational complexity grows exponentially with the number of robots, coupled methods are limited to a simple problem involving a small number of robots [18].

In [11, 14, 19], they attempt to combine the merits of the above two methods, to reduce the computational complexity while guaranteeing completeness. In [19], to reduce the computational complexity, they create probabilistic roadmaps which reduce the size

of the search space. Nevertheless, the method is appropriate for only a small number of robots. In [11], they proposed an algorithm in which they first find a path of each robot independently, then to avoid collision, start configurations and goal configurations of all robots are changed temporarily so that there are no overlapping paths. Since the path of each robot is computed independently from other robots, the computational complexity is low. However, if the number of robots is large compared with the size of the roadmap, changing start and goal configurations may not be plausible and then, completeness cannot be guaranteed. In [14], first the roadmap is transformed to a graph consisting of nodes and edges. Then, to avoid collision, initially all robots move to leaf nodes which are the nodes with only one incident edge and then they move to each goal according to priority. Although the method guarantees completeness and reduces the computational complexity due to simplicity, it cannot be applied to the case that the number of robots is larger than the number of leaf nodes of the graph. As the size of the roadmap increases, the required empty space to guarantee completeness becomes larger to satisfy the conditions in [11, 14].

On the other hand, robots are used in many industrial fields to increase work productivity as shown in [4, 10]. Also, recently needs for robots have been increased to provide better service [1]. Specifically, in shipbuilding, robots are used to increase work productivity and quality [6]. As the number of robots placed at work sites increases, work productivity and quality can be increased. However, the increase of robots makes empty space small and makes it difficult for robots to find collision-free paths reaching to their goals. Thus, work productivity and quality can be degraded. In such an environment, to guarantee high work productivity and quality, a path planning algorithm suitable for small empty space due to a crowd of robots is needed. The previous researches [11, 14] are not appropriate in that they require large empty space to guarantee finding collision-free paths.

In this paper, we present a path planning method for multiple robots which guarantees completeness with a simple method even though the migration of robots is difficult due to small empty space and due to a crowd of robots. For this purpose, the roadmap is transformed to a graph with cycles. Then, we formally show that representation of the roadmap as a graph can guarantee completeness although empty space is small. Moreover, we show that representation of the roadmap as a graph with cycles can provide scalability for a large number of robots with low computational complexity. Then, we propose an algorithm which guarantees completeness and scalability for a large number of robots in the complex environment easily. Then, through examples including practical application, we show that (1) when the roadmap is not crowded with robots, the proposed algorithm finds collision-free paths with computational complexity lower than or equal to the previous results (2) although previous results cannot guarantee completeness due to a crowd of robots in the roadmap, the proposed method can guarantee completeness.

2. Map Representation with Cycles and Nodes.

As shown in [14], the roadmap is represented as a graph without cycles in which each space is represented as a node and the connecting line between two nodes is represented as an edge. Although the method can guarantee completeness simply, it holds only for the case that the number of leaf nodes is larger than the number of robots. Unlike the method in [14], we represent the roadmap as a graph with cycles and show that forming cycles guarantees completeness for more robots than without forming cycles.

Before presenting the main results, we introduce assumptions about the environment which are also used in [11, 14].

Assumptions about environment

- Each node is occupied by only one robot.

- To avoid collision, only one robot can move for each time interval.
- Robots can turn in place.

Note that from the above assumptions, if there are one or more empty nodes in a cycle, just by rotating in the cycle, any robots in the cycle can move to any node in the cycle while keeping its relative position to other robots. In the following theorem, we use this property.

Theorem 2.1. *Suppose that the roadmap is transformed to a graph with n connected cycles C_i, $i = 1, \cdots, n$ and n sets of leaf nodes L_i connected to C_i. Let the number of nodes in C_i be N_{C_i} and the number of nodes in L_i be N_{L_i}. Let R be the number of robots in the roadmap. If $R \leq 2$, there are at least one or more paths such that the robots can reach their goals. If $R > 2$, the sufficient and necessary condition of the existence of collision-free paths irrespectively of their start configurations is*

$$\begin{cases} N_{C_1} + N_{L_1} - 2 & \geq R \quad for \;\; n = 1 \\ \sum_{i=1}^{n} (N_{C_i} + N_{L_i}) - 3 & \geq R \quad for \;\; n > 1 \end{cases} \tag{1}$$

Proof: When $R \leq 2$, if the roadmap can be changed to a graph with cycles, robots can always avoid collision in cycles, and thus robots can move to any node in the graph irrespectively of their start configuration. Thus, robots can move to their goals irrespectively of their start configuration.

When $R > 2$, if robots are placed in a cycle, they cannot switch their position in the cycle. In the following, we derive the sufficient condition and necessary condition for the existence of collision-free paths irrespectively of robots' start configuration.

