УДК 519.17

Стёпкин А.В.

[1] *заведующий лабораториями факультета экономики и управления, ГВУЗ «ДГПУ»*

e-mail: stepkin.andrey@rambler.ru

ВОЗМОЖНОСТЬ И СЛОЖНОСТЬ РАСПОЗНАВАНИЯ ГРАФОВ КОЛЛЕКТИВОМ АГЕНТОВ

В работе рассматривается задача распознавания конечного графа коллективом агентов. Два агента-исследователя одновременно передвигаются по графу, считывают и изменяют отметки на элементах графа, передают необходимую информацию агенту-экспериментатору, который и распознает исследованный граф. Предложен алгоритм кубической (от числа вершин графа) временной и квадратической емкостной сложностей, который распознает любой конечный неориентированный граф без петель и кратных ребер. Для распознавания графа каждому агенту необходимо 2 различные краски (всего 3 краски). Метод основан на методе обхода графа в глубину.

Ключевые слова: *распознавание графа, обход в глубину, агент-исследователь, агент-экспериментатор.*

Введение

В наше время актуальным вопросом кибернетики является развитие такого направления, как теория дискретных динамических систем. В общей схеме Глушкова – Летичевского эта система представляется в виде модели взаимодействия управляющей и управляемой систем. Подобное взаимодействие рассматривается в [1], в предположении, что один агент-исследователь (АИ) передвигается по неизвестному графу и обменивается данными с агентом-экспериментатором (АЭ), который восстанавливает граф по данным, полученным от АИ.

В данной работе, интересующее нас взаимодействие, рассмотрено в предположении, что два АИ перемещаются по неизвестной среде, заданной конечным неориентированным графом [2]. АИ могут окрашивать вершины графа, ребра и инциденторы, воспринимать эти отметки и на их основании осуществлять перемещение. Информацию о своих действиях они передают АЭ, который и строит представление исследуемого графа.

Полученный алгоритм использует результаты и обозначения из [3].

Основные определения и обозначения

В работе рассматриваются конечные, неориентированные графы без петель и кратных ребер. Пусть $G = (V, E)$ – граф, где V – множество вершин, E – множество ребер (двухэлементных подмножеств (v, u), где $v, u \in V$). Тройку $((v, u), u)$ будем называть инцидентором ребра (v, u) и вершины u. Множество таких троек обозначим I. Множество $L = V \cup E \cup I$ назовем множеством элементов графа G. Функцией раскраски графа G назовем отображение $\mu : L \to \{w, r, y, ry, b\}$, где w интерпретируется как белый цвет, r – красный, y – желтый, ry – красно - желтый b – черный. Пара (G, μ) называется раскрашенным графом. Последовательность $u_1, u_2, ...u_k$ попарно смежных вершин называется путем в графе G, а k – длинной пути. При условии $u_1 = u_k$ путь называется циклом. Окрестностью $Q(v)$ вершины v будем называть множество элементов графа, состоящее из вершины v, всех вершин u смежных с v, всех ребер (v, u) и всех инциденторов $((v, u), v)$, $((v, u), u)$. Мощность множеств вершин V и ребер E обозначим через n и m соответственно. Ясно что $m \leqslant \frac{n(n-1)}{2}$. Изоморфизмом графа G и графа H назовем такую биекцию $\varphi : V_G \to V_H$, что $(v, u) \in E_G$ точно тогда, когда $(\varphi(v), \varphi(u)) \in E_H$. Таким образом, изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов.

Агенты A и B передвигаются по графу из вершины v в вершину u по ребру (v,u), могут изменять окраску вершин v, u, ребра (v,u), инциденторов $((v, u), v)$,$((v, u), u)$. Находясь в вершине v, АИ воспринимает метки всех элементов окрестности $Q(v)$. И на основании этих меток определяет, по какому ребру будет перемещаться, и как будет окрашивать элементы графа. АЭ передает, принимает и идентифицирует сообщения, полученные от АИ, обладает конечной, неограниченно растущей внутренней памятью, в которой фиксируется результат функционирования АИ на каждом шаге и строится представление графа G, вначале неизвестного агентам, списками ребер и вершин.

