УДК 519.6

©2009. **И.С. Грунский, А.В. Стёпкин**

РАСПОЗНАВАНИЕ КОНЕЧНОГО ГРАФА КОЛЛЕКТИВОМ АГЕНТОВ

Рассматривается задача распознавания неизвестного графа коллективом агентов. Два агента-исследователя передвигаются по графу, изменяют и считывают метки на элементах графа и передают информацию агенту-экспериментатору, который строит представление исследуемого графа. Предложен алгоритм, который распознает любой конечный неориентированный граф. Для распознавания графа агентам требуется 2 различные краски, кубическое (от числа вершин графа) число шагов и квадратичная память. Метод основан на методе обхода графа в глубину.

Введение. Основной проблемой компьютерной науки, является проблема взаимодействия управляющей и управляемой систем, т.е. взаимодействия управляющего автомата (агента) и его операционной среды [1]. В рассмотренном ниже случае взаимодействие этих систем представляется как процесс перемещения агентов по графу среды [2], который является топологической моделью среды [3]. В этом случае агенту доступна информация только о связях между различными областями среды и недоступна метрическая и алгоритмическая информация о данной среде. Зачастую подобная ситуация возникает в роботике [4]. Топологическая модель представляет собой граф с помеченными различными способами вершинами, дугами, инциденторами. Выделяют три основные задачи исследования среды агентом: 1) самолокализация агента; 2) контроль соответствия модели среды и самой среды; 3) построение агентом модели карты среды [4]. Значительное количество работ (см. обзоры [5–7]) посвящено задаче распознавания среды, известен ряд алгоритмов распознавания, получено много результатов о возможности, невозможности и сложности такого распознавания с помощью тех или иных средств агента. Однако на наш взгляд, мало исследована возможность и сложность распознавания графа более чем одним агентом.

В настоящей работе предложен новый алгоритм распознавания. В нем используется три агента: два агента-исследователя (АИ), которые перемещаются по графу, и агент-экспериментатор (АЭ), который принимает сообщения АИ и по ним строит представление исследуемого графа, основное внимание уделено анализу того, что и как распознается на каждом шаге алгоритма и выделению элементарного эксперимента. Алгоритм основан на методе обхода графа вглубь и при этом агенты-исследователи специальным образом окрашивают элементы графа. Алгоритм, на наш взгляд, является базовым и на его основе авторы надеются создать новые более эффективные алгоритмы, которые позволят улучшить результаты, полученные с помощью аналогичного алгоритма [8].

1. Основные определения и обозначения.

1.1. Раскрашенные графы. Рассматриваются конечные, неориентированные

графы без петель и кратных ребер. Все неопределяемые понятия общеизвестны, их можно найти, например, в [9-11]. Пусть $G = (V, E)$ – граф, у которого V – множество вершин, E – множество ребер, т.е. 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. Пара (G, μ) называется раскрашенным графом. Последовательность $u_1, u_2, ...u_k$ попарно смежных вершин называется путем в графе G, а k – длиной пути. При $u_1 = u_k$ этот путь называется 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 $((v, u), v), ((v, u), u)$. 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.

1.2. Мобильные агенты. Мобильные агенты A и B характеризуются следующими свойствами. Они передвигаются по графу из вершины v в вершину u по ребру (v, u). При этом агенты могут изменять окраску вершин v, u, ребра (v, u), инцидентов $((v, u), v)$,$((v, u), u)$. Находясь в вершине v агенты A и B воспринимают метки всех элементов окрестности $O(v)$ и на этом 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 вначале неизвестного агентам, списками ребер и вершин.

1.3. Постановка задачи. Требуется разработать такой алгоритм функционирования АИ(передвижения по графу, раскраски его элементов и передачи информации АЭ) и АЭ(восстановления графа по данным полученным от АИ), что АИ будучи помещены в произвольные вершины произвольного неизвестного АИ и АЭ графа G, все элементы которого окрашены цветом w, через конечное число шагов обойдут его, пошагово передавая АЭ информацию, по которой АЭ через конечное число шагов восстановит граф H, изоморфный G, т.е. распознает G.

2. Распознавание графа.

2.1. Стратегия распознавания графа. Предлагаемый алгоритм основан на стратегии поиска в глубину. Эта стратегия такова: агенты идут “вглубь”, 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 метим все перешейки из текущей вершины в чужую область и сообщаем второму АИ, через АЭ о необходимости вернуться и распознать помеченные перешейки, также передается информация о количестве помеченных

перешейков. Пока второй агент не распознает все перешейки, выбор дальнейшего пути перемещения первым агентом изменяется. В таком случае поиск перешейков выполняется непосредственно перед проверкой выполнения условий, связанных с распознаванием дерева, построенного данным АИ. Если отсутствуют все возможные варианты перемещения, наличие которых проверяется до проверки на наличие перешейка, то при обнаружении его, текущий агент останавливается до того момента, пока все помеченные перешейки не будут распознаны вторым агентом.

Известен ряд алгоритмов поиска в глубину для известного графа [9–11]. Предлагаемая ниже стратегия обладает рядом особенностей. 1) Граф G агентам не известен. 2) При прохождении вершин графа G агенты создают неявную нумерацию пройденных вершин: при первом посещении вершины она окрашивается красным цветом, если это агент A и желтым цветом в случае агента B и ей фактически ставится в соответствие номер, равный значению переменной $Сч\_A$ для агента A или $Сч\_B$ для агента B. Агент A ведет нумерацию нечетными числами, а агент B четными, другими словами переменные $Сч\_A$ и $Сч\_B$ принимают соответственно нечетные и четные значения. На основе этой нумерации и происходит восстановление графа путем построения графа H изоморфного G. В процессе поиска агенты строят неявные деревья поиска в глубину и относительно этих деревьев все ребра разделяются на древесные, т.е. принадлежащие деревьям и окрашиваемые при первом прохождении по ним в красный (желтый) цвет, обратные – не принадлежащие дереву и окрашиваемые при первом прохождении в черный цвет и ребра перешейки, которые соединяют деревья. Древесные ребра проходятся, по крайней мере, 2 раза и при последнем проходе окрашиваются агентом в черный цвет, обратные проходятся один раз, а ребра перешейки проходятся каждым агентом два раза, первый прошедший по нему агент метит перешеек, окрашивая его в красный цвет в случае агента A или в желтый для агента B, второй прошедший по нему агент распознает перешеек и красит его в черный цвет.

Древесные ребра распознаются агентами при первом проходе по ним. Красные (желтые) вершины графа G, на каждом шаге алгоритма, образуют красный (желтый) путь. С помощью этого пути распознаются обратные ребра, при проходе в новую вершину красный (желтый) путь удлиняется, при проходе назад (в случае если это не возврат для распознавания перешейков) – укорачивается, при распознавании обратного ребра – не изменяется. Вершина, у которой все инцидентные ребра распознаны красится черным. Алгоритм оканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины черными.

2.2. Распознавание графа. Весь процесс распознавания состоит из двух принципиально разных алгоритмов: 1) "Обход"; 2) "Восстановление". Первый алгоритм описывает обход неизвестного графа G агентами-исследователями, с целью проведения серии элементарных экспериментов и передачи информации АЭ. Второй алгоритм представляет собой анализ результатов элементарных экспериментов и их объединение, в результате которого будет построен граф H, изоморфный распознаваемому графу G. Стоит отметить, что совместная работа алгоритмов осуществляется в следующей последовательности: сначала ходит агент A, АЭ обрабатывает

полученные от него данные, далее ходит агент B, АЭ обрабатывает полученные данные и т.д. Агенты исследователи помещаются в различные вершины графа G, эти вершины сразу окрашиваются в красный и желтый цвет для агентов A и B соответственно.

2.3. 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 окрашены краской b, агент A находится в исходной вершине v;

Данные: v – рабочая вершина графа G, в которой находится агент.

Алгоритм:

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

Рассмотрим процедуры, используемые в данном алгоритме.

МЕТИМ_ ПЕР_ A(v)

1. if в $O(v)$ обнаружено ребро, у которого $(\mu(v,u) = w) and (\mu(u) = y)$ then do
2. Агент A выполняет процедуру $МЕТИМ\_AB(v)$;
3. go to 1 данной процедуры;
4. end do;
5. else do
6. Агент A запрашивает у АЭ значение переменной E;
7. if E=0 then $ВЫБОР\_ХОДА\_A(v)$;
8. else do
9. Агент A выполняет процедуру $ФИКС\_A(v)$;
10. go to 2 алгоритма обхода;
11. end do;
12. end do;

ВЫБОР_ ХОДА_ A(v)

1. if в $O(v)$ 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;
4. end do;
5. else if в $O(v)$ есть ребро, у которого $(\mu(v,u) = w) and (\mu(u) = w)$then do
6. Агент выполняет процедуру $ВПЕРЕД\_A(v)$;
7. go to 2 алгоритма обхода;
8. end do;
9. else if в $O(v)$ есть ребро, у которого$(\mu(v,u) = w) and (\mu(u) = y)$ then do

10. Агент выполняет процедуру $СТОИТ\_А\ (v)$;
11. go to 2 алгоритма обхода;
12. end do;
13. else if в $O(v)$ есть ребро, у которого $(\mu(v,u)=y)$ then do
14. Агент A выполняет процедуру $СТОИТ\_А(v)$;
15. go to 2 алгоритма обхода;
16. end do;
17. else if в $O(v)$ есть ребро, у которого $(\mu(v,u)=r)and(\mu(u)=y)$then do
18. Агент A выполняет процедуру $СТОИТ\_А\ (v)$;
19. go to 4 алгоритма обхода;
20. end do;
21. else if в $O(v)$ есть ребро,т.ч.$(\mu(v,u)=r)and(\mu(v)=r)and(\mu((v,u),v)=r)$ then do
22. Агент A выполняет процедуру $НАЗАД\_А(v)$;
23. go to 2 алгоритма обхода;
24. end do;
25. else агент A выполняет процедуру $СТОП\_А$;

РАСП_ ПЕР_ А(v)

1. if в $O(v)$ есть ребро, у которого $(\mu(v,u)=y)$ then do
2. Агент A выполняет процедуру $РАСП\_АВВ(v)$;
3. Агент A запрашивает у АЭ значение переменной K;
4. if $K\neq 0$ then go to 1 данной процедуры;
5. else do
6. Агент A выполняет процедуру $ОБН\_А(v)$;
7. if в $O(v)$ есть ребро,т.ч.$(\mu(v,u)=r)and(\mu(u)=r)and(\mu((v,u),)=r)$ then do
8. Агент A выполняет процедуру $ВПЕРЕД\_АR(v)$;
9. go to 7 данной процедуры;
10. end do;
11. else go to 2 алгоритма обхода;
12. end do;
13. end do;
14. else do
15. Агент A выполняет процедуру $ОТСТУП\_А(v)$;
16. Перемещаемся в строку 1 данной процедуры;
17. end do;

ВПЕРЕД_ А(v): агент A выбирает из $O(v)$ ребро (v,u) у которого $\mu(v,u)=w$, $\mu(u)=w$, переходит по нему в вершину u, окрашивая $\mu(v,u):=r$, $\mu(u):=r$, $\mu((v,u),u):=r$, выполняет $v:=u$ и записывает в список M: ВПЕРЕД_А;

НАЗАД_ А(v): агент A выбирает из $O(v)$ произвольное ребро (v,u) у которого $((\mu(v,u)=r)and(\mu((v,u),v)=r)and(\mu(v)=r)$ и переходит по нему в вершину u окрашивая $\mu(v):=b,\mu(v,u):=b$, выполняет $v:=u$ и записывает в M: НАЗАД_А;

СТОИТ_ А(v): агент A не выполняет никаких действий, запись в M: СТОИТ_А;

МЕТИМ_ АВ(v): агент A выбирает из $O(v)$ произвольное ребро (v,u), у которого $(\mu(v,u)=w)and(\mu(u)=y)$, переходит по нему в вершину u, окрашивая $\mu(v,u):=r$, $\mu((v,u),u):=r$, выполняет $v:=u$ и записывает в M: ВПЕРЕД_АВ, на следующем шаге A выбирает из $O(v)$ ребро (v,u), у которого $((\mu(v,u)=r)and(\mu(v)=y))$, и переходит по нему в вершину u, выполняет $v:=u$ и записывает в M: НАЗАД_АВ;
РАСП_ АВВ(v): агент A выбирает из $O(v)$ ребро (v,u), у которого $\mu(v,u)=y$, и переходит по нему в вершину u, окрашивая $\mu(v,u):=r, \mu((v,u),u):=b$, выполняет $v:=u$ и записывает в список M: ВПЕРЕД_АВВ, на следующем шаге агент A выбирает из $O(v)$ ребро (v,u), у которого $((\mu(v,u)=r)and(\mu((v,u),v)=b))$, и переходит по нему в вершину u, окрашивая $\mu(v,u):=b$, выполняет $v:=u$ и записывает в список M: НАЗАД_АВВ;
ВПЕРЕД_ AR(v): агент A выбирает из $O(v)$ произвольное ребро (v,u), у которого $(\mu(v,u)=r)and(\mu(u)=r)and(\mu((v,u),u)=r)$, и переходит по нему в вершину u, выполняет $v:=u$ и записывает в M: ВПЕРЕД_AR;
ОТСТУП_ А(v): агент A выбирает из $O(v)$ произвольное ребро (v,u), у которого $((\mu(v,u)=r)and(\mu((v,u),v)=r))$, и переходит по нему в вершину u, выполняет $v:=u$ и записывает в M: ОТСТУП_А;
ФИКС_ А(v): агент A записывает в M: ФИКС_А;
ОБН_ А(v): Агент A записывает в M: ОБН_А;
РАСП_ А(v)
1. A выбирает из $O(v)$ 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 выбирает из $O(u)$ произвольное ребро (u,l), у которого $(\mu(u,l)=r)and(\mu((u,l),l)=r)and(\mu(l)=r)$, и переходит по нему в вершину l;
4. u:=l;
5. Агент A записывает в M: РАСП_А;
6. 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
7. агент A переходит по ребру (u,l) в вершину l;
8. u:=l;
9. Агент A записывает в M: ОТСТУПИЛ_А;
10. end do;
11. Агент A записывает в M: РЕБРО_РАСПОЗНАНО_А;

Алгоритм работы агента B аналогичен алгоритму работы агента A.

2.4. Алгоритм "Восстановление".

Вход: списки сообщений M и N от АИ.

Выход: список вершин V_H и ребер E_H графа H, изоморфного G.

Данные: V_H, E_H – списки вершин и ребер графа H, изоморфного графу G. *Сч_ А, Сч_ В* – счетчики числа посещенных вершин графа G агентов A и B соответственно. *r(1)r(2)..r(t)* – список номеров вершин красного пути, t – длина этого списка. *y(1)y(2)..y(p)* – список номеров вершин желтого пути, p – длина этого списка. *N_ А* – переменная, в которой хранится номер вершины, из которой агент последний

раз помечал перешейки. N_B – переменная, в которой хранится номер вершины, из которой агент последний раз помечал перешейки. F – количество перешейков из вершины N_A, помеченных для распознавания. K – количество перешейков из вершины N_B, помеченных для распознавания. E – переменная, в которой 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 – счетчик, используемый агентом A при подсчете номера второй вершины перешейка и номера второй вершины обратного ребра. j – счетчик, используемый агентом B при подсчете номера второй вершины перешейка и номера второй вершины обратного ребра. Mes – текущее сообщение. AN – переменная, в которой значение "1" дает агенту A сигнал для возврата и распознавания 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 распознавания помеченных агентом A перешейков, значение "0" позволяет A работать дальше в обычном режиме.

Алгоритм:

1. *Сч_ A*:=1;
2. *Сч_ B*:=2;
3. Обнуляем массивы: *AN, BN, N_ A, N_ B, N, M, F, K, E, L, i, j,* E_H;
4. *t*:=1;
5. *p*:=1;
6. *r(t):= Сч_ A*;
7. *y(p):= Сч_ B*;
8. $V_H := \{1, 2\}$;
9. While $(M \neq \varnothing) and (N \neq \varnothing)$ do
10. if $M \neq \varnothing$ then do Прочитать в Mes сообщение и удалить его из M;
11. *ОБР_ СП_ A*(); end do;
12. if $N \neq \varnothing$ then do Прочитать в Mes сообщение и удалить его из N;
13. *ОБР_ СП_ B*(); end do;
14. end do;
15. Печать V_H, E_H.

Рассмотрим используемые процедуры.

***ОБР_ СП_ A*()**:

1. if Mes = "ВПЕРЕД_A" then *ВПЕРЕД_ A*() ;
2. if Mes = "ВПЕРЕД_AB" then *ВПЕРЕД_ AB*();
3. if Mes = "ВПЕРЕД_ABB" then *ВПЕРЕД_ ABB*();
4. if Mes = "НАЗАД_A" then *НАЗАД_ A*();
5. if Mes = "НАЗАД_AB" then *НАЗАД_ AB*();
6. if Mes = "НАЗАД_ABB" then *НАЗАД_ ABB*();
7. if Mes = "ФИКС_A" then *ФИКС_ A*();
8. if Mes = "ОБН_A" then *ОБН_ A*();
9. if Mes = "РАСП_A" then *РАСП_ A*();

10. if Mes = "ОТСТУП_А" then $ОТСТУП\_A()$;
ВПЕРЕД_ А(): выполняется серия операций: *Сч_ А:=Сч_ А+2*; $t := t + 1$;
r(t):=Сч_ А; $V_H := V_H \cup \{Сч\_A\}$; $E_H := E_H \cup \{r(t\text{-}1), r(t)\}$;
НАЗАД_ А(): из списка *r(1)..r(t)* удаляется элемент $r(t)$; $t := t - 1$;
ВПЕРЕД_ АВ(): F присваивается значение $F + 1$;
НАЗАД_ АВ(): E присваивается значение 1;
ВПЕРЕД_ АВВ(): $E_H := E_H \cup \{N\_A, r(t) - i\}$;
НАЗАД_ АВВ(): K присваивается значение $K - 1$;
ОТСТУП_ А(): i присваивается значение $i + 2$;
ФИКС_ А(): выполняется серия операций: *N_ А:=Сч_ А*; *BN:=1*; $E := 0$;
ОБН_ А(): отменяется команда "назад" для $A(AN := 0)$, обнуляется параметр i;
РАСП_ А():
1. Прочитать в Mes сообщение из M и удалить его;
2. While Mes ="$ОТСТУПИЛ\_A$" do
3. $i := i + 2$;
4. end do;
5. $E_H := E_H \cup \{r(t), r(t)\text{-}i\}$;
6. $i := 0$;

Процедуры обработки списка сообщений от агента B аналогичны.

3. Свойства алгоритма распознавания. В начале алгоритма, если $n \geq 3$, то как минимум по одному разу выполнятся процедуры: $ВПЕРЕД\_A(v)$, $ВПЕРЕД\_A()$, поскольку в рассматриваемых графах простых циклов длины меньшей 3 нет. При выполнении агентом A процедуры $ВПЕРЕД\_A(v)$ он посещает новую вершину графа G, процедурой $ВПЕРЕД\_A()$ агента АЭ создается новая вершина графа H. Следовательно, в процессе выполнения алгоритма устанавливается неявная нумерация $\varphi : V_G \to V_H$ вершин графа G в вершины графа H, где устанавливается равенство $\varphi(v) = t$, когда вершина v красится в красный цвет и $t =$*Сч_ А* или равенство $\varphi(s) = p$, когда вершина s красится в желтый цвет и $p =$*Сч_ В*. Эта нумерация является биекцией, поскольку в связном графе G все вершины достижимы из начальных вершин, и поэтