UDC 519.2
DOI: https://doi.org/10.17721/1812-5409.2025/2.33

Andrij STOPKIN, PhD (Phys. & Math.), Assoc. Prof.
ORCID ID: 0000-0002-6130-9920
e-mail: stepkin.andrej@gmail.com
SHEI "Donbas State Pedagogical University", Dnipro, Ukraine

Tetiana TURKA, PhD (Phys. & Math.), Assoc. Prof.
ORCID ID: 0000-0001-6445-2223
e-mail: tvturka@gmail.com
SHEI "Donbas State Pedagogical University", Dnipro, Ukraine

ALGORITHM FOR RECOGNIZING SIMPLE GRAPHS BY A COLLECTIVE OF AGENTS

Abstract. ***These days, automation and robotization across various processes are developing rapidly. Naturally, this trend also extends to exploring environments where human operation is more difficult – or even dangerous – than operating under the same conditions with specialized robotic systems. This makes research on exploring unknown graphs (map construction) by mobile agents particularly relevant. This paper proposes a new algorithm for exploring finite, undirected, simple graphs by two different types of agents. One agent, the researcher, moves through the graph, reads and modifies marks on graph elements, and transmits information about its actions to the other agent – the experimenter. The researcher's operation is based on a depth-first graph traversal method. The researcher agent has finite memory that does not depend on the dimension of the graph. The experimenter agent operates outside the explored graph. Its main task is to build an internal representation of the explored graph (as a list of edges and nodes) based on information received from the researcher agent. The experimenter agent has finite but unboundedly growing memory, the size of which depends on the number of nodes in the explored graph. This paper examines in detail the researcher's modes of operation, indicating the priority of their activation depending on local conditions, and lists the messages exchanged by the agents during the algorithm. The complete algorithm for the experimenter to process the received messages – on the basis of which graph exploration takes place – is also provided. Additionally, the paper analyzes the time, space, and communication complexity of the algorithm and the number of edge traversals performed by the researcher during graph traversal. It is shown that the proposed algorithm has quadratic (from the number of nodes of the explored graph) time, space and communication complexity. The upper estimate of the number of transitions along the edges performed by the researcher agent is estimated as $O(n^2)$, where n is the number of nodes in the graph. For the work of proposed graph exploration algorithm, the researcher agent needs two paints of different colors and one stone.***

K e y w o r d s : ***undirected graphs, graph exploration, agent collective, algorithm complexity, depth-first traversal method.***

AMS 2020 classification: 05C85 68R10.

Introduction

A pressing problem in computer science, which has received considerable attention, is the interaction between a controlling system and a controlled system. Many works over the past decades have been devoted to solving this problem (Dopp, 1971; Dudek et al., 1993; Stepkin, 2015; Nagavarapu et al., 2020), under the assumption that it involves moving mobile agents through some environment toward a specific goal. It is quite clear that an agent's movement in an operational environment is possible only if a complete model of the chosen environment exists. In modeling environments, several approaches have been identified, one of which is topological. Under this approach, a roaming agent has access to information about connections between different regions of the environment but lacks metric and algorithmic information about it. Such a situation often arises in robotics (Dudek, 2024), and the topological model is taken to be a graph, equipped with additional information on edges, vertices, and incidences (points where edges meet vertices of the graph). Typically, three main tasks of environment exploration by an agent are distinguished: (1) agent self-localization within the environment; (2) verification of the model's correspondence to the actual environment; (3) construction by the agent of a map model of the environment. A significant number of papers (Fraigniaud et al., 2004) are devoted to the problem of environment recognition (map model construction), and several recognition algorithms are known, yielding many results on the feasibility and complexity of such recognition using various agent capabilities. However, the problem of graph recognition by teams of agents remains under-investigated, making studies aimed at recognizing (mapping) unknown graphs by various teams of mobile agents especially relevant.

