УДК 519.1:519.7:007.5
DOI https://doi.org/10.32782/2663-5941/2024.5.1/43

Стьопкін А.В.
ДВНЗ «Донбаський державний педагогічний університет»

АЛГОРИТМ РОЗПІЗНАВАННЯ ПРОСТИХ НЕОРІЄНТОВАНИХ ГРАФІВ КОЛЕКТИВОМ АГЕНТІВ

Стаття присвячена проблемі розпізнавання неорієнтованих графів без петель та кратних ребер колективом агентів. У статті запропоновано новий алгоритм розпізнавання графів колективом агентів, що складається з двох агентів-дослідників, які мають скінчену пам'ять (об'єм пам'яті агентів не залежить від числа вершин досліджуваного ними графа), і для розпізнавання графу використовують по дві фарби різного кольору кожен (всього три різні фарби) і одного агента-експериментатора, який знаходиться поза межами графу та володіє скінченою, необмежено зростаючої внутрішньою пам'яттю. Агенти-дослідники можуть пересуватися по графу, зчитувати і змінювати мітки елементів графа, а також передавати необхідну інформацію агенту-експериментатору. Агент-експериментатор – це нерухомий агент в пам'яті якого на кожному кроці фіксується результат функціонування агентів-дослідників і поступово вибудовується представлення досліджуваного графа списком ребер і списком вершин.

У статті розкрито режими роботи агентів-дослідників із зазначенням порядку їх активації та детально розглянуто повідомлення, якими обмінюються агенти між собою в процесі роботи. Запропоновано повний алгоритм роботи агента-експериментатора з детальним описом процедур обробки отриманих від агентів-дослідників повідомлень, на підставі яких і відбувається розпізнавання досліджуваного графа (побудова карти графа). Також в статті проведено аналіз часової, ємнісної, комунікаційної складності побудованого алгоритму та проаналізовано кількість переходів по ребрах, які виконують агенти-дослідники.

З'ясовано, що представлений алгоритм розпізнавання графів має квадратичні (від числа вершин досліджуваного графа) часову, ємнісну і комунікаційну складності, а також кількість переходів по ребрах, які виконують агенти-дослідники. Робота запропонованого алгоритму розпізнавання графів ґрунтується на методі обходу графа в глибину.

Ключові слова: *розпізнавання графів, прості скінчені графи складність алгоритму, обхід в глибину, колектив агентів.*

Постановка проблеми. В наш час досить стрімко розвивається такий напрям кібернетики як робототехніка, яка дозволяє автоматизувати деякі процеси та спростити людині виконання важкої або навіть небезпечної роботи. Беручи до уваги те, що в світі, у зв'язку з недоступністю для людини, наявна велика кількість недосліджених середовищ [1] стає очевидним, що вирішення задачі дослідження середовищ роботами залишається актуальною проблемою [2–4], причому використання колективу агентів [5–7] підвищує ймовірність вирішення поставленої задачі, а можливо навіть покращить час роботи алгоритму.

Аналіз останніх досліджень і публікацій.

Початком досліджень в цьому напрямку прийнято вважати роботу К. Шеннона [8], в якій розглядалася задача пошуку автоматом заданої цілі в лабіринті. Активне вивчення поведінки автоматів в лабіринтах починається після появи роботи К. Деппа [9], який описує обхід шахових лабіринтів скінченими автоматами. Далі в роботі [10] проводилися дослідження в галузі аналізу властивостей невідомого середовища при різних способах взаємодії автомата з операційним середовищем, а також при різній апріорній інформації про нього.

До вирішення проблеми розпізнавання графів визначено ряд підходів, запропоновано ряд алгоритмів блукання агента по графу і способів позначення елементів графа фарбами або камінням, що дозволяють розпізнати досліджуваний граф з точністю до ізоморфізму. Так, в роботі [11], пропонується метод, в якому агент, при переміщенні, мітить тільки інцидентори (точка з'єднання ребер з вершинами) пройдених ребер. Для розпізнавання ребер агент переходить по ним, потім найкоротшим шляхом повертається в початкову вершину. При цьому запам'ятовує мітки пройдених інциденторов, які однозначно визначають вершину, в яку потрапив агент.

Починаючи з 1993 року публікуються роботи по дослідженню графів роєм агентів [11, 12].

