УДК 519.7

АЛГОРИТМ РАСПОЗНАВАНИЯ ГРАФОВ ТРЕМЯ АГЕНТАМИ

А. В. Стёпкин
Славянский государственный педагогический университет, г. Славянск

В работе рассматривается задача распознавания неизвестного конечного графа коллективом агентов. Два агента-исследователя одновременно передвигаются по графу, считывают и изменяют метки элементов графа, передают необходимую информацию агенту-экспериментатору, который строит представление исследуемого графа. Предложен алгоритм кубической (от числа вершин графа) временной и квадратической емкостной сложностей, который распознает любой конечный неориентированный граф. Для распознавания графа каждому агенту требуется 2 различные краски (всего 3 краски). Метод основан на методе обхода графа в глубину.

Ключевые слова: распознавание графа, обход графа, агенты.

Введение. Важной и актуальной проблемой компьютерной науки является проблема взаимодействия управляющей и управляемой систем [1]. Ранее подобное взаимодействие было рассмотрено в [2], в предположении, что оно представлено передвижением одного агента-исследователя (АИ) по неизвестному графу и обменом данными с агентом-экспериментатором (АЭ), который и производил восстановление графа по данным, полученным от АИ.

Целенаправленное перемещение агента в операционной среде (ОС) невозможно, пока не будет сформирована достаточно полная её модель. В вопросе моделирования ОС определен ряд подходов, одним из которых является топологический [3], при котором агенту доступна информация о связях между различными областями среды и недоступна метрическая и алгоритмическая информация о среде. Зачастую подобная ситуация возникает в роботике [4]. Топологическая модель представляет собой граф, оснащенный дополнительной информацией на ребрах, в вершинах и инцидентора.

Данная работа посвящена решению задачи, в предположении, что взаимодействие управляющей и управляемой систем представляется процессом перемещения двух АИ по неизвестной среде, заданной конечным графом [5]. А суть взаимодействия заключается в обмене данными АИ с АЭ, на основе которого возможно восстановление графа.

Основные определения и обозначения. Требуется разработать такой алгоритм функционирования АИ и АЭ, что АИ, будучи помещены в произвольные, но не совпадающие вершины произвольного неизвестного агентам-исследователям и агенту-экспериментатору графа G, все элементы которого окрашены цветом w, через конечное число шагов обойдут его, передавая АЭ информацию. АЭ, в свою очередь, используя эту информацию, через конечное число шагов восстановит граф H, изоморфный G, то есть распознает G.

В работе рассматриваются конечные, неориентированные графы без петель и кратных ребер. Все неопределяемые понятия общеизвестны, с ними можно ознакомиться, например, в [6-8]. Пусть $G=(V,E)$ – граф, у которого V – множество вершин, E - множество ребер (то есть множество двухэлементных подмножеств (u,v), где $u,v\in V$). Тройку $((u,v),v)$ будем называть инцидентором («точкой прикосновения») ребра (u,v) и вершины v. Множество таких троек обозначим I. Множество $L=V\cup E\cup I$ назовем множеством элементов графа G. Функцией раскраски графа G назовем отображение $\mu: L\to\{w,r,y,b\}$, где w интерпретируется как белый цвет (краска), r – красный, y – желтый, b – черный. Пару (G,μ) называют раскрашенным графом. Последовательность $u_1,u_2,...u_k$ попарно смежных вершин называют путем в графе G, а k - длиной пути. При условии $u_1=u_k$ путь называют циклом. Окрестностью $O(v)$ вершины v будем называть множество элементов графа, состоящее из вершины v, всех вершин u смежных с v, всех ребер (v,u) и всех инцидентов $((v,u),v),((v,u),u)$. Число вершин V и ребер E обозначим через n и m соответственно. Ясно что $m\le n(n-1)/2$. Изоморфизмом между графами G и H назовем такую биекцию $\phi: V_G\to V_H$, что $(v,u)\in E_G$ тогда и только тогда, когда $(\phi(v),\phi(u))\in E_H$. Таким образом, изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов.

