Graph Exploration by Energy-Sharing Mobile Agents*

J. Czyzowicz[1], S. Dobrev[2], R. Killick[3], E. Kranakis[3], D. Krizanc[4], L. Narayanan[5], J. Opatrny[5], D. Pankratov[5], and S. Shende[6]

[1] Département d'Informatique, Université du Québec en Outaouais, Canada
[2] Slovak Academy of Sciences, Bratislava, Slovakia
[3] School of Computer Science, Carleton University, Ottawa, Canada
[4] Dept.of Mathematics & Computer Science, Wesleyan University, Middletown CT, USA
[5] Department of Computer Science and Software Engineering, Concordia University, Canada
[6] Department of Computer Science, Rutgers University, USA

Abstract We consider the problem of collective exploration of a known n-node edge-weighted graph by k mobile agents that have limited energy but are capable of energy transfers. The agents are initially placed at an arbitrary subset of nodes in the graph, and each agent has an initial, possibly different, amount of energy. The goal of the exploration problem is for every edge in the graph to be traversed by at least one agent. The amount of energy used by an agent to travel distance x is proportional to x. In our model, the agents can *share* energy when co-located: when two agents meet, one can transfer part of its energy to the other.

For an n-node path, we give an $O(n+k)$ time algorithm that either finds an exploration strategy, or reports that one does not exist. For an n-node tree with ℓ leaves, we give an $O(n+\ell k^2)$ algorithm that finds an exploration strategy if one exists. Finally, for the general graph case, we show that the problem of deciding if exploration is possible by energy-sharing agents is NP-hard, even for 3-regular graphs. In addition, we show that it is always possible to find an exploration strategy if the total energy of the agents is at least twice the total weight of the edges; moreover, this is asymptotically optimal.

Key words and phrases. Energy, Exploration, Graph, Mobile Agent, Path, Sharing, Tree.

1 Introduction

The emergence of swarm robotics has inspired a number of investigations into the capabilities of a collection of autonomous mobile robots (or agents), each with limited capabilities. Such agents cooperate and work collaboratively to achieve complex tasks such as pattern formation, object clustering and assembly, search,

* Research supported in part by NSERC grants and NSF grant AF-1813940.

and exploration. Collaboration on such tasks is achieved by, for example, decomposing the task at hand into smaller tasks which can be performed by individual agents. The benefits of the collaborative paradigm are manifold: smaller task completion time, fault tolerance, and the lower build cost and energy-efficiency of a collection of smaller agents as compared to larger more complex agents. Somewhat surprisingly, for example, a recent paper [16] shows that two agents can search for a target at an unknown location on the line with lower total energy costs than a single agent.

In this paper, we study the problem of collective exploration of a known edge-weighted graph by n mobile agents initially placed at arbitrary nodes of the graph. Many variants of the graph exploration problem have been studied previously; see Section 1.2 for a description of some of the related work. For our work, the goal of exploration is that *every edge* of the graph must be traversed by at least one agent. The weight of an edge is called its *length*. Every agent is equipped with a *battery/energy container* that has an initial amount of energy; the initial energies of different agents can be different. We assume that moving length x depletes the battery of an agent by exactly x.

Clearly then, for exploration to be possible, the sum of the initial energies of all agents has to be at least $\mathcal{E}$, the sum of all edge weights. However total energy $\mathcal{E}$ may not be sufficient; the *initial placement* of the agents plays a role in deciding if exploration is possible with the given energies. To see this, consider exploration by 2 agents of a path with 4 nodes, where each of the 3 edges has length 1. If the agents are initially placed at the two endpoints of the path, then total energy 3 suffices to explore the path. However if the two agents are initially placed at the middle two nodes of the path, it is not difficult to see that total energy 4 is necessary to complete the exploration.

In addition to initial placement of agents, and the total amount of energy, the initial *energy distribution* also affects the existence of an exploration strategy. To see this, suppose the 2 agents are placed at the middle nodes of the 4-node path. Consider first an energy distribution in which both agents have initial energy 2. Then one exploration strategy would be for both agents to explore half of the center edge, and turn around to travel to the endpoint. Next consider an energy distribution in which agent 1 has energy $3 + \epsilon$ for some $0 < \epsilon \leq 1$ and the agent 2 has energy $1 - \epsilon$. It is easy to see that exploration is impossible, even though the total energy of both agents is the same as in the first distribution.

Recently, several researchers have proposed a new mechanism to aid collaboration: *the capability to share energy*. In other words, when two agents meet, one can transfer a portion of its energy to the other. It is interesting to investigate what tasks might be made possible with this new capability, given the same initial amounts of energies. In [6, 12–14, 26], researchers have studied the problems of data delivery, broadcast, and convergecast by energy-sharing mobile agents.

In the example described above, where agent 1 has energy $3 + \epsilon$ for some $0 < \epsilon \leq 1$ and the agent 2 has energy $1 - \epsilon$, if energy transfer is allowed, agent 1 (with the higher energy) can first go to the endpoint closer to its initial position, then turn around, reach agent 2, and transfer its remaining energy ϵ to agent

2. This enables agent 2 to reach the other endpoint, thereby completing the exploration.

This simple example shows that energy-sharing capabilities make graph exploration possible in situations where it would have been impossible otherwise. Note that an algorithm for exploration with energy sharing requires not only an assignment of trajectories to agents that collectively explore the entire graph, but also an achievable schedule of energy transfers. In this paper, we are interested in exploration strategies for edge-weighted graphs by energy-sharing mobile agents. We give a precise definition of our model and the collaborative exploration problem below.

1.1 Model

We are given a *weighted graph* $G = (V, E)$ where V is a set of n vertices (or nodes), E a set of m edges, and each edge $a_i \in E$ is assigned a real number w_i, denoting its *length*. We have k *mobile agents* (or robots) $r_1, r_2, \ldots, r_k$ placed at some of the vertices of the graph. We allow more than one agent to be located in the same place. Each mobile agent (or agent for short) r_i can move with speed 1, and initially possesses a specific amount of *energy* equal to e_i for its moves. An agent can move in any direction along the edges of the graph G, it can stop if needed, and it can reverse its direction of moving either at a vertex, or after traversing a part of an edge. The energy consumed by a moving agent is linearly proportional to the distance x traveled; to simplify notation it is assumed to be equal to x. An agent can move only if its energy is greater than zero.

An important feature of our model is the possibility of *energy sharing* between agents: when two agents, say r_i and r_j, $i \neq j$, meet at some time at some location in the graph, agent r_i can transfer a portion of its energy to r_j. More specifically, if e'_i and e'_j are the energy levels of r_i and r_j at the time they meet then r_i can transfer to r_j energy $0 < e \leq e'_i$ and thus their energies will become $e'_i - e$ and $e'_j + e$, respectively.

In our model, each agent is assigned a *trajectory* to follow. We define a trajectory of an agent to be a sequence of edges or parts of edges that starts at the agent's initial position and forms a continuous walk in the graph. In addition, a trajectory specifies a *schedule* of energy transfers, i.e., all points on this walk (could be points different from vertices) where the agent is to receive/transfer energy from/to other agents, and for each such point the amounts of energy involved. We call a set of trajectories *valid* if the schedules of energy transfers among trajectories match, and energy levels are sufficient for the movement of agents. More specifically, for every transfer point on a trajectory of agent r_i where energy is to be received/transferred, there is exactly one agent r_j, $j \neq i$, whose trajectory contains the same transfer point transferring/receiving that amount of energy to/from r_i, and the transfers can scheduled on a time line. Furthermore, the energy of an agent, initially and after any energy transfer, must be always

sufficient to continue to move along its assigned trajectory. We are interested in solving the following general problem of collaborative exploration:

Graph Exploration Problem: Given a *weighted graph* $G = (V, E)$ and k mobile agents $r_1, r_2, \ldots, r_k$ together with their respective initial energies $e_1, e_2, \ldots, e_k$ and positions $s_1, s_2, \ldots, s_k$ in the graph, find a valid set of trajectories that *explore* (or cover) all edges of the graph.

1.2 Related work

The problems of exploration and searching have been investigated for over fifty years. The studied environments were usually graphs (e.g.[1, 18, 23, 27]) and geometric two-dimensional terrains (e.g.[2, 4, 17]). The goal of such research was most often the minimization of the time of the search/exploration that was proportional to the distance travelled by the searcher. The task of searching consists of finding the target placed at an unknown position of the environment. The environment itself was sometimes known in advance (cf. [4, 8, 14, 23]) but most research assumed only its partial knowledge, e.g. the type of graph, the upper bound on its size or its node degree, etc. Remarkably, there exist hundreds of papers for search in an environment as simple as a line (cf. [3]). The task of exploration consisted of constructing a traversal of the entire environment, e.g. in order to construct its map (see [22, 27]). It is worth noting that performing a complete graph traversal does not result in acquiring the knowledge of the map (see [10]).

Most of the early research on search and exploration has been done for the case of a single searcher. When a team of collaborating searchers (also called agents or robots) is available, the main challenge is usually to partition the task among the team members and synchronize their efforts using available means of communication, cf. [5, 9, 19, 21]. Unfortunately, for the centralized setting, already in the case of two robots in the tree environment known in advance, minimizing its exploration time is NP-hard, e.g., see [21].

The case of robots that can share energy has been recently studied for the tasks of data communication, [6, 12–14, 26]. In this research the robots are distributed in different places of the network, each robot initially possessing some amount of energy, possibly distinct for different robots. The energy is used proportionally to the distance travelled by the robot. The simplest communication task is *data delivery* (see [6, 7, 11, 12]), where the data packet originally placed in some initial position in the environment has to be carried by the collaborating robots into the target place. Remarkably, when the robots cannot share energy, data delivery is an NP-hard problem even for the line network, (see [11]). When the robots are allowed to exchange a portion of energy while they meet in the tree of n nodes, [12] gives the $O(n)$-time solution for the data delivery. For energy sharing robots, the authors of [13] study the *broadcast problem*, where a single packet of data has to be carried to all nodes of the tree network, while [12] investigates also the *convergecast problem*, where the data from all tree nodes need to be accumulated in the memory of the same robot. In both cases efficient

communication algorithms are proposed. A byproduct of [14] is an optimal exploration algorithm in the special case when all robots are initially positioned at the same node of the tree. When the energy sharing robots have small limited memory, able to carry only one or two data packets at a time, the simplest case of the data delivery problem is shown to be NP-hard in [6]. Further, in [20] bounds are proved in an energy model where robots can communicate when they are in the same node and the goal of robot team is to jointly explore an unknown tree.

1.3 Results of the paper

We start in Section 2 with exploration of a path. Given an initial placement and energy distribution for k energy-sharing agents on an n-node path, we give an $O(n+k)$ algorithm to generate a set of valid trajectories whenever the exploration of the path is possible. We also show that a path can always be explored if the total energy of energy-sharing agents is $\frac{3}{2}$ times the total weight of edges in the path. In contrast, we show that there are energy configurations for which any total amount of energy is insufficient for path exploration *without* energy sharing.

In Section 3 we study exploration of trees. We first observe that without energy sharing the exploration of trees is NP-complete. Then, for an n-node tree, we give an $O(n + \ell k^2)$ algorithm that finds an exploration strategy if one exists, where ℓ is the number of leaves in the tree.

In Section 4, we consider exploration of general graphs. We show that the problem is NP-hard even for 3-regular graphs. In addition, we show that it is always possible to find an exploration strategy if the total energy of the agents is at least twice the total weight of the edges; moreover, this is asymptotically optimal, even for trees.

Therefore our results show that allowing energy to be shared between agents makes exploration possible in many situations when it would not be possible without sharing energy. Furthermore, the total energy needed for exploration is at most twice (at most 3/2) the total weight of the edges in the graph (path respectively), while there is no upper bound on the total energy needed for exploration if agents cannot share energy, even when the graph to be explored is a path. Due to space limitations, all missing proofs can be found in [15].

2 Exploring a Path

In this section we consider the case when the graph is a simple path on n nodes; without loss of generality, we assume that the path is embedded in the horizontal line segment $[0, 1]$, and we will refer to the movements of agents in their trajectories as being left/right movements on the segment. Clearly, in case the graph is given in the usual graph representation, this embedding can be obtained in $O(n+k)$ time. The path exploration problem can therefore be restated as follows:

Problem 1 (Segment Exploration). Given mobile agents $r_1, r_2, \ldots, r_k$ with energies $e_1, e_2, \ldots, e_k$, located initially in positions $0 \leq s_1 \leq s_2 \leq \cdots \leq s_k \leq 1$ of

a line segment $[0, 1]$, respectively, find a set of valid trajectories of these agents that explore the segment, if possible.

