УДК 519.1

Стёпкин А.В.

[1] *заведующий лабораториями факультета экономики и управления, СГПУ*

e-mail: stepkin.andrey@rambler.ru

АЛГОРИТМ РАСПОЗНАВАНИЯ ГРАФА ТРЕМЯ АГЕНТАМИ

Рассматривается проблема распознавания конечных неориентированных графов тремя агентами. Получен алгоритм распознавания, временная и емкостная сложности которого равны $O(n^2)$. При работе два агента, передвигающиеся по графу, используют по две различные краски (всего три краски).

Ключевые слова: *распознавание графа, обход в глубину, обратное ребро, перешеек.*

Введение

Актуальным направлением математической кибернетики является теория дискретных динамических систем [1]. В общей схеме Глушкова – Летичевского эта система представлена в виде модели взаимодействия управляющей и управляемой систем [2]. Ранее подобное взаимодействие было рассмотрено в [3], в предположении, что оно представлено поочередным перемещением двух агентов-исследователей (АИ) по неизвестной среде, заданной конечным графом [2], и обменом данными с агентом-экспериментатором (АЭ), который восстанавливает исследуемый граф, по данным полученным от АИ.

Данная работа посвящена исследованию рассматриваемого взаимодействия, в предположении, что оно представлено процессом одновременного перемещения двух АИ A и B по неизвестной среде, заданной конечным графом (АИ могут окрашивать вершины, ребра и инциденторы графа, воспринимать эти отметки и на их основании осуществлять перемещение), и обменом данными с АЭ (восстанавливает граф, по данным полученным от АИ и передает АИ информацию, необходимую им для функционирования). В работе предложен алгоритм построения маршрутов АИ по графу, позволяющих АЭ точно восстановить граф среды. Для этого у каждого АИ есть две краски: у A это r и b, у B – y и b. Алгоритм основан на методе обхода графа в глубину. При его описании используются результаты и обозначения из [3].

Основным результатом исследования является оптимизация алгоритма распознавания перешейков и обратных ребер, что позволило, в итоге, улучшить временную сложность с $O\left(n^3\right)$ [3] до $O\left(n^2\right)$, где n – число вершин графа.

Основные определения и обозначения

В работе рассматриваются конечные, неориентированные графы без петель и кратных ребер. Понятия, которые не рассмотрены ниже, общеизвестны и с ними можно ознакомиться в [4-6] 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 инцидентов $((v, u), v), ((v, u), u)$. Мощность множеств вершин V и ребер E обозначим через n и m соответственно. Ясно что $m \leq \frac{n(n-1)}{2}$. 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), инцидентов $((v, u), v)$,$((v, u), u)$. 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.

Основна частина

Стратегия решения задачи аналогична стратегии, рассмотренной в [7], со значительными изменениями в режимах работы АИ с перешейками и обратными ребрами. Рассмотрим эти режимы более подробно.

1) Если, при движении вперед, в вершине v было обнаружено обратное ребро, то АИ прекращает работу в обычном режиме и переключается в режим *распознавания обратных ребер*. АИ красит в «свой» цвет ближние

инциденторы всех обратных ребер инцидентных вершине v. Завершив покраску инциденторов, АИ передвигается назад по своему пути, до обнаружения вершины инцидентной помеченному обратному ребру (под помеченным обратным ребром понимается белое ребро, у которого дальний инцидентор и дальняя вершина окрашены в «свой» цвет), переходит по этому ребру, окрашивая его в черный цвет. На этом этапе возможны два случая: 1.1) Распознаны не все, помеченные рассматриваемым АИ, обратные ребра. В этом случае АИ возвращается назад по пройденному на предыдущем шаге ребру, окрашивая в черный цвет ближний инцидентор, и продолжает движение назад по своему пути, до обнаружения следующей вершины, инцидентной помеченному обратному ребру. 1.2) Распознаны все, помеченные рассматриваемым АИ, обратные ребра. В этом случае АИ окрашивает ближний инцидентор ребра, по которому он перешел на предыдущем шаге, в черный цвет и завершает работу в режиме распознавания обратных ребер.

2) Если в процессе обхода графа в вершине v был обнаружен перешеек, то при условии, что все ранее помеченные данным АИ перешейки были распознаны, агент переключается в *режим пометки перешейков*. В этом режиме АИ окрашивает ближние инциденторы всех перешейков, инцидентных вершине v, в черный цвет. В данном режиме работы агент A имеет приоритет над агентом B, поэтому в ситуации, когда оба агента одновременно обнаружат один и тот же перешеек, он будет помечен агентом A. На каждом шаге АИ обмениваются данными с АЭ. По завершению режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков.

