Semi-Uniform Deployment of Mobile Robots in Perfect ℓ-ary Trees

Masahiro Shibata
Graduate School of Computer Science and Systems Engineering
Kyushu Institute of Technology
Fukuoka, Japan
shibata@csn.kyutech.ac.jp

Sébastien Tixeuil
Sorbonne Université
CNRS, LIP6
Paris, France
Sebastien.Tixeuil@lip6.fr

***Abstract*—In this paper, we consider the problem of semi-uniform deployment for mobile robots in perfect ℓ-ary trees, where every intermediate node has ℓ children, and all leaf nodes have the same depth. This problem requires robots to spread in the tree so that, for some positive integer d and some fixed integer s $(0 \leq s \leq d-1)$, each node of depth $s+dj$ $(j \geq 0)$ is occupied by a robot. In other words, after semi-uniform deployment is achieved, nodes of depth $s, s+d, s+2d, \ldots$ are occupied by a robot. Robots have an infinite visibility range but are opaque, that is, robot r_i cannot observe some robot r_j if there exists another robot r_k in the path between r_i and r_j. In addition, each robot can emit a light color visible to itself and other robots, taken from a set of κ colors, at each time step. Then, we clarify the relationship between the number of available light colors and the solvability of the semi-uniform deployment problem. First, we consider robots with the minimum number of available light colors, that is, robots with $\kappa = 1$ (in this case, robots are oblivious). In this setting, we show that there is no collision-free algorithm to solve the semi-uniform deployment problem with explicit termination. Next, we relax the number of available light colors, that is, we consider robots with $\kappa = 2$. In this setting, we propose a collision-free algorithm that can solve the problem with explicit termination. Thus, our algorithm is optimal with respect to the number of light colors. In addition, to the best of our knowledge, this paper is the first to report research considering (a variant of) uniform deployment in graphs other than rings or grids.**

***Index Terms*—mobile robot, semi-uniform deployment, visible lights**

I. Introduction

Background. Studies for mobile robot networks have emerged recently in the field of Distributed Computing. Robots aim to achieve some tasks with limited capabilities. Most studies assume that robots are identical (they execute the same algorithm and cannot be distinguished by their appearance) and oblivious (they cannot remember their past actions). In addition, it is assumed that robots cannot communicate with other robots explicitly. Instead, the communication is done implicitly by having each robot observe the positions of others.

Since Suzuki and Yamashita presented a pioneering work [1], using the above robots, many problems have been studied in continuous environments (*a.k.a.* two- or three-dimensional Euclidean space) [1], or in discrete environments (*a.k.a.* graphs) [2]. For example, the *uniform deployment* (or *uniform scattering*) problem has been studied as a fundamental problem for coordination of robots. This problem requires robots to spread uniformly in the network. So far, the uniform deployment problem for mobile robots has been considered in rings (*i.e.*, the discrete model) [3], cycles (*i.e.*, the continuous model) [4], and grid networks (again, the discrete model) [5]–[7]. In grids, it was argued you can achieve uniform deployment faster [6] using *luminous robots* (uniform deployment in grids is nevertheless feasible by oblivious robots [5]). A luminous robot is equipped with a device that can emit a non-volatile single color (from a constant number of colors) to other robots at a given time. Since the light color is non-volatile, it can be used as a constant space memory. The notion of luminous robots was introduced by Das. et al. [8] with the initial goal to circumvent impossibility results that hold for oblivious robots. In [9], D'Emidio et al. consider the solvability of several problems for luminous robots in the graph environment, focusing on the relationship between synchronicity and light availability. Recently, Poudel and Sharma [7] improved the time complexity of uniform deployment on grids for robots without light colors (*i.e., oblivious robots)*. A separate track of research considered the uniform deployment problem in ring networks for another mobile entity called *mobile agents* [10]–[12], which have persistent memory but cannot observe others' positions unless they are located on the same node. Like aforementioned works, although uniform deployment has been considered in various settings, to the best of our knowledge, it was not considered in graphs other than rings or grids.

Our Contribution. In this paper, we consider a variant of the uniform deployment problem and investigate its solvability in graphs other than rings or grids. Concretely, we consider the *semi-uniform deployment* (or *semi-uniform scattering*) problem of mobile robots in perfect ℓ-ary trees, where every intermediate node has ℓ children, and all leaf nodes have the same depth. This problem requires robots to spread in the tree so that, for some positive integer d and some fixed integer s $(0 \leq s \leq d-1)$, each node of depth $s+dj$ $(j \geq 0)$ is occupied by a robot. An example is given in Fig. 1, where we denote by n and k the number of nodes and the number of robots, respectively. We assume that robots are *semi-synchronous* (SSYNC), that is, in each time step, a non-empty subset of robots are activated and they take an action synchronously and concurrently. In addition, we assume