A trajectory t_i of agent r_i explores a closed sub-segment a_i of $[0, 1]$ containing s_i. Let b_i^ℓ, b_i^r be the left, right end point of this sub-segment. We want to find a valid set of trajectories, i.e., a set that explores the line segment $[0, 1]$, and there exists a schedule for energy transfers such that every agent has enough energy to follow its trajectory.

We first observe that in the case of exploring a line segment with the possibility of energy sharing some assumptions on the shape of valid trajectories can be made without loss of generality.

1. The segments $a_1, a_2, \ldots, a_k$ explore (or cover) $[0, 1]$ and they don't overlap, i.e., $b_1^\ell = 0$, $b_n^r = 1$, and $b_i^r = b_{i+1}^\ell$ for $1 \leq i \leq k - 1$.
2. Trajectory t_i starts at s_i, goes straight to one of the endpoints of a_i. When both endpoints are different from s_i, it turns around and goes straight the other endpoint of a_i. Thus, in this case the trajectory *covers doubly* a sub-segment between s_i and the endpoint where it turns around, and the trajectory has a *doubly covered part* and a *singly covered part*.
3. A transfer of energy between two agents r_i and r_{i+1} may occur only at their *meeting point* b_i^r. Thus at b_i^r exactly one of the following occurs:
   (a) There is no energy transfer.
   (b) There is energy transfer from r_i to r_{i+1}. In that case t_{i+1} does not end at that point, it ends at b_{i+1}^r, and either $b_{i+1}^\ell = s_{i+1}$ or b_{i+1}^ℓ is a point where the trajectory t_{i+1} turns around to the right.
   (c) There is energy transfer from r_{i+1} to r_i. In that case t_i does not end at that point, it ends at b_i^ℓ and either $b_i^r = s_i$ or b_i^r is a point where the trajectory t_i turns around to the left.

The next lemma, stated without proof, specifies two additional restrictions that can be imposed on the nature of valid trajectories that will be applied by our algorithm.

Lemma 1. *Assume that the segment* $[0, 1]$ *can be explored by a set of valid trajectories* $T = \{t_1, t_2, \ldots, t_k\}$ *of the agents. Then there is a* canonical *set of valid trajectories* $T' = \{t'_1, t'_2, \ldots, t'_k\}$ *that explore the segment such that*

(i) If agent r_i receives energy from a right (left) neighbour then it receives it at its initial position s_i, and its trajectory may only go in straight line segment from s_i to the left (right).
(ii) For each trajectory, its singly covered part is at least as long as its doubly covered part.

We now describe a recursive, linear time algorithm for Problem 1 to find canonical trajectories as described in Lemma 1 above. The trajectories are assigned to agents from left to right, determining whether more energy needs to be transferred to complete the coverage on the left, or some surplus energy is to be transfused to agents on the right. If i is the index of the leftmost agent not

used in the exploration of initial sub-segment $[0, \ell_i]$ and if tr_i is the energy deficit (negative) or surplus (positive) left after exploring this sub-segment using only agents 1 through $(i-1)$, then the procedure call PATH(i, ℓ_i, tr_i) decides whether a solution to the exploration problem for the remaining sub-segment $[\ell_i, 1]$ is possible via canonical trajectories. It does so by greedily deploying agent i to use the least amount of energy to cover at least the segment $[\ell_i, s_i]$: if this does not lead to energy deficit, then the trajectory of agent r_i is allowed to cover as much of the segment $[s_i, s_{i+1}]$ as it can, which determines the position ℓ_{i+1}.

Procedure PATH(i, ℓ_i, tr_i)

if $tr_i \leq 0$ **then**
  if $e_i < s_i - \ell_i + tr_i$ **then** ▷ **Case 1.1** (deficit increases)
    # r_i waits to receive energy $|tr_{i+1}|$ from r_{i+1}. Trajectory t_i is from s_i to ℓ_i. Transfer $|tr_i|$ of energy to r_{i-1}
    $\ell_{i+1} \leftarrow s_i$
    $tr_{i+1} \leftarrow e_i + tr_i - (s_i - \ell_i)$
  else if $e_i \geq s_i - \ell_i + tr_i$ **then** ▷ **Case 1.2** (deficit eliminated)
    # Using the values of ℓ_i, s_i and $e_i - |tr_i|$, select a canonical trajectory t_i originating in s_i, located between ℓ_i and s_{i+1} as in **Cases 1.2.1** to **1.2.3**. Transfer $|tr_i|$ of energy to r_{i-1}
    $tr_{i+1} \leftarrow e_i + tr_i - length(t_i)$
    $\ell_{i+1} \leftarrow right\_endpoint(t_i)$
else ▷ **Case 2**: (Energy surplus)
  # A surplus of energy at ℓ_i implies that $\ell_i = s_i$. The trajectory t_i of r_i is "from s_i to the right, but at most to s_{i-1}".
  $\ell_{i+1} \leftarrow s_i + \min\{s_{i+1} - s_i, s_i + e_i + tr_i\}$
  $tr_{i+1} \leftarrow tr_i + e_i - (\min\{s_{i+1} - s_i, s_i + e_i + tr_i\} - s_i))$
if $(i < k)$ **then** return PATH$(i+1, \ell_{i+1}, tr_{i+1})$
else if $(\ell_{k+1} < 1)$ or $(tr_{k+1} < 0)$ **then** ▷ Insufficient energy
  return with **solvable** $\leftarrow$ false
else
  return with **solvable** $\leftarrow$ true

Then, a recursive call to PATH is made with arguments $i+1$, ℓ_{i+1} and the resulting energy deficit or surplus tr_{i+1}. For an easier understanding of the algorithm, the description below is annotated in detail and Figure 1 provides an example for each case encountered in the algorithm.

We use the following lemma to simplify the proof of the main theorem of this section.

Lemma 2. *Let $t_1, t_2, \ldots, t_{i-1}$ be the trajectories and ℓ_i, tr_i be the values established by Procedure* PATH *after $i-1$ recursive calls, $0 \leq i \leq k+1$. These trajectories explore the segment $[0, \ell_i]$ using in total energy $(\sum_{j=1}^{i-1} e_j) - tr_i$ and this is the minimum energy required by the agents to explore the segment $[0, \ell_i]$.*

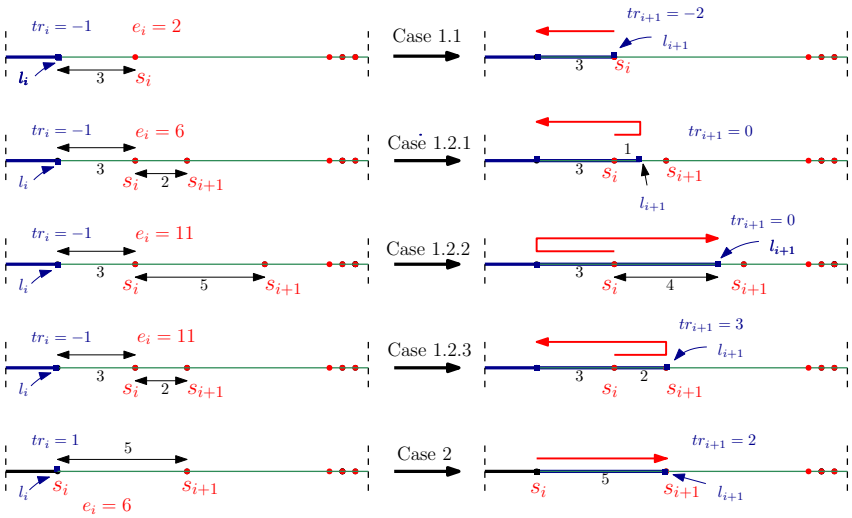

Fig. 1. Assignment of a trajectory to agent r_i by Procedure PATH. The trajectory established in each case is in red on the right part of the figure. Cases 1.1 to 1.2.3 deal with an energy deficit prior to an assignment of a trajectory to r_i, and Case 2 deals with a surplus of energy. In Cases 1.2.1 to 1.2.3 the deficit is eliminated.

Furthermore, if $tr_i < 0$ *(deficit) then* $\ell_i = s_{i-1}$*, if* $tr_i > 0$ *(surplus) then* $\ell_i = s_i$*, and if* $tr_i = 0$ *then* $s_{i-1} \leq \ell_i \leq s_i$*.*

Proof. The proof is done by induction on i and omitted for lack of space. □

Theorem 1. *Assume we are given mobile agents* $r_1, r_2, \ldots, r_k$ *with energies* $e_1, e_2, \ldots, e_k$*, located initially in positions* $s_1 \leq s_2 \leq \cdots \leq s_k$ *of a line segment* $[0, 1]$ *respectively. Let* r_{k+1} *be an additional "dummy agent" at position 1 with zero energy. Then the procedure call* PATH$(1, 0, 0)$ *on this instance runs in* $O(k)$ *time and terminates with* ***solvable*** *being true if and only if there are trajectories of agents that explore the line segment* $[0, 1]$*.*

Proof. It is clear from the description of the algorithm that it is linear in k. When the algorithm terminates with **solvable** being true, it is straightforward to see that a schedule can be determined for the agents, that creates a *valid* trajectory for each agent. To wit, in Round 1 all agents that receive no energy follow their trajectory and do the energy transfers as calculated. Notice that energy is received by agents when in their initial positions.
In Round i all agents that receive energy in Round $i - 1$ follow their trajectory and do the energy transfers as calculated. Therefore, if the algorithm terminates with **solvable** being true, the exploration of segment $[0, 1]$ is possible, and our algorithm returns valid trajectories for the agents to achieve this coverage. Thus we only need to show that the exploration of the segment $[0, 1]$ is not possible when the algorithm terminates with **solvable** being false. In fact, **solvable** is

set to false in the algorithm only if either $tr_{k+1} < 0$ or $\ell_{k+1} < 1$. By Lemma 2, if $tr_{k+1} < 0$ then we can cover the segment up to $l_{k+1} = s_k \leq 1$ but we need more than the given $(\sum_{j=1}^{k} e_j)$ in energy. If $tr_{k+1} = 0$ and $\ell_{k+1} < 1$ then $[0, \ell_{k+1}]$ is the maximum segment that can be explored by the agents using $(\sum_{j=1}^{k} e_j)$ energy. In both cases, the exploration of the entire segment $[0, 1]$ is impossible with the given initial positions and energies. □

Consider the case when we apply our algorithm to an input instance with the sum of energies $\sum_1^k e_i \geq \frac{3}{2}$. Since in each trajectory the part covered doubly is less or equal the part covered singly, the energy deficit/surplus tr_{k+1} obtained after assigning a trajectory to agent r_k is at most $\sum_1^k e_i - \frac{3}{2}$, and it cannot be negative. Also ℓ_{k+1} cannot be less than 1 since there would be an unused surplus of energy of at least $\sum_1^k e_i - \frac{3\ell_{k+1}}{2} > 0$. Thus with $\sum_1^k e_i \geq \frac{3}{2}$ the algorithm terminates with valid exploration trajectories for the segment.

On the other hand when the input instance consists of a single agent r_1 located at point 0.5, the energy needed by r_1 to cover the segment $[0, 1]$ is equal to $\frac{3}{2}$.

Corollary 1. *The segment* $[0, 1]$ *can always be explored by* k *agents with canonical trajectories if the sum of their initial energies is at least* $\frac{3}{2}$*, but exploration may be impossible in some instances if the sum is less than* $\frac{3}{2}$*.*

Remark 1. If agents *cannot share energy*, regardless of k, exploration is impossible in an input instance where all k agents are co-located at 0, each with energy equal to $1-\epsilon$, which gives total energy greater than $k-1$. Thus, without energy-sharing, there is no upper bound on the total energy of agents that guarantees exploration of a path. We have also constructed a linear algorithm for the exploration of a path by agents that cannot share energy, however we cannot include it in this paper due to the limit on the number of pages.

3 Exploring a Tree

In this section we consider a restricted case of the graph exploration problem, specifically when the graph is a tree. First we observe that, *without energy exchange*, there is a straightforward reduction from the partition problem showing that the exploration is NP-hard even on a star graphs: Given an instance of the partition problem $S = \{a_1, a_2, \ldots, a_n\}$, let $T = \sum_{i=1}^{n} a_i/2$. We construct a star graph, with $n + 2$ edges incident on the central node. Of these, n edges have weight $a_1/2, a_2/2, \ldots, a_n/2$ respectively, and two additional edges each have cost T. Assume two agents are at the central node of the star graph with energy $3T/2$ each. Then there is a partition of the set S if and only if the there is an exploration strategy (without energy sharing) for the two agents on the star graph. However, for energy-sharing agents located on a tree we derive below a polynomial exploration algorithm (see also [21]).