Sufficiency: Suppose that $\sum_{i=1}^{n} N_{L_i} = 0$. Let $n = 2$. Suppose that $R = \sum_{i=1}^{2} N_{C_i} - 3$. There are two cycles C_1, C_2 as in the following figure:

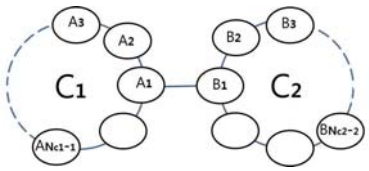


FIGURE 1. Two connected cycles with $R = \sum_{i=1}^{2} N_{C_i} - 3$

The left cycle C_1 has N_{C_1} nodes and $N_{C_1} - 1$ robots A_1, $A_2, \cdots$, $A_{N_{C1}-1}$ and the right cycle C_2 has N_{C_2} nodes and $N_{C_2} - 2$ robots B_1, $B_2, \cdots$, $B_{N_{C2}-2}$. From the structure of cycles, robots can rotate in the cycle while keeping the relative order of robots along the clockwise or the counter clockwise direction only if there is one or more nodes in cycle is empty. Thus, in Figure 1, a robot A_i, $i \in [1, N_{C_1} - 1]$, can always move to either the left side node or the right side node of B_j, $j \in [1, \cdots, N_{C_2} - 2]$. Then, although A_i moves to C_2, there is 1 empty node in C_2. Thus by rotating the robots in C_2, A_i can move to the desired node while keeping the relative position with other robots. Since A_i does not indicate a particular robot rather indicates anyone of robots, the above principle is applied to any robot in C_1. Thus, all robots can move to the desired goals. If there are more than 3 empty nodes in two cycles, robots can move more freely than for the case of 3 empty nodes, obviously all robots can move to their desired goals.

Now suppose that the left cycle has N_{C_1} nodes and N_{C_1} robots and right cycle has N_{C_2} nodes and $N_{C_2} - 3$ robots. To move a robot in C_1 to C_2, the robot can be placed at the node connected to C_2. For this purpose, it is required that robots in C_1 can rotate. However, since there is no empty node in C_1, the robot placed at the node connected to

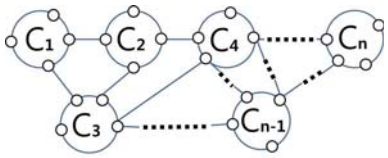


FIGURE 2. Multiple connected cycles with $R = \sum_{i=1}^{n} N_{C_i} - 3$

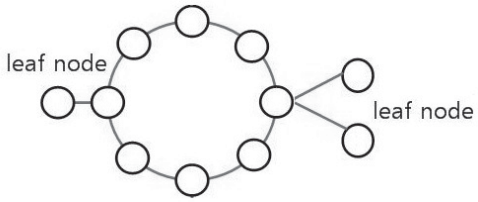


FIGURE 3. One cycle and leaf nodes connected to cycle

C_2 should be moved. Then, the situation becomes the same as the left cycle has N_{C_1} nodes and $N_{C_1} - 1$ robots and the right cycle has N_{C_2} nodes and $N_{C_2} - 2$ robots. Thus, the above principle can be applied, and thus, all robots can move to their desired goals. As it follows from the above, we can see that for $n = 2$ without leaf, if

$$\sum_{i=1}^{2} N_{C_i} - 3 \geq R \tag{2}$$

all robots can move to their goals without collision.

Let $n \geq 3$ and $\sum_{i=1}^{n} N_{L_i} = 0$. Suppose that $R = \sum_{i=1}^{n} N_{C_i} - 3$ and cycles are connected as shown in Figure 2.

From the result of $n = 2$, if there are 3 empty nodes in neighboring two cycles, robots in the cycles can move to any node in the two cycles. On the other hand, suppose that the number of empty nodes in neighboring two cycles is smaller than 3. Since $R = \sum_{i=1}^{n} N_{C_i} - 3$, there are empty nodes outside the two cycles, and by moving some robots in the two cycles to other cycles, 3 empty nodes can be obtained in the neighboring two cycles. Thus, robots which tend to move towards their goals can move to the adjacent cycle in the direction of their goals. By repeating this procedure, robots can reach their goals. Thus, the robots can move to their goals irrespectively of their start configuration if

$$\sum_{i=1}^{n} N_{C_i} - 3 \geq R \quad \text{for } n \geq 2 \tag{3}$$

Now, suppose that there is a cycle C_1 connected to a set of leaf nodes L_1 and $N_{L_1} \neq 0$ as shown in Figure 3. If there are one or more empty nodes in C_1 when a robot R_{L_1} in a set L_1 moves to C_1, R_{L_1} can move to any place in the cycle by rotating in the cycle. Then, by rotating robots in the cycle, any other robots in the cycle can move to the leaf node in which R_{L_1} was firstly placed. Moreover, by moving a robot in C_1 to a node of L_1 and then again moving to C_1, the relative position of the robot to other robots can be switched. Thus, all robots can move to any place. Then, we can easily see that if

$$N_{C_1} + N_{L_1} - 2 \geq R \tag{4}$$

all robots can move to any node without collision.