Мобильные 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 вер-

© Стёпкин А. В.

шине v, агенты воспринимают метки всех элементов окрестности $O(v)$ и на их основании определяют по какому ребру (v,u) они будут перемещаться и как будут перекрашивать элементы $v,u,(v,u),((v,u),v),((v,u),u)$. АЭ передает, принимает и идентифицирует сообщения АИ, обладает конечной, неограниченно растущей внутренней памятью, в которой фиксируется результат функционирования АИ на каждом шаге, и, кроме того, постепенно выстраивается представление графа G (изначально неизвестного агентам) списками ребер и вершин.

Отметим, что работа алгоритма осуществляется следующим образом: АИ помещаются в различные вершины графа G, эти вершины сразу окрашиваются в красный и желтый цвета для агентов A и B соответственно. АИ передвигаются одновременно, пошагово передавая АЭ информацию, АЭ в свою очередь обрабатывает полученные от них данные.

Стратегия решения задачи. Рассматриваемый алгоритм основан на стратегии поиска в глубину. Эта стратегия такова: агенты идут «в глубину», пока это возможно, возвращаются назад, ищут другой путь с еще не посещенными вершинами и не пройденными ребрами. В случае обнаружения смежной вершины, окрашенной в «чужой» цвет, агент метит все перешейки из текущей вершины в чужую область и сообщает второму АИ, через агента-экспериментатора, о необходимости распознавания помеченных перешейков. Пока второй АИ распознает перешейки, первый АИ не может метить новые перешейки. В случае отсутствия других возможных вариантов перемещения, кроме как пометить новый перешеек, первый АИ останавливается до того момента, пока второй АИ не распознает все помеченные перешейки.

Предлагаемая стратегия обладает рядом особенностей: 1) Граф G агентам не известен. 2) При обходе графа G, агенты создают неявную нумерацию пройденных вершин: при первом посещении вершины она окрашивается агентом A в красный цвет (в желтый цвет в случае агента B) и ей фактически ставится в соответствие номер, равный значению переменной $Сч\_A$ для агента A ($Сч\_B$ для агента B). Отметим, что $Сч\_A$ и $Сч\_B$ принимают соответственно нечетные и четные значения. На основе нумерации и происходит восстановление графа путем построения графа H изоморфного G. В процессе обхода агенты строят неявные деревья поиска в глубину. Относительно этих деревьев все ребра разделяются на древесные (окрашиваются при первом прохождении по ним красным либо желтым цветом), обратные (не принадлежат дереву и окрашиваются при первом прохождении в черный цвет) и ребра перешейки (соединяют деревья, построенные разными агентами). Древесные ребра проходятся как минимум 2 раза и при последнем проходе окрашиваются агентами в черный цвет. Обратные ребра проходятся один раз. А вот ребра перешейки проходятся каждым агентом по два раза: первый прошедший по нему агент метит перешеек, окрашивая его в красный цвет в случае агента A (в желтый цвет в случае агента B), второй прошедший по нему агент распознает перешеек и красит его в черный цвет.

Красные (желтые) вершины графа G, на каждом шаге алгоритма, образуют красный (желтый) путь. При проходе в новую вершину красный (желтый) путь удлиняется, при проходе назад (в случае если это не возврат для распознания перешейков) – укорачивается, при распознавании обратного ребра и перешейков – не изменяется. Вершина, у которой все инцидентные ребра распознаны, окрашивается в черный цвет. Алгоритм заканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины черными.

Обход графа. В работе АИ можно выделить 5 режимов:

1*) Обычный режим.* АИ движется вперед по белым вершинам, окрашивая вершины, соединяющие их ребра и дальние инциденторы в «свой» цвет. Если нет возможных путей перемещения, то АИ возвращается назад, окрашивая пройденные вершины, ребра и ближние инциденторы в черный цвет. Вернувшись в начальную вершину, АИ завершает работу. На каждом шаге АИ обменивается данными с АЭ.