Let T be an edge-weighted tree with k agents distributed across the n nodes of T with possibly several agents per node each of them with some non-negative

energy. To simplify the design of our algorithm, we first preprocess the tree to transform it to a rooted binary tree where all the agents are located only at the leaves of the tree. We obtain such a tree from the initial tree T in four stages: (a) by taking all the agents at every non-leaf vertex v and shifting them to a new leaf node l_v that is added to the tree and connected to v via a zero-weight edge, (b) by repeatedly splitting vertices of degree more than 3 into trees of maximum degree 3 using zero-weight edges, (c) by collapsing any path with internal vertices of degree 2 into an edge whose weight equals the cumulative weight of the path, and (d) by converting the resulting 3-regular unrooted tree into a rooted one by splitting one of its edges and making its midpoint be the root of the tree. Without loss of generality, we will denote this new, rooted tree as T. We note these preprocessing steps have complexity $O(n+k)$. Our problem can be stated as follows:

Problem 2 (Tree Exploration). Let T be a rooted, edge-weighted binary tree obtained by preprocessing an unrooted edge-weighted tree with k agents at its nodes, so that all the agents are now located at leaves in T and have their given initial energies. For every node v, let a_v be the initial number of agents inside subtree T_v and let e_v be the sum of their initial energies. Let $w_e \geq 0$ be the weight of edge e. If possible, find a set of valid trajectories for the agents that explore every edge of T using only the given initial energies.

Now, consider any *feasible* exploration for the tree witnessed by a set of trajectories for the agents. The successful exploration of a subtree T_v may either have necessitated additional energy brought into the subtree from outside, or there may be a surplus of energy that could have gone out of the subtree to explore other parts of the tree. Also, exploration may have needed a transfer of agents into the subtree (over and above its a_v agents) or may have been accomplished with some agents made available to leave the subtree and contribute to the exploration of the rest of the tree.

We formalize this idea as follows. Let $B[v,i]$ denote the *maximum* possible total *surplus energy* that can *leave* the subtree T_v after it is fully explored so that i agents can *leave* the subtree with this total energy. Note that i counts only the *balance* of agents that depart from the tree, not individual arrivals and departures. Thus, we allow for i being negative (i.e., i agents enter the the tree on balance), or $B[v,i]$ being negative (i.e., $-B[v,i]$ amount of energy is needed to be brought in from *outside* T_v to explore it fully). We remark that:

(i) when $i \leq 0$ and $B[v,i] \geq 0$, it means that an agent carrying excess energy *must* leave T_v but nevertheless, the overall balance of agents entering/leaving T_v is non-positive.
(ii) the $B[v,i]$ values do not take into consideration the energy expenditure that would be required to explore the edge from v to its parent node in the tree.
(iii) for node v, the value of i can only be in the interval $[-k+a_v,\ a_v]$, because, on balance, at most a_v agents can leave T_v and at most $k-a_v$ can enter it.

In order to simplify the calculation of $B[]$, we extend the definition of $B[]$ to edges as well: Let $e=(u,v)$ where u is the parent of v. As described above,

$B[v, *]$ denotes the surplus energy leaving subtree T_v. $B[e, i]$ will denote the surplus energy available at u *along edge* e with a balance of i agents that could transit through u from the direction of v. We observe that agents *do* spend energy traversing e itself and can also stop in the middle of e, and hence the values $B[e, *]$ and $B[v, *]$ are different. Since node u has exactly two child edges below it, we can compute the $B[u, *]$ values by suitably combining the $B[]$ values of these edges.

It remains to show how to calculate $B[v, *]$ and $B[e, *]$ for each v and e. In principle, we can consider all the possibilities of what the agents can do, but in reality it is enough to consider only the best possible activity with the desired balance of agents. The calculation is performed by calling procedures HANDLEVERTEX and HANDLEEDGE(described below in pseudocode) respectively for each vertex and for each edge of T. The computation proceeds in a bottom-up manner starting from leaves, with the boolean arrays **done**$[v]$ and **done**$[e]$ being used to ensure this flow. In order to simplify the presentation in HANDLEEDGE, we assume that the assignment $B[e, i] \leftarrow y$ is shorthand for $B[e, i] \leftarrow \max(B[e, i],\ y)$ (with the initial values $B[e, *]$ being initialized to $-\infty$).

Our main result in this section is the following:

Theorem 2. *After transforming an unrooted tree with a specified distribution of initial agent locations and energies, procedure* HANDLEVERTEX*, when applied to the root of the resulting rooted binary tree, correctly solves the tree exploration problem for the original tree in* $O(n+\ell k^2)$ *time. It does so by correctly computing the optimal* $B[v, i]$ *values for every vertex* v *and all relevant* i *values for that vertex, in conjunction with procedure* HANDLEEDGE *that correctly computes the optimal* $B[e, i]$ *for every edge* e *and all relevant* i *values for that edge.*

Procedure HANDLEVERTEX(v)

```
1: if v is a leaf then
2:     for i = −k + a_v to a_v do
3:         B[v, i] ← e_v
4: else                                   ▷ v has two child edges e′ and e″
5:     wait until done[e′] and done[e″]
6:     for i = −k + a_v to a_v do
7:         B[v, i] ← max_{i′+i″=i}(B[e′, i′] + B[e″, i″])
8: done[v] ←true
9: if v is the root then
10:    if there is B[v, i] ≥ 0 with i ≥ 0 then
11:        return solvable
12:    else
13:        return not solvable
```

The proof of Theorem 2 hinges on an inductive argument that shows that the $B[]$ values are correctly computed in a bottom-up manner starting at the leaves

Procedure HANDLEEDGE(e)

