MATHEMATICAL MODELING AND COMPUTING, Vol. 12, No. 1, pp. 75–82 (2025)

Finite graph exploration by a mobile agent

Stopkin A. V.

SHEI "Donbas State Pedagogical University",
13 Naukova Str., 49020, Dnipro, Ukraine

(Received 24 September 2024; Revised 16 December 2024; Accepted 19 December 2024)

The paper considers the task of finite connected graph exploration by a mobile agent. The mobile agent moves along the graph, reads and changes marks of the graph elements, and explores the graph based on this information. A new algorithm for finite undirected graph exploration of time complexity $O(n^3)$, space complexity $O(n^2)$ and the upper estimate of the number of transitions along the edges made by the agent $O(n^3)$ is proposed. For the algorithm to work, the agent needs one color. The algorithm is based on depth-first traversal method.

Keywords: *mobile agent; finite graph; graph exploration.*

2010 MSC: 05C85, 68R10 **DOI:** 10.23939/mmc2025.01.075

1. Introduction

The problem of computer science, the solution of which is paid special attention, is the problem of interaction between the control and controlled systems, for example, the interaction of the control machine, its agent and the operating environment of this agent [1]. In the case considered below, the interaction of these systems is presented as the process of moving the agent through the graph of the environment [2].

Until a sufficiently complete model of the operating environment is formed, it is impossible to purposefully move the agent through it. In the issue of modeling of operating environments, a number of approaches have been identified, one of which is topological [3]. In this case, the agent only has information about the relationships between the different regions of the environment, and the metric and algorithmic information about the environment is not available. Often, a similar situation arises in robotics [4].

2. Review of studies and publications

The beginning of research in this direction is considered to be the work of K. Shannon [5], in which the task of searching for a given target in the maze with a maze-solving machine was considered.

An active study of the behavior of automata in mazes begins after the appearance of the work of K. Dopp [6] describing the traversing of chess mazes with end automata. Over time, the conditions of the problem expanded, and the machine already needed to visit all the nodes and/or edges of the explored graph (maze). Later, [7] studies were carried out in the field of analyzing the properties of an unknown environment under various methods of interaction of the machine with the operating environment, as well as various prior information about it.

Graph analysis includes a number of particular problems. Let us consider the main ones: the problem of self-localization (determining the node of the graph in which the automat is located initially), the problem of map control (checking the isomorphism of the explored graph and the standard graph (map)) and the problem of complete exploration of the graph (constructing the reference graph (map) of the explored graph). A number of approaches to solving the latter problem have been determined, a number of algorithms for wandering the agent through the graph and methods for marking the elements of the graph with colors or stones, which allows us to explore the graph with accuracy to isomorphism, have been proposed.

So, in the work [7], a method is proposed in which the agent, when moving, marks only the incidentors (the point of connection of the edges with the nodes) of the passed edges. To explore the edges, the agent moves along them, then returns to the starting node in the shortest way. At the same time, it remembers the marks of the past incidentors, which clearly determine the node into which the agent fell, passing along the exploring edge.

The work [8] proposes an algorithm based on a depth-first search strategy. The algorithm has a number of features: firstly, the graph under study is completely unknown to the agent and the agent at each step has information about the colors of elements from the vicinity of the working node; secondly, when passing the nodes of the graph, the agent creates implicit numbering (numbers are not applied to the nodes) of the passed nodes: when the node is first visited, it is colored in red and it is actually matched with a number equal to the value of a special variable from the agent's memory. Based on this numbering, the graph is explored by constructing the graph which is isomorphic to the original.