3) Получив от АЭ команду о необходимости распознавания перешейков, АИ переключается в *режим распознавания перешейков*. Если в этот момент агент работает в режиме распознавания обратного ребра, то АИ переключится в режим распознавания перешейков, лишь по завершению распознавания обратного ребра. При переключении в этот режим АИ проверяет наличие из вершины, в которой он находится, других возможных путей перемещения кроме как движение назад по своему пути. Если такие пути есть, то АИ возвращается назад по своему пути, ничего не окрашивая, до обнаружения ближайшей вершины, инцидентной помеченному перешейку (под помеченным перешейком понимается белое ребро, у которого дальний инцидентор окрашен в черный цвет, а дальняя вершина окрашена в «чужой» цвет). Если же таких путей нет, то возвращаясь назад по своему пути, АИ окрашивает его в черный цвет до тех пор, пока не появится вершина с другими возможными путями перемещения, далее АИ возвращается назад по своему пути до обнаружения ближайшей вершины, инцидентной помеченному перешейку. При

обнаружении помеченного перешейка возможны два случая: 3.1) *Помечено один перешеек.* АИ окрашивает ближний инцидентор помеченного перешейка в черный цвет. Далее движется вперед по своему пути, пока не вернется в вершину, в которой было произведено переключение в режим распознавания перешейков. 3.2) *Помечено не менее двух перешейков.* АИ окрашивает ближний инцидентор помеченного перешейка в «свой» цвет. Далее АИ движется назад по своему пути, пока не будет найден следующий помеченный перешеек. При обнаружении помеченного перешейка возможно два варианта: 3.2.1) *Следующий помеченный перешеек не последний.* АИ окрашивает ближний инцидентор в черный цвет. На следующем шаге АИ снова возвращается назад по своему пути до следующего помеченного перешейка. 3.2.2) *Следующий помеченный перешеек последний.* АИ окрашивает ближний инцидентор в «свой» цвет. На следующем шаге АИ переходит по последнему перешейку в чужую область, окрашивая ближний инцидентор в черный цвет. На следующем шаге АИ переходит по первому распознанному перешейку в свою область, окрашивая дальний инцидентор в черный цвет. Далее АИ возвращается в вершину, в которой было произведено переключение в режим распознания перешейков.

4) *Одновременное попадание двух АИ в одну белую вершину.* При одновременном попадании двух АИ в одну белую вершину, каждый АИ окрашивает вершину наполовину, и она становится красно-желтой. Агент B на следующем шаге отступает назад по своему пути. При этом удаляет краску с ближнего инцидентора и ребра, окрашивает дальний инцидентор в черный цвет и переходит в режим пометки перешейков (при этом ребро, по которому он вернулся уже посчитано как первый перешеек, а длина желтого пути уменьшена на одну вершину). Агент A видит разноцветную вершину как свою, но при распознавании окрашивает в черный цвет обе половинки.

Алгоритмы обхода и восстановления. Распознавание графа проводится посредством двух типов алгоритмов: «Обход» и «Восстановление». Первый тип алгоритма описывает обход неизвестного графа G агентами-исследователями, с целью проведения серии элементарных экспериментов и обменом данными с АЭ. Второй тип алгоритма описывает анализ агентом-экспериментатором результатов элементарных экспериментов и их объединение, в результате которого будет построен граф H, изоморфный распознаваемому графу G.

Алгоритм работы агента A: *Вход*: граф G неизвестный АИ и АЭ, все элементы графа G окрашены краской w, агент A помещен в произвольную вершину v.

Выход: все элементы графа G (кроме перешейков, обнаруженных в процессе работы агентов-исследователей), которые попадут в область работы агента A, окрашены краской b, агент A находится в исходной вершине v, пошагово выданы команды АЭ.

Данные: v – рабочая вершина графа G, в которой находится агент 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_*$A(v)$; *end do*;
7. *else* *РАСП_ПЕР_*$A(v)$;

Рассмотрим процедуры, используемые в данном алгоритме.

*МЕТИМ_ПЕР_*$A(v)$:

1. *if* в $Q(v)$ есть ребро (v,u), у которого
$(\mu((v,u),v) = w)\, and\, (\mu((v,u),u) = w)\, and\, (\mu(u) = y)$ *then do*
2. агент A выполняет процедуру *МЕТИМ_АВ*(v);
3. *go to* 1 данной процедуры; *end do*;
5. *else do*
6. запрос E;
7. *if* $E = 0$ *then do* *ВЫБОР_ХОДА_*$A(v)$;
8. *else do*
9. агент A выполняет процедуру *ФИКС_*$A(v)$;
10. *go to* 2 алгоритма обхода; *end do*;
11. *end do*;

При выполнении процедуры МЕТИМ_АВ(v), агент A выбирает из окрестности $Q(v)$ ребро (v,u), которое удовлетворяет следующему условию $(\mu((v,u),v) = w)\, and\, (\mu((v,u),u) = w)\, and\, (\mu(u) = y)$, окрашивает ближний инцидентор $\mu((v,u),v) := b$ и записывает в список M сообщение: МЕТИМ_АВ.

*ВЫБОР_ХОДА_*$A(v)$:

1. *if* в $Q(v)$ есть ребро, у которого $(\mu(v,u) = w) and (\mu(u) = \mu(v) = r)$ *then do*
2. агент A выполняет процедуру *РАСП_*$A(v)$;
3. *go to* 2 алгоритма обхода; *end do*;
4. *else if* в $Q(v)$ есть ребро, у которого $(\mu(v,u) = w)\, and\, (\mu(u) = w)$ *then do*
5. агент A выполняет процедуру *ВПЕРЕД_*$A(v)$;
6. *go to* 2 алгоритма обхода; *end do*;
7. *else if* в $Q(v)$ есть ребро, у которого

$(\mu((v,u),v)=w)\,and\,(\mu((v,u),u)=w)\,and\,(\mu(u)=y)\,then\ do$
8. агент A выполняет процедуру $СТОИТ\_A(v)$;
9. *go to* 2 алгоритма обхода; *end do*;
10. *else if* в $Q(v)$ есть ребро, у которого
$((\mu((v,u),v)=w)and(\mu((v,u),u)=b)and(\mu(u)=y))or(\mu(v,u)=y)\ \ then\ do$
11. агент A выполняет процедуру $СТОИТ\_A(v)$;
12. *go to* 2 алгоритма обхода; *end do*;
13. *else if* в $Q(v)$ есть ребро, у которого
$(\mu(v)=r)\,and\,(\mu((v,u),v)=r)\,and\,(\mu(v,u)=r)\ \ then\ do$
14. агент A выполняет процедуру $НАЗАД\_A(v)$;
15. *go to* 2 алгоритма обхода; *end do*;
16. *else* агент A выполняет процедуру $СТОП\_A(v)$;

$РАСП\_A(v)$:
1. *while* в $Q(v)$ есть ребро, у которого $((\mu(v)=\mu(u)=r)and(\mu(v,u)=w))\ \ do$
2. агент A красит $(\mu((v,u),v)):=r$;
3. агент A записывает в M: МЕТКА_ОР_А; *end do*;
4. агент A выбирает из окрестности $Q(v)$ ребро (v,u), у которого $(\mu((v,u),v)=r)and(\mu(v,u)=r)and(\mu(u)=r)$, переходит по нему в вершину u;
5. $v:=u$;
6. агент A записывает в M: ОТСТУП_А;
7. *if* в окрестности $Q(v)$ нет ребра (v,u), у которого $(\mu(v,u)=w)and$ $and(\mu((v,u),u)=r)and(\mu(v)=\mu(u)=r)\ \ then\ go\ to$ 4 данной процедуры;
8. *else do*
9. агент A переходит по ребру (v,u), красит $\mu(v,u):=b$;
10. $v:=u$;
11. агент A записывает в список M: ВПЕРЕД_ОР_А; *end do*;
12. запрос $UDOBR\_A$;
13. *if* $UDOBR\_A$ = TRUE *then do*
14. агент A выбирает из $Q(v)$ ребро (v,u), у которого $(\mu(v,u)=b)\,and$ $and\,(\mu((v,u),v)=r)\,and\,(\mu(v)=\mu(u)=r)$ и красит $\mu((v,u),v):=b$;
15. агент A записывает в список M: РЕБРА_РАСПОЗНАНЫ_А; *end do*;
16. *else do*
17. агент A выбирает из $Q(v)$ ребро (v,u), у которого $(\mu(v,u)=b)and$ $and(\mu((v,u),v)=r)and(\mu(v)=\mu(u)=r)$ и переходит по нему в вершину u;
18. агент A красит $\mu((v,u),v):=b$;
19. $v:=u$;
20. *go to* 4 данной процедуры; *end do*;

При выполнении процедуры $ВПЕРЕД\_A(v)$, агент A выбирает из

окрестности $Q(v)$ ребро (v,u), у которого $(\mu(v,u) = w) and (\mu(u) = w)$ и переходит по нему в вершину u. При этом окрашивает $\mu(v,u) := r$, $\mu((v,u),u) := r, \mu(u) := r$, выполняет присваивание $v := u$ и записывает в список M сообщение: ВПЕРЕД_А.

Выполняя процедуру $СТОИТ\_A(v)$, агент A не выполняет никаких действий.

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

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

Выполняя процедуру $ФИКС\_A(v)$, агент A записывает в список M сообщение: $ФИКС\_A$.

$РАСП\_ПЕР\_A(v)$:

1. *if* в $Q(v)$ не обнаружено ребра, у которого
   $(\mu((v,u),v) = w)\, and\, (\mu((v,u),u) = b)\, and\, (\mu(v,u) = w)$ *then do*
2. *if* в $Q(v)$ обнаружено ребро, у которого $((\mu(v,u) = w) and (\mu(u) = w)) or$
   $or((\mu(v,u) = r) and (\mu((v,u),u) = r)) or ((\mu(v,u) = w) and (\mu(v) = \mu(u) = r))\, or$
   $or((\mu((v,u),u) = w) and (\mu(u) = y)) or ((\mu((v,u),v) = r) and$
   $and (\mu((v,u),u) = b) and (\mu(v,u) = w))$ *then do*
3. агент A выполняет процедуру $ОТСТУП\_A(v)$;
4. *go to 1* данной процедуры; *end do*;
5. *else do*
6. агент A выполняет процедуру $НАЗАД\_A(v)$;
7. *go to 1* данной процедуры; *end do*;
8. *end do*;
9. *else do*
10. запрос UDP_B;
11. *if* $UDP\_B = TRUE$ *then* $РАСП\_AB(v)$;
12. *else* $РАСП\_ABb(v)$;
13. запрос K;
14. *if* $K \neq 0$ *then go to* 1 данной процедуры;
15. *else do*
16. агент A выполняет процедуру $ОБН\_A(v)$;
17. *if* в $Q(v)$ обнаружено ребро, у которого