Require: $e = (u, v)$ where v is a child of u

1: wait until done[v]
2: **for** $i = -k + a_v$ to a_v **do**
3: **if** $B[v, i] \leq 0$ **then**
4: **if** $i < 0$ **then**
5: $B[e, i] \leftarrow B[v, i] - |i|w_e$ ▷ Case 1a
6: **else**
7: $B[e, i] \leftarrow B[v, i] - (i + 2)w_e$ ▷ Cases 2a, 3a and 4a
8: **else if** $i \leq 0$ **then** ▷ $B[v, i] > 0$
9: **if** $B[v, i] > (2 + |i|)w_e$ **then**
10: $B[e, i] \leftarrow B[v, i] - (2 + |i|)w_e$ ▷ Cases 1c and 2c
11: **else if** $i < 0$ **then**
12: $B[e, i] \leftarrow i(w_e - B[v, i]/(2 + |i|))$ ▷ Case 1b
13: **else** ▷ $i = 0$
14: $B[e, -1] \leftarrow (B[v, 0] - 2w_e)/2$ ▷ Case 2b
15: $B[e, 0] \leftarrow B[v, 0] - 2w_e$ ▷ Case 2b'
16: **else** ▷ $i > 0$ and $B[v, i] > 0$
17: **if** $B[v, i] \geq iw_e$ **then**
18: $B[e, i] \leftarrow B[v, i] - iw_e$ ▷ Cases 3c and 4c
19: **else** ▷ $i > 0$ and $B[v, i] < iw_e$
20: $B[e, i] \leftarrow -(i + 2)(w_e - B[v, i]/i)$ ▷ Cases 3b" and 4b
21: **if** $i = 1$ **then**
22: $B[e, -1] \leftarrow B[v, 1] - w_e$ ▷ Case 3b
23: $B[e, 0] \leftarrow 2(B[v, 1] - w_e)$ ▷ Case 3b'
24: done[e] ←**true**

and working our way up the tree. The base case for the induction is for the leaf nodes, and follows directly from the construction (line 3 of HANDLEVERTEX: all the energy in a leaf node is surplus and can be utilized to explore the rest of the tree).

Our induction hypothesis is established by proving two concomitant lemmas.

Lemma 3. *Let $e = (u, v)$ be an edge in T with u being the parent of v. If the values $B[v, *]$ have been correctly computed, then procedure* HANDLEEDGE *correctly computes $B[e, *]$, where $*$ stands for all relevant values of i.*

Lemma 4. *Let v be an internal vertex with two child edges e' and e''. If $B[e', *]$ and $B[e'', *]$ have been correctly computed, then procedure* HANDLEVERTEX *correctly computes $B[v, *]$ where $*$ stands for all relevant values of i.*

It is easy to see after the $O(n + k)$ time preprocessing step to convert the original tree into a full binary tree, each leaf of the tree and subsequently, each

edge of the tree can be processed in $O(k)$ time. To obtain the $B[]$ values for any internal node, we need to combine the k-vectors associated with the child edges (in step 7 of HandleVertex). Observe that there are two classes of internal nodes in the converted tree. The first class correspond to $O(k)$ nodes that contained a subset of agents in the original tree. Each such node, generated in stage (a) of the conversion phase, has a child that is a leaf in the converted tree (containing the same subset of agents). The second class contains the internal nodes generated in stage (b) of the conversion phase, as well as the nodes that were of degree at least 3 in the original tree, and the root obtained in stage (d). Observe that there are at most ℓ nodes in the second class. Let v be an internal node of the first class and let v' be its child that was added in preprocessing that has taken the v's agents. By construction, all the $B[v', *]$ values are equal to $e_{v'}$, and since the weight of e' is 0, $B[e', *] = e_{v'}$ as well. Hence, for such a node v we have $B[v, a_{e'} + a_{e''} - i] = e_{v'} + \max_{j=0}^{i} B[e'', a_{e''} - i]$ which can be for $i = 0$ to k computed in time $O(k)$. Consequently, the overall complexity of HandleVertex called for all $O(k)$ internal nodes of class one is $O(k^2)$. On the other hand, for each of $O(\ell)$ internal nodes from class two, the max function from step 7 of HandleVertex may be computed in $O(k)$ time, which leads to $O(\ell k^2)$ overall complexity of all the calls of the HandleVertex procedure for nodes of the second class. Applying Lemma 4 to the root of the tree, it is clear that the algorithm correctly decides whether or not the binary tree can be explored fully, and the computation takes $O(\ell k^2)$ time after the preprocessing step.

We further remark that if exploration is indeed feasible then in the same time complexity, standard post-processing, top-down techniques can be used to recover the trajectories (and the schedules) for the agents from the computed $B[]$ values by combining the schedules locally computed at each vertex and edge. The choice of the root vertex is arbitrary (after preprocessing) and does not influence the decision outcome – however, it does influence the computed schedule and the amount of energy left in the root, and there may be multiple feasible solutions (one for each $i \geq 0$ in the root's $B[]$ values). This completes the proof of Theorem 2.

4 General Graphs

Unfortunately, while the exploration problem for segments and trees admits very efficient solutions, for general graphs, exploration becomes intractable (unless P=NP). Indeed, we show that the graph exploration problem is NP-hard even in the case of 3-regular graphs by using a reduction from the Hamiltonian cycle problem. We also give an approximation algorithm.

4.1 NP hardness for 3-regular graphs

Let G be a 3-regular graph on n nodes. We construct a graph M by replacing each edge $e = (u, v)$ of G by a meta-edge gadget $m(u, v)$ from Figure 2 case a),

where a and b are chosen so that $a > 5nb$ where n is the number of vertices of G. In addition, each meta-vertex (i.e., an image $m(v)$ of a vertex v of G) starts with one agent and $3a + 5b$ energy.

Note that the overall energy is $(3a + 5b)n$, while the overall weight of the edges ($3n/2$ of them in a 3-regular graphs) is $3n(a + b)$. Hence, a length at most $2bn$ can be crossed twice. As $a > 5bn$, this means no a-edge is crossed twice and at most $2n$ of b-edges are crossed twice.