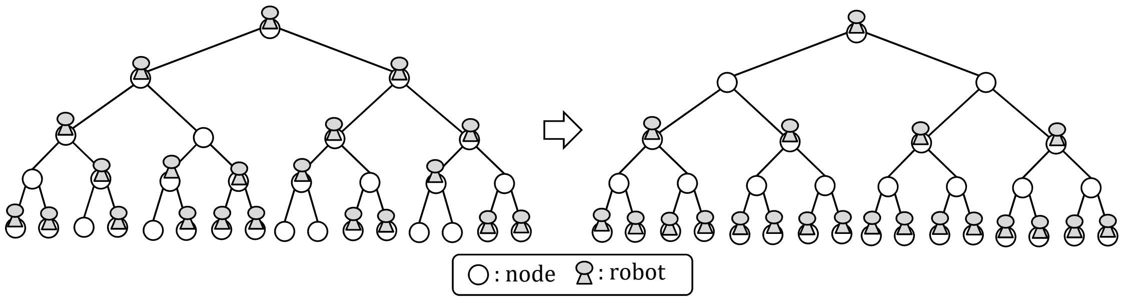

Fig. 1. An example of the semi-uniform deployment problem ($\ell = 2, n = 31, k = 21, d = 2, s = 0$).

that robots have an infinite visibility range but are opaque, that is, robot r_i cannot observe some robot r_j if there exists another robot r_k in the path between r_i and r_j. Moreover, each robot can emit a light color visible to itself and other robots, taken from a set of κ colors, at each time step. Then, we clarify the relationship between the number of available light colors and the solvability of the semi-uniform deployment problem. First, we consider robots with the minimum number of available light colors, that is, robots with $\kappa = 1$ (in this case, robots are oblivious). In this setting, we show that there is no collision-free algorithm to solve the semi-uniform deployment problem with explicit termination. Next, we relax the number of available light colors, that is, we consider robots with $\kappa = 2$. In this setting, we propose a collision-free algorithm that can solve the problem with explicit termination. Thus, our algorithm is optimal with respect to the number of light colors.

II. Model

System models. A *perfect ℓ-ary tree network* T is represented by a tuple $T = (V, E)$, where V is a set of nodes (vertices) and E is a set of edges. We denote by n $(= |V|)$ the number of nodes. In a perfect ℓ-ary tree, there exist three types of nodes: a *root node* v_r with degree ℓ (it has ℓ children), intermediate nodes whose degree is $\ell + 1$ (it has one parent and ℓ children), and leaf nodes whose degree is 1 (it has one parent). From some node, we call the direction toward (resp., away from) the root node the *up direction* (resp., *down direction*). A *subtree* rooted at node v is a part of T that comprises v, and all nodes in v's down direction. The *path* $P(v_0, v_t) = (v_0, v_1, \ldots, v_t)$ with length t is a sequence of nodes from v_0 to v_t such that $\{v_i, v_{i+1}\} \in E$ $(0 \leq i < t)$ and $v_i \neq v_j$ if $i \neq j$. Notice that $P(u, v)$ is unique in a tree for any $u, v \in V$. The *distance* from u to v is the length of the path from u to v, and denoted by $dis(u, v)$. The *depth* of node v is defined as the distance $dis(v_r, v)$ from the root node v_r to v. Notice that the depth of the root node v_r is 0. The *level* of tree T is defined as the maximum depth among leaf nodes. In a perfect ℓ-ary tree, all leaf nodes have the same depth, and hence when the level of the tree is h, the number n of nodes in the tree is $n = \sum_{i=0}^{h} \ell^i$. Let d_v be the degree of node v. We assume that nodes have no distinct IDs (*i.e.,* they are anonymous), but each edge e incident to v is uniquely labeled at v with a label from the set $\{1, 2 \ldots, d_v\}$. We call this label *port number*. Since each edge connects two nodes, it has two port numbers. Port labelings are common to robots, but they are $local$, that is, there is no coherence between the two port numbers in the edge connecting two nodes.

We consider a set of k robots with the following characteristics and capabilities. Robots are *identical*, that is, robots execute the same algorithm. Robots are *luminous*, that is, each robot has a device that can emit a light color (or state) visible to itself and other robots. A robot can choose the color of its light from a discrete set *Col*. When the set *Col* is finite, we denote by κ the number of available colors (*i.e.,* $\kappa = |Col|$). Notice that, when $\kappa = 1$, robots are equivalent to oblivious robots. Robots have knowledge of k and n, and have a *common sense of direction*, that is, each robot r knows the direction toward the root node v_r for each node observable by r. Robots have no other persistent memory and cannot remember the history of past actions. Robots cannot communicate with other robots explicitly, but they can communicate implicitly by observing positions and light colors of other robots (for collecting information), and by changing their light colors and moving (for sending information). In addition, we assume that robots have an infinite visibility range but are *opaque*, that is, robot r_i cannot observe some robot r_j if there exists another robot r_k in the path from r_i to r_j. An example is given in Fig. 2. Numbers at each edge endpoint represent port numbers. In the figure, robot r_i cannot observe r_h because another robot r_j exists in the path from r_i and r_h. Robot r_i can observe the area of solid nodes, edges, and robots, and port numbers within the area, but it cannot observe the dotted area. We call the observable area of r_i the *view* of r_i.