To address the problem of graph recognition by a collective of agents, a number of approaches have been defined and various algorithms proposed for moving a team of agents through a graph and marking its elements with paints or stones, enabling recognition of the target graph up to isomorphism. The first works on graph exploration by an agent swarm appeared in the 1990s (Dudek et al., 1993; Dudek et al., 1998). In these publications, all agents start from a single vertex. They schedule their next meeting time, divide the available edges among themselves, and disperse along them. After a predetermined interval, having traversed part of the graph, the agents return to the starting vertex to merge their partial maps. They then set the next meeting time and again spread out through the graph. This process repeats until a complete map of the explored graph is built. Similar swarm-based algorithms were proposed in (Wang, Jenkin, & Dymond, 2009; Zhang, 2010). By contrast, in (Das et al., 2007) an algorithm is presented in which agents start from different vertices, and communication occurs via tokens that agents leave at graph vertices.

In (Stepkin, 2015), the problem of recognizing finite, simple, undirected graphs by a collective of agents is examined. Two researcher agents move through the graph simultaneously, read and modify labels on graph elements, and send the necessary information to an experimenter agent, which constructs the map of the explored graph in its memory as lists of edges and vertices. For recognition, each moving agent uses two different paints (three colors in total). The method is based on a depth-first search traversal.

In the paper (Nagavarapu, Vachhani & Sinha, 2016), a decentralized approach to map-building of graphs by a collective of agents is proposed, in which agents avoid collisions by using marks placed by agents on previously visited vertices of the graph. In this algorithm, decision-making lies directly with each agent and does not depend on the actions of other agents.

ISSN 2518-1378 (Print), ISSN 2708-6038 (Online)

By contrast, in the paper (Banfi et al., 2018), agents exchange information about completed tasks with a base station via a dedicated network. In other words, to organize coordinated action among agents, it is first necessary to establish a network that meets certain conditions.

The object of our research is the problem of constructing maps of the explored graphs by a collective of mobile agents.

The goal of the research is to develop a new algorithm for recognizing simple undirected graphs using two agents, serving as a base algorithm for further scaling to a larger number of agents traversing the graph.

1. Main Results

We consider connected, finite, undirected simple graphs (without loops or multiple edges). Let $G = (V, E)$ – be a graph with vertex set V and edge set E. An edge is a two-element subset $\{u, v\}$ of V; its endpoints u, v are called adjacent vertices, and the edge (u, v) is said to be incident to both u, v. We may denote the edge either as (u, v) or (v, u). A triple $((v, u), u)$ is called the incidence (or point of contact) of edge (v, u) with vertex u. The incidence $((v, u), u)$ is called the far incidence of v, and $((v, u), v)$ the near incidence. Denote by I the set of all such triples. The set $L = V \cup E \cup I$ is called the set of elements of the graph G. A coloring function of the graph G is a surjective map $\mu: L \to \{w, r, b\}$, where w is white, r is red, and b is black. The pair (G, μ) is called a colored graph. A sequence $u_1, u_2, \dots, u_k$ of pairwise adjacent vertices is called a path in G, and k is the length of that path. If $u_1 = u_k$, the path is called a cycle. The neighborhood $Q(v)$ of a vertex v is the set of graph elements consisting of the vertex v, all vertices u adjacent to v, all edges (v, u), and all incidences $\big((v, u), v\big)$ and $\big((v, u), u\big)$. We denote the power of the sets of vertices V and edges E by n and m, respectively. It is clear that $m \le \frac{n(n-1)}{2}$. An isomorphism between graphs G and H is a bijection $\varphi: V_G \to V_H$ such that $(v, u) \in E_G$ if and only if $\big(\varphi(v), \varphi(u)\big) \in E_H$. Thus, isomorphic colored graphs differ only by renaming of vertices and recoloring of their elements.

In this paper, we study a team of two agents: a researcher and an experimenter. The researcher is capable of moving along edges of the graph and recoloring graph elements. The experimenter agent builds and stores the graph map in its memory based on messages received from the researcher. The researcher agent has finite memory that does not depend on the dimension of the graph under investigation; the experimenter agent, in turn, has finite but unboundedly growing internal memory.