$(\mu(v,u)=w)\,and((((\mu((v,u),v)=r)\,and\,(\mu((v,u),u)=b))\,or$
$or\,((\mu((v,u),v)=b)\,and\,(\mu((v,u),u)=r)))\;\;then\;do$

18. агент A выполняет процедуру $ВПЕРЕД\_AR\_N(v)$;
19. *go to* 17 данной процедуры; *end do*;
20. *if* в $Q(v)$ есть ребро, у которого
$(\mu(v,u)=r)\,and\,(\mu((v,u),u)=r)\,and\,(\mu(u)=r)\;\;then\;do$
21. агент A выполняет процедуру $ВПЕРЕД\_AR(v)$;
22. *go to* 20 данной процедуры; *end do*;
23. *else go to* 2 алгоритма обхода; *end do*;
24. *end do*;

Выполняя процедуру $ОТСТУП\_A(v)$, агент A выбирает из окрестности $Q(v)$ ребро (v,u), у которого $(\mu((v,u),v)=r)\,and\,(\mu(v,u)=r)$ и переходит по нему в вершину u, выполняет присваивание $v:=u$ и записывает в список M сообщение: ОТСТУП_А.

При выполнении процедуры $РАСП\_AB(v)$, агент A выбирает из окрестности $Q(v)$ произвольное ребро (v,u), для которого выполняется условие $(\mu((v,u),v)=w)\,and\,(\mu((v,u),u)=b)\,and\,(\mu(v,u)=w)$, окрашивает $\mu((v,u),v):=r$ и записывает в список M сообщение: РАСП_АВ.

Процедура $РАСП\_ABb(v)$ аналогична процедуре $РАСП\_AB(v)$, с отличием лишь в том, что инцидентор окрашивается следующим образом $\mu((v,u),v):=b$.

При выполнении процедуры $ОБН\_A(v)$, агент A записывает в список M сообщение: ОБН_А;

Выполняя процедуру $ВПЕРЕД\_AR\_N(v)$, агент A выбирает из окрестности $Q(v)$ произвольное ребро (v,u), которое удовлетворяет условию $(\mu(v,u)=w)and(((\mu((v,u),v)=r)and(\mu((v,u),u)=b))or((\mu((v,u),v)=b)and(\mu((v,u),u)=r)))$, переходит по нему в вершину u, окрашивая $\mu((v,u),v):=b, \mu((v,u),u):=b$ и выполняет присваивание $v:=u$.

При выполнении процедуры $ВПЕРЕД\_AR(v)$, агент A выбирает из окрестности $Q(v)$ произвольное ребро (v,u), для которого выполняется условие $(\mu(v,u)=r)\,and\,(\mu((v,u),u)=r)\,and\,(\mu(u)=r)$, переходит по нему в вершину u и выполняет присваивание $v:=u$.

Алгоритм работы агента B: *Вход*: граф G неизвестный АИ и АЭ, все элементы графа G окрашены краской w, агент B помещен в произвольную вершину s.

Выход: все элементы графа G (кроме перешейков, обнаруженных в процессе работы агентов-исследователей), которые попадут в область работы агента B, окрашены краской b, агент B находится в исходной вершине s, пошагово

выданы команды АЭ.

Данные: s – рабочая вершина графа G, в которой находится агент 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_В*(s);
7. агент B выполняет процедуру *МЕТИМ_ПЕР_В*(s); *end do*;
8. *else if* $AN = 0$ *then МЕТИМ_ПЕР_В*(s);
9. *else ВЫБОР_ХОДА_В*(s); *end do*;
10. *else РАСП_ПЕР_В*(s);

Процедуры агента B, которые не рассматриваются ниже, аналогичны процедурам агента A.

Выполняя процедуру *ВОЗВРАТ_В*(s), агент B выбирает из окрестности $Q(s)$ ребро, у которого $(\mu((s,z),s) = y)\,and\,(\mu(s,z) = y)$ и переходит по нему в вершину z, удаляя краску $(\mu((s,z),s) := w)\,and\,(\mu(s,z) := w)$. При переходе по ребру, агент B окрашивает $\mu((s,z),z) := b$, выполняет присваивание $s := z$ и записывает в список N сообщение: ВОЗВРАТ_В.