Each robot executes an algorithm by repeating three-phases cycles: Look, Compute, and Move (LCM). During the *Look* phase, the robot observes positions and light colors of robots within its view. During the *Compute* phase, the robot computes its next light color and movement according to the observation in the Look phase. The robot may change its light color at the end of the Compute phase. If the robot decides to move, it moves to an adjacent node during the *Move* phase. In this paper, we assume that robots are *semi-synchronous* (SSYNC), that is, in each cycle, a *scheduler* activates a non-empty subset of robots and the activated robots execute an LCM cycle synchronously and concurrently. We assume that the scheduler

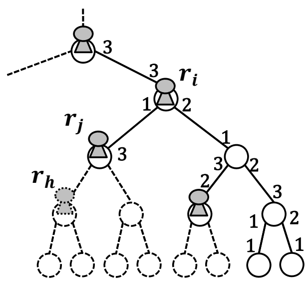

Fig. 2. An example of a view by robot r_i.

is *fair*, that is, each robot is activated infinitely often. We consider the scheduler as an adversary. That is, we assume that the scheduler is omniscient (it knows robot positions, colors, algorithms, etc.), and tries to activate robots in such a way that they fail to execute the task.

A *configuration* $\mathcal{C}$ of the system is defined by the position and light color of all robots. In *initial configuration* C_0, all robots emit the same light color (or they are in the same state) and placed at distinct nodes (however, their placement is decided by the adversary). We call a node hosting a robot (resp., not hosting a robot) a *robot node* (resp., an *empty node*). For an infinite sequence of configurations $E = C_0, C_1, \ldots, C_t, \ldots$, we say E is a *fair execution* from initial configuration C_0 if, for every instant t, C_{t+1} is obtained from C_t after a fair scheduler activates a non-empty subset of robots and they execute an LCM cycle. We say C_i is the i-th configuration of execution E. In addition, during the execution of the algorithm, a *collision* is not allowed. Here, a collision represents a situation such that two robots traverse the same edge from different directions or several robots exist at the same node. Concretely, the following three behaviors are not allowed: *(a)* some robot r_i (resp., r_j) staying at node v_p (resp., v_q) moves to v_q (resp., v_p), *(b)* some robot r_i staying at node v_p remains at v_p and robot r_j staying at node v_q moves to v_p, and *(c)* several robots move to the same empty node. The rationale for avoiding collisions is to prevent two robots from occupying the same node. Otherwise, it leads to a situation such that a deterministic algorithm cannot recover from (assuming the adversary always simultaneously activates all robots at a given node, the system would behave as a system with strictly less robots, and hence robots cannot solve the semi-uniform deployment problem).

The problem to be solved. We assume that there exist $k = \sum_{j=0}^{1+\lfloor (h-s)/d \rfloor} \ell^{s+dj}$ robots that occupy distinct nodes in a perfect ℓ-ary tree of level h for some integers d and s ($2 \leq d \leq \lfloor h/2 \rfloor, 0 \leq s \leq d-1$). Then, the semi-uniform deployment problem in a perfect ℓ-ary tree of level h requires that each node of depth $s + dj\,(0 \leq j \leq 1 + \lfloor (h-s)/d \rfloor)$ is occupied by a robot like Fig. 1. After the deployment, the distance from some robot r to any of *adjacent up (or down) robot* r' is d. Here, we say that robot r' is the adjacent up (or down) robot of r when r' exists in r's up (or down) direction, and no robot exists in the path $P(r, r')$. We define the problem is as follows.

Definition 1. *We assume that there exist* $k = \sum_{j=0}^{1+\lfloor (h-s)/d \rfloor} \ell^{s+dj}$ *robots that occupy in a perfect* ℓ*-ary tree of level* h *for some integers* d *and* s *(*$2 \leq d \leq \lfloor h/2 \rfloor, 0 \leq s \leq d-1$*). Then, an explicitly terminating algorithm solves the semi-uniform deployment problem in a perfect* ℓ*-ary tree if and only if* (i) *each robot eventually enters a state in which it does not move nor change its state, whatever it observes, and* (ii) *after all robots terminate algorithm execution, every node of depth* $s + dj\,(0 \leq j \leq 1 + \lfloor (h-j)/d \rfloor)$ *is occupied by one robot.*

III. Impossibility result

In this section, we show that, when $\kappa = 1$, deterministic terminating semi-uniform deployment is not achievable.