Основная часть

Целью работы является разработка такого алгоритма работы агентов, что АИ, будучи помещены в произвольные не совпадающие вершины неизвестного агентам конечного графа G, все элементы которого окрашены цветом w, через конечное число шагов обойдут его, передавая АЭ информацию о своих действиях. АЭ используя эту информацию, восстановит граф H, изоморфный G, то есть распознает граф G.

Рассмотрим подробнее алгоритмы функционирования агентов.

Алгоритм работы агента A:

1. Агент A красит $(\mu(v) := r)$;
2. запрос AN;
3. *if* $AN \neq 1$ *then do*
4. запрос BN;
5. *if* $BN = 0$ *then* $МЕТИМ\_ПЕР\_A(v)$;
6. *else* $ВЫБОР\_ХОДА\_A(v)$;
7. *end do*;
8. *else* $РАСП\_ПЕР\_A(v)$;

Все, процедуры, которые не описаны ниже, представлены в [3].

$РАСП\_ПЕР\_A(v)$:

1. $Z := K$;
2. *if* в $O(v)$ нет ребра, у которого $(\mu(v,u) = y)$ *then do*
3. агент A выполняет $ОТСТУП\_A(v)$;
4. *go to* 2 данной процедуры;
5. *end do*;
6. *else do*
7. *if* $(((K = Z)\, or\, (K = 1))\, and\, (Z \neq 1))$ *then* $РАСП\_ABB(v)$;
8. *else* $РАСП\_ABBb(v)$;
9. запрос K;
10. *if* $K \neq 0$ *then go to* 2 данной процедуры;
11. *else* агент A выполняет процедуру $ОБН\_A(v)$;
12. *if* $Z \neq 1$ *then do*
13. *if* в $O(v)$ есть ребро, у которого $(\mu\,((v,u),v) = r)\, and\, (\mu\,(v,u) = b)\, and\, (\mu\,((v,u),u) = r)$ *then do*
14. агент A выполняет процедуру $ВПЕРЕД\_AR\_N(v)$;
15. *go to* 13 данной процедуры;
16. *end do*;
17. *else if* в $O(v)$ есть ребро, у которого $(\mu\,(v,u) = r)\, and\, (\mu\,(u) = r)\, and\, (\mu\,((v,u),u) = r)$ *then do*
18. агент A выполняет процедуру $ВПЕРЕД\_\ AR(v)$;
19. *go to* 17 данной процедуры;
20. *end do*;
21. *else go to* 2 алгоритма обхода;
22. *end do*;
23. *else go to* 17 данной процедуры;
24. *end do*;

При выполнении процедуры $НАЗАД\_A(v)$, агент A выбирает из окрестности $O(v)$ произвольное ребро (v,u), для которого выполняется условие $((\mu(v,u)=r)\,and\,(\mu((v,u),v)=r))\,and\,(\mu(v)=r)$, и переходит по нему в вершину u. При этом, окрашивает $\mu(v):=b,\ \mu(v,u):=b,\ \mu((v,u),v):=b$, выполняет присваивание $v:=u$ и записывает в список M сообщение: $НАЗАД\_A$.