МЕТИМ_ПЕР_В (s):

1. *if* в $Q(s)$ есть ребро (s,z), у которого $(\mu((s,z),s)=w)and(\mu((s,z),z)=w)and$ $and((\mu(z)=r)or(\mu(z)=ry))$ *then do*
2. *if* в вершине z ребра (s,z) находится агент A *then do*
3. агент B выполняет процедуру *СТОИТ_В*(s);
4. *go to* 1 данной процедуры; *end do*;
5. *else do*
6. агент B выполняет процедуру *МЕТИМ_ВА*(s);
7. *go to* 1 данной процедуры; *end do*;
8. *end do*;
9. *else do*
10. запрос L;
11. *if* $L = 0$ *then ВЫБОР_ХОДА_В*(s);
12. *else do*
13. агент B выполняет процедуру *ФИКС_В*(s);
14. *go to* 2 алгоритма обхода; *end do*;
15. *end do*;

Выполняя процедуру *МЕТИМ_ВА*(s), агент B выбирает из окрестности $Q(s)$ ребро (s,z), у которого $(\mu((s,z),s)=w)and(\mu((s,z),z)=w)and((\mu(z)=r)or$

$or(\mu(z)=ry))$, окрашивает $\mu((s,z),s):=b$ и записывает в список N сообщение: МЕТИМ_ВА.

ВЫБОР_ХОДА_B(s):

1. *if* в $Q(s)$ есть ребро, у которого $(\mu(s,z)=w)and(\mu(s)=\mu(z)=y)$ *then do*
2. агент B выполняет процедуру *РАСП_B*(s);
3. *go to* 2 алгоритма обхода; *end do*;
4. *else if* в $Q(s)$ обнаружено ребро, у которого $(\mu(s,z)=w)and(\mu(z)=w)$ *then do*
5. агент B выполняет процедуру *ВПЕРЕД_B*(s);
6. *go to* 2 алгоритма обхода; *end do*;
7. *else if* в $Q(s)$ есть ребро, у которого $(\mu((s,z),s)=w)and(\mu((s,z),z)=w)and$ $and((\mu(z)=r)or(\mu(z)=ry))$ *then do*
8. агент B выполняет процедуру *СТОИТ_B*(s);
9. *go to* 2 алгоритма обхода; *end do*;
10. *else if* в $Q(s)$ есть ребро, у которого $(\mu(s,z)=w)and(\mu((s,z),s)=w)and$ $and(\mu((s,z),z)=b)$ *then do*
11. агент B выполняет процедуру *СТОИТ_B*(s);
12. *go to* 2 алгоритма обхода; *end do*;
13. *else if* в $Q(s)$ есть ребро, у которого $(\mu(s,z)=y)and(\mu(s)=y)and$ $and(\mu((s,z),s)=y)$ *then do*
14. агент B выполняет процедуру *НАЗАД_B*(s);
15. *go to* 2 алгоритма обхода; *end do*;
16. *else* агент B выполняет процедуру *СТОП_B*;

РАСП_ПЕР_B(s):

1. *if* в $Q(s)$ не обнаружено ребра, у которого $(\mu((s,z),s)=w)\,and$ $and\,(\mu((s,z),z)=b)\,and\,(\mu(s,z)=w)$ *then do*
2. *if* в $Q(s)$ обнаружено ребро, у которого $((\mu(s,z)=w)and(\mu(z)=w))or$ $or((\mu(s,z)=y)and(\mu((s,z),z)=y))or((\mu(s,z)=w)and(\mu(s)=\mu(z)=y))or$ $or((\mu((s,z),z)=w)and((\mu(z)=r)or(\mu(z)=ry)))or$ $or((\mu((s,z),s)=y)and(\mu((s,z),z)=b)and(\mu(s,z)=w))$ *then do*
3. агент B выполняет процедуру *ОТСТУП_B*(s);
4. *go to 1* данной процедуры; *end do*;
5. *else do*
6. агент B выполняет процедуру *НАЗАД_B*(s);
7. *go to 1* данной процедуры; *end do*;
8. *end do*;
9. *else do*
10. запрос UDP_B;
11. *if* $UDP\_B=TRUE$ *then РАСП_ВА*(s);

12. *else* $РАСП\_BAb(s)$;
13. запрос F;
14. *if* $F \neq 0$ *then go to* 1 данной процедуры;
15. *else do*
16. агент B выполняет процедуру $ОБН\_B(s)$;
17. *if* в $Q(s)$ обнаружено ребро, у которого
$(\mu((s,z),s) = w)\, and((\mu((s,z),s) = y)\, and\, (\mu((s,z),z) = b))\, or$
$or\, ((\mu((s,z),s) = b)\, and\, (\mu((s,z),z) = y)))$ *then do*
18. агент B выполняет процедуру $ВПЕРЕД\_BR\_N(s)$;
19. *go to* 17 данной процедуры; *end do*;
20. *if* в $Q(s)$ есть ребро, у которого
$(\mu(s,z) = y)\, and\, (\mu((s,z),z) = y)\, and\, (\mu(z) = y)$ *then do*
21. агент B выполняет процедуру $ВПЕРЕД\_BR(s)$;
22. *go to* 20 данной процедуры; *end do*;
23. *else go to* 2 алгоритма обхода; *end do*;
24. *end do*;

Выполняя процедуру $РАСП\_BA(s)$, агент B выбирает из окрестности $Q(s)$ произвольное ребро (s,z), для которого выполняется условие $(\mu((s,z),s) = w)\, and\, (\mu((s,z),z) = b)\, and\, ((\mu(s,z) = w))$, окрашивает $\mu((s,z),s) := y$ и записывает в список N сообщение: РАСП_ВА.