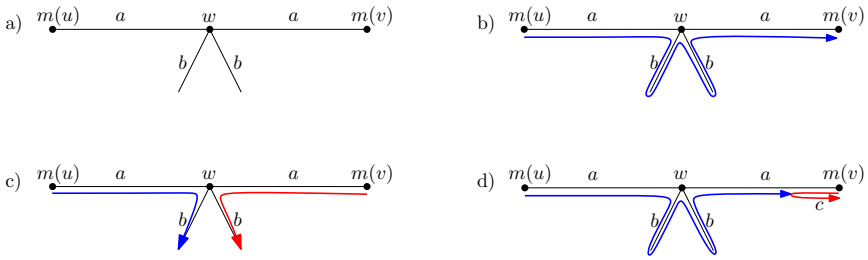

Fig. 2. a) the meta-edge gadget, b) covering gadget using one agent c) efficient covering of the gadget using two agents d) covering gadget using two agents so that at least one agent exits the gadget

Lemma 5. *If only one agent x enters (w.l.o.g. from $m(u)$) a meta-edge $e = m(u, v)$, a total of $2a + 4b$ energy is spent ensuring e is fully explored. In such case x ends up in $m(v)$.*

Lemma 6. *Assume two agents x and y enter a meta-edge $e = m(u, v)$ from $m(u)$ and $m(v)$, respectively, and ensure e is fully explored. If no agent leaves e then the total energy spent in e is at least $2a + 2b$. If one or both agents leave e then all the leaving agents leave to the same meta-vertex, and the total energy spent in e is more than $2a + 4b$.*

Note that the second case of Lemma 6 is worse than Lemma 5 in terms of total energy expenditure, due to the extra cost of c (or $2c$). Still, it might be justified if the blue agent does not have enough energy for case b), or if the red agent does not have enough energy for case c).

Lets call a meta-edge *light* if it is covered by the first part of Lemma 6 (Figure 2, case c)), otherwise it is *heavy*. Observe that each light meta-edge consumes 2 agents, while each heavy meta-edge consumes an excess of $2b$ energy compared to the weight of its edges. This yields:

Lemma 7. *If there is an exploration strategy for the input graph, the number of heavy edges is exactly n.*

By Lemma 5 and the second part of Lemma 6, a heavy meta-edge is traversed in one direction – lets call the halves *outgoing* and *incoming*.

Consider the directed graph $H = (V, E')$ formed by the heavy meta-edges, i.e., $e = (u, v) \in E'$ iff e is a heavy meta-edge from $m(u)$ to $m(v)$.

Lemma 8. *Each vertex of H has at least one incoming edge.*

Proof. As both light and outgoing edges consume agents, if v had no incoming edge, it would need 3 agents. Since each vertex starts with 1 agent, it is not possible for v to have 3 agents without an agent coming via an incoming edge. □

Because the number of heavy edges is exactly n, the heavy edges form a vertex-disjoint (vertex) cycle cover of H, i.e., each meta-vertex has one incoming, one outgoing and one light edge.

The problem is that the disjoint cycle cover is solvable in polynomial time. Hence, we need to modify the input so that there is a solution to the exploration problem if and only if the heavy edges form a single cycle of length n.

This is done by modifying the input into I as follows:

- the three edges incident to the initial vertex have weights adjusted to $a + \epsilon n$ for some small $\epsilon < b/3n$,
- the energy at the initial vertex is $3a + 5b + (2n)\epsilon$
- the energy at all other meta-vertices is $3a + 5b + \epsilon$

Lemma 9. *If the graph G is Hamiltonian then there is an exploration solution to the modified input I.*

Proof. Select the direction of the Hamiltonian cycle, its edges will be the heavy meta-edges. The *explorer* agent starting in the initial vertex takes all the energy available there and follows the Hamiltonian cycle. It collects $2a + 4b + \epsilon$ energy in each of the meta-vertices it crosses, while spending $2a + 4b$ on each heavy meta-edge. The agents located at other meta-vertices wake-up when the explorer arrives, take $a + b$ energy and explore half of the incident light meta-edge.

Note that when the explorer reaches i-th meta-vertex (not counting the initiator), it has $a + b + \epsilon n + i - 1$) energy remaining, except when it returns to the initiator, when it has only $a + b + \epsilon n$ energy as the last meta-edge it crossed has an extra ϵn cost. This is just sufficient to cover half of the incident light meta-edge, which is the last part not yet covered. □

Theorem 3. *The exploration problem is NP-hard for 3-regular graphs.*

Proof. Suppose there is an exploration solution for I. We claim that then the graph G is Hamiltonian. Note that since the sum of all ϵ is less than b, Lemma 7 and Lemma 8 still hold. Hence, the only way meta-edges incident to the starting vertex can be covered is if the agent returning to the starting vertex carries $a + b + \epsilon n$ energy. As the agent can gain only ϵ energy in each vertex it crosses (the remainder is used-up on crossing the heavy meta-edge and covering half of the incident light meta-edge), it needs to visit all n meta-vertices in order to collect sufficient energy, i.e., it has performed Hamiltonian cycle. The theorem then follows since the Hamiltonian cycle problem for 3-regular graphs is known to be NP-complete. □

4.2 An Approximation Algorithm for General Graphs

Even though the graph exploration problem is NP-hard, it is still possible to obtain an efficient approximation algorithm for exploring arbitrary graphs that has an energy-competitive ratio at most 2. Specifically, the algorithm uses at most twice as much energy as the cumulative sum of the edge weights of the graph. First we state without proof a well-known result for agents on a cycle for which the reader is referred to [25][Paragraph 3, Problem 21].

Lemma 10. *For any cycle and any initial positions of the agents there is an algorithm which explores the cycle if and only if the given sum of the energies of the agents is not less than the length of the cycle.*

Lemma 10 has some important consequences. Recall that a graph G is Eulerian if it is connected and all its vertices have even degrees.

Theorem 4. *For any Eulerian graph and any initial positions of the agents there is an algorithm which explores the graph provided that the sum of the energies of the agents at the start is not less than the sum of the edges of the given graph. Moreover, the algorithm has optimal energy consumption.*

Proof. Assume we have agents in such a graph so that the sum of the energies of the agents is at least equal to the sum of the length of the edges of the graph. Since the graph is Eulerian we can construct a cycle which traverses all the edges of the graph exactly once. By Lemma 10 there is an algorithm which assigns trajectories to the given sequence of agents and covers the entire graph. This proves Theorem 4. □

Theorem 5. *Any graph can be explored by energy-sharing agents if the sum of their initial energies is at least twice the sum of edge weights. Moreover, this constant* 2 *cannot be improved even for trees.*

Proof. The original graph, say $G = (V, E)$, is not necessarily Eulerian. However, by doubling the edges of the graph we generate an Eulerian graph $G' = (V, E')$. The sum of the weights of the edges of G' is equal to twice the sum of the weights of the edges of G. Theorem 4 now proves that the graph G can be explored if the sum of their initial energies is at least twice the sum of edge weights in the graph.