Theorem 1. *When* $\kappa = 1$*, deterministic robots cannot solve the semi-uniform deployment problem with explicit termination.*

Proof. Let us consider the configuration like Fig. 3(a). Note that this configuration is an allowed initial configuration as all robots occupy distinct nodes. In this configuration, in order to achieve semi-uniform deployment, robot r_i must move and visit node v_t (the tree rooted at v_t requires one more robot, and collisions –hence bypassing– are forbidden). However, r_i cannot distinguish the configuration of Fig. 3(a) from a semi-uniform deployment-achieved configuration like Fig. 3(b) due to opacity (its view is the same in both situations). Obviously, in the second situation, r_i has no other choice but to terminate, as no robot as a reason to move. If r_i moves to v_t in configuration like Fig. 3(b), explicit termination is never achieved. If r_i explicitly terminates in configuration like Fig. 3(a), semi-uniform deployment is never achieved. Therefore, the theorem follows. ☐

IV. Proposed algorithm

In this section, we propose an explicitly terminating and collision-free deterministic algorithm to solve the semi-uniform deployment problem for the case of $\kappa = 2$. By Theorem 1, this algorithm is optimal with respect to the number of light colors. In the initial configuration, each robot emits the same light color, say M, and robots are placed at distinct nodes by an adversary. Each robot uses two kinds of light colors: M (Moving) or T (Terminated). Each robot with light color T means that it terminated the algorithm execution and is staying at its destination node (*i.e.,* the robot never leaves its current node anymore). Hence, in the following we explain the behavior of robots with light color M.

Each robot r_i with light color M first looks for a landmark of semi-uniform deployment, such as a leaf node of the tree or a robot with light color T. Since robots are opaque but have an infinite visibility range, at least one robot with light color M can observe a leaf node from any initial configuration. After r_i finds a leaf node (or a robot with light color T), it calculates the locations of destination-candidate nodes for semi-uniform deployment within its view, based on the location of a landmark,

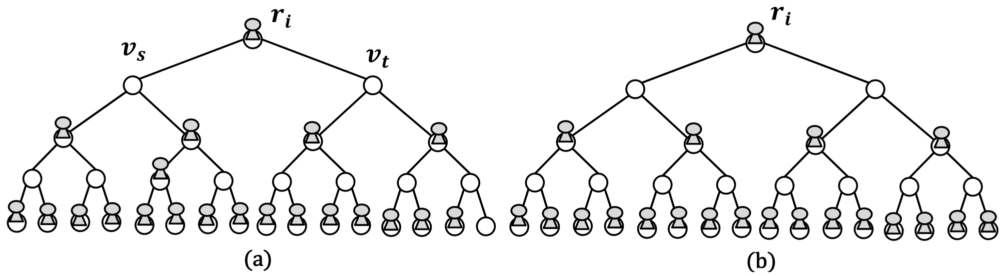

Fig. 3. Configurations such that r_i cannot detect whether it can explicitly terminate the algorithm execution or not.

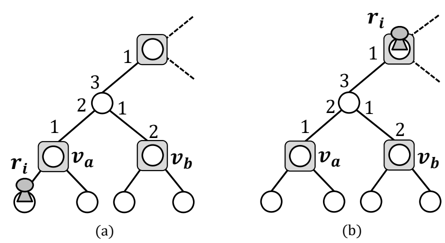

Fig. 4. Examples of selecting a destination node.

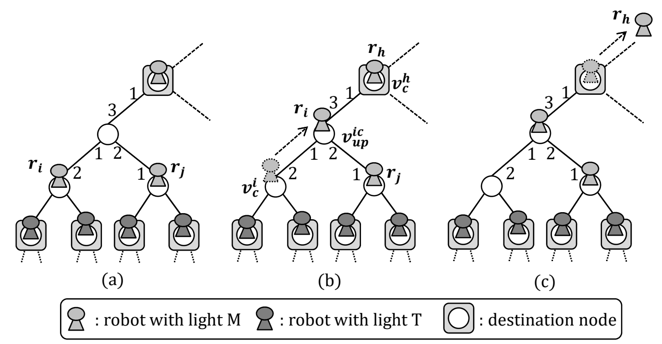

Fig. 5. Movement examples of a robot when it cannot find an empty destination node or it already stays at its destination node ($d = 3$).

the values of n and k, and the fact that it is on a perfect ℓ-ary tree. Let D_i be the set of destination-candidate nodes visible to r_i. Then, among D_i, r_i determines its (temporary) destination node v_d^i as the node with the largest depth (*i.e.,* a node that is the farthest from the root node v_r), and with the shortest length and the smallest port sequence from r_i. Examples are given in Fig. 4. We omit port numbers unrelated to selection of the destination node. In Fig. 4 (a), there exist two destination-candidate nodes v_a and v_b, each of which has the same largest depth. In this case, v_a is selected as the destination node of r_i because it is closer to r_i than v_b. In Fig. 4 (b), there exist two destination-candidate nodes v_a and v_b, each of which has the same largest depth and the same distance to r_i. In this case, v_b is selected because the port sequence 1312 from r_i to v_b is smaller than the sequence 1321 from r_i to v_a. Notice that the reason of why a node with the largest depth is preferentially selected is that there are more destination nodes with a larger depth than destination nodes with a smaller depth due to the tree structure, and hence staying at a destination node with a larger depth can avoid preventing other robots from moving to their destination nodes. Also notice that the destination node for each robot may change when its view changes by its and other robots' movements.