At the start of its operation, the researcher agent is placed at an arbitrary vertex of the graph G and paints it in its own color. The experimenter agent at that moment adds a new vertex to the set V_H in its memory. The researcher agent moves through the graph from vertex v to vertex u along edge (v, u), and can change the coloring of vertices v, u, edges (v, u), and incidences $\big((v, u), v\big), \big((v, u), u\big)$. While at vertex v, the researcher agent can perceive the labels of all elements in the neighborhood $Q(v)$, and based on this information determines which edge it will move along next and how it will color the graph's elements. The experimenter agent, in turn, sends, receives, and identifies messages received from the researcher agent and records in its memory the result of the researcher agent's operation at each step. Based on this information, the experimenter agent builds a map of the graph G, initially unknown to the agents, as lists of edges and vertices.

Note that the proposed algorithm has several features: first, the graph G is unknown to the agents; second, the proposed method of recognizing the graph is based on a depth-first search strategy; third, during the traversal of graph G, the agents create an implicit numbering of visited vertices. Based on the constructed numbering, the recognition of graph G occurs by building a graph H isomorphic to G. In the course of traversal, the researcher agent builds an implicit depth-first search tree, all edges of which are divided into those that are colored red upon first traversal and belong to the tree itself, and those that do not belong to the tree and are colored black upon first traversal–back edges. Tree edges during the execution of the algorithm are traversed at least twice, and on the last traversal they are colored black by the agent. Back edges are traversed from one to two times.

Red vertices of graph G at each step of the algorithm form a red path. When moving into a new vertex, the red path is extended; when moving back, it is shortened; when recognizing a back edge, it does not change. A vertex whose all incident edges have been recognized is colored black. The algorithm finishes when the red path becomes empty and all vertices are black.

The operation of the researcher agent can be divided into two modes: Normal mode. The researcher agent moves forward over white vertices, coloring the vertices, the edges connecting those vertices, and the far incidences red. If in the neighborhood of the current vertex there are no white edges or vertices – that is, no possible paths of movement – then the researcher agent returns back, coloring the traversed vertices, edges, and near incidences black. Upon returning to the starting vertex, the researcher agent terminates. At each step, the researcher agent exchanges data with the experimenter agent. Recognition of back edges. If, while moving forward at a vertex v, a back edge is detected, then the researcher agent stops working in normal mode and switches to back-edge recognition mode. The researcher agent leaves a stone in the current vertex. After that, it begins to move backward along its path until it finds the vertex incident to the back edge (this refers to a white edge whose far endpoint is colored red and in which the stone is located) and traverses that edge, coloring it black. Then it returns to the previous vertex and continues moving backward. At this stage, two cases are possible: 1) the researcher agent finds another back edge. In this case, the researcher agent repeats the back-edge recognition procedure. 2) The researcher agent reaches the starting vertex, which means that all marked back edges have been recognized. After this, the agent returns to the end of the red path where the stone is located, picks up the stone, and continues working in normal mode.

Let us examine in more detail the agents' algorithms.

Algorithm of the researcher agent:

1. $\mu(v) := r$;
2. if $\exists (v, u) \in Q(v) | (\mu(v, u) = w)\ and\ (\mu(u) = \mu(v) = r)$ then EXPL_IE(v);
3. else if $\exists (v, u) \in Q(v) | (\mu(v, u) = w)\ and\ (\mu(u) = w)$ then do
4. FORWARD(v);
5. go to 2;
6. end do;
7. else if $\exists (v, u) \in Q(v) | \big(\mu\big((v, u), v\big) = r\big)\ and\ (\mu(v, u) = r)$ then do