Now there are two or more cycles C_i, $i = 1, \cdots, n$ and $\sum_{i=1}^{n} L_i \neq 0$. Denote a pair (C_i, L_i) be S_i. Suppose that S_l, $l \in \{1, \cdots, n\}$ and the other set S_k where $l \neq k \in \{1, \cdots, n\}$ are adjacent. Then, from (2) and (4), a robot in S_l can move to any node in C_k if

$$(N_{C_l} + N_{L_l}) + (N_{C_k} + N_{L_k}) - 3 \geq \text{ total number of robots in } S_l \text{ and } S_k \tag{5}$$

On the other hand, from (4), a robot in C_k can move to any node in L_k if

$$(N_{C_l} + N_{L_l}) + (N_{C_k} + N_{L_k}) - 2 \geq \text{ total number of robots in } S_l \text{ and } S_k \tag{6}$$

Note that from (5) and (6), if (5) is satisfied, any robot in S_l can move to any nodes in S_k. Then, we can consider S_i as a cycle, and then by applying the same principle used to drive (3), we can see that if $\sum_{i=1}^{n}(N_{C_i} + N_{L_i}) - 3 \geq R$, any robot can move to any node without collision.

Necessity: We show the necessary condition by mathematical induction. First, for 2 cycles and one or zero leaf node, we show that if (1) is not satisfied, then robots cannot move to their goals irrespectively of their start configuration. First, we assume that for n cycles and m leaf nodes, if (1) is not satisfied, then robots cannot move to their goals irrespectively of their start configuration. Then, we need to show that when the number of cycles increases to $n + 1$ or the number of leaf nodes increases to $m + 1$, if (1) is not satisfied, then robots cannot move to their goals irrespectively of their start configuration. Thus, by this induction method, we show that for any number of cycles and leaf nodes, robots cannot move to their goals irrespectively of their start configuration if (1) is not satisfied.

Consider that there are 2 cycles C_1, C_2 as follows: C_1 has N_{C_1} nodes and $N_{C_1} - 1$ robots $A_1, \cdots, A_{N_{C_1}-1}$ and C_2 has N_{C_2} nodes and $N_{C_2} - 1$ robots, $B_1, \cdots, B_{N_{C_2}-1}$. That is, $N_{C_1} + N_{C_2} - 2 = R$. Suppose that the goal of A_i, $i \in \{1, \cdots, N_{C_1} - 1\}$ is a node in C_2 which is not the node connected to C_1. If A_i moves to the node in C_2 which is connected to C_1, there is no empty node and then the robots cannot rotate in C_2, and thus A_i cannot move to other node in C_2. To prevent this situation, when A_i is transferred to C_2, there is at least one empty node in C_2, and it means that there are at least 2 empty nodes in C_2 before A_i moves to C_2. To make 2 empty nodes in C_2, one of $B_1, \cdots, B_{N_{C_2}-1}$ should move to C_1 before A_i move to C_2. However, then the entrance of C_1 is blocked and A_i cannot move to C_2. Thus, in any cases, A_i cannot move to a node in C_2 which is not the node connected to C_1. Since the same principle holds for other robots, all robots cannot move to their goals irrespectively of their start configuration.

Now suppose that $\sum_{i=1}^{2} N_{L_i} = 1$ and $\sum_{i=1}^{2} N_{C_i} - 1 = R$. Comparing to the case that there is no leaf node, it is clear that the relation $\sum_{i=1}^{2} N_{C_i} + \sum_{i=1}^{2} N_{L_i} - 2 = R$ is not changed. Thus, a robot should be placed in the leaf node and the leaf node cannot provide empty space. The situation is the same as the leaf node does not exist and thus robots in C_1 (or C_2) cannot move to the nodes in C_2 (or C_1). It means that robots cannot move to their goals irrespectively of their start configuration.