Since 1993, works have been published on the study of graphs by a swarm of agents [9, 10]. The algorithm for swarm operation is as follows. All agents (the memory of each agent can accommodate the full map of the graph under study) begin work from one node. Having agreed in advance on the time of the next meeting, the agents divide the available edges among themselves and diverge on them. After traversing a certain part of the graph, the agents return to the start node to combine the received maps. Then they agree on the next meeting and again diverge and so it continues until the graph is fully explored. Similar algorithms using swarm of agents were considered in works of H. Wang [11] and C. Zhang [12]. But, for example, in the work of S. Das [13], a team of agents is used (the memory of each agent can accommodate the complete map of the graph under study), which begin work from different nodes of the graph and interact with each other by means of tokens written by agents to the nodes of the graph. When several agents get the same node, each of them knows how many agents are at the node and can exchange information with any of them, thus combining the necessary parts of the maps.

In work [14], the algorithm of work of a team of agents with additional requirements imposed on the explored graph, used agents and the channel of communication between agents is considered. For example, each node of the explored graph must have a memory that allows you to write a number which is equal to the number of nodes of this graph.

Also, each agent has the ability to read and send the numbers of all nodes from the vicinity of the node in which it is located, which contains the entailed dependence of the memory of the agent on the dimension of the explored graph. So, one team of agents could only explore those graphs for which he had enough memory. It is also worth noting that each agent, after explored its subgraph, begins to look for new subgraphs for exploration.

In work [15], the algorithm of the work of a team consisting of two agents was considered. It is similar to the algorithm [14], but the memory of the nodes is now reduced to the possibility of storing two marks and does not depend on the dimension of the graph. Also, agents do not need to pass node numbers. This makes the memory of the agent independent of the dimension of the explored graph. So, one team of agents can explore graphs of any dimension.

The work [16], it is proposed a decentralized approach to the graph exploration by a team of agents, in which agents can avoid collisions without having a direct connection with each other. Information exchange between robots takes place by means of beacons installed in previously visited nodes of the graph. It is worth noting that in this algorithm, decision-making lies directly on each agent and does not depend on the actions of other agents. The proposed decentralized method ensures that the unknown environment exploration is completed in a finite number of steps. The structure of the explored environment is built gradually during the work of agents.

But, for example, in the work [17], agents exchange information about the work done with the base station through a special network. This network has some restrictions on data exchange. That is, in order to coordinate actions between agents, it is first necessary to form a network that satisfies these conditions. In other words, communication with the base station takes place at predetermined locations. The peculiarity of the article is asynchronous strategies working with arbitrary communication models. Asynchronism refers to the ability to give instructions to subgroups of agents when they are ready to accept them.

The article [18] attempts to develop general concepts and requirements for the graph exploration by multi-agent systems. The proposed representation for a multi-agent system is aimed at providing general conditions for declaring the completion of a graph exploration and the completion of such an exploration in the finite time. The paper proposes modifications of the incident matrix for the organized exchange of information between agents. The general algorithm of work of the team of agents researching the graph is also presented. The algorithm is based on the proposed generalized structure and the modified incident matrix.

The work [19] considers the algorithm of work of the team of agents consisting of two research agents and one experimental agent. The algorithm is based on the depth-first traversal method. When the algorithm works, the research agents build implicit numbering on their nodes (numbers are not applied to the nodes) on the basis of which the experimental agent builds a graph in its memory that is isomorphic to the explored one. Each node of the graph must have the same memory for the graph of any dimension, sufficient to store the inks applied (no more than two colors at a time). The memory of research agents does not depend on the dimension of the graph, which allows one team to explore a graph of any dimension.

3. Problem statement

As it can be seen from the publications, the problem of graphs exploration is relevant. This is not surprising, taking into account that an infinite number of unexplored environments remain around us [20]. All this makes the tasks of conducting a systematic research of graph exploration experiments, that is, creating agent routes along an unknown graph, marking its elements, collecting and processing local graph information and methods of constructing a graph on this information, accurate to the marks on graph elements, relevant. As well as tasks aimed at finding methods for optimizing resource consumption, time consumption, load on the communication channel, etc., when graphs exploring.

The purpose of the work is to create an effective method and to build an appropriate algorithm for unknown graphs exploration using a wandering agent. As well as a study of temporal complexity, capacitive complexity and an estimate of the number of transitions along the edges made by the agent for the graph exploration.

4. Basic definitions and symbols

The paper considers simple finite undirected connected graphs. Let $G = (V, E)$ be a connected finite undirected graph without loops and multiple edges, where V is the set of nodes, E is the set of edges (two-element subsets (u, v), where $u, v \in V$). The three $((u, v), v)$ we shall call the incidentor (a contact point) of the edge (u, v) and the node v. Through I we denote a lot of all the incidentors of the graph. A set $L = V \cup E \cup I$ is called a set of graph G elements. A surjective map $\mu\colon L \to \{w, b\}$, where w is interpreted as white, and b is black, we call the coloring function of the graph G. A pair (G, μ) is called a colored graph. A sequence $u_1, u_2, \ldots, u_k$ of pairwise adjacent nodes of the graph G is called a length path of k. The vicinity $Q(v)$ of the node v we call the set of elements of the graph, consisting of a node v, all nodes u adjacent to v, all edges (v, u) and all the incidentors $((v, u), v)$, $((v, u), u)$. The cardinality of the sets of nodes V and edges E we shall denote through n and m, respectively. It is clear that $m \leqslant \frac{n(n-1)}{2}$. We call the isomorphism of the graph G and the graph H such a bijection $\varphi\colon V_G \to V_H$ that $(v, u) \in E_G$ exactly when $(\varphi(v), \varphi(u)) \in E_H$. Thus, isomorphic graphs are equal to within the designation of nodes and the coloring of their elements.

To graph exploration, one mobile agent is used, which can move around the graph, color the graph elements and read marks on the graph elements, as well as build a representation of the explored graph in its memory. The agent has a finite, unrestricted growing internal memory, in which the result of functioning at each step is fixed and, in addition, a representation of the graph G, initially unknown to the agent, is built with lists of edges and nodes. The agent has one black paint.

At the beginning of work, the agent is placed in an arbitrary node of the graph G, colors this node black and places it in a set of nodes V_H. The agent moves along the graph from node v to node u along the edge (v, u), can change the color of the nodes v, u, edges (v, u) and incidentors $((v, u), v)$, $((v, u), u)$. Being at the node v, the agent perceives the marks of all elements of the vicinity $Q(v)$ and based on this information determines along which edge the agent will move further, and how it will color the elements of the graph.

The proposed algorithm is based on a depth-first search strategy [21], has cubic temporal complexity and quadratic capacitive complexity, while the upper estimate of the number of transitions along the edges performed by the agent is estimated as $O(n^3)$.

5. Algorithm of the agent operation

At the beginning of the operation of the algorithm, all elements of the graph are colored white. During the operation, the graph elements can be colored by the agent. The elements of the graph, depending on their color, will be called white or black.

Let us consider in more detail the algorithm of functioning of the agent. At the beginning of the operation, the agent enters an arbitrary node of the graph, which is immediately colored black by it and added to the list of visited nodes, which is stored in the agent's memory. Then the agent moves along the white nodes, coloring these nodes, edges (which connect these nodes) and the distant incidentors black. At each step, node and edge lists are updated in the agent's memory.

Algorithm 1 Algorithm of the agent operation

```
1: initialization: $ct := 1$, $i := 0$, $E_H := \varnothing$, $t := 1$, $work(t) := ct$, $V_H := \{1\}$;
2: begin
3:     $\mu(v) := b$;
4:     if $\exists(v, u) \in Q(v) \mid (\mu(v, u) = w)$ and $(\mu(u) = \mu(v) = b)$ then do
5:       $EXPL\_IE(v)$;
6:       if $\exists(v, u) \in Q(v) \mid (\mu((v, u), v) = w)$ and $(\mu((v, u), u) = b)$ and $(\mu(u) = b)$ then do
7:           $FORWARD\_IE(v)$;
8:           go to 6;
9:           end do;
10:      else do
11:          $ADD\_IE(v)$;
12:          go to 3;
13:          end do;
14:    end do;
15:    else if $\exists(v, u) \in Q(v) \mid (\mu(v, u) = w)$ and $(\mu(u) = w)$ then do
16:              $FORWARD(v)$;
17:              go to 3;
18:              end do;
19:      else if $\exists(v, u) \in Q(v) \mid (\mu((v, u), v) = b)$ and $(\mu((v, u), u) = w)$ then do
20:                  $BACK(v)$;
21:                  go to 3;
22:                  end do;
23:              else $STOP(v)$;
24: print $V_H$, $E_H$;
25: end.
```

If, in the process of moving forward along the white nodes, the agent meets the back edge (a white edge with white near and far incidentors, which leads to the already visited node), then it moves along this edge, coloring it and the far incidentor black. Then the agent returns to the node from which it began the exploration of the back edge, along the previously traveled path, counting the number of steps taken. The steps are counted to determine the number of the node in which the agent got, making a transition along the back edge. Returning to the node from which the transition on the back edge was made, the agent determines the number of the far node of the back edge, adds it to the list of edges and colors the near incidentor of this edge black.

When an agent enters a node in the vicinity of which there are no white nodes or unexplored back edges, it begins to move back along a previously explored path in search of unexplored edges and nodes. When moving back, the agent colors black the distant incidentors of the edges along which it makes the transition. If when moving back, the agent returned to the initial node and did not find unexplored nodes and edges, then this means that the entire graph is explored. The agent finishes the operation and at that moment its memory already contains a list of nodes and a list of edges of the graph.

```
$EXPL\_IE(v)$:
1. The agent selects from $Q(v)$ the edge $(v,u)$ which has $(\mu(v,u)=w)$ and $(\mu(u)=\mu(v)=b)$;
2. The agent moves along the edge $(v,u)$ to the node $u$ coloring $\mu((v,u),v):=b$, $\mu(v,u):=b$, $\mu((v,u),u):=b$;
3. $v:=u$.
$FORWARD\_IE(v)$:
1. The agent selects from $Q(v)$ the edge $(v,u)$ which has $(\mu((v,u),v)=w)$ and $(\mu((v,u),u)=b)$ and $(\mu(u)=b)$;
2. The agent moves along the edge $(v,u)$ to the node $u$;
3. $v:=u$;
4. $i:=i+1$.
$ADD\_IE(v)$:
1. $E_H := E_H \cup \{(work(t), work(t-i))\}$;
2. $i:=0$.
$FORWARD(v)$:
1. The agent selects from $Q(v)$ the edge $(v,u)$ which has $(\mu(v,u)=w)$ and $(\mu(u)=w)$;
2. The agent moves along the edge $(v,u)$ to the node $u$ coloring $\mu(v,u):=b$, $\mu((v,u),u):=b$, $\mu(u):=b$;
3. $v:=u$;
4. $ct:=ct+1$;
5. $t:=t+1$;
6. $work(t):=ct$;
7. $V_H := V_H \cup \{ct\}$;
8. $E_H := E_H \cup \{(work(t-1), work(t))\}$.
$BACK(v)$:
1. The agent selects from $Q(v)$ the edge $(v,u)$ which has $(\mu((v,u),v)=b)$ and $(\mu((v,u),u)=w)$;
2. The agent moves along the edge $(v,u)$ to the node $u$ coloring $\mu((v,u),u):=b$;
3. $v:=u$;
4. delete $work(t)$;
5. $t:=t-1$.
$STOP(v)$:
1. The agent finishes the operation.
```

6. Analysis of the exploration algorithm

The procedure *FORWARD*(v) is performed when the agent visits the white nodes of the explored graph G. This procedure creates one new node of the graph H. Thus, the process of executing the described algorithm induces the mapping $\varphi\colon V_G \to V_H$. Moreover, $\varphi(v) = ct$. The specified mapping φ is a bijection because in the connected graph G all nodes are reachable from the starting node.

When performing the procedure $FORWARD(v)$, the agent explores the tree edge (v, u) and numbers the node u in such a way, that the edge (v, u) uniquely corresponds to the edge $(\varphi(v), \varphi(u))$ of the graph H. When performing the procedure $ADD\_IE(v)$, the agent explores the back edge (v, u) of the graph G and perfectly matches it the edge $(\varphi(v), \varphi(u))$ of the graph H. Therefore, φ is an isomorphism of the graph G to the graph H.

Theorem 1. *By performing an exploration algorithm on the graph G, the agent explores the graph with accuracy to isomorphism.*

Now we define the time and space complexity of the algorithm in a uniform scale [22]. We shall also assess the number of agent transitions along the edges. From the description of the algorithm, it follows that at each step of the algorithm, the working path is a simple path connecting the starting node v with the number $\varphi(v) = 1$ to the node u with the number $\varphi(u) = ct$. Therefore, the length of the working path does not exceed n.

Note that for a single moving forward or backward (procedures: $FORWARD(v)$, $BACK(v)$), the agent spends one move and passes one edge. When single exploration of back edges (procedures: $EXPL\_IE(v)$, $FORWARD\_IE(v)$, $ADD\_IE(v)$), the agent passes one back edge, and no more than $n - 1$ edges of the working path. By exploring the back edge, the agent, in fact, goes through a cycle consisting of the back edge and some final segment of the working path connecting the nodes which are incident to the back edge.

When calculating the time complexity of the algorithm, we assume that the initialization of the algorithm and the choice of one of the possible procedures take a certain constant number of units of time. We also consider that the moving along the edge by the agent is carried out in a time equal to some constant. Moreover, the total time spent on the analysis of the vicinity of the working node $Q(v)$ and the selection of the necessary edges is estimated as $O(n^2)$. Then the time complexity of the algorithm is determined by the ratios:

1. Initialization is performed once and its asymptotic complexity is $O(1)$;
2. The procedure $FORWARD(v)$ is executed no more than $n - 1$ times and the total execution time is evaluated as $O(n)$.
3. The procedure $BACK(v)$ is executed no more than $n - 1$ times and the total execution time is evaluated as $O(n)$.
4. The procedure $EXPL\_IE(v)$ is executed no more than m times and the total execution time is evaluated as $O(n^2)$.
5. The procedure $FORWARD\_IE(v)$ is executed no more than $n \cdot (n - 1) \cdot O(n)$ times and the total execution time is evaluated as $O(n^3)$.
6. The procedure $ADD\_IE(v)$ is executed no more than m times and the total execution time is evaluated as $O(n^2)$.
7. The procedure $STOP(v)$ is executed once and its asymptotic complexity is $O(1)$.

Therefore, the upper estimate of the number of edge $M(n)$ transitions performed by the agent satisfies the ratio: $M(n) = O(n^3)$.

The total time complexity $T(n)$ of the algorithm satisfies the ratio: $T(n) = O(n^3)$.

The space complexity $S(n)$ of the algorithm is determined by the complexity of the lists V_H, E_H, $work(1), \ldots, work(t)$, whose complexity is accordingly determined by the magnitudes $O(n)$, $O(n^2)$, $O(n)$. Therefore $S(n) = O(n^2)$.

Theorem 2. *The time complexity of the exploration algorithm is $O(n^3)$, the space complexity is $O(n^2)$, and the upper estimate of the number of transitions along the edges made by the agent is $O(n^3)$. In this case, the algorithm uses 1 color.*

7. Conclusions

The paper proposes a new algorithm for finite undirected connected graph exploration of time complexity $O(n^3)$, space complexity $O(n^2)$ and the upper estimate of the number of transitions along the edges made by the agent $O(n^3)$. The agent has finite memory at each step, an unlimitedly growing internal memory overall, and uses one color.

[1] Glushkov V. M., Letichevsky A. A. Theory of Discrete Converters (Selected Topics of Algebra and Logic). Novosibirsk, Nauka. 5–20 (1973).

[2] Stepkin A. V. Finite Graphs Exploration by Three Agents. Artificial Intelligence. **2**, 84–93 (2011).

[3] Kuipers B. The spatial semantic hierarchy. Artificial Intelligence. **119** (1–2), 191–233 (2000).

[4] Dudek G., Jenkin M. Computational principles of mobile robotics. Cambridge, Cambridge University Press (2000).

[5] Shannon C. E. Presentation of a Maze-Solving Machine. Cybernetics, Transactions of the 8th Conference, Editor: H. von Foerster. 173–180 (1951).

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

[7] Dudek G., Jenkin M., Milios E., Wilkes D. Map validation in a graphlike world. Proceedings of the 13th International Joint Conference on Artifical Intelligence (Chambery, France, August 1993). San Fransisco, Morgan Kaufmann Publishers Inc. 1648–1653 (1993).

[8] Grunskiy I. S., Tatarinov E. A. Recognition of a finite graph by an agent wandering on it. Bulletin of DonNU. **1**, 492–497 (2009).

[9] Dudek G., Jenkin M., Milios E., Wilkes D. A taxonomy for swarm robots. Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93). 441–447 (1993).

[10] Dudek G., Jenkin M., Milios E., Wilkes D. Topological exploration with multiple robots. Proceedings of the 7th International Symposium on Robotics with Application (ISORA). Anchorage, Alaska, USA (1998).

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

[12] Zhang C. Parallelizing Depth-First Search for Robotic Graph Exploration. Harvard College, Cambridge, Massachusetts (2010).

[13] Das S., Flocchini P., Kutten S., Nayak A., Santoro N. Map construction of unknown graphs by multiple agents. Theoretical Computer Science. **385** (1–3), 34–48 (2007).

[14] Stepkin A. Using a Collective of Agents for Exploration of Undirected Graphs. Cybernetics and Systems Analysis. **51** (2), 223–233 (2015).

[15] Stepkin A. Exploration of the graph structure by two agents. Proceedings of IAMM of NASU. **30**, 111–121 (2016).

[16] Nagavarapu S. C., Vachhani L., Sinha A. Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach. Journal of Intelligent & Robotic Systems. **83**, 503–523 (2016).

[17] Banfi J., Quattrini Li A., Rekleitis I., Amigoni F., Basilico N. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. Autonomous Robots. **42**, 875–894 (2018).

[18] Nagavarapu S. C., Vachhani L., Sinha A., Buriuly S. Generalizing Multi-agent Graph Exploration Techniques. International Journal of Control, Automation and Systems. **19**, 491–504 (2021).

[19] Stepkin A. V. Possibility and complexity of graph exploration by three agents. Tavricheskiy Vestnik Informatics and Mathematics. **1** (20), 88–98 (2012).

[20] Goth G. Exploring new frontiers. Communications of the ACM. **52** (11), 21–23 (2009).

[21] Cormen T., Leiserson Ch., Rivest R., Stein C. Introduction to Algorithms. MIT Press (2009).

[22] Aho A., Hopcroft J., Ullman J. Design and Analysis of Computer Algorithms. Addison-Wesley (1974).

Розпізнавання скінченого графу мобільним агентом

Стьопкін А. В.

ДВНЗ "Донбаський державний педагогічний університет",
вул. Наукова, 13, 49020, Дніпро, Україна

В роботі розглядається завдання розпізнавання скінчених зв'язних графів мобільним агентом. Мобільний агент пересувається по графу, зчитує та змінює мітки елементів графа та на підставі цієї інформації розпізнає досліджуваний граф. Запропоновано новий алгоритм розпізнавання скінчених неорієнтованих графів часової складності — $O(n^3)$, ємнісної складності — $O(n^2)$ і верхньою оцінкою кількості переходів по ребрам здійснюваних агентом — $O(n^3)$. Для роботи алгоритму агенту потрібна одна фарба. Алгоритм грунтується на методі обходу графа в глибину.

Ключові слова: *мобільний агент; скінчений граф; розпізнавання графів.*