8. BACK(v);
9. go to 2;
10. end do;
11. else STOP(v);

EXPL_IE(v):
1. agent leaves the stone $\mu(v) := l$;
2. if $\exists(v,u) \in Q(v) | \big(\mu\big((v,u),v\big) = r\big)\ and\ (\mu(v,u) = r)\ and\ \big(\mu\big((v,u),u\big) = w\big)$
3. then do
4. agent selects $(v,u) \in Q(v) | \big(\mu\big((v,u),v\big) = r\big)\ and\ (\mu(v,u) = r)\ and\ \big(\mu\big((v,u),u\big) = w\big)$ follows it to the vertex u;
5. $v := u$;
6. agent appends RETURN to its message list M;
7. go to 2 of this procedure;
8. end do;
9. if $\exists(v,u) \in Q(v) | (\mu(v) = r)\ and\ (\mu(v,u) = w)\ and\ (\mu(u) = l)$
10. then do
11. agent moves along edge (v,u), coloring: $\mu\big((v,u),v\big) := r, \mu(v,u) := r, \mu\big((v,u),v\big) := r$;
12. $v := u$;
13. agent appends FORWARD_IE to its message list M;
14. agent selects any edge $(v,u) \in Q(v)\ such\ that: \big(\mu\big((v,u),v\big) = r\big)\ and\ (\mu(v,u) = r)\ and\ \big(\mu\big((v,u),u\big) = r\big)$, moves along it to the vertex u, coloring $\big(\mu\big((v,u),v\big) = b\big)\ and\ (\mu(v,u) = b)\ and\ \big(\mu\big((v,u),u\big) = b\big)$;
15. go to 2 of this procedure;
16. end do;
17. else go to 2 of this procedure;
18. if $\exists(v,u) \in Q(v) | \big(\mu\big((v,u),v\big) = w\big)\ and\ (\mu(v,u) = r)\ and\ \big(\mu\big((v,u),u\big) = r\big)$
19. then do
20. agent selects $(v,u) \in Q(v)\ such\ that: \big(\mu\big((v,u),v\big) = w\big)\ and\ (\mu(v,u) = r)\ and\ \big(\mu\big((v,u),u\big) = r\big)$ and moves through it to vertex u;
21. $v := u$;
22. go to 18 of this procedure;
23. end do;
24. else do
25. agent picks up the stone $\mu(v) := r$;
26. agent appends RECOGNIZED_IE to its message list M;
27. end do

During the FORWARD(v) procedure, the researcher agent selects from the neighborhood $Q(v)$ an arbitrary edge (v,u) for which $(\mu(v,u) = w)\ and\ (\mu(u) = w)$, then moves along it to vertex u. In doing so, it colors $(\mu(v,u) := r)\ and\ \big(\mu\big((v,u),u\big) := r\big)\ and\ (\mu(u) := r)$, performs the assignment $v := u$, and appends the message FORWARD to the list M.

During the BACK(v) procedure, the researcher agent selects from the neighborhood $Q(v)$ an edge for which $(\mu(v) = r)\ and\ \big(\mu\big((v,u),v\big) = r\big)\ and\ (\mu(v,u) = r)$, then moves along it to vertex u. In doing so, it colors $\mu(v) := b$, $\mu\big((v,u),v\big) := b$, $\mu(v,u) := b$, performs the assignment $v := u$, and appends the message BACK to the list M.

During the STOP(v) procedure, the researcher agent colors $\mu(v) := b$, appends the message STOP to the list M, and terminates.

Algorithm of the experimenter agent:

Input: list of messages M from the researcher agent.

Output: lists of vertices V_H and edges E_H of the graph H, which is isomorphic to G.

Data: V_H, E_H – lists of vertices and edges of graph H, isomorphic to G. ct – a counter of how many vertices of G have been visited by the researcher agent. M – the list of messages from the researcher agent. i – a counter used to determine the indices of far endpoints of back edges after a stone has been placed at their common vertex. $STOP$ – a variable used by the researcher agent to signal to the experimenter agent that the graph traversal is complete. $r(1) \dots r(t)$ – the list of indices of the vertices on the red path, where t is the length of this list. Mes – a variable that holds the current message.

Algorithm of the experimenter agent:
1. $ct := 1, M := \emptyset, i := 0, E_H := \emptyset, STOP := 0$, t:= 1, r(t):= ct, $V_H := \{1\}, Mes :=$ "";
2. while $STOP = 0$ do
3. if $M \neq \emptyset$,
4. then do
5. read the next message into Mes and remove it from the queue M;
6. LIST_PROCESSING();
7. end do;
8. end do;
9. print V_H, E_H.

LIST_PROCESSING():
1. $if Mes := "FORWARD"; then\ FORWARD();$
2. $if Mes := "BACK"\ then\ BACK();$

3. $if Mes \coloneqq$ "$RETURN$" $then\ RETURN();$
4. $if Mes \coloneqq$ "$FORWARD\_IE$" $then\ FORWARD\_IE();$
5. $if Mes \coloneqq$ "$RECOGNIZED\_IE$" $then\ RECOGNIZED\_IE();$
6. $if Mes \coloneqq$ "$STOP$" $then\ STOP();$

FORWARD():
1. $ct \coloneqq ct + 1;$
2. $t \coloneqq t + 1;$
3. $r(t) \coloneqq ct;$
4. $V_H \coloneqq V_H \cup \{ct\};$
5. $E_H \coloneqq E_H \cup \{r(t-1), r(t)\};$

BACK():
1. Delete the element $r(t)$ from the list $r(1) \dots r(t)$;
2. $t \coloneqq t - 1;$

RETURN():
1. $i \coloneqq i + 1;$

FORWARD_IE():
1. $E_H \coloneqq E_H \cup \{r(t), r(t-i)\};$

RECOGNIZED_IE():
1. $i \coloneqq 0;$

STOP():
1. $STOP \coloneqq 1;$

3. Algorithm Analysis

Lemma 1. *Two agents, while executing the recognition algorithm on a graph, uniquely identify (recognize) all back edges.*

Proof. During the EXPL_IE(v) procedure, the agents recognize back edges. Specifically, when moving forward, if the researcher agent notices a back edge originating from the current vertex, it marks that vertex with a stone l (see line 1 of EXPL_IE(v)) and begins moving backward along the red path. At each step, it sends the message "RETURN" to the experimenter agent (line 6 of EXPL_IE(v)). Based on this "RETURN" message, the experimenter agent's RETURN() procedure counts how many edges of the red path have been traversed backward.

If at some vertex during this backward traversal the researcher agent detects a white edge whose far endpoint is marked by the stone l, then it traverses that edge, coloring the near incidence, the edge itself, and the far incidence all in red (line 11 of EXPL_IE(v)), and sends the message "FORWARD_IE" to the experimenter agent (line 13 of EXPL_IE(v)). Upon receiving "FORWARD_IE," the experimenter agent's FORWARD_IE() procedure determines the previously unknown implicit index of the second endpoint of the back edge. Since the agents already know the implicit index of the vertex where the stone was placed, the experimenter agent can add this back edge to the edge set of graph H using the same procedure.

After handling the back edge, the researcher agent returns to the previous vertex, coloring the near incidence, the edge, and the far incidence through which it traversed all in black (line 14 of EXPL_IE(v)) and continues moving backward along the red path until it reaches its very beginning, thereby recognizing all back edges incident to the vertex marked by the stone.

Having reached the start of the red path, the researcher agent then returns to its end and sends the message "RECOGNIZED_IE" to the experimenter agent (line 26 of EXPL_IE(v)). Upon receiving this message, the experimenter agent's RECOGNIZED_IE() procedure resets its counter of traversals to zero.

Thus, for each back edge incident to the vertex marked by the stone (in fact, this is the vertex at the end of the red path at that stage of the graph traversal), the agents count the number of transitions along vertices of the red path required to reach the second endpoint of the given back edge and uniquely determine its index. This, in turn, allows them to unambiguously identify the back edge and add it to the edge set E_H of graph H. Q.E.D.

Theorem 1. *Two agents, after executing the recognition algorithm on a graph, will recognize any graph up to isomorphism.*

Proof. At the beginning of the recognition algorithm (for $n \geq 3$), each time the researcher agent visits a white vertex of the explored graph G, the procedure FORWARD(v) is invoked at least twice. Each invocation of FORWARD() creates a new vertex in graph H. Thus, the execution of this algorithm induces a mapping $\varphi: V_G \to V_H$ from the vertices of G to the vertices of H. Moreover, the equality $\varphi(v) = ct$ s established when vertex v is colored red.