2) *Распознавание обратных ребер.* Если при движении вперед было обнаружено обратное ребро, то агент прекращает работу в обычном режиме. АИ переходит по обратному ребру, окрашивая его в черный цвет, и по своему пути возвращается в вершину, в которой сменил режим работы. Достигнув этой вершины, агент переключается в обычный режим работы. На каждом шаге АИ обменивается данными с АЭ.

3) *Пометка перешейков.* Если в процессе обхода графа 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 переходит по перешейку в чужую область, окрашивая ребро и дальний инцидентор в «свой» цвет. На следующем шаге возвращается по этому перешейку, ничего не окрашивая, и ищет другие перешейки из этой вершины. Пометив все перешейки, АИ сообщает об этом АЭ, который в свою очередь, дает команду второму АИ о необходимости распознавания помеченных перешейков. По завершению режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков.

4) *Распознавание перешейков.* 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 за-

вершению распознавания обратного ребра. В этом режиме АИ возвращается назад по своему пути до обнаружения ближайшей вершины, инцидентной помеченному перешейку. Под помеченным перешейком понимается ребро, у которого ближний инцидентор, ребро и дальняя вершина этого ребра окрашены в «чужой» цвет. Далее возможны два случая: *Помечено один перешеек.* АИ переходит по нему в чужую область, окрашивая перешеек в «свой» цвет, а дальний инцидентор в черный. На следующем шаге возвращается по этому перешейку в свою область, окрашивая дальний инцидентор и перешеек в черный цвет. Далее движется вперед по своему пути, пока не вернется в вершину, в которой было произведено переключение в режим распознавания перешейков. *Помечено не менее двух перешейков.* АИ переходит по первому найденному помеченному перешейку в чужую область, окрашивая его в «свой» цвет, а дальний инцидентор в черный. На следующем шаге АИ возвращается по этому перешейку в свою область, окрашивая его в черный цвет, а оба инцидентора в красный. Далее АИ движется назад по своему пути, пока не будет найден следующий помеченный перешеек. Далее возможно два варианта.

4.1. *Следующий помеченный перешеек не последний.* АИ переходит по найденному перешейку, окрашивая его в «свой» цвет, а дальний инцидентор в черный. На следующем шаге АИ возвращается по этому перешейку в свою область, окрашивая его и оба инцидентора в черный цвет. И снова возвращается назад по своему пути до следующего помеченного перешейка.

4.2. *Следующий помеченный перешеек последний.* АИ переходит по найденному перешейку, окрашивая его в «свой» цвет, а дальний инцидентор в черный. На следующем шаге АИ возвращается по этому перешейку в свою область, окрашивая его в черный цвет, а оба инцидентора в красный.

АИ переходит по последнему перешейку в чужую область, окрашивая инциденторы в черный цвет. На следующем шаге АИ переходит по первому распознанному перешейку в свою область, окрашивая инциденторы в черный цвет. Далее АИ возвращается в вершину, в которой было произведено переключение в режим распознания перешейков.

5) *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 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. Опишем подробно алгоритмы, реализующие описанную выше стратегию. Процесс распознавания состоит из двух принципиально разных типов алгоритмов: «Обход» и «Восстановление». Первый тип алгоритма описывает обход неизвестного графа G агентами-исследователями, с целью проведения серии элементарных экспериментов и передачи информации АЭ. Второй тип алгоритма представляет собой анализ результатов элементарных экспериментов и их объединение, в результате которого будет построен граф H, изоморфный распознаваемому графу G.

Рассмотрим непосредственно сами алгоритмы.

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, 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.

Алгоритм:

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_А*(v);
6. *else ВЫБОР_ХОДА_А*(v);
7. *end do*;
8. *else РАСП_ПЕР_А*(v);

Все, процедуры, которые не описаны ниже, представлены в [9].