Now, suppose that for n cycles with $\sum_{i=1}^{n} N_{L_i} = m$, if $\sum_{i=1}^{n} N_{C_i} + \sum_{i=1}^{n} N_{L_i} - 2 = R$ holds, robots cannot move to their goals irrespectively of their start configuration. Increase the number of cycles to $(n + 1)$ while the relation $\sum_{i=1}^{n} N_{L_i} = m$ remains the same. Then, if $\sum_{i=1}^{n+1} N_{C_i} + \sum_{i=1}^{n+1} N_{L_i} - 2 = R$, it corresponds to the case that a cycle full of robots is appended compared to the case of n cycles and thus the appended cycle cannot provide any empty nodes. The situation is the same as before the cycle is appended. Thus, robots

cannot move to their goals irrespectively of their start configuration. On the other hand, increase the number of leaf nodes to $m+1$ while the number of cycles remains as n, that is, n cycles with $\sum_{i=1}^{n} N_{L_i} = m+1$. Then, if $\sum_{i=1}^{n} N_{C_i} + \sum_{i=1}^{n} N_{L_i} - 2 = R$, it corresponds to the situation that a leaf node full of robots is appended compared to the case of n cycles with $\sum_{i=1}^{n} N_{L_i} = m$ and thus the appended node cannot provide any empty nodes. The situation is the same as before the leaf node is appended. Then, robots cannot move to their goals irrespectively of their start configuration.

Thus, by mathematical induction, when there are two or more cycles, robots cannot move to their goals irrespectively of their start configuration if $\sum_{i=1}^{n} N_{C_i} + \sum_{i=1}^{n} N_{L_i} - 2 = R$. Thus, if $\sum_{i=1}^{n} N_{C_i} + \sum_{i=1}^{n} N_{L_i} - 2 < R$, the number of empty nodes is smaller so that robots cannot move to their goal irrespectively of their start configuration. Thus, robots cannot move to their goal irrespectively of their start configuration if $\sum_{i=1}^{n} N_{C_i} + \sum_{i=1}^{n} N_{L_i} - 2 \leq R$.

Now, by using Theorem 2.1, we show that representing the roadmap as a graph with cycles guarantees the existence of collision-free paths for more or equal to the number of robots representing the roadmap as a graph without cycles.

Theorem 2.2. *Represent the roadmap as a graph of tree structure with the method in* [14]. *Then, transform the the graph except leaf nodes to cycles. Then, the transformed graph with cycles guarantees the existence of collision-free paths for more or equal to the number of robots using the graph without cycles shown in* [14].

Proof: In [14], when the roadmap is represented as a graph of tree structure, collision-free paths for R robots exist if $L - 1 \geq R$ where L is the number of leaf nodes. On the other hand, when there is one or more cycle, from Theorem 2.1, there are collision-free paths for R robots if $N_C + L - 3 \geq R$ where N_C is the total number of nodes in cycles. If $N_C = 2$, then $L - 1 \geq R$ and $N_C + L - 3 \geq R$ are the same but if $N_C \geq 3$, we have $N_C + L - 3 > L - 1$. Thus, if each cycle has at least 3 nodes, representing the roadmap as a graph with cycles guarantees existence of collision-free paths for a large number of robots compared with the result in [14].

Remark 2.1. *If the roadmap is crowded with robots, it is not likely to satisfy* $L - 1 \geq R$. *On the other hand, for the large roadmap, 3 empty nodes turns out to be small space in the large roadmap. Thus, representing the road map as a graph with cycles is appropriate to find collision-free paths when the empty space is small due to a crowd of robots.*

3. **Two Phase Path Planning Algorithm.** In this section, to obtain collision-free paths, we propose a path planning algorithm. The proposed algorithm consists of two phases. As described in detail later, the algorithm is applied to individual robots sequentially according to the predefined priority. In Phase 1, all robots whose goals are leaf nodes move to their goals sequentially according to priority, and then in Phase 2, other robots move to their goals according to priority. We summarize some definitions and settings to be used in the algorithm.

Definitions and Settings: To begin with, we use a word 'CL-set' as a set of a cycle and leaf nodes connected to the cycle.

r_i: In two phases, r_i denotes robot which moves to its goal currently.

depth: When the distance from a predetermined node in a cycle to a node N in the cycle is measured in the clockwise direction, if the distance is large, we say that the depth of the node N is deep, otherwise we say that the depth of the node is swallow.

connection node of S for r_i: When a robot r_i moves to a CL-set S from outside S, there is a node in S which the robot should pass by first. The node is called as the *connection node of S for* r_i.

left (*right*) entry node of S for r_i: When a robot r_i moves to a CL-set S from outside of S, there are two nodes in S adjacent to the connection node of S for r_i. For r_i, the two nodes are placed at the left and right side of the connection node of S respectively. The $left$ ($right$) side node is called as the *left* (*right*) entry node of S for r_i.

outnode(S_1, S_2): When a robot r_i moves from a CL-set S_1 to a CL-set S_2, the node in S_1 connected to S_2 is called as the *outnode*(S_1, S_2).

current CL-set(r_i): The CL-set in which r_i is located.

next CL-set(r_i): The CL-set adjacent to the current CL-set in the direction of r_i's goal.

cnset(r_i): Function such that $cnset(r_i) : r_i \rightarrow$ [current CL-set(r_i), next CL-set(r_i)].

goal CL-set(r_i): The CL-set in which the goal of r_i is located.