This mapping naturally establishes an implicit numbering of the vertices of graph G. Moreover, the mapping φ is a bijection because we consider only connected graphs, and in such graphs every vertex is reachable from any starting vertex. Therefore, all vertices are visited by the researcher agent, and upon its first visit, the agents assign it an implicit number.

From the algorithm's description, it is clear that the researcher agent traverses every edge of G, since after the algorithm completes, all edges of the graph are colored black.

When executing the procedure FORWARD(), the experimenter agent recognizes a tree edge (v, u) and numbers vertex u so that the edge in graph G corresponds uniquely to edge $(\varphi(v), \varphi(u))$ in graph H. When executing the procedure FORWARD_IE(), the experimenter agent recognizes a back edge (v, u) of G and associates it uniquely with edge $(\varphi(v), \varphi(u))$ in H. Since all edges of the graph are divided exclusively into tree edges and back edges, it follows that the mapping φ is an isomorphism from G to H. Q.E.D.

Theorem 2. *The time, space, and communication complexities of the recognition algorithm, as well as the number of edge traversals performed by the researcher agent, are bounded above by як $O(n^2)$, where n is the number of vertices in the graph. Two different-colored paints suffice for recognition.*

Proof. Let us examine more closely the properties of the red path. From the algorithm's description, at each step the red path is a simple path that starts at the vertex v with $\varphi(v) = 1$ and ends at the vertex u with $\varphi(u) = ct$. Hence, the length of the red path does not exceed n. Each time the procedures FORWARD(v) and BACK(v) are executed, the researcher agent traverses exactly one edge. In a single run of the back-edge recognition mode, the researcher agent recognizes at most $n - 2$

back edges, during which it: spends one move to place the stone in a vertex; traverses at most $n-1$ edges of the red path twice (once moving backward and once returning to the last red vertex); traverses at most $n-2$ back edges twice while coloring them.

When analyzing the time complexity of the algorithm, we assume that algorithm initialization and the choice of any one of the possible procedures each take some constant amount of time. We also assume that traversing an edge by the researcher agent and processing the command by the experimenter agent – received at that step from the researcher agent – each take time equal to some constant. We further assume that the time to send a single message equals a constant that does not exceed the time it takes for the researcher agent to traverse one edge. Moreover, the total time spent on analyzing the neighborhood $Q(v)$ of vertex v and choosing the necessary edges is bounded above by $O(n^2)$.

Then the time complexity of the algorithm is determined by the following relations:

1. The procedures FORWARD(v) and BACK(v) are each executed no more than $2 \times (n-1)$ times, and the total time for their execution is $O(n)$.

2. The time spent recognizing back edges is $(n-2) \times \big(2 \times (n-1) + 2 \times (n-2)\big)$ steps, i.e. essentially $O(n) \times O(n)$ steps. Therefore the total time for executing the back-edge recognition procedure is $O(n^2)$.

Hence, the upper bound on the number of edge traversals performed by the researcher agent is $O(n) + O(n^2) = O(n^2)$, and the overall time complexity $T(n)$ of the algorithm satisfies $T(n) = O(n^2)$.

The space complexity $S(n)$ of the algorithm is determined by the combined sizes of the lists E_H, V_H, and $r(1),\ldots,r(t)$, whose complexities are $O(n^2)$, $O(n)$, and $O(n)$, respectively.

Thus, the upper bound on the space complexity of the algorithm satisfies $S(n) = O(n^2)$.

When calculating the communication complexity of the algorithm, noting that the researcher agent can send the experimenter agent only one of six possible messages (FORWARD, BACK, RETURN, FORWARD_IE, RECOGNIZED_IE, STOP), we assume that each message occupies one unit of memory.