Алгоритм роботи рою полягає в наступному: всі агенти починають роботу з однієї вершини. Домовившись заздалегідь про час наступної зустрічі, агенти ділять доступні ребра між собою і розходяться по ним. Після обходу певної частини графа агенти повертаються у вихідну вершину для об'єднання отриманих карт. Далі вони домовляються про наступну зустріч і знову розходяться і так триває до повного розпізнавання графа. Аналогічні алгоритми з використанням рою агентів розглядалися в роботах [5] і [6]. А ось, наприклад, в роботі [13] використовується колектив агентів, які починають роботу з різних вершин графа і взаємодіють між собою за допомогою токенів, що записуються в вершини графа.

В роботі [4] пропонується децентралізований підхід до дослідження графів колективом агентів, в якому агенти можуть уникати зіткнень, не маючи між собою прямого зв'язку (обмін інформацією між роботами відбувається за допомогою міток, встановлених в раніше відвіданих вершинах графа). Варто відзначити, що в цьому алгоритмі прийняття рішень лежить безпосередньо на кожному агенті і не залежить від дій інших агентів.

А ось, наприклад, в роботі [2] агенти обмінюються інформацією про виконану роботу з базовою станцією через спеціальну мережу. Ця мережа має деякі обмеження щодо обміну даними. Тобто для координації дій між агентами потрібно спочатку сформувати мережу, що задовольняє певним умовам. Особливість дослідження – асинхронні стратегії, що працюють з довільними моделями комунікації.

У статті [3] зроблено спробу розробки загальних концепцій і вимог для дослідження графів колективами агентів. Запропоноване представлення мультиагентної системи, спрямоване на надання загальних умов для оголошення завершення дослідження графа і завершення такого дослідження за скінчений час.

Постановка завдання. У відомих роботах мало уваги приділено дослідженню обміну інформацією між агентами, а також його складності і можливого впливу на часову та ємнісну складності. Це робить актуальними дослідження експериментів по розпізнаванню графа колективом агентів, а також завдання, спрямовані на пошук методів оптимізації витрат ресурсів та часу, а також навантаження на канал зв'язку при розпізнаванні графів.

Мета роботи – створення ефективного методу і побудова відповідного алгоритму розпізнавання невідомих простих графів за допомогою колективу агентів [7, 14–15], а також дослідження часової, ємнісної та комунікаційної складностей побудованого алгоритму.

Виклад основного матеріалу. В роботі розглядаються скінчені, неорієнтовані, прості графи. Нехай $G=(V,E)$ – граф, у якого V – множина вершин, E – множина ребер (двоелементних підмножин (v,u), де $u,v\in V$). Трійку $((v,u),u)$ будемо називати інцидентором (точкою дотику) ребра (v,u) та вершини u. Множину таких трійок позначимо I. Множину $L=V\cup E\cup I$ назвемо множиною елементів графа G. Функцією розфарбування графа G назвемо сюр'єктивне відображення $\mu: L\to\{w,r,y,ry,b\}$, де w – білий колір, r – червоний, y – жовтий, ry – червоно-жовтий а b – чорний. Пара (G,μ) називається розфарбованим графом. Послідовність $u_1,u_2,...u_k$ попарно суміжних вершин називається шляхом в графі G, а k – довжиною шляху. За умови $u_1=u_k$ шлях називається циклом. Околом $Q(v)$ вершини v будемо називати множину елементів графа, що складається з вершини v, всіх вершин u суміжних з v, всіх ребер (v,u) та всіх інциденторів $((v,u),v),((v,u),u)$. Потужність множин вершин V і ребер E позначимо через n і m відповідно. Зрозуміло, що $m\le\frac{n(n-1)}{2}$. Ізоморфізмом графа G і графа H назвемо таку бієкцію: $\phi: V_G\to V_H$, що $(v,u)\in E_G$ тоді й тільки тоді, коли $(\phi(v),\phi(u))\in E_H$. Таким чином, ізоморфні графи рівні з точністю до позначення вершин і розфарбування їх елементів.

Колектив агентів складається з агентів двох типів:

– агент-дослідник (АД) – це агент, який може переміщуватися по скінченому графу, в процесі роботи може фарбувати вершини, ребра і інциденти графа, а також сприймати ці позначки, також АД може відправляти повідомлення агенту-експериментатору;

– агент-експериментатор (АЕ) – це нерухомий агент, який в процесі роботи може передавати, приймати і ідентифікувати повідомлення АД, володіє скінченою, необмежено зростаючої внутрішньою пам'яттю, в якій фіксується результат функціонування АД на кожному кроці, і, крім того, поступово вибудовується представлення графа списками ребер і вершин.

На початку роботи АД A та B розміщуються в довільні неспівпадаючі вершини графа G. Агенти йдуть «в глибину», поки це можливо, повертаються назад, шукають інший шлях з ще не відвіданими вершинами та не пройденими ребрами. У випадку знаходження суміжної вершини, яка має «чужий» колір, агент мітить всі

перешийки (ребра, що з'єднують підграфи, які розпізнаються різними АД) з поточної вершини в чужу область і сповіщає іншого АД, через АЕ, про необхідність розпізнавання помічених перешийків. Поки другий АД розпізнає перешийки, перший АД не може мітити нові перешийки. У випадку відсутності інших можливих варіантів руху, крім як помітити новий перешийок, перший АД зупиняється до того моменту, поки другий АД не розпізнає всі помічені перешийки. При переході в білі вершини графа АД A і B фарбують його елементи в червоний та жовтий кольори відповідно. При русі «в глибину» червоний (жовтий) шлях подовжується, при русі назад зменшується. При русі АД назад, для розпізнавання зворотних ребер або перешийків, довжина шляху не змінюється. Вершина, з якої можливий лише рух назад по своєму шляху, або зовсім відсутні варіанти переміщення, забарвлюється в чорний колір. Таким чином формуються червоний і жовтий шляхи кожного з агентів. Алгоритм закінчує роботу, коли червоний і жовтий шляху стають порожніми, а всі вершини чорними. Виконуючи обхід графа агенти створюють неявну нумерацію відвіданих вершин. Перший раз відвідавши вершину агент A фарбує її в червоний колір (агент B – в жовтий) та АЕ ставить їй у відповідність номер, рівний значенню змінної Ct_A (Ct_B для агента B). Зазначимо, що змінні Ct_A і Ct_B зберігаються та оброблюються АЕ і можуть приймати лише непарні або парні значення відповідно. Розпізнавання графу G відбувається на основі створеної агентами нумерації шляхом побудови графа H ізоморфного G.

Далі розглянемо режими в яких можуть функціонувати АД. При описі режимів, в дужках вказуються повідомлення, що відправляються АД у відповідній ситуації із зазначенням агента, який відправляє повідомлення («Message_A»; «Message_B»).

1. *Звичайний режим роботи.* Працюючи в цьому режимі, АД рухається по вершинах, пофарбованих у білий колір, фарбуючи ці вершини, ребра, що їх з'єднують та дальні інцидентори в «свій» колір («Forward_A»; «Forward_B»). Якщо білі вершини і інші можливі шляхи переміщення відсутні, то АД повертається назад по своєму шляху, фарбуючи пройдені вершини, ребра, що їх з'єднують і ближні інцидентори в чорний колір («Back_A»; «Back _B»). АД завершує роботу тоді, коли його вихідна вершина забарвлюється в чорний колір («Stop_A», «Stop_B»).

2. *Режим розпізнавання зворотних ребер.* Якщо, при русі вперед, в вершині було виявлено біле ребро, у якого дальня вершина пофарбована в «свій» колір (зворотнє ребро), то АД перемикається в режим розпізнавання зворотних ребер і фарбує в «свій» колір ближні інцидентори всіх зворотних ребер, інцидентних поточній вершині («Mark_BE_A»; «Mark_BE_B»). Завершивши фарбування інцидентерів, АД рухається назад по своєму шляху на кожному кроці відправляючи повідомлення («Stepback_A»; «Stepback_B»), до виявлення вершини інцидентної позначеному раніше зворотному ребру, переходить по цьому ребру, фарбуючи його в чорний колір («Forward_BE_A»; «Forward_BE_B»). Якщо залишилися нерозпізнані зворотні ребра, то АД повертається назад по пройденому на попередньому кроці ребру, фарбуючи в чорний колір ближній інцидентор, і продовжує пошук зворотних ребер. Після виявлення останнього зворотного ребра АД забарвлює ближній його інцидентор, в чорний колір («Recognized_BE_A»; «Recognized_BE_B») і завершує роботу в режимі розпізнавання зворотних ребер.

3. *Режим помітки перешийків.* Якщо в процесі обходу графа в вершині v було знайдено перешийок, то за умови, що всі раніше позначені даним АД перешийки були розпізнані другим АД, агент перемикається в режим помітки перешийків. Якщо ж другий АД ще не розпізнав всі помічені перешийки, то перший АД не може мітити нові перешийки і у випадку, коли у першого АД відсутні інші можливі варіанти руху, крім як відмітити новий перешийок, він зупиняється до того моменту, поки другий АД не розпізнає всі помічені перешийки. Працюючи в цьому режимі АД фарбує ближні інцидентори всіх перешийків інцидентних вершині v в чорний колір («Mark_AB»; «Mark_BA»). Коли всі перешийки помічені робота АД в даному режимі завершується («Fix_A»; «Fix_B»). По завершенню режиму помітки перешийків АЕ містить інформацію про кількість помічених перешийків. В даному режимі роботи агент A має пріоритет над агентом B, тому в ситуації, коли обидва АД одночасно виявлять один і той же перешийок, він буде помічений агентом A.

4. *Режим розпізнавання перешийків.* Отримавши від АЕ команду про необхідність розпізнавання перешийків, АД перемикається в режим розпізнавання перешийків. Якщо в цей момент АД працює в режимі розпізнавання зворотних ребер, то він не переключиться в режим розпізнавання перешийків до завершення поточного режиму роботи. При перемиканні в цей режим АД перевіряє наявність з вершини, в якій він знаходиться, інших можливих шляхів переміщення

крім повернення назад по своєму шляху. Якщо такі шляхи є, то АД повертається назад по своєму шляху, нічого не фарбуючи («Stepback_A»; «Stepback_B»), до виявлення найближчої вершини, інцидентної білому ребру, у якого дальній інцидентор забарвлений в чорний колір, а дальня вершина – в «чужий» колір. Якщо ж таких шляхів немає, то повертаючись назад по своєму шляху, АД забарвлює його в чорний колір («Back_A»; «Back_B») до тих пір, поки не потрапить в вершину інцидентну позначеному перешийку або ж в вершину з іншими можливими шляхами переміщення. У другому випадку АД продовжує повертатися назад по своєму шляху, нічого не фарбуючи (кожен крок назад, відправляючи повідомлення «Stepback_A»; «Stepback_B») до виявлення найближчої вершини, інцидентної позначеному перешийку.

Якщо в процесі помітки перешийків було помічено один перешийок, то АД фарбує ближній інцидентор поміченого перешийка в чорний колір («Recognized_AB»; «Recognized_BA») і далі рухається вперед в кінець шляху «свого» кольору.

Якщо ж було позначено не менше двох перешийків, то АД забарвлює ближній інцидентор поміченого перешийка в «свій» колір («Recognized_AB»; «Recognized_BA»). Далі АД рухається назад по своєму шляху («Stepback_A»; «Stepback_B»), поки не буде знайдений наступний позначений перешийок. В цьому випадку, якщо позначений перешийок виявився не останнім, то АД забарвлює ближній інцидентор в чорний колір («Recognized_AB»; «Recognized_BA») і на наступному кроці АД знову повертається назад по своєму шляху до наступного поміченого перешийка («Stepback_A»; «Stepback_B»). Якщо ж позначений перешийок виявився останнім, то АД фарбує ближній інцидентор в «свій» колір («Recognized_AB»; «Recognized_BA»). На наступному кроці АД повідомляє АЕ про розпізнавання всіх помічених іншим АД перешийків («Settozero_A»; «Settozero_B»). Далі переходить по останньому перешийку в чужу область, забарвлюючи ближній інцидентор в чорний колір. На наступному кроці АД переходить по першому розпізнаному перешийку в свою область, фарбуючи дальній інцидентор в чорний колір. Далі АД рухається вперед в кінець шляху «свого» кольору.

5. *Одночасне потрапляння двох АД в одну білу вершину.* У разі одночасного потрапляння обох АД в одну і ту ж білу вершину кожен з них фарбує вершину наполовину, і вона стає червоно-жовтою. На наступному кроці агент B робить крок назад по своєму шляху, видаляє фарбу, нанесену на попередньому кроці з ближнього інцидентора і ребра, фарбує дальній інцидентор в чорний колір («Return_B») і переходить в режим помітки перешийків, причому ребро, по якому він здійснив перехід, вже пораховано як перший перешийок, а довжина жовтого шляху зменшена на одну вершину за рахунок обробки АЕ повідомлення «Return_B». Звернемо увагу, що агент A під час роботи на своєму шляху обробляє червоно-жовту вершину як вершину червоного кольору.