After selection of the destination node, robots try to move to and stay at their destination nodes in a bottom-up manner. Each robot r_i determines its behavior depending on which direction its destination node v_d^i exists from r_i. For explanation, we consider the following four cases in this order: *(a)* r_i cannot find an empty destination-candidate node, *(b)* r_i is already staying at v_d^i, *(c)* v_d^i is in r_i's up direction, and *(d)* v_d^i is in r_i's down direction. In case *(a)*, it means that existence of robots within r_i's view prevents r_i from finding a destination node due to opacity. For example, in Fig. 5(a), although nodes with squares are destination nodes, they are already occupied by robots and robots r_i and r_j cannot stay at any empty destination-candidate node. To inform such a situation, letting r_h be the robot staying at a node with the smallest depth within views of r_i and r_j, they try to move up until they reach the child node of r_h's currently staying node v_c^h. Let v_{up}^{ic} be the adjacent up node of r_i's currently staying node v_c^i. During the movement, since a collision is not allowed, each robot r_i moves up only when there is no other robot at v_{up}^{ic}'s child nodes or when the port sequence from v_{up}^{ic} to v_c^i is the lexicographically smallest among port sequences from v_{up}^{ic} to a child node with a robot (Fig. 5(b)). Notice that, if we assume ASYNC robots, it is possible that several robots staying at child nodes of some node v decide to move up to v at different timings but reach v at the same time, which causes a collision. Also, from Fig. 5(a) to (b), if we assume that port labelings are not common to robots, both r_i and r_j may recognize that each of them has the lexicographically smallest port sequence from v_{up}^{ic}, which also causes a collision. Hence, in this paper we assume that robots are SSYNC and port labelings are common to robots. When r_i continues to move up and eventually reaches v_c^h's child node (node v_{up}^{ic} in case of Fig. 5(b)), r_h detects the fact and tries to move up, which is explained next.

In case *(b)* (*i.e.,* when r_i is already staying at v_d^i), r_i determines its behavior depending on the configuration of the subtree rooted at v_d^i. First, we consider the case that v_d^i is a deepest destination node in the tree. In this case, when there is no robot in r_i's down direction, r_i changes its light color to T

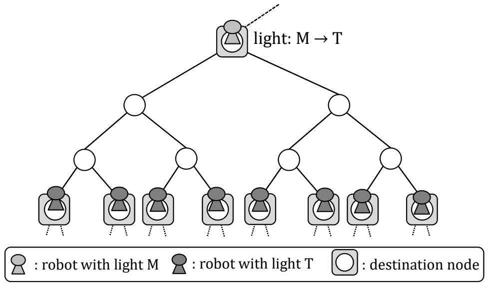

Fig. 6. Movement example of a robot when it already stays at a destination node and terminates the algorithm execution ($d = 3$).

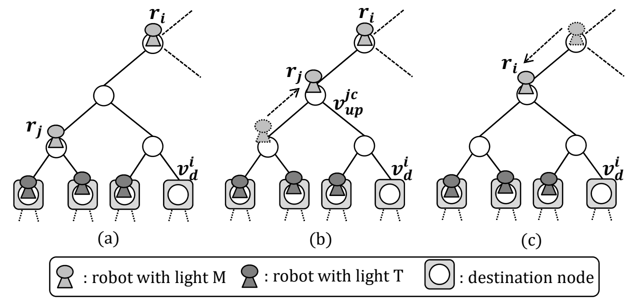

Fig. 7. Movement examples of robot r_i when its destination node is in r_i's down direction ($d > 3$).

and terminates the algorithm execution. Otherwise, *i.e.,* when there is a robot r_j in r_i's down direction, it means that r_j cannot stay at any empty destination-candidate node similarly to Fig. 5(a). In this case, r_j eventually reaches v_d^i's child node as explained in case *(a)*. After that, r_i tries to move up with avoiding a collision as explained above. Next, we consider the case that v_d^i is a non-deepest destination node in the tree. In this case, r_i keeps staying at v_d^i until *(i)* there exists a robot r_j at v_d^i's child node or *(ii)* all nodes with distance d in v_d^i's down direction are occupied by a robot with light color T. When *(i)*, as explained in case *(a)*, it is possible that r_j cannot stay at any empty destination-candidate node. Hence, r_i tries to move up with avoiding a collision like robot r_h in Fig. 5(c). When *(ii)*, it means that all robots in r_i's down direction correctly reached their destination nodes and terminated the algorithm executions. Hence, r_i also changes its light color to T and terminates the algorithm execution (Fig. 6).