$РАСП\_A(v)$:

1. Агент A выбирает из окрестности $O(v)$ ребро (v,u), у которого $(\mu(v)=\mu(u)=r)\,and\,(\mu(v,u)=w)$ и переходит по нему в вершину u;
2. агент A красит $\mu(v,u):=b$;
3. агент A записывает в список M сообщение: $ОБРАТНОЕ\_РЕБРО\_A$;
4. *while* в $O(u)$ есть ребро (u,l), у которого $(\mu(u,l)=r)\,and\,(\mu((u,l),l)=r)\,and\,(\mu(l)=r)$ *do*
5. агент A переходит по ребру (u,l) в вершину l;
6. $u:=l$;
7. агент A записывает в список M: $ОТСТУПИЛ\_A$;
8. *end do*;
9. агент A записывает в список M сообщение: $РЕБРО\_РАСПОЗНАНО\_A$;

При выполнении процедуры $РАСП\_ABB(v)$, агент A выбирает из окрестности $O(v)$ ребро (v,u), для которого выполняется условие $\mu(v,u)=y$ и переходит в вершину u, окрашивая $\mu(v,u):=r,\ \mu((v,u),u):=b$. Выполняет $v:=u$ и записывает в список M сообщение: $ВПЕРЕД\_ABB$. После чего агент A выбирает из окрестности $O(v)$ ребро (v,u), у которого $((\mu(v,u)=r)\,and\,(\mu((v,u),v)=b))$, и переходит по нему в вершину u, окрашивая $\mu((v,u),v):=r,\ \mu(v,u):=b, \mu((v,u),u):=r$, выполняет присваивание $v:=u$ и записывает в список M сообщение: $НАЗАД\_ABB$.

Процедура $РАСП\_ABBb(v)$ аналогична процедуре $РАСП\_ABB(v)$, с отличием в том, что выполняя возврат по перешейку в свою область, агент A окрашивает ребро и дальний инцидентор следующим образом $\mu(v,u):=b,\ \mu((v,u),u):=b$.

Выполняя процедуру $ВПЕРЕД\_AR\_N(v)$, агент A выбирает из окрестности $O(v)$ произвольное ребро (v,u), для которого выполняется условие $(\mu((v,u),v)=r)\,and\,(\mu(v,u)=b)\,and\,(\mu((v,u),u)=r)$, переходит по нему в вершину u, окрашивая $\mu((v,u),v):=b,\ \mu((v,u),u):=b$. После чего выполняет присваивание $v:=u$ и записывает в список M сообщение: $ВПЕРЕД\_AR\_N$.

При выполнении процедуры $СТОП\_A(v)$, агент A окрашивает $\mu\ (v) := b$ и завершает работу.

Алгоритм работы агента B:

1. Агент B красит $(\mu\,(s) := y)$;
2. запрос BN;
3. *if* $BN \neq 1$ *then do*
4. Запрос AN;
5. *if* $\mu(s) = ry$ *then do*
6. агент B выполняет процедуру $ВОЗВРАТ\_B\,(s)$;
7. агент B выполняет процедуру $МЕТИМ\_ПЕР\_B\,(s)$;
8. *end do*;
9. *else if* $AN = 0$ *then* $МЕТИМ\_ПЕР\_B\,(s)$;
10. *else* $ВЫБОР\_ХОДА\_B\,(s)$;
11. *end do*;
12. *else* $РАСП\_ПЕР\_B\,(s)$;

Процедуры агента B, не рассмотренные ниже, аналогичны процедурам агента A.

При выполнении процедуры $МЕТИМ\_ПЕР\_B(s)$, агент B проверяет наличие в $O\,(s)$ ребра (v, u), для которого выполняется условие $(\mu\,(s{,}z) = w)\,and\,((\mu\,(z) = r)\,or\,(\mu\,(z) = ry))$ (1). Если такое ребро обнаружено и в вершине z стоит агент A, то агент B выполняет процедуру $СТОИТ\_B(s)$ и возвращается в строку 7 алгоритма обхода. Если в вершине z нет агента A, то агент B выполняет процедуру $МЕТИМ\_BA\,(s)$ и возвращается в начало рассматриваемой процедуры. Если в $O\,(s)$ нет ребра, удовлетворяющего условию (1), то агент B запрашивает значение переменной L. При этом: если $L = 0$, то агент B выполняет процедуру $ВЫБОР\_ХОДА\_B\,(s)$, иначе агент B выполняет процедуру $ФИКС\_B\,(s)$ и возвращается в строку 2 алгоритма обхода.

$ВЫБОР\_ХОДА\_B\,(s)$:

1. *if* в $O\,(s)$ обнаружено ребро, у которого $(\mu\,(s{,}z) = w)\,and\,(\mu\,(z) = \mu\,(s) = y)$ *then do*
2. агент B выполняет процедуру $РАСП\_B(s)$;
3. *go to* 2 алгоритма обхода;
4. *end do*;
5. *else if* в $O\,(s)$ обнаружено ребро, у которого $(\mu\,(s{,}z) = w)\,and\,(\mu\,(z) = w)$ *then do*
6. агент B выполняет процедуру $ВПЕРЕД\_B(s)$;
7. *go to* 2 алгоритма обхода;

8. *end do*;
9. *else if* в $O\,(s)$ есть ребро, у которого $(\mu\,(s,z) = w)\, and\, ((\mu\,(z) = r)\, or\, (\mu\,(z) = ry))$ *then do*
10. агент B выполняет процедуру $СТОИТ\_B(\,s)$;
11. *go to* 2 алгоритма обхода;
12. *end do*;
13. *else if* в $O\,(s)$ есть ребро, у которого $(\mu\,(s, z) = r)$ *then do*
14. агент B выполняет процедуру $СТОИТ\_B(\,s)$;
15. *go to* 2 алгоритма обхода;
16. *end do*;
17. *else if* в $O\,(s)$ есть ребро, у которого $((\mu\,(s,z) = y)\, and\, ((\mu\,(z) = r)\, or\, (\mu\,(z) = ry)))$ *then do*
18. агент B выполняет процедуру $СТОИТ\_B(\,s)$;
19. *go to* 4 алгоритма обхода;
20. *end do*;
21. *else if* в $O\,(s)$ есть ребро, у которого $(\mu\,(s, z) = y)\, and\, (\mu\,(s) = y)\, and\, (\mu\,((s, z)\,, s) = y)$ *then do*
22. агент B выполняет процедуру $НАЗАД\_B(\,s)$;
23. *go to* 2 алгоритма обхода;
24. *end do*;
25. *else* агент B выполняет процедуру $СТОП\_B$;

Выполняя процедуру $МЕТИМ\_BA\,(s)$, агент B выбирает из окрестности $O\,(s)$ произвольное ребро (s, z), для которого выполняется условие $((\mu\,(s, z) = w)\, and\, ((\mu\,(z) = r)\, or\, (\mu\,(z) = ry)))$, переходит по нему в вершину z, окрашивая $\mu\,(s, z) := y, \mu\,((s, z)\,, z) := y$, выполняет присваивание $s := z$ и записывает в список N: *ВПЕРЕД_ВА*. Далее B выбирает из окрестности $O\,(s)$ ребро, у которого $((\mu\,(s,\mathrm{z}) = y)\, and\, ((\mu(s) = r)\, or\, (\mu(s) = ry)))$, переходит по нему в вершину z, выполняет $s := z$ и записывает в список N сообщение: *НАЗАД_ВА*.

При выполнении процедуры $ВОЗВРАТ\_B\,(s)$, агент B выбирает из $O\,(s)$ ребро, у которого $(\mu\,(s, z) = y)\, and\, (\mu\,((s, z)\,, s) = y)$, переходит по нему в вершину z, выполняет присваивание $s := z$ и записывает в список N: *ВОЗВРАТ_B*.

Алгоритм «Восстановление» и процедуры, не рассмотренные ниже, изложены в [3] с поправкой, что при использовании цикла с предусловием, условие имеет вид: $(M \neq \varnothing)\, or\, (N \neq \varnothing)$.

$ОБР\_СП\_А()$:
1. *if* Mes = «ВПЕРЕД_А» *then* $ВПЕРЕД\_А()$;
2. *if* Mes = «ВПЕРЕД_АВ» *then* $ВПЕРЕД\_АВ()$;
3. *if* Mes = «ВПЕРЕД_АВВ» *then* $ВПЕРЕД\_АВВ()$;
4. *if* Mes = «НАЗАД_А» *then* $НАЗАД\_А()$;
5. *if* Mes = «НАЗАД_АВ» *then* $НАЗАД\_АВ()$;
6. *if* Mes = «НАЗАД_АВВ» *then* $НАЗАД\_АВВ()$;
7. *if* Mes = «ФИКС_А» *then* $ФИКС\_А()$;
8. *if* Mes = «ОБН_А» *then* $ОБН\_А()$;
9. *if* Mes = «ОТСТУПИЛ_А» *then* $ОТСТУПИЛ\_А()$;
10. *if* Mes = «РЕБРО_РАСПОЗНАНО_А» *then* $РЕБРО\_РАСПОЗНАНО\_А()$;
11. *if* Mes = «ОТСТУП_А» *then* $ОТСТУП\_А()$;

$ОТСТУП\_А()$: $i := i + 1$.
$ВПЕРЕД\_АВВ()$: $E_H := E_H \cup \{(N\_B, r(t-i))\}$.
$ОТСТУПИЛ\_А()$: $i := i + 1$.
$РЕБРО\_РАСПОЗНАНО\_А()$: $E_H := E_H \cup \{(r(t), r(t-i))\}$; $i := 0$.

Процедуры работы со списком команд агента B, которые не рассмотрены ниже, аналогичны процедурам работы со списком команд агента A. Процедура $ОБР\_СП\_В()$ аналогична рассмотренной $ОБР\_СП\_А()$, только добавлено условие:
if Mes=«ВОЗВРАТ_В» *then* ВОЗВРАТ_В().
$ВОЗВРАТ\_В()$: $E_H := E_H \backslash \{(y(p-1), y(p))\}$; $V_H := V_H \backslash \{Сч\_В\}$; $Сч\_В := Сч\_В - 2$; $p := p - 1$; $y(p) := Сч\_В$; $L := 1$; $K := K + 1$.

Рассмотрим основные свойства алгоритма. В начале алгоритма, при $n \geqslant 3$, как минимум, по одному разу выполнятся следующие процедуры: $ВПЕРЕД\_А(v)$, $ВПЕРЕД\_А()$ и $ВПЕРЕД\_В(s)$, $ВПЕРЕД\_В()$. При выполнении процедур $ВПЕРЕД\_А(v)$ и $ВПЕРЕД\_В(s)$ АИ посещают новую вершину графа G. Процедурами агента АЭ $ВПЕРЕД\_А()$ и $ВПЕРЕД\_В()$ создается новая вершина графа H. При одновременном попадании двух АИ в одну белую вершину процедурами $ВПЕРЕД\_А()$ и $ВПЕРЕД\_В()$ будет создано две новые вершины графа H. Одна из вершин (дублирующая вершину созданную агентом A) будет удалена командой $ВОЗВРАТ\_В()$. Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\varphi : V_G \to V_H$ вершин графа G в вершины графа H. Причем $\varphi(v) = t$ (когда вершина v окрашена в красный цвет и $t = Сч\_А$) и $\varphi(s) = p$ (когда вершина s окрашена в желтый цвет и $p = Сч\_В$). Указанное отображение естественным образом устанавливает неявную нумерацию вершин графа G.

Более того, отображение φ является биекцией, поскольку в связном графе G все вершины достижимы из начальных вершин. Поэтому все вершины посещаются агентами, то есть окрашиваются в красный и желтый цвета.