infreedom(S): Function such that $infreedom(S) = true$ if the number of empty nodes in $S \geq 2$, and $infreedom(S) = false$ otherwise. This function is used to check whether robots in the CL-set S can move to any nodes in S. If the condition for $true$ is satisfied, it is clear that robots in the CL-set S can move to any node in S because the relation (1) for $n = 1$ is satisfied for the nodes in S.

betfreedom(S_1, S_2): Function such that $betfreedom(S_1, S_2) = true$ if the total number of empty nodes in S_1 and $S_2 \geq 3$ and $betfreedom(S_1, S_2) = false$ otherwise. This function is used to check whether robots in the CL-sets S_1 and S_2 can move to any nodes in S_1 and S_2. If the condition for $true$ is satisfied, it is clear that robots in the CL-sets S_1 and S_2 can move to any node in S_1 and S_2 because the relation (1) for $n > 1$ is satisfied for the nodes in S_1 and S_2.

moveout$(S_1, \cdots, S_k)$: Select any other robot than r_i in one of $S_1, \cdots, S_k$, and move it out from $S_1, \cdots, S_k$.

movetoentry(S): Move r_i to one of the $left$ ($right$) entry nodes in S which are adjacent to next CL-set(r_i).

prior(A): Give priority to robots in a set A. Robots whose goals at leaf nodes should be given higher priority than robots whose goals are cycles. If A is the set of robots whose goals are leaf nodes, assign higher priority to a robot whose goal is connected to a cycle which is nearer to the end of the graph. For the robots whose goals are connected to the same cycle, priority is given randomly. If A is the set of robots whose goals are not leaf nodes, assign higher priority to a robot whose goal is in a cycle which is nearer to the end of the graph. For the robots whose goals are in the same cycle, higher priority is given to the robots whose goals are deeper.

insert (*next CL-set*(r_i)): For the robots in the cycle of the *next CL-set*(r_i), rotate the robots keeping their relative position to the other robots so that the robot whose depth of goal is the deepest among robots whose depth of goal is more shallow than r_i should be placed at the left entry node of the *next CL-set*(r_i). Analogously, the robot whose depth of goal is the shallowest among robots whose depth of goal is deeper than robot i should be placed at the right entry node of the *next CL-set*(r_i), and the connection node of the *next CL-set*(r_i) should be empty.

Recall that robots whose goals are leaf nodes should be given higher priority than robots whose goals are cycles. Since robots whose goals are leaf nodes move in Phase 1 and robots whose goals are leaf nodes move in Phase 2, Phase 2 begins after Phase 1 ends.

A. Phase 1

In this phase, all robots whose goals are leaf nodes move to their goals sequentially according to priority. Since higher priority is given to a robot whose goal is connected to cycle and the cycle is nearer to the end of the graph and leaf nodes are end nodes in the

roadmap, if robots whose goals are leaf nodes move earlier than other robots, they do not block other robots' paths, and thus, complexity of the algorithm decreases. Robots move to their goals sequentially according to their priority. From the proof of Theorem 2.1, to move a robot from a set and to another set, betfreedom should be *true* and to move a robot from a leaf node to the connected cycle or from a cycle to the connected leaf node, infreedom should be *true*. If it is not satisfied, other robots should move to other sets to make the values true. Remaining details are given in the Pseudocode for Phase 1.

Pseudocode for Phase 1

```
A : Set of robots whose goals are leaf nodes
prior(A)
In order of priority, name robots as r_1, r_2 ··· , r_n,
that is r_1's priority is the highest and and r_n's priority is the lowest.
for i = 1 to n
   [current CL-set(r_i)), next CL-set(R_i)] = cnset(r_i)
   while ( next CL-set(r_i) ≠ goal CL-set(r_i))
        infree = infreedom(current CL-set(r_i))
        betfree = betfreedom(current CL-set(r_i), next CL-set(r_i))
         if (betfree = false)
           while (betfree = false)
                moveout(current CL-set(r_i), next CL-set(r_i))
           end while
         else
           if (infree = false)
                moveout(current CL-set(r_i))
           end if
           movetoentry(current CL-set(r_i))
         end if
         infree = infreedom(next CL-set(r_i))
         if (infree = false)
           outnode_Rand(r_i) = randomly selected one node
                                of outnodes(current CL-set(r_i), next CL-set(r_i))
           while (infree = false)
                moveout(next CL-set(r_i), outnode_Rand(r_i))
                infree = infreedom(next CL-set(r_i))
           end while
         end if
         move r_i to next CL-set(r_i)
         [current CL-set(r_i), next CL-set(r_i)] = cnset(r_i)
   end while
   moveout(goal of r_i)
   rotate r_i in the current cycle and then move r_i to its goal
end for
```

B. Phase 2

In the second phase, robots whose goals are not leaf nodes move to their goals sequentially in order of priority. Since all robots whose goals are leaf nodes are placed in their goals through Phase 1, the leaf nodes can be neglected as if these nodes do not exist. Detailed algorithm is given in the Pseudocode for Phase 2.