In case *(c)* (*i.e.,* v_d^i is in r_i's up direction), r_i tries to move up with avoiding a collision as explained above. Finally, in case *(d)* (*i.e.,* v_d^i is in r_i's down direction), basically r_i tries to move down to reach v_d^i. However, moving down without any additional rule may cause a collision. For example, in Fig. 7(a), robots r_i and r_j recognize v_d^i as a common destination node and r_j tries to move to the up adjacent node v_{up}^{jc} since there is no other robot at v_{up}^{jc}'s child nodes (Fig. 7(b)). Hence, a collision occurs if r_i also moves down. To avoid this, letting T_d^i be the subtree rooted at v_c^i's child node existing on $P(v_c^i, v_d^i)$, r_i tries to move down only when there is no robot in T_d^i or all robots existing in T_d^i emit the same light color T (Fig. 7(c)), which guarantees that the robots in T_d^i do not move anymore. If any of the above conditions is not satisfied, r_i keeps staying at v_c^i. Notice that, while waiting, the destination node of r_i may change to a node existing in r_i's up direction. In this case, r_i applies the behavior in case (c) and tries to move up with avoiding a collision. By these behaviors, with avoiding a collision and a deadlock, all robots eventually reach their destination nodes and they achieve semi-uniform deployment.

An execution example for the case of $d = 2$ is given in Fig. 8. In this figure, each robot observes at least one leaf node and so all robots already know positions of destination nodes. From (a) to (b), robot r_3 already stays at a destination node and there is no robot in r_3's down direction. Hence, r_3 changes its light color to T and terminates the algorithm execution there. In addition, r_1 and r_2 have the same destination node existing in their up direction. In this case, since the port sequence from the destination to r_1 is smaller than that from the destination to r_2, r_1 moves to the destination node and r_2 keeps staying at the current node. Moreover, r_4 and r_5 have the same destination node v_d^{45}. In this case, since the destination node is in the up (resp., down) direction from r_5 (resp., r_4), r_5 moves up to the destination and r_4 keeps staying at the current node. Thereafter, r_5 changes its light color to T and terminates the algorithm execution. From (b) to (c), r_4 moves to the adjacent up node (root node) and checks behaviors of r_1 (and r_2). On the other hand, r_1 moves up because it already stays its destination node but there is another robot r_2 at a child node of r_1's currently staying node. From (c) to (d), r_2 moves up to its destination node, changes its light color to T, and terminates algorithm execution. From (d) to (e), r_1 moves down to its destination node, changes its light color to T, and terminates algorithm execution. From (e) to (f), r_4 recognizes that all nodes with distance 2 ($= d$) are occupied by a robot with light color T, and hence it also changes its light color to T and terminates the algorithm execution. Then, robots achieve semi-uniform deployment.

The pseudocode is described in Algorithm 1. In the algorithm, robots use procedure *Up*() to try to move up with avoiding a collision (lines 26 – 29). To show the correctness of the proposed algorithm, we have the following lemmas.

Lemma 1. *Unless all deepest destination nodes in the tree are occupied by a robot with light color T, at least one robot with light color M recognizes some deepest destination node not occupied by a robot with light color T as a possible destination node.*

Proof. We show the proof by contradiction, that is, we assume that no robot with light color M recognizes some deepest destination node in the tree as a possible destination node. Let r_i be a robot observing some leaf node, v_c^i be the node where r_i is currently staying, and v_d^i be a deepest destination node that is nearest to r_i and not occupied by a robot with light color T (r_i may or may not recognize the existence of v_d^i). Notice that at least one robot can observe a leaf node since robots are opaque but have an infinite visibility range.

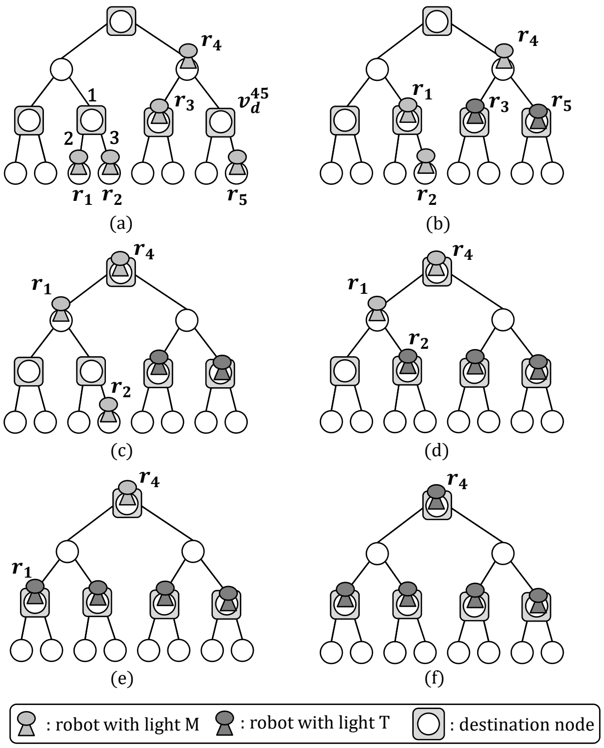

Fig. 8. Movement examples to avoid a situation such that some robot cannot stay at any destination node ($d = 2$).