Процедура $РАСП\_BAb(s)$ аналогична процедуре $РАСП\_BA(s)$, с отличием лишь в том, что инцидентор окрашивается следующим образом $\mu((s,z),s) := b$.

Алгоритм восстановления: *Вход*: списки сообщений M и N от АИ. *Выход*: список вершин V_H и ребер E_H графа H, изоморфного графу G. *Данные*: V_H, E_H списки вершин и ребер графа H, изоморфного графу G. $Сч\_A$, $Сч\_B$ – счетчики числа посещенных вершин графа G агентами A и B соответственно. AN – переменная, в которой значение «1» дает агенту A сигнал для возврата и распознания помеченных агентом B перешейков, значение «0» позволяет агенту A работать дальше в обычном режиме. BN – переменная, в которой значение «1» дает агенту B сигнал для возврата и распознания помеченных агентом A перешейков, значение «0» позволяет агенту B работать дальше в обычном режиме. N_A, N_B – переменные, в которых хранятся номера вершин, из которых агенты A и B соответственно, последний раз помечали перешейки. F – количество перешейков из вершины N_A, помеченных для распознания. K – количество перешейков из вершины N_B, помеченных для распознания. M, N – списки сообщений от агентов A и B соответственно. E – переменная, в кото-

рой делается отметка о том, был ли на предыдущем шаге агентом A помечен перешеек (значение «1») или нет (значение «0»). L – переменная, в которой делается отметка о том, был ли на предыдущем шаге агентом B помечен перешеек (значение «1») или нет (значение «0»). i, j – счетчики используемые агентами A и B соответственно при подсчете номеров вторых вершин помеченных перешейков и номеров вторых вершин помеченных обратных ребер. $STOP\_A, STOP\_B$ – переменные, используемые агентами A и B соответственно, для сигнализации АЭ, о завершении распознавания своей области. UDP_A, UDP_B – логические переменные, используемые агентами A и B соответственно, для определения способа окраски инциденторов, рассматриваемого в определенный момент, перешейка. $UDOBR\_A, UDOBR\_B$ – логические переменные, используемые агентами A и B соответственно для определения является ли рассматриваемое обратное ребро последним из помеченных. $KOBR\_A, KOBR\_B$ – переменные, в которые агенты A и B соответственно записывают количество помеченных обратных ребер. $r(1), r(2), ..., r(t)$ – список номеров вершин красного пути, где t – длинна этого списка. $y(1), y(2), ..., y(p)$ – список номеров вершин желтого пути, где p – длинна этого списка. Mes – текущее сообщение.

1. $Сч\_A := 1$;
2. $Сч\_B := 2$;
3. $AN \coloneqq 0, BN \coloneqq 0, N\_A \coloneqq 0, N\_B \coloneqq 0, F \coloneqq 0, K \coloneqq 0, M \coloneqq \varnothing, N \coloneqq \varnothing, E \coloneqq 0,$ $L \coloneqq 0, i \coloneqq 0, j \coloneqq 0, E_H \coloneqq \varnothing, STOP\_A \coloneqq 0, STOP\_B \coloneqq 0, UDP\_A \coloneqq FALSE,$ $UDP\_B \coloneqq FALSE, UDOBR\_A \coloneqq FALSE, UDOBR\_B \coloneqq FALSE,$ $KOBR\_A \coloneqq 0, KOBR\_B \coloneqq 0$;
4. $t := 1$;
5. $p := 1$;
6. $r(t) :=$ $Сч\_A$;
7. $y(p) :=$ $Сч\_B$;
8. $V_H := \{1, 2\}$;
9. *while* $(STOP\_A = 0)$ *or* $(STOP\_B = 0)$ *do*
10. *if* $M \neq \varnothing$ *then do*
11. прочитать в Mes сообщение и удалить его из очереди M;
12. *ОБР_СП_A*(); *end do*;
13. *if* $N \neq \varnothing$ *then do*
14. прочитать в Mes сообщение и удалить его из очереди N;
15. *ОБР_СП_B*(); *end do*;
16. *end do*;
17. печать V_H, E_H.