Pseudocode for Phase 2

```
A : Set of robots whose goals are not leaf nodes
prior(A)
In order of priority, name robots as r_1, r_2 ··· , r_m,
that is r_1's priority is the highest and and r_m's priority is the lowest.
for i = 1 to m
  [current CL-set(r_i)), next CL-set(r_i)] = cnset(r_i)
  If current CL-set(r_i) = goal CL-set(r_i)
    If (robots in current CL-set(r_i) whose goals are in current CL-set(r_i)
                                          are not arranged in order of their depth)
        move r_i to one of adjacent CL-sets
        next CL-set(r_i) = goal CL-set(r_i)
    else
        continue
    end if
  end if
  while (next CL-set(r_i) ≠ goal CL-set(r_i))
      infree = infreedom(current CL-set(r_i))
      betfree = betfreedom(current CL-set(r_i), next CL-set(r_i))
      if (betfree = false)
        while (betfree = false)
            moveout(current CL-set(r_i), next CL-set(r_i))
        end while
      else
        if (infree = false)
            moveout(current CL-set(r_i))
        end if
        movetoentry(current CL-set(r_i))
      end if
      infree = infreedom(next CL-set(r_i))
      if (infree = false)
        outnode_Rand(r_i) = randomly selected one node
                          of outnodes(current CL-set(r_i), next CL-set(r_i))
        while (infree = false)
            moveout(next CL-set(r_i), outnode_Rand(r_i))
            infree = infreedom(current CL-set(r_i))
        end while
      end if
      move(next CL-set(r_i))
      [current CL-set(r_i), next CL-set(r_i)] = cnset(r_i)
  end while
  insert (next CL-set(r_i))
end for
All robots rotate in the cycle while keeping their relative positions to move
 their goals
```

After Phase 1 and Phase 2, all robots arrive to their respective goals completely. Let the complexity of A^* algorithm be C. Note that when there is only one robot, path planning is executed with the same manner of A^* algorithm. Throughout two phases, each robot's path is computed only once since robots move to their goal sequentially and robots which

already arrived to their goals do not block other robots' paths. Then, although the number of robots increases, other robots' paths need not be considered when computing the path for each robot. Thus, the total computational complexity of two phases becomes $O(RC)$ for R robots.

Remark 3.1. *Note that the proposed algorithm guarantees completeness with low computational complexity which increases linearly as the number of robots increases. This achievement is obtained by introducing cycles which allows robots in a cycle move to any place in the cycle just by rotation along the cycle even though other robots are on the cycle. From this property, if the road map is represented as a graph with cycles, when a robot moves to its goal, the robot has only to know the cycles on the path reaching to goal irrespective of the paths of other robots. Thus, the computational complexity is that of the decoupled method.*

4. **Examples.** To show the effectiveness of the proposed method, we compare performances of the proposed method with the methods in [11, 14] for two scenarios. In the first scenario, if the proposed method and the methods in [11, 14] can guarantee completeness, the proposed method exhibits efficiency higher than or equal to the methods in [11, 14] in the sense that the total length of the robots' movement is shorter than or equal to those obtained from [11, 14] and the computational complexity is lower than or equal to those obtained from [11, 14] also. Nowadays, as presented in [6], robots are used in shipbuilding to improve work productivity and quality. In the second scenario, we consider the path planning problem for robots in shipyard as a practical example, where we show that the proposed method can guarantee completeness with computational complexity $O(RC)$ even the methods in [11, 14] cannot guarantee completeness. The comparison results are summarized in Table 1.

i) 1st scenario

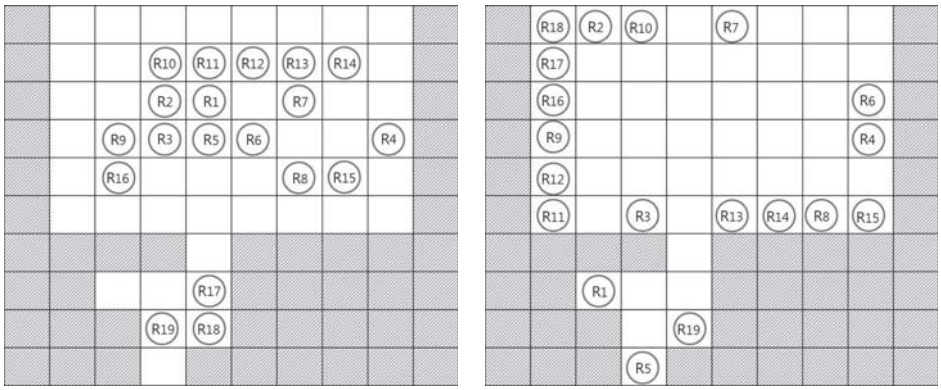
Initially 19 robots $R_1, \cdots, R_{19}$ are located as shown in Figure 4(a) and their goal configuration is shown in Figure 4(b). Squares with slashes are the spaces which are filled with fixed obstacles and thus robots cannot move to those squares. Squares without slashes are free spaces. Since there can be two or more cycles and the number of the empty nodes is larger than 3, the proposed method can be applied. Also, the methods in [11, 14] can be applied because the number of leaf nodes can be larger than the number of robots. Let the length of squares' side be 1. As summarized in Table 1, the proposed algorithm guarantees completeness with performance better than or nearly equal to the methods in [11, 14].