РАСП_ПЕР_А(v):

1. $Z := K$;
2. *if* в $O(v)$ нет ребра, у которого $(\mu(v,u) = y)$ *then do*
3. Агент A выполняет *ОТСТУП_А*(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* агент A выполняет *РАСП_АВВ*(v);
8. *else* агент A выполняет процедуру *РАСП_АВВb*(v);
9. Агент A запрашивает у АЭ значение переменной K;
10. *if* $K\neq 0$ *then go to* 2 данной процедуры;
11. *else* Агент A выполняет процедуру *ОБН_А*(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_А*(v), агент A выбирает из окрестности $O(v)$ ребро, 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, производит окрашивание $\mu(v):=b,\ \mu(v,u):=b, \mu((v,u),v):=b$, выполняет присвоение $v:=u$ и записывает в список M сообщение: *НАЗАД_А*;

РАСП_А(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_А*;
4. *while* в $O(u)$ существует 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* сообщение: *ОТСТУПИЛ_А*;
8. *end do*;
9. Агент A записывает в список *M* сообщение: *РЕБРО_РАСПОЗНАНО_А*;

Выполняя процедуру *РАСП_АВВ*(v), агент A выбирает из окрестности $O(v)$ ребро (v,u), для которого выполняется условие $\mu(v,u)=y$ и переходит по нему в вершину u, производя окрашивание $\mu(v,u):=r,\ \mu((v,u),u):=b$. Выполняет присвоение $v:=u$ и записывает в список M сообщение: *ВПЕРЕД_АВВ*. После чего агент A выбирает из окрестности $O(v)$ ребро, для которого выполняется условие $((\mu(v,u)=r)and(\mu((v,u),v)=b))$, и переходит по нему в вершину u, окрашивая $\mu((v,u),v):=r, \mu(v,u):=b, \mu((v,u),u):=r$, выполняет присвоение $v:=u$ и записывает в список M сообщение: *НАЗАД_АВВ*.

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

При выполнении процедуры *ВПЕРЕД_AR_N*(v), агент A выбирает из окрестности $O(v)$ ребро, удовлетворяющее условию $(\mu((v,u),v)=r)and(\mu(v,u)=b)and(\mu((v,u),u)=r)$, переходит по нему в вершину u, производя окрашивание $\mu((v,u),v):=b,\ \mu((v,u),u):=b$. После чего выполняет присвоение $v:=u$ и записывает в список M сообщение: *ВПЕРЕД_AR_N*.

Выполняя процедуру *СТОП_А*(v), агент A окрашивает $\mu(v):=b$ и завершает работу.

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, 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.

Алгоритм:

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);
8. *end do*;
9. *else if* $AN = 0$ *then МЕТИМ_ПЕР_В*(s);
10. *else ВЫБОР_ХОДА_В*(s);
11. *end do*;
12. *else РАСП_ПЕР_В*(s);

Все процедуры агента B, которые не рассмотрены ниже, аналогичны процедурам агента A.

Выполняя процедуру *МЕТИМ_ПЕР_В*(s), агент B проверяет наличие в окрестности $O(s)$ ребра (s, z), 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 находится агент A, то агент B выполняет процедуру *СТОИТ_ В*(z) и возвращается в строку 7 АО. Если же в вершине z нет агента A, то агент B выполняет процедуру *МЕТИМ_ ВА*(z) и возвращается в начало данной процедуры.

Если в окрестности $O(s)$ не обнаружено ребра, удовлетворяющего условию (1), то агент B запрашивает значение переменной L. При этом: если $L = 0$, то агент B выполняет процедуру *ВЫБОР_ХОДА_В*(z), иначе агент B выполняет процедуру *ФИКС_В*(z) и возвращается в строку 2 АО.

ВЫБОР_ХОДА_В(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_В*(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_В*(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_В*(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_В*(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_В*(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_В*(s);
23. *go to* 2 алгоритма обхода;
24. *end do*;
25. *else* агент B выполняет процедуру *СТОП_В*;

Выполняя процедуру *МЕТИМ_ВА*(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 присвоение $s := z$ и записывает в список N сообщение: *ВПЕРЕД_ВА*. Далее агент B выбирает из окрестности $O(s)$ ребро (s, z), удовлетворяющее условию $((\mu(s, z) = y) and ((\mu(s) = r) or (\mu(s) = ry)))$, переходит по нему в вершину z, выполняет присвоение $s := z$ и записывает в список N сообщение: *НАЗАД_ВА*.