Из описания алгоритма следует, что агенты АИ проходят все ребра графа G, поскольку при окончании алгоритма все ребра становятся черными. При выполнении процедуры *ВПЕРЕД_А*() или *ВПЕРЕД_В*() АЭ распознает древесное ребро (v, u) и так нумерует вершину u, что ребру (v, u) однозначно соответствует ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *РЕБРО_РАСПОЗНАНО_А*() или *РЕБРО_РАСПОЗНАНО_В*() агент АЭ распознает обратное ребро (v, u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *ФИКС_А*(), *ВПЕРЕД_АВВ*() или процедур *ФИКС_В*(), *ВПЕРЕД_ВАА*(), АЭ распознает перешеек (v, u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. Следовательно, φ является изоморфизмом графа G на граф H.

Таким образом, выполняя алгоритм распознавания, агенты распознают любой граф G с точностью до изоморфизма.

Подсчитаем временную и емкостную сложность в равномерной шкале [4]. Для этого рассмотрим свойства красного и желтого путей. Из описания алгоритма следует, что на каждом шаге алгоритма красный (желтый) путь – это простой путь, соединяющий начальную вершину v (вершину s – в случае агента B) с номером $\varphi(v) = 1$ $(\varphi(s) = 2)$ с вершиной u (z) с номером $\varphi(u) = Сч\_A$ $(\varphi(z) = Сч\_B)$. Следовательно, общая длина красного и желтого пути не превосходит n.

При выполнении процедур *ВПЕРЕД_А*(v), *ВПЕРЕД_В*(s) и *НАЗАД_А*(v), *НАЗАД_В*(s) агенты АИ проходят одно ребро. При выполнении процедур *РАСП_А*(v), *РАСП_В*(s) агенты АИ проходят одно обратное ребро и не более $n-2$ (изначально одна вершина уже окрашена в «чужой» цвет) ребер красного (желтого) пути. При выполнении процедур *РАСП_А*(v), *РАСП_В*(s) агенты АИ проходят фактически цикл, состоящий из обратного ребра и некоторого конечного отрезка красного (желтого) пути, соединяющего вершины инцидентные обратному ребру. При выполнении процедур *МЕТИМ_АВ*(v), *МЕТИМ_ВА*(s) и *РАСП_АВВ*(v), *РАСП_АВВb*(v), *РАСП_ВАА*(s), *РАСП_ВААb*(s) оба АИ проходят один и тот же перешеек, сначала в одном направлении, а потом в обратном. Выполняя процедуры *ВПЕРЕД_AR*(v), *ВПЕРЕД_BR*(s) и процедуры *ОТСТУП_А*(v), *ОТСТУП_В*(s) агенты АИ проходят одно красное (желтое) ребро. При выполнении процедур *ВПЕРЕД_AR_N*(v), *ВПЕ-*

РЕД_BR_N (s) АИ проходят одно черное ребро. При выполнении процедур *ФИКС_A* (v) *ФИКС_B* (s) и *ОБН_A* (v) *ОБН_B* (s) агенты АИ не передвигаются, а только делают записи в свой список команд для АЭ, на что так же уходит один ход.

При подсчете временной сложности алгоритма будем считать, что инициализация алгоритма, анализ окрестности $O(v)$ рабочей вершины и выбор одной из возможных процедур занимают некоторое постоянное число единиц времени. Так же будем считать, что выбор ребер, проход по ним АИ и обработка команд АЭ полученных на данном этапе от АИ осуществляется за 1 единицу времени. Тогда временная сложность алгоритма определяется следующими соотношениями:

1. Инициализация выполняется один раз и ее асимптотическая сложность равна $O(1)$;
2. процедуры *ВПЕРЕД_A*(v) и *ВПЕРЕД_B*(s) выполняются не более чем $n-1$ раз, и общее время их выполнения оценивается как $O(n)$;
3. аналогично общее время выполнения процедур *НАЗАД_A*(v) и *НАЗАД_B*(s), оценивается как $O(n)$;
4. на выполнение процедур *МЕТИМ_AB*(v) и *МЕТИМ_BA*(s) уходит время, которое оценивается как $2 \times O(n) \times n$, то есть как $O\left(n^2\right)$;
5. на выполнение процедур *РАСП_ABB* (v), *РАСП_ABBb* (v) и процедур *РАСП_BAA* (s), *РАСП_BAAb* (s) уходит время, оцениваемое как $O\left(n^2\right)$;
6. процедуры *ВПЕРЕД_AR*(v) и *ВПЕРЕД_BR*(s) выполняются за время, оцениваемое как $O(n) \times n$, то есть как $O\left(n^2\right)$;
7. процедуры *ВПЕРЕД_AR_N*(v) и *ВПЕРЕД_BR_N*(s) выполняются за время, оцениваемое как $O(n)$;
8. аналогично процедуры *ОТСТУП_A*(v) и *ОТСТУП_B*(s) выполняются за время, оцениваемое как $O(n) \times n$, то есть как $O\left(n^2\right)$;
9. время, затрачиваемое на процедуры *ФИКС_A*(v) и *ФИКС_B*(s), оценивается как $O(n)$;
10. время, затрачиваемое на процедуры *ОБН_A*(v) и *ОБН_B*(s), оценивается как $O(n)$;
11. выполнение процедур *РАСП_A*(v) и *РАСП_B*(s), оценивается как $O(n) \times m$, то есть как $O\left(n^3\right)$;
12. время выполнения процедур *СТОИТ_A*(v) и *СТОИТ_B*(s) в общей сложности для всех четырёх возможных случаев, оценивается как $O(n) + O\left(n^2\right) = O\left(n^2\right)$.

Следовательно, суммарная временная сложность $T(n)$ алгоритма удовлетворяет соотношению: $T(n) = O\left(n^3\right)$.

Емкостная сложность $S(n)$ алгоритма определяется сложностью списков $V_H, E_H, r(1)...r(t), y(1)...y(p)$, сложность которых соответственно определяется величинами $O(n), O\left(n^2\right), O(n), O(n)$. Следовательно: $S(n) = O\left(n^2\right)$.

Таким образом, временная сложность алгоритма распознавания равна $O\left(n^3\right)$, а емкостная - $O\left(n^2\right)$. При этом алгоритм использует 3 краски.

Теорема 1. *Выполняя полученный алгоритм распознавания, агенты распознают любой граф G с точностью до изоморфизма.*

Теорема 2. *Временная сложность алгоритма равна $O\left(n^3\right)$, а емкостная – $O\left(n^2\right)$, где n – число вершин графа. При этом используется 3 краски.*

Заключение

Предложен алгоритм распознавания графа среды временной сложности $O\left(n^3\right)$ и емкостной – $O\left(n^2\right)$. АИ имеют память, ограниченную n, и используют по две краски. Алгоритм показывает, что при распознавании графа двумя АИ асимптотические временная и емкостная сложности остаются такими же, как при распознавании графа одним АИ [1]. Временная сложность, в лучшем случае, может быть понижена в 2 раза.

На основе данного исследования автор надеется создать, более эффективные алгоритмы, которые позволят улучшить результаты, полученные с помощью аналогичного алгоритма [3].

Литература

[1] *Грунский И.С.* Распознавание конечного графа блуждающим по нему агентом / И.С. Грунский, Е.А. Татаринов // Вестник Донецкого университета. Серия А. Естественные науки. — 2009. — Вып. 1. — С. 492 – 497.

[2] *Кудрявцев В.Б.* Введение в теорию автоматов / В.Б. Кудрявцев, С.В. Алешин, А.С. Подкозлин. — М.: Наука, 1985. — 320 с.

[3] *Грунский И. С.* Распознавание конечного графа коллективом агентов / И.С. Грунский, А.В. Стёпкин // Труды ИПММ НАН Украины. — 2009. — Т. 19. — С. 43 – 52.

[4] *Ахо А.* Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман — М.: Мир, 1979. — 536 с.