ii) 2nd scenario

Consider the shipbuilding yard as shown in Figure 5(a). Figure 5(a) can be schematized as shown in Figure 5(b). As shown in Figure 5, robots can move on some regions of the

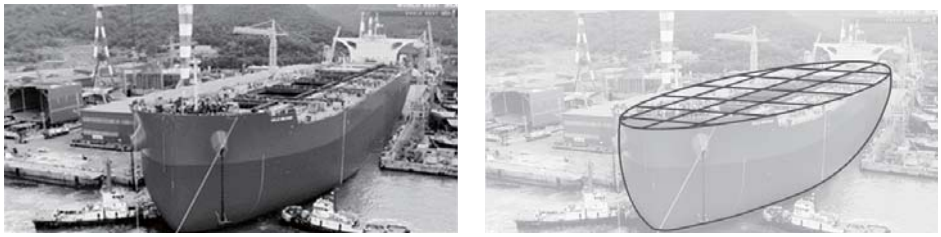
TABLE 1. Performance comparison summary for two scenarios

Scenario	Method	Completeness	Computational complexity	Total movement length of robots
Scenario 1	[11]	guaranteed	21C	93
	[14]	guaranteed	29C	196
	Proposed	guaranteed	22C	93
Scenario 2	[11]	not guaranteed	–	–
	[14]	not guaranteed	–	–
	Proposed	guaranteed	38C	897



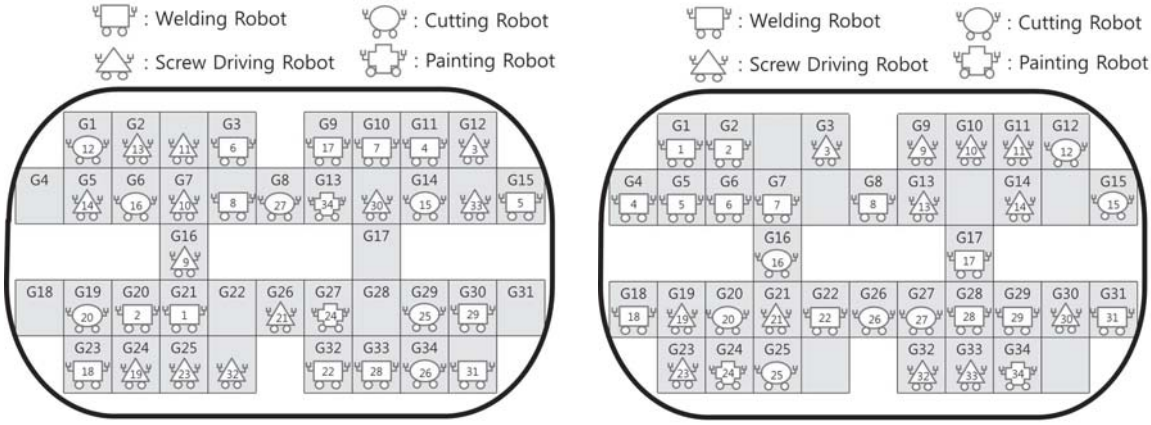
(a) (b)

FIGURE 4. 1st scenario: (a) start configuration, (b) desired final configuration



(a) (b)

FIGURE 5. (a) Shipbuilding yard, (b) schematized shipbuilding yard



(a) (b)

FIGURE 6. 2nd scenario: (a) start configuration, (b) desired final configuration

ship but cannot move on the other regions. Suppose that robots are doing their tasks on the deck. Then, the deck of the ship can be drawn as a roadmap as shown in Figure 6. In Figure 6, the outer line represents the edge of the deck, the gray blocks represent the region where robots can move and the other regions where robots cannot move. Let there be 4 types of tasks, welding, screw driving, cutting, and painting but robots are specialized in only one of the types. Suppose that for each block, 4 types of tasks should be executed and at each block, robots have just finished their tasks at their current positions as shown in Figure 6(a). Then, Figure 6(b) represents their next goal positions where their new

tasks should be executed. Each block can be represented as a node, and then there are 6 empty nodes in the map. Thus, the condition in Theorem 2.1 is satisfied and by applying the proposed algorithm, all robots can reach their goal positions without collision. Let the length of squares' side be 1. As shown in Table 1, proposed algorithm guarantees completeness with computational complexity $O(RC)$ even though the methods in [11, 14] cannot guarantee completeness.