If v_d^i is within r_i's current view, r_i can recognize v_d^i as a destination node since r_i can observe a leaf node. Thus, by the hypothesis of the contradiction, it is necessary that v_c^i has a larger depth than v_d^i and there exists at least one robot in the path from v_c^i to v_d^i. Among the robots, let r_j be the robot nearest to v_d^i. We assume that r_j is at v_d^i's child node existing in the path from v_d^i to v_c^i (the other cases can be treated similarly). Then, when r_j observes a leaf node, r_j can recognize v_d^i as a destination node. Hence, by the hypothesis of the contradiction, robot(s) are placed at descendant nodes of r_j's currently staying node v_c^j so that r_j cannot observe any leaf nodes in the down direction. Then, the way of placing such robots with the minimum number of robots is to place a robot at each child node of v_c^j. Moreover, when r_j observes a leaf node via the up direction, *i.e.,* a leaf node whose path between the leaf node and v_c^j includes the root node, r_j can also recognize v_d^i as a destination node. To prevent this, it is necessary that *(1)* any child node $v_{child}^{deepest}$ of some deepest destination node in the tree is occupied by a robot and *(2)* any child node of $v_{child}^{deepest}$ is also occupied by a robot. However, letting dep_{max} be the depth of a deepest destination node, the required number of robots is at least the number of nodes with depth $dep_{max}+1$ or $dep_{max}+2$, which is clearly more than the actual number of robots, which is a contradiction. Therefore, the lemma follows. □

Lemma 2. *All deepest destination nodes are eventually occupied by a robot with light color T.*

Algorithm 1 Behavior of robot r_i with light color M (v_c^i is the current node of r_i)

if there is no leaf node or a robot with light color T within its view **then**
  Up()
else
  Calculate set D_i of destination-candidate nodes within its view by n, k, and a position of a leaf node or a robot with light color T
  if $D_i = \emptyset$ **then**
    Up()
  else
    Determine its destination node v_d^i among D_i as the node with the largest depth and with the shortest length and the smallest port sequence from r_i
    if r_i is already staying at v_d^i **then**
      if a robot exists at v_c^i's (or v_d^i's) child node **then**
        Up()
      else if ((r_i cannot observe a leaf node) $\wedge$ (all nodes with distance d existing in v_d^i's down direction are occupied by a robot with light color T)) $\vee$ ((r_i can observe a leaf node) $\wedge$ (there is no robot other than r_i in the subtree rooted at v_d^i)) **then**
        Change its light color to T and terminate the algorithm execution
      else
        Keep staying at v_c^i
      end if
    else if v_d^i is in r_i's up direction **then**
      Up()
    else if v_d^i is in r_i's down direction **then**
      Let T_d^i be the subtree rooted at v_c^i's child node existing in $P(v_c^i, v_d^i)$
      if (there is no robot in T_d^i) $\vee$ (all the robots existing in T_d^i emit the same light color T) **then** move to the down adjacent node via $P(v_c^i, v_d^i)$
      else Keep staying at v_c^i
    end if
  end if
end if

Procedure *Up*()
Let v_{up}^{ic} be the v_c^i's adjacent up node
if (there is no other robot at v_{up}^{ic}'s child nodes) $\vee$ (the port sequence from v_{up}^{ic} to v_c^i is the lexicographically smallest among port sequences from v_{up}^{ic} to a child node with a robot) **then** moves to v_{up}^{ic}
else keep staying at v_c^i

Proof. By Lemma 1, at least one robot r_i with light color M recognizes some node within its view as a deepest and possible destination node v_d^i. In the proof, we show that v_d^i is eventually occupied by a robot with light color T, and the remaining proof (*i.e.,* all other deepest destination nodes are occupied) can be shown in a similar way. Let v_c^i be r_i's currently staying node. We consider the case that the depth of v_c^i is *(a)* larger than that of v_d^i, *(b)* the same with that of v_d^i (*i.e.,* r_i is already staying at v_d^i), and *(c)* smaller than that of v_d^i in this order. First, *(a)* if v_c^i has a larger depth than v_d^i, then v_d^i is in r_i's up direction and r_i tries to move up to reach v_d^i. If there is no other robot in the subtree rooted at v_d^i, by lines 12 – 13 and 17 – 18 in Algorithm 1, r_i can continue to move up, reach v_d^i, change its light color to T, and terminate the algorithm execution. If another robot r_j exist in the subtree rooted at v_d^i, r_i and r_j try to move up with avoiding a collision by Procedure *Up*(). Without loss of generality, we assume that r_i first reaches v_d^i. Then, by lines 1 – 2 in Algorithm 1, r_j eventually reaches v_d^i's child node. After that, r_i detects the fact and eventually moves up (lines 9 – 11 of Algorithm 1). Finally, r_j reaches v_d^i, changes its light color to T, and terminates the algorithm execution (the case when more than two robots exist in the subtree rooted at v_d^i is treated similarly).

Next, we consider the case that *(b)* r_i already stays at v_d^i. In this case, when there is no other robot in the subtree rooted at v_d^i, r_i can change its light color to T and terminate the algorithm execution. Otherwise (*i.e.,* when other robots exist in the subtree rooted at v_d^i), as explained in case *(a)*, r_i eventually moves up and some robot in the subtree rooted at v_d^i eventually reaches v_d^i, changes its light color to T, and terminates the algorithm execution.