ОБР_СП_А():
1. *if* $Mes =$ "ВПЕРЕД_А" *then* *ВПЕРЕД_А*();
2. *if* $Mes =$ "МЕТИМ_АВ" *then* *МЕТИМ_АВ*();
3. *if* $Mes =$ "НАЗАД_А" *then* *НАЗАД_А*();
4. *if* $Mes =$ "РАСП_АВ" *then* *РАСП_АВ*();
5. *if* $Mes =$ "ФИКС_А" *then* *ФИКС_А*();
6. *if* $Mes =$ "ОБН_А" *then* *ОБН_А*();
7. *if* $Mes =$ "РЕБРА_РАСПОЗНАНЫ_А" *then* *РЕБРА_РАСПОЗНАНЫ_А*();
8. *if* $Mes =$ "ОТСТУП_А" *then* *ОТСТУП_А*();
9. *if* $Mes =$ "МЕТКА_ОР_А" *then* *МЕТКА_ОР_А*();
10. *if* $Mes =$ "ВПЕРЕД_ОР_А" *then* *ВПЕРЕД_ОР_А*();
11. *if* $Mes =$ "СТОП_А" *then* *СТОП_А*().

ВПЕРЕД_А(): выполняются операции: $Сч\_А \coloneqq Сч\_А + 2$; $t \coloneqq t+1$; $r(t) \coloneqq Сч\_А$;
$V_H \coloneqq V_H \cup \{Сч\_А\}$; $E_H \coloneqq E_H \cup \{(r(t-1), r(t))\}$;
МЕТИМ_АВ(): $F := F + 1$; $E := 1$;
НАЗАД_А(): из списка $r(1), ..., r(t)$ удаляется элемент $r(t)$; $t := t - 1$;
РАСП_АВ(): $E_H := E_H \cup \{(N\_B, r(t-i))\}$; $K := K - 1$;
$UDP\_B := (((K = Z)\, or\, (K = 1))\, and\, (Z \neq 1))$;
ФИКС_А(): $N\_A := Сч\_А$; $BN := 1$; $E := 0$; $Q := F$;
$UDP\_A := (((F = Q)\, or\, (F = 1))\, and\, (Q \neq 1))$;
ОБН_А(): $AN := 0$; $i := 0$;
РЕБРА_РАСПОЗНАНЫ_А(): $i := 0$;
ОТСТУП_А(): $i := i + 1$;
МЕТКА_ОР_А(): $KOBR\_A := KOBR\_A + 1$;
ВПЕРЕД_ОР_А(): $KOBR\_A \coloneqq KOBR\_A - 1$; $UDOBR\_A \coloneqq (KOBR\_A = 0)$;
$E_H \coloneqq E_H \cup \{(r(t), r(t-i))\}$;
СТОП_А(): $STOP\_A := 1$;

Процедуры работы со списком команд от агента B, которые не рассмотрены ниже, аналогичны процедурам работы со списком команд от агента A.

ОБР_СП_В():
1. *if* $Mes =$ "ВПЕРЕД_В" *then* *ВПЕРЕД_В*();
2. *if* $Mes =$ "МЕТИМ_ВА" *then* *МЕТИМ_ВА*();
3. *if* $Mes =$ "НАЗАД_В" *then* *НАЗАД_В*();
4. *if* $Mes =$ "РАСП_ВА" *then* *РАСП_ВА*();
5. *if* $Mes =$ "ФИКС_В" *then* *ФИКС_В*();
6. *if* $Mes =$ "ОБН_В" *then* *ОБН_В*();
7. *if* $Mes =$ "РЕБРА_РАСПОЗНАНЫ_В" *then* *РЕБРА_РАСПОЗНАНЫ_В*();
8. *if* $Mes =$ "ОТСТУП_В" *then* *ОТСТУП_В*();

9. *if* $Mes =$ "МЕТКА_ОР_В" *then* *МЕТКА_ОР_В*();
10. *if* $Mes =$ "ВПЕРЕД_ОР_В" *then* *ВПЕРЕД_ОР_В*();
11. *if* $Mes =$ "ВОЗВРАТ_В" *then* *ВОЗВРАТ_В*();
12. *if* $Mes =$ "СТОП_В" *then* *СТОП_В*().

ВОЗВРАТ_В(): $E_H := E_H \backslash \{(y(p-1), y(p))\}$; $V_H := V_H \backslash \{ Сч\_B \}$;
$Сч\_B := Сч\_B - 2$; $p := p - 1$; $y(p) := Сч\_B$; $L := 1$; $K := K + 1$.

Свойства алгоритма распознавания.

В начале работы алгоритма распознавания, при $n \geq 3$, как минимум, по одному разу выполняются процедуры: ВПЕРЕД_А(v), ВПЕРЕД_А() и ВПЕРЕД_В(s), ВПЕРЕД_В(). Выполняя процедуры ВПЕРЕД_А(v) и ВПЕРЕД_В(s) АИ посещают белые вершины исследуемого графа G. Процедурами агента АЭ ВПЕРЕД_А() и ВПЕРЕД_В() создаются две новые вершины (по одной вершине для каждой из процедур) графа H.