При выполнении процедуры *ВОЗВРАТ_В*(*s*) агент B выбирает из окрестности $O(s)$ ребро, для которого выполняется условие $(\mu(s,z)=y)and(\mu((s,z),s)=y)$, переходит по нему в вершину z, выполняет присвоение $s:=z$ и записывает в список N сообщение: *ВОЗВРАТ_В*.

Алгоритм «Восстановление» и процедуры, которые не рассмотрены ниже, изложены в [9] 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 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* = "*ВПЕРЕД_ВА*" *then ВПЕРЕД_ВА*();
3. *if Mes* = "*ВПЕРЕД_ВАА*" *then ВПЕРЕД_ВАА*();
4. *if Mes* = "*НАЗАД_В*" *then НАЗАД_В* ();
5. *if Mes* = "*НАЗАД_ВА*" *then НАЗАД_ВА*();
6. *if Mes* = "*НАЗАД_ВАА*" *then НАЗАД_ВАА*();
7. *if Mes* = "*ФИКС_В*" *then ФИКС_В*();
8. *if Mes* = "*ОБН_В*" *then ОБН_В*();
9. *if Mes* = "*ОТСТУПИЛ_В* " *then ОТСТУПИЛ_В*();
10. *if Mes* = "*РЕБРО_РАСПОЗНАНО_В*" *then РЕБРО_РАСПОЗНАНО_В*();
11. *if Mes* = "*ОТСТУП_В*" *then ОТСТУП_В*();
12. *if Mes* ="*ВОЗВРАТ_В*" *then ВОЗВРАТ_В*().

ВОЗВРАТ_В(): $L := 1; Сч\_B := Сч\_B - 2$;

Свойства алгоритма распознавания. В начале алгоритма, при $n \geq 3$, как минимум, по одному разу выполнятся 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 $t = Сч\_A$) 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 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 *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 [6]. 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 (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, потом в обратном. Выполняя процедуры *ВПЕРЕД_AR*(v), *ВПЕРЕД_BR*(s) и *ОТСТУП_А*(v), *ОТСТУП_В*(s) агенты АИ проходят одно красное (желтое) ребро. При выполнении процедур *ВПЕРЕД_AR_N*(v), *ВПЕРЕД_BR_N*(s) АИ проходят одно черное ребро. При выполнении процедур *ФИКС_А*(v) *ФИКС_В*(s) и *ОБН_А*(v) *ОБН_В*(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_А*(v) и *ВПЕРЕД_В*(s) выполняются не более чем $n-1$ раз, и общее время их выполнения оценивается как $O(n)$;
3. аналогично общее время выполнения процедур *НАЗАД_А*(v) и *НАЗАД_В*(s), оценивается как $O(n)$;
4. на выполнение процедур *МЕТИМ_АВ*(v) и *МЕТИМ_ВА*(s) уходит время, которое оценивается как $2\times O(n)\times n$, то есть как $O(n^2)$;
5. аналогично выполнение процедур *РАСП_АВВ*(v), *РАСП_АВВb*(v) и *РАСП_ВАА*(s), *РАСП_ВААb*(s) занимает время, оцениваемое как $O(n^2)$;
6. процедуры *ВПЕРЕД_AR*(v) и *ВПЕРЕД_BR*(s) выполняются за время, оцениваемое как $O(n)\times n$, то есть как $O(n^2)$;
7. процедуры *ВПЕРЕД_AR_N*(v) и *ВПЕРЕД_BR_N*(s) выполняются за время, оцениваемое как $O(n)$;
8. аналогично процедуры *ОТСТУП_А*(v) и *ОТСТУП_В*(s) выполняются за время, оцениваемое как $O(n)\times n$, то есть как $O(n^2)$;
9. время, затрачиваемое на процедуры *ФИКС_А*(v) и *ФИКС_В*(s), оценивается как $O(n)$;
10. время, затрачиваемое на процедуры *ОБН_А*(v) и *ОБН_В*(s), оценивается как $O(n)$;
11. выполнение процедур *РАСП_А*(v) и *РАСП_В*(s), оценивается как $O(n)\times m$, то есть как $O(n^3)$;
12. время выполнения процедур *СТОИТ_А*(v) и *СТОИТ_В*(s) в общей сложности для всех четырёх возможных случаев, оценивается как $O(n)+O(n^2)=O(n^2)$;

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

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 $O(n)$, $O(n^2)$, $O(n), O(n)$. Следовательно, $S(n) = O(n^2)$.