Пріоритетність перемикання АД в один з можливих режимів роботи розподіляється наступним чином: насамперед ми повинні перевірити наявність помічених перешийків, тому найвищий пріоритет має режим розпізнавання перешийків; далі ми перевіряємо чи немає з поточної вершини перешийків, які можна помітити для розпізнавання агентом A, тому другим за пріоритетом перемикання буде режим помітки перешийків; далі ми перевіряємо наявність зворотних ребер з поточної вершини, відповідно третім за пріоритетом у нас буде режим розпізнавання зворотних ребер; останнім по пріоритетності є звичайний режим роботи. Режим роботи АД у разі одночасного потрапляння їх в одну білу вершину не розглядається так як агент A в цьому випадку продовжує роботу в звичайному режимі, а для агента B в цей момент, вибір іншого режиму роботи буде недоступний.

Алгоритм роботи агента-експериментатора.

Вхід: списки повідомлень *SS_A* і *SS_B* від АД.

Вихід: список вершин V_H і ребер E_H графа H, ізоморфного графу G.

Дані: V_H, E_H списки вершин і ребер графа H. *Ct_A*, *Ct_B* – лічильники числа відвіданих вершин графа G агентами A і B відповідно. AN, BN – змінні, в яких значення «1» дає агентам A, B відповідно, сигнал для повернення та розпізнавання перешийків, значення «0» повідомляє агентам, що перешийки помічені для конкретного агента на розпізнавання відсутні. *N_A*, *N_B* – змінні, в яких зберігаються номери вершин, з яких агенти A і B відповідно, в останній раз помічали перешийки. F, K – кількість перешийків, помічених для розпізнавання, з вершин *N_A*, *N_B* відповідно. Q, Z – змінні, що використовуються в деяких підпрограмах роботи з перешийками, для зберігання початкових значень змінних F, K відповідно. *SP_A*, *SP_A* – списки повідомлень агентів A і B відповідно. E, L – змінні, в яких ставиться помітка про те, чи був на попередньому кроці агентом A чи B відповідно, помічений перешийок (значення «1») чи ні (значення «0»). i, j – лічильники, що використовуються агентами A і B відповідно при визначенні номерів других вершин помічених перешийків чи номерів

других вершин помічених зворотніх ребер. *ST_A*, *ST_B* – змінні, що використовуються агентами A і B відповідно, для повідомлення АЕ, про завершення розпізнавання свого підграфу. *UDP_A*, *UDP_B* – логічні змінні, що використовуються агентами A і B відповідно, для визначення способу фарбування інциденторів, перешейку, який розглядається в конкретний момент. *UDOBR_A*, *UDOBR_B* – логічні змінні, що використовуються агентами A і B відповідно для визначення чи є зворотне ребро, що розглядається в даний момент останнім з помічених. *KOBR_A*, *KOBR_B* – змінні, в які агенти A і B відповідно записують кількість помічених зворотних ребер. $r(1), r(2), ...r(t)$ – список номерів вершин червоного шляху, де t – довжина цього списку. $y(1), y(2), ...y(p)$ – список номерів вершин жовтого шляху, де p – довжина цього списку. *Mes* – повідомлення, яке обробляється в даний момент.

1. $AN := 0$; $BN := 0$; $N\_A := 0$; $N\_B := 0$; $F := 0$; $K := 0$; $SS\_A := \varnothing$; $SS\_B := \varnothing$; $E := 0$; $L := 0$; $i := 0$; $j := 0$; $E_H := \varnothing$; $Ct\_A := 1$; $Ct\_B := 2$; $V_H := \{Ct\_A, Ct\_B\}$; $t := 1$; $p := 1$; $r(t) := Ct\_A$; $y(p) := Ct\_B$; *UDP_A* := *False*; *UDP_B* := *False*; *UDOBR_A* := *False; UDOBR_B* := *False;* $KOBR\_A := 0$; $KOBR\_B := 0$; $STOP\_A := 0$; $STOP\_B := 0$;
2. *while* (*STOP_A*=0) *or* (*STOP_B*=0) *do*
3. *if SS_A*≠∅ *then do*
4. *читаємо повідомлення в Mes*;
5. *SS_A* :=*SS_A* \ {*Mes*};
6. *List_processing_A*();
7. e*nd do*;
8. *if SS_B*≠∅ *then do*
9. *читаємо повідомлення в Mes*;
10. *SS_B* :=*SS_B* \ {*Mes*};
11. *List_processing_B*();
12. *end do*;
13. e*nd do*;
14. друк V_H, E_H.