Since in normal mode, at each step exactly one message (FORWARD, BACK, or STOP) is transmitted, the total volume of information sent in this mode is bounded above by $2 \times O(n)$. In back-edge recognition mode, while the researcher agent moves backward along the red path, it sends one message (RETURN) at each step, so the total volume of information sent in that mode is also bounded above by $n \times O(n)$. At the first traversal of a back edge, one message (FORWARD_IE) is sent, so the total volume of information in that part is also $n \times O(n)$. No messages are sent when returning along a back edge. Finally, one message (RECOGNIZED_IE) is sent when returning to the vertex containing the stone, so that volume is $n \times O(1)$.

Thus, the total volume of information transmitted by the agents is
$K(n) = 2 \times O(n) + n \times (O(n) + O(n) + O(1)) = O(n^2)$ Q.E.D.

Discussion and conclusions

The paper proposes a new algorithm for recognizing finite, undirected, simple graphs (without loops or multiple edges) using two agents. One agent – the researcher agent – traverses the graph, reads and updates labels on graph elements, and transmits information about its actions to the other agent, the experimenter agent. The researcher's operation is based on a depth-first search traversal. The researcher has finite memory that does not depend on the graph's dimension. The second agent – the experimenter – operates outside the explored graph. Its primary task is to construct an internal representation of the graph in its memory based on information received from the researcher. The experimenter's memory is finite at each step but unbounded overall, growing with the number of vertices in the graph.

The paper also analyzes the time, space, and communication complexities of the proposed algorithm and examines the number of edge traversals performed by the researcher during the graph traversal. It is shown that the algorithm's time, space, and communication complexities are all quadratic in the number of vertices n. The upper bound on the number of edge traversals by the researcher is $O(n^2)$. To execute this graph-recognition algorithm, the researcher needs only two different-colored paints and one stone.

The results of this study are planned to be scaled up to a larger number of agents to develop more efficient graph-recognition algorithms.

Authors' contributions: Andrij Stopkin – conceptualization, formal analysis, methodology, drafting of the original manuscript, source analysis, preparation of the literature/theoretical background review; Tetiana Turka – data validation, writing – review and editing.

Acknowledgment. The authors is grateful to the referee for comments and suggestions that improved the paper.

Sources of funding. This study did not receive any grant from a funding institution in the public, commercial, or non-commercial sectors.

References

Banfi, J., Quattrini Li, A., Rekleitis, I., Amigoni, F., & Basilico, N. (2018). Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *Autonomous Robots*, 42, 875–894. https://doi.org/10.1007/s10514-017-9652-y

Das, S., Flocchini, P., Kutten, S., Nayak, A., & Santoro, N. (2007). Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1–3), 34–48. https://doi.org/10.1016/j.tcs.2007.05.011

Dopp, K. (1971). Automaten in labirinthen. *Electronische Informationsverarbeitung und Kybernetik*, 7(2), 79–94.

Dudek, G., & Jenkin, M. (2024). *Computational principles of mobile robotics* (3rd ed.). Cambridge University Press. https://doi.org/10.1017/9781108682404

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1993). Map validation in a graphlike world. *International Joint Conference on Artifical Intelligence: Proceedings of the 13th International Conference, Chambery (France), San Fransisco (USA),* (pp. 1648–1653).

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1998). Topological exploration with multiple robots. *Robotics with Application (ISORA): Proceedings of the 7th International Symposium, Alaska.*

Fraigniaud, P., Jecincas, D., Peer, G., Pelc, A., & Peleg, D. (2004). Graph Exploration by a finite automaton. *Proceedings of the 29th International Symposium on Mathematical Foundation of Computer Science (MFCS)*, (pp. 451–462).