При одновременном попадании двух АИ в одну белую вершину процедурами ВПЕРЕД_А() и ВПЕРЕД_В() будет создано две новые вершины графа H. Вершина созданная агентом B, на следующем шаге будет удалена командой ВОЗВРАТ_В(), так как она дублирует вершину, созданную агентом A. 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 $t = Сч\_A$ и $\varphi(s) = p$ (когда вершина s окрашена в желтый цвет и $p = Сч\_B$). 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 АИ проходят все ребра графа G, поскольку по 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 *ВПЕРЕД_ОР_А*() или *ВПЕРЕД_ОР_В*() АЭ распознает обратное ребро (v, u) графа G и ставит ему в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *РАСП_АВ*() или *РАСП_ВА*() 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.

Теорема 1. *Выполнив алгоритм распознавания, агенты распознают граф G с точностью до изоморфизма.*

Подсчитаем временную и емкостную сложности алгоритма в равномерной шкале [6]. Рассмотрим подробнее свойства красного и желтого путей. Из описания алгоритма следует, что на каждом шаге алгоритма красный (желтый) путь – это простой путь, соединяющий начальную вершину v (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_A(v), ВПЕРЕД_B(s) и НАЗАД_A(v), НАЗАД_B(s) АИ проходят одно ребро. При выполнении процедур РАСП_A(v), РАСП_B(s) АИ проходят не более $n-2$ (изначально одна вершина уже окрашена в «чужой» цвет) ребер красного (желтого) пути. При выполнении процедур МЕТИМ_AB(v), МЕТИМ_BA(s), РАСП_AB(v), РАСП_ABb(v), РАСП_BA(s) и РАСП_BAb(s) АИ не совершают перехода по перешейку, а просто окрашивают его ближний инцидентор. Выполняя процедуры ВПЕРЕД_AR(v), ВПЕРЕД_BR(s) и ОТСТУП_A(v), ОТСТУП_B(s) АИ проходят одно красное (желтое) ребро. При выполнении процедур ВПЕРЕД_AR_N(v), ВПЕРЕД_BR_N(s) АИ проходят один перешеек. При выполнении процедур ФИКС_A(v) ФИКС_B(s) и ОБН_A(v) ОБН_B(s) 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 $Q(v)$ 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. Процедуры ВПЕРЕД_A(v), ВПЕРЕД_B(s), НАЗАД_A(v) и НАЗАД_B(s) выполняются не более чем $2\times(n-1)$ раз, общее время их выполнения оценивается как $O(n)$. 2. На выполнение процедур МЕТИМ_AB(v), МЕТИМ_BA(s), РАСП_AB(v), РАСП_ABb(v), РАСП_BA(s) и РАСП_BAb(s) уходит время, которое оценивается как $2\times O(n)\times n$, то есть как $O\left(n^2\right)$. 3. Каждая из пар процедур ВПЕРЕД_AR(v), ВПЕРЕД_BR(s) и ОТСТУП_A(v), ОТСТУП_B(s) выполняются за время, оцениваемое как $O(n)\times n$, то есть как $O\left(n^2\right)$. 4. На выполнение процедур ВПЕРЕД_AR_N(v), ВПЕРЕД_BR_N(s), ФИКС_A(v), ФИКС_B(s), ОБН_A(v) и ОБН_B(s) уходит время, оцениваемое как $3\times O(n)$, то есть как $O(n)$. 5. Время, затрачиваемое на выполнение процедур РАСП_A(v) и РАСП_B(s), оценивается как $O(n)\times n$, то есть как $O\left(n^2\right)$. 6. Время вы-

полнения процедур СТОИТ_А(v) и СТОИТ_В(s) в общей сложности оценивается как $O(n)+O\left(n^2\right)=O\left(n^2\right)$. Следовательно, суммарная временная сложность $T(n)$ алгоритма удовлетворяет соотношению: $T(n)=O\left(n^2\right)$. 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: $S\,(n)=O\left(n^2\right)$.

Теорема 2. *Временная и емкостная сложности алгоритма равны* $O\left(n^2\right)$, *где* n *– число вершин графа, при этом алгоритм использует 3 краски.*

Заключение

В работе предложен новый алгоритм распознавания графа среды временной и емкостной сложностей $O\left(n^2\right)$. АИ имеют конечную память, независимую от n, и используют по две краски каждый (всего три краски).

Литература

[1] *Кудрявцев В.Б.* О поведении автоматов в лабиринтах / В.Б. Кудрявцев, Щ. Ушчумлич, Г. Калибарда // Дискретная математика. – 1992. – Т. 4, № 3. – С. 3 – 28.

[2] *Кудрявцев В.Б.* Введение в теорию автоматов / В.Б. Кудрявцев, С.В. Алешин, А.С. Подколзин. – 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] *Касьянов В. Н.* Графы в программировании: обработка, визуализация и применение / В.Н. Касьянов, В.А. Евстигнеев. – СПБ.: БХВ – Петербург, 2003. – 1104 с.

[5] *Кормен Т.* Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО, 2001. – 960 с.

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

[7] *Стёпкин А.В.* Распознавание конечных графов тремя агентами / А.В. Стёпкин // Искусственный интеллект. – 2011. – №2. – С. 84 – 93.