5. **Conclusion.** In this paper, a multiple robot path planning problem is presented. First, we find out the existence condition of collision-free paths irrespective of their start configuration. We show that representing the roadmap as a graph with cycles guarantees the existence of collision-free paths for more robots than that in the existing methods. We quantitatively show the relation between the node number of the roadmap and the number of robots for the existence of collision-free paths. We conclude that as the number of robots increases, representing the roadmap as a graph with cycles is more beneficial than the existing methods. Finally, we propose an algorithm guaranteeing completeness and scalability.

Acknowledgements. This research is supported by NRRC for RIT (NIPA-2012-H1502-12-1002).

REFERENCES

[1] G. Bekey and J. Yuh, The status of robotics, *IEEE Robotics & Automation Magazine*, vol.15, no.1, pp.80-86, 2008.
[2] M. Bennewitz, W. Burgard and S. Thrun, Optimizing schedules for prioritized path planning of multi-robot systems, *Proc. of IEEE Int. Conf. Robot. Autom.*, pp.271-276, 2001.
[3] M. Bennewitz, W. Burgardm and S. Thrun, Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, *Robotics and Autonomous Systems*, vol.41, no.2, pp.89-99, 2002.
[4] G. Biegelbauer, A. Pichler, M. Vincze and C. L. Nielsen, The inverse approach of flex paint, *IEEE Robotics & Autonomation Magazine*, vol.12, no.3, pp.89-99, 2002.
[5] C. Clark, S. Rock and J. C. Latombe, Motion planning for multiple robot systems using dynamic networks, *Proc. of IEEE Int. Conf. Robot. Autom.*, pp.4222-4227, 2003.
[6] P. G. de Santos, M. A. Armada and M. A. Jimenez, Ship bulding with ROWER, *IEEE Robotics & Automation Magazine*, vol.7, no.4, pp.35-43, 2000.
[7] Y. Guo and L. Parker, A distributed and optimal motion planning approach for multiple mobile robot, *Proc. of IEEE Int. Conf. Robot. Autom.*, pp.2612-2619, 2002.
[8] L. Kavraki, P. Svestka, J.-C. Latombe and M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. on Robotics and Automation.*, vol.12, no.4, pp.566-580, 1996.
[9] S. LaValle and J. Kuffner, Rapidly-exploring random trees: Progress and prospects, *Workshop on the Algorithmic Foundations of Robotics*, pp.293-308, 2000.
[10] J. N. Pires, A. Loureiro, T. Godinho, P. Ferreria, B. Fernanodo and J. Morgado, Welding robots, *IEEE Robotics & Automation Magazine*, vol.10, no.2, pp.45-55, 2003.
[11] J.-H. Oh, J.-H. Park and J.-T. Lim, Centralized decoupled path planning algorithm for multiple robots using the temporary goal configurations, *The 3rd International Conference on CIMSIM*, pp.206-209, 2011.
[12] L. E. Parker, *Path Planning and Motion Coordination in Multiple Mobile Robot Teams*, Springer, 2009.
[13] D. Parsons and J. Canny, A motion planner for multiple mobile robots, *Proc. of IEEE International Conference on Robotics and Automation*, pp.8-13, 1990.
[14] M. Peasgood, C. M. Clark and J. McPhee, A complete and scalable strategy for coordinating multiple robots within roadmaps, *IEEE Trans. on Robot.*, vol.24, no.2, pp.283-292, 2008.
[15] J. Peng and S. Akella, Coordinating multiple robots with kinodynamic constraints along specified paths, *Int. J. Robot. Res.*, vol.24, no.4, pp.295-310, 2005.

[16] M. Saha and P. Isto, Multi-robot motion planning by incremental coordination, *Proc. of IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp.5960-5963, 2006.

[17] G. Sanchez and J. C. Latombe, Using a PRM planner to compare centralized and decoupled planning for multi-robot systems, *IEEE Int. Conf. Robot. Autom.*, pp.2112-2119, 2002.

[18] T. Siméon, S. Leroy and J.-P. Laumond, Path coordination for multiple mobile robots: A resolution complete algorithm, *IEEE Trans. on Robot. Autom.*, vol.18, no.1, pp.42-49, 2002.

[19] P. Svestka and M. Overmars, Coordinated path planning for multiple robots, *Robotics and Autonomous Systems*, vol.23, pp.125-152, 1998.

[20] K.-H. C. Wang and A. Botea, Tractable multi-agent path planning on grid maps, *Proc. of the International Joint Conference on Artificial Intelligence*, pp.1870-1875, 2009.