Finally, we consider the case that *(c)* v_c^i has a smaller depth than v_d^i. In this case, let v_{down}^i be the adjacent down node of v_c^i existing in the path from v_c^i to v_d^i. Then, when there is no robot with light color M in the subtree rooted at v_{down}^i, by lines 12 – 13 and 19 – 21 in Algorithm 1, r_i can continue to move down, reach v_d^i, change its light color to T, and terminate the algorithm execution. Otherwise (*i.e.,* when other robots with light color M exist in the subtree rooted at v_{down}^i), r_i keeps staying at the current node and checks their behaviors by line 22 of Algorithm 1, and some robot among them or r_i eventually reaches v_d^i, changes its light color to T, and terminates the algorithm execution, as explained in case *(a)* and *(b)*. Therefore, without a collision and a deadlock during the algorithm execution, v_d^i is eventually occupied by a robot with light color T. Thus, the lemma follows. □

Lemma 3. *All destination nodes are eventually occupied by a robot with light color T.*

Proof. By Lemma 2, all deepest destination nodes are eventually occupied by a robot with light color T. Then, by considering robots' behaviors and the similar discussion of the proof of Lemma 2, from destination nodes with a larger depth to destination nodes with a smaller depth, they are occupied by a robot with light color T one by one. Eventually, all destination nodes are occupied with a robot light color T. Thus, the lemma follows. □

By Lemmas 1, 2, and 3, we have the following theorem.

Theorem 2. *When $\kappa = 2$, Algorithm 1 solves the semi-uniform deployment problem with explicit termination.*

V. Conclusion

In this paper, we considered the problem of semi-uniform deployment for luminous and opaque robots in perfect ℓ-ary trees, and clarified the relationship between the number κ of available light colors and the solvability of the problem. First, when $\kappa = 1$ (*i.e.,* robots are oblivious), there is no collision-free algorithm to solve the semi-uniform deployment problem with explicit termination. Next, when $\kappa = 2$, robots can achieve collision-free semi-uniform deployment with explicit termination (and our proof is constructive, as we provide a deterministic algorithm for this task). So, with respect to the number of light colors, our algorithm is optimal. Interestingly, the situation strongly differs from the case of grids, where oblivious solutions are available [5], [7]. In trees, we exhibited an infinite family (the perfect ℓ-ary trees) that precludes oblivious solutions (but allows luminous ones).

An interesting directions for future research are as follows. First, we will consider whether or not oblivious robots can achieve semi-uniform deployment *without* explicit termination (that is, the robots eventually form a semi-uniform deployment, but are unaware the task is complete). Next, we will consider whether or not robots with weaker capability can solve the problem, *e.g.,* ASYNC robots and/or robots without chirality. Finally, we will extend the problem to trees of arbitrary shape (*i.e.*, not just perfect ℓ-ary trees).

Acknowledgements.: This work was partially supported by JSPS KAKENHI Grant Number 18K18031, 20KK0232, and 21K17706. This work was partially funded by the ANR project SAPPORO, ref. 2019-CE25-0005-1.

References

[1] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing, 28(4):1347–1363, 1999.

[2] S. Cicerone, G. Di Stefano, and A. Navarra. Asynchronous robots on graphs: Gathering. In Distributed Computing by Mobile Entities, Current Research in Moving and Computing, volume 11340, pages 184–217. Springer, 2019.

[3] Y. Elor and A. M. Bruckstein. Uniform multi-agent deployment on a ring. Theoretical Computer Science, 412(8-10):783–795, 2011.

[4] P. Flocchini, G. Prencipe, and N. Santoro. Self-deployment of mobile sensors on a ring. Theoretical Computer Science, 402(1):67–80, 2008.

[5] L. Barriere, P. Flocchini, E. Mesa-Barrameda, and N. Santoro. Uniform scattering of autonomous mobile robots in a grid. International Journal of Foundations of Computer Science, 22(03):679–697, 2011.

[6] P. Poudel and G. Sharma. Time-optimal uniform scattering in a grid. ICDCN, pages 228–237, 2019.

[7] P. Poudel and G. Sharma. Fast uniform scattering on a grid for asynchronous oblivious robots. SSS, pages 211–228, 2020.

[8] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. Theoretical Computer Science, 609:171–184, 2016.

[9] M. D'Emidio, G. Di Stefano, D. Frigioni, and A. Navarra. Characterizing the computational power of mobile robots on graphs and implications for the euclidean plane. Information and Computation, 263:57–74, 2018.

[10] M. Shibata, T. Mega, F. Ooshita, H. Kakugawa, and T. Masuzawa. Uniform deployment of mobile agents in asynchronous rings. Journal of Parallel and Distributed Computing, 119:92–106, 2018.

[11] M. Shibata, H. Kakugawa, and T. Masuzawa. Space-efficient uniform deployment of mobile agents in asynchronous unidirectional rings. Theoretical Computer Science, 809:357–371, 2020.

[12] M. Shibata, Y. Sudo, J. Nakamura, and Y. Kim. Uniform deployment of mobile agents in dynamic rings. SSS, pages 248–263, 2020.