Nagavarapu, S. C., Vachhani, L., Sinha, A., & Buriuly, S. (2020). Generalizing Multi-agent Graph Exploration Techniques. *International Journal of Control, Automation and Systems,* 19, 491–504. Springer. https://doi.org/10.1007/s12555-019-0067-8

Nagavarapu, S. C., Vachhani, L. & Sinha, A. (2016). Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach. *Journal of Intelligent & Robotic Systems*, 83, 503–523. Springer. https://doi.org/10.1007/s10846-015-0309-9

Stepkin, A. (2015). Using a Collective of Agents for Exploration of Undirected Graphs. *Cybernetics and Systems Analysis*, 51(2), 223–233. https://doi.org/10.1007/s10559-015-9715-z

Wang, H., Jenkin, M., & Dymond, P. (2009). It can be beneficial to be 'lazy' when exploring graph-like worlds with multiple robots. *Advances in Computer Science and Engineering (ACSE): In Proceedings of the IASTED International Conference*, (pp. 55–60).

Zhang, C. (2010). *Parallelizing Depth-First Search for Robotic Graph Exploration*. Harvard University.

Отримано редакцією журналу / Received: 16.06.25
Прорецензовано / Revised: 19.07.25
Схвалено до друку / Accepted: 10.10.25

Андрій СТЬОПКІН, канд. фіз.-мат. наук, доц.
ORCID ID: 0000-0002-6130-9920
e-mail: stepkin.andrej@gmail.com
ДВНЗ "Донбаський державний педагогічний університет", Дніпро, Україна

Тетяна ТУРКА, канд. фіз.-мат. наук, доц.
ORCID ID: 0000-0001-6445-2223
e-mail: tvturka@gmail.com
ДВНЗ "Донбаський державний педагогічний університет", Дніпро, Україна

АЛГОРИТМ РОЗПІЗНАВАННЯ ПРОСТИХ ГРАФІВ КОЛЕКТИВОМ АГЕНТІВ

Нині досить динамічно розвиваються напрямки автоматизації та роботизації різних процесів. Звісно, це стосується і дослідження різноманітних середовищ, де організація роботи людини є складнішою або навіть небезпечнішою, ніж організація роботи в тих самих умовах спеціальних роботизованих систем. Це робить актуальними дослідження, спрямовані на розпізнавання (побудову мапи) невідомих графів мобільними агентами. У представленій роботі запропоновано новий алгоритм розпізнавання скінчених, неорієнтованих графів без петель і кратних ребер за допомогою двох агентів різного типу. Один з агентів – агент-дослідник, що пересувається по графу, зчитує та змінює мітки елементів графа і передає інформацію про свої дії іншому агенту – агенту-експериментатору. Робота агента-дослідника базується на методі обходу графа у глибину. Агент-дослідник має скінчену пам'ять, що не залежить від розмірності графа. Другий агент – це агент-експериментатор, що перебуває поза межами досліджуваного графа. Головним завданням цього агента є побудова представлення досліджуваного графа (у вигляді переліку ребер і вершин) у своїй пам'яті на основі інформації, яку він отримує від агента-дослідника. Агент-експериментатор має скінчену внутрішню пам'ять, що необмежено зростає, розмірність якої залежить від кількості вершин графа над розпізнаванням якого працюють агенти.

Також детально розглянуто режими роботи агента-дослідника із зазначенням пріоритетності їхньої активації залежно від навколишніх умов і перелік повідомлень, якими обмінюються агенти у процесі роботи алгоритму. Наведено повний алгоритм роботи агента-експериментатора з обробки отриманих повідомлень, на основі яких і відбувається розпізнавання графа. Виконано аналіз часової, ємнісної, комунікаційної складності побудованого алгоритму та проаналізовано кількість переходів по ребрах, які виконує агент-дослідник під час обходу графа. З'ясовано, що запропонований алгоритм має квадратичні (від кількості вершин досліджуваного графа) часову, ємнісну та комунікаційну складності. Верхню оцінку кількості переходів по ребрах, які здійснює агент-дослідник, оцінено як O(n^2), де n – кількість вершин у графі. Для роботи запропонованого алгоритму розпізнавання графа агенту-досліднику необхідно дві фарби різного кольору й один камінь.

Ключові слова: ***неорієнтовані графи, розпізнавання графів, колектив агентів, складність алгоритму, обхід у глибину.***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; in the decision to publish the results.