Consider a star graph with $2k$ leaves, all edges are of weight 1, for a total weight of $2k$. Assume that we have k agents in the leaves of the star, and one agent in the center of the star. Agents in the leaves have energy 0, and the agent in the center $4k - 1$. To explore the graph the center agent can traverse $2k - 1$ edges of the star twice and one edge once. It is easy to see that no other strategy can do it with less energy. Thus the energy of the middle agent cannot be any lower, and asymptotically, total energy in this instance equal to $4k - 1$ approaches the double of the cost of the edges. □

Remark 2. An improvement of the competitive ratio 2 of Theorem 5 is possible in specific cases by using a Chinese Postman Tour [24] (also known as Route Inspection Problem). Namely, in polynomial time we can compute the minimum sum of edges that have to be duplicated so as to make the graph Eulerian, and this additional sum of energies is sufficient for the exploration.

Assume we have a fixed configuration C of k agents $r_1, r_2, \ldots, r_k$ in a given graph G. We say that energy assignment $E = e_1, e_2, \ldots, e_k$ to agents in configuration C is *minimal* if exploration is possible with energies in E, but impossible when the energy level of any one agent is decreased. Let $|E| = \sum_{i=1}^{k} e_i$. We now investigate how large the ratio $|E_1|/|E_2|$ can be for two minimal assignments E_1 and E_2 of a given configuration C,

Claim. For any configuration of agent C and any two minimal energy assignments E_1 and E_2 the ratio $|E_1|/|E_2| \leq 2$, and this is asymptotically optimal.

5 Conclusion

We studied graph exploration by a group of mobile agents which can share energy resources when they are co-located. We focused on the problem of deciding whether or not it is possible to find trajectories for a group of agents initially placed in arbitrary positions with initial energies so as to explore the given weighted graph. The problem was shown to be NP-hard for 3-regular graphs while for general graphs it is possible to obtain an efficient approximation algorithm that has an energy-competitive ratio at most 2 (and this is shown to be asymptotically optimal). We also gave efficient algorithms for the decision problem for paths, trees, and Eulerian graphs. The problem considered is versatile and our study holds promising directions for additional research and interesting open problems remain by considering exploration with 1) optimal total energy consumption, 2) agents with limited battery capacity, 3) energy optimal placement of mobile agents, 4) time vs energy consumption tradeoffs for mobile agents with given speeds, 5) additional topologies, as well as 6) combinations of these.

References

1. S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. Computing*, 29(4):1164–1188, 2000.
2. S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32(1):123–143, 2002.
3. S. Alpern and S. Gal. *The theory of search games and rendezvous*. Springer, 2003.
4. R. Baeza Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.
5. R. Baeza-Yates and R. Schott. Parallel searching in the plane. *Comp. Geom.*, 5(3):143–154, 1995.
6. E. Bampas, S. Das, D. Dereniowski, and C. Karousatou. Collaborative delivery by energy-sharing low-power mobile robots. In *Proc. of ALGOSENSORS*, pages 1–12, 2017.

7. A. Bärtschi. *Efficient Delivery with Mobile Agents*. PhD thesis, ETH Zurich, 2017.
8. A. Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, 1964.
9. W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.
10. J. Chalopin, S. Das, and A. Kosowski. Constructing a map of an anonymous graph: Applications of universal sequences. In *Proc. of OPODIS*, pages 119–134. Springer, 2010.
11. J. Chalopin, R. Jacob, M. Mihalák, and P. Widmayer. Data delivery by energy-constrained mobile agents on a line. In *Proc. of ICALP*, pages 423–434. Springer, 2014.
12. J. Czyzowicz, K. Diks, J. Moussi, and W. Rytter. Communication problems for mobile agents exchanging energy. In *Proc. of SIROCCO*, pages 275–288, 2016.
13. J. Czyzowicz, K. Diks, J. Moussi, and W. Rytter. Broadcast with energy-exchanging mobile agents distributed on a tree. In *Proc. of SIROCCO*, pages 209–225, 2018.
14. J. Czyzowicz, K. Diks, J. Moussi, and W. Rytter. Energy-optimal broadcast and exploration in a tree using mobile agents. *Theoretical Computer Science*, 795:362–374, 2019.
15. J. Czyzowicz, S. Dobrev, R. Killick, E. Kranakis, L. Narayanan, J. Opatrny, D. Pankratov, and S. Shende. Graph exploration by energy-sharing mobile agents. *arxiv.org/pdf/2102.13062*, 2021.
16. J. Czyzowicz, K. Georgiou, R. Killick, E. Kranakis, D. Krizanc, M. Lafond, L. Narayanan, J. Opatrny, and S.M. Shende. Energy consumption of group search on a line. In *Proc. of ICALP*, pages 137:1–137:15, 2019.
17. X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proc. of FOCS*, pages 298–303. IEEE Computer Society, 1991.
18. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *J. of Graph Theory*, 32(3):265–297, 1999.
19. D. Dereniowski, Y. Disser, A. Kosowski, D. Pajak, and P. Uznański. Fast collaborative graph exploration. *Information and Computation*, 243:37–49, 2015.
20. Miroslaw Dynia, Jakub Łopuszański, and Christian Schindelhauer. Why robots need maps. In *International Colloquium on Structural Information and Communication Complexity*, pages 41–50. Springer, 2007.
21. P. Fraigniaud, L. Gasieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. *Networks: An International Journal*, 48(3):166–177, 2006.
22. J. M. Kleinberg. On-line search in a simple polygon. In *Proc. of SODA*, pages 8–15, 1994.
23. E. Koutsoupias, C. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. of ICALP*, pages 280–289. Springer, 1996.
24. M.-K. Kwan. Graphic programming using odd or even points. *Acta Mathematica Sinica (MR 0162630. Translated in Chinese Mathematics 1: 273–277, 1962)*, 10:263–266, 1960.
25. L. Lovász. *Combinatorial Problems and Exercises*. Elsevier, 1979.
26. J. Moussi. *Data communication problems using mobile agents exchanging energy*. PhD thesis, Université du Québec en Outaouais, 2018.
27. P. Panaite and A. Pelc. Exploring unknown undirected graphs. *J. of Algorithms*, 33(2):281–295, 1999.