List_processing_A():

1. *if Mes* = ”Forward_A” *then Forward_A*();
2. *if Mes* = ”Back_A” *then Back_A*();
3. *if Mes* = ”Stop_A” *then Stop_A*();
4. *if Mes* = ”Mark_BE_A” *then Mark_BE_A*();
5. *if Mes* = ”Stepback_A” *then Stepback_A*();
6. *if Mes* = ”Forward_BE_A” *then Forward_BE_A*();
7. *if Mes* = ”Recognized_BE_A” *then Recognized_BE_A*();
8. *if Mes* = ”Mark_AB” *then Mark_AB*();
9. *if Mes* = ”Fix_A” *then Fix_A*();
10. *if Mes* = ”Recognized_AB” *then Recognized_AB*();
11. *if Mes* = ”Settozero_A” *then Settozero_A*();

Forward_A():

$Ct\_A := Ct\_A + 2$; $t := t + 1$; $r(t) := Ct\_A$; $V_H := V_H \cup \{Ct\_A\}$ $E_H := E_H \cup \{(r(t-1), r(t))\}$;

Back_A(): зі списку $r(1), r(2), ...r(t)$ видаляється елемент $r(t)$; $t := t - 1$;

Stop_A(): $ST\_A := 1$;

Mark_BE_A(): $KOBR\_A := KOBR\_A + 1$;

Stepback_A(): $i := i + 1$;

Forward_BE_A():

$E_H := E_H \cup \{(r(t), r(t-i))\}$;

Recognized_BE_A(): $i := 0$;

Mark_AB(): $F := F + 1$; $E := 1$;

Fix_A():

$N\_A := Ct\_A$; $BN := 1$; $E := 0$; $Q := F$; *UDP_A* := $(((F = Q)$ *or* $(F = 1))$ *and* $(Q \neq 1))$:

Recognized_AB():

$E_H := E_H \cup \{(N\_B, r(t-i))\}$; $K := K - 1$;

UDP_B := $(((K = Z)$ *or* $(K = 1))$ *and* $(Z \neq 1))$;

Settozero_A(): $AN := 0$; $i := 0$;

Процедури роботи зі списком повідомлень від агента B, які не розглядаються нижче, аналогічні до процедур роботи зі списком повідомлень від агента A.

Процедура *List_processing_B*() така ж як процедура *List_processing_A*() за виключенням додаткової перевірки умови і виклику підпрограми: «*if Mes* = ”Return_B” *then Return_B*();», яка стосується лише агента B і відноситься до режиму одночасного попадання двох АД в одну білу вершину.

Return_B():

$E_H := E_H \setminus \{(y(p-1), y(p))\}$; $V_H := V_H \setminus \{Ct\_B\}$; $Ct\_B := Ct\_B - 2$; $p := p - 1$; $y(p) := Ct\_B$; $L := 1$; $K := K + 1$;

Властивості алгоритму розпізнавання

За умови, що $n \geq 3$ як мінімум, по одному разу виконаються процедури: *Forward_A*() і *Forward_B*(). В кожній з процедур створюється по одній вершині графа H. При одночасному попаданні двох АД в одну білу вершину цими процедурами буде створено дві нові вершини графа *H*. Вершина створена агентом B, на наступному кроці буде видалена процедурою *Return_B*(), так як вона дублює вершину створену агентом A. Таким чином, процес виконання описаного алгоритму індукує відображення $\phi : V_G \to V_H$ вершин графа G у вершини графа H. Причому $\phi(v) = t$ (коли вершина v пофарбована в червоний колір і $t = Ct\_A$) і $\phi(s) = p$ (коли вершина s має жовтий колір і $p = t = Ct\_B$). Вказане відображення природним чином встановлює неявну нумерацію вершин графа G. Більш того, відображення ϕ є бієкцією, оскільки в зв’язному графі G всі вершини

досяжні з початкових вершин. А це значить, що всі вершини відвідуються агентами, тобто фарбуються в червоний та жовтий кольори. При виконанні процедури *Forward_A*() чи *Forward_B()* АЕ розпізнає ребро (v,u) і так нумерує вершину u, що ребру (v,u) однозначно відповідає ребро $(\phi(v),\phi(u))$ графа H. При виконанні процедур *Forward_BE_A*() чи *Forward_BE_B*() АЕ розпізнає зворотнє ребро (v,u) графа G і ставить йому в однозначну відповідність ребро $(\phi(v),\phi(u))$ графа H. При виконанні процедур *Recognized_AB*() або *Recognized_BA*() АЕ розпізнає перешийок (v,u) графа G і ставить йому в однозначну відповідність ребро $(\phi(v),\phi(u))$ графа H. Відповідно, ϕ є ізоморфізмом графа G на граф H.