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

Выводы. В работе предложен новый алгоритм точного распознавания графа среды временной сложности $O(n^3)$, а емкостной - $O(n^2)$. АИ имеют память, ограниченную числом n, и используют по две краски каждый. Этот алгоритм показывает, что при распознавании графа двумя АИ асимптотическая временная и асимптотическая емкостная сложности остаются такими же, как при распознавании графа одним АИ [2]. Временная сложность алгоритма, в лучшем случае, может быть понижена в 2 раза.

На основе даного исследования предполагается создать новые, более эффективные алгоритмы, которые позволят улучшить результаты, полученные с помощью аналогичного алгоритма [9].

РЕЗЮМЕ

В роботі розглядається задача розпізнавання невідомого скінченного графа колективом агентів. Два агента-дослідника одночасно рухаються графом, зчитують та змінюють помітки елементів графа, передають необхідну інформацію агенту-експериментатору, який будує уявлення про досліджуваний граф. Запропоновано алгоритм кубічної (від числа вершин графа) часової та квадратичної ємнісної складностей, який розпізнає будь-який скінченний неорієнтований граф. Для розпізнавання графа кожному агенту необхідно 2 різні фарби (усього 3 фарби). Метод базується на методі обходу графа в глибину.

Ключові слова: розпізнавання графа, обхід графа, агенти.

SUMMARY

Problem of exploration finite unknown graph by a collective of agents is considered in this work. Two agents-researchers simultaneously move on graph, they read and change marks of graph elements, transfer the information to the agent-experimenter. It builds explored graph representation. The algorithm cubic (from amount of nodes of the graph) time and quadratic capacity complexities is proposed. It recognizes any finite non-oriented graph. For graph exploration each agent needs two different marks (in total three colors). The method is based on depth-first traversal method.

Keywords: graph exploration, graph traversal, agents.

СПИСОК ЛИТЕРАТУРЫ

1. Капитонова Ю. В. Математическая теория проектирования вычислительных систем / Ю. В. Капитонова, А. А. Летичевский. – М.: Наука, 1988. – 296 с.
2. Грунский И. С. Распознавание конечного графа блуждающим по нему агентом / И. С. Грунский, Е. А. Татаринов // Вестник Донецкого университета. Сер. А. Естественные науки. – 2009. – № 1. – С. 492-497.
3. Kuipers B. The spatial semantic hierarchy / B. Kuipers. // Artificial Intelligence. – 2000. – Vol. 119, No 1-2. – P. 191-233.
4. Dudek G. Computational principles of mobile robotics / G. Dudek, M. Jenkin. – Cambridge Univ. press, Cambridge. 2000. – 280 p.
5. Кудрявцев В. Б. Введение в теорию автоматов./ В. Б. Кудрявцев, С. В. Алешин, А. С. Подкозлин. – М.: Наука, 1985. – 320 с.
6. Ахо А. Построение и анализ вычислительных алгоритмов./ А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М.: Мир, 1979. – 536 с.
7. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р Ривест. – М.: МЦНМО, 2001. – 960 с.
8. Касьянов В. Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. – СПБ.: БХВ-Петербург, 2003. – 1104 с.
9. Стёпкин А. В. Распознавание конечного графа коллективом агентов / А. В. Стёпкин // Інформатика та комп'ютерні технології-2009: матеріали V міжнар. наук.-тех. конф. студентів, аспірантів та молодих вчених (м. Донецьк, 24-26 лист. 2009 р.). − Донецьк, 2009. – Т. 2 – С. 126-131.

Поступила в редакцию 18.10.2010 г.