Теорема 1. Три агенти, виконавши алгоритм розпізнавання на графі G, розпізнають цей граф з точністю до ізоморфізму.

Підрахуємо часову, ємнісну та комунікаційну складності алгоритмів у рівномірній шкалі. При підрахунку часової складності алгоритму будемо вважати, що ініціалізація алгоритму, аналіз околу робочих вершин та вибір однієї з можливих процедур займають деяке постійне число одиниць часу. Так ми будемо вважати, що вибір ребер, прохід по ним АД та обробка повідомлень АЕ, отриманих на даному етапі від АД, здійснюється за одиницю часу. З опису алгоритму слідує, що на кожному кроці алгоритму червоний (жовтий) шлях – це звичайний шлях, що з'єднує початкову вершину v (s – у випадку агента B) з номером $\phi(v)=1$ ($\phi(s)=2$ – у випадку агента B) з вершиною u (z – у випадку агента B) з номером $\phi(u)$= *Ct_A* ($\phi(z)$= *Ct_B* – у випадку агента B). Отже, загальна довжина червоного та жовтого шляху не перевищує n.

Кожного разу при виконанні процедур звичайного режиму роботи кожен АД проходить одне ребро. Що означає, що процедури звичайного режиму роботи виконуються не більше ніж $2\times(n-1)$ рази (враховуючи, що АД в цьому режимі пройде по кожному ребру свого шляху два рази), а значить загальний час їх виконання оцінюється як $O(n)$.

При однократному виконанні процедур режиму розпізнавання зворотніх ребер АД помічає не більше $n-2$ (так як граф простий і як мінімум одна вершина пофарбована в «чужий» колір) зворотніх ребер, по одному разу проходить не більш ніж $n-2$ ребер червоного (жовтого) шляху, а також по два рази проходиться не більше $n-2$ зворотніх ребер. Час, що витрачається при роботі у режимі розпізнавання зворотніх ребер, оцінюється як $4\times n\times O(n)$, тобто як $O(n^2)$.

При виконанні процедур режиму розпізнавання перешийків АД не здійснює переходів по перешийкам, а просто фарбує їх ближні інциденторі, після чого, надсилає повідомлення АЕ про завершення режиму розпізнавання перешийків, на що також йде один хід. Тобто на роботу в цьому режимі йде час, що оцінюється як $n\times O(n)+O(n)$, тобто як $O(n^2)$.

При одноразовому виконанні процедури режиму розпізнавання перешийків АД проходять не більше $n-2$ ребер червоного (жовтого) шляху, після чого, помічаючи перешейки як розпізнані, АД не виконують переходу по перешийку, а просто фарбують його ближній інцидентор. Закінчивши фарбування всіх помічених перешийків АД повідомляє про це АЕ, на що також йде один хід. Повертаючись після розпізнавання всіх перешийків АД може пройти максимум по одному разу два перешийка, а також не більше ніж $n-2$ ребер червоного (жовтого) шляху. Час виконання цього режиму оцінюється як $O(n^2)+O(n^2)+O(n)+O(n)+O(n^2)$, тобто як $O(n^2)$.

Час простою агентів в очікуванні в цілому оцінюється як $O(n^2)$. Отже, сумарна часова складність алгоритму задовольняє співвідношенню: $T(n)=O(n^2)$. В свою чергу верхня оцінка числа переходів по ребрах $M(n)$, які здійснюють АД складає $M(n)=O(n^3)$.

Ємнісна складність алгоритму визначається складністю списків $V_H, E_H, r(1)...r(t), y(1)...y(p)$, складність яких відповідно визначається величинами $O(n), O(n^2), O(n), O(n)$. Отже, $S(n)=O(n^2)$.

На кожному кроці алгоритму АД можуть відправити АЕ не більше чотирьох повідомлень порізно або одночасно і отримати від нього не більше трьох повідомлень кожен, тому об'єм переданої агентами інформації оцінюється як $14\times O(n^2)$. Отже, $K(n)=O(n^2)$.

З огляду на описані вище припущення про спосіб підрахунку часової складності, має місце наступна теорема.

Теорема 2. Часова, ємнісна, комунікаційна складності алгоритму та верхня оцінка числа переходів по ребрах дорівнюють $O(n^2)$, де n – число вершин графа, при цьому в алгоритмі використовується 3 фарби різного кольору.

Висновки. У роботі запропоновано новий метод і відповідний алгоритм розпізнавання графа колективом агентів. Основними перевагами побудованого алгоритму є те, що вдалося оптимізувати роботу агентів таким чином, що стало можливим використання АД зі скінченою пам'яттю, яка не залежить від розміру графа. Скінчену, необмежено зростаючу внутрішню пам'ять в нашому алгоритмі має тільки нерухомий АЕ, в пам'яті якого і відбувається побудова карти графа.

Також нами були досліджені часова, ємнісна, комунікаційна складності отриманого алгоритму та верхня оцінка числа переходу по ребрах, які здійснюють АД.

Список літератури:

1. Goth G. Exploring new frontiers. *Communications of the ACM*. 2009. 52(11). P. 21–23.

2. Banfi, J., Quattrini Li, A., Rekleitis, I. et al. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *Autonomous Robots*. 2018. 42 P. 875–894. https://doi.org/10.1007/s10514-017-9652-y

3. Nagavarapu, S.C., Vachhani, L., Sinha, A. et al. Generalizing Multi-agent Graph Exploration Techniques. *International Journal of Control, Automation and Systems*. 2020. https://doi.org/10.1007/s12555-019-0067-8

4. 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*. 2016. 83. P. 503–523 https://doi.org/10.1007/s10846-015-0309-9

5. Wang H., Jenkin M., Dymond P. 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. 2009. P. 55–60.

6. Zhang C. Parallelizing Depth-First Search for Robotic Graph Exploration. Harvard College, Cambridge, Massachusetts. 2010.

7. Stepkin A.V. Using a collective of agents for exploration of undirected graphs. *Cybernetics and Systems Analysis*. 2015. T.51, №2. P. 223–233.

8. Shannon C.E. Presentation of a maze-solving machine. Cybernetics Trans, of the 8 th Conf. of the JosiahMacy Jr. Found / Editor: H. Foerster. 1951. P. 173–180.

9. Dopp K. Automaten in labirinthen. *Electronische Informationsverarbeitung und Kybernetik*. 1971. V.7, № 2. P. 79–94.

10. Dudek G., Jenkin M. Computational principles of mobile robotics. *Cambridge Univ. press*. 2000. 280 p.

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

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

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

14. Стьопкін А.В., Кіт М.Ю. Алгоритми розпізнавання графів колективом агентів. *Збірник наукових праць фізико-математичного факультету ДДПУ*. 2023. Вип. 13. С. 85–89.

15. Стьопкін А.В., Кіт М.Ю. Алгоритми розпізнавання графів колективом агентів та ознайомлення з ними в позакласній роботі з інформатики в закладах загальної середньої освіти. Матеріали математичного семінару до 85-річного ювілею академіка НАН України А.М. Самойленка (02.01.1938–04.12.2020). – Магдебург. – 2023. – С. 88.

Stopkin A.V. ALGORITHM FOR EXPLORATION OF A SIMPLE UNDIRECTED GRAPH BY A COLLECTIVE OF AGENTS

The article is devoted to the problem of a simple graphs exploration by a collective of agents. The article proposes a new algorithm for graphs exploration by a collective of agents, which consist of two agents-researchers with finite memory (the memory of the agents does not exceed the number of nodes of the explored graph), which use two different colors (in total three colors) for a graph exploration and one agent-experimenter, which is located outside an explored graph and has a finite, infinitely growing internal memory. The agents-researchers can move around the graph, read and change the labels of the graph elements, and also transmit the necessary information to the agent-experimenter. An agent-experimenter is a stationary agent in whose memory the result of the functioning of agents-researchers is fixed at each step and the representation of the investigated graph is gradually built up with a list of edges and a list of nodes.

The article describes the modes of operation of the agents-researchers with the order of their activation, and the messages that the agents exchange with each other in the process of work are discussed in detail. The complete algorithm of the agent-experimenter work is presented with a detailed description of the procedures for processing the messages received from the agents-researchers, on the basis of which a researched graph is explored (construction a map of a graph). The article also analyzes the time, capacity, and communication complexity of the built algorithm, as well as analyzes the number of transitions along edges performed by agents-researchers.

It was found that the presented algorithm of a graph exploration has quadratic (from the number of nodes of the explored graph) time, capacity, and communication complexities, as well as the number of transitions along the edges performed by agents-researchers. The algorithm of a graph exploration is based on the depth-first traversal method.

Key words: *graph exploration, finite simple graphs, complexity of the algorithm, depth-first traversal method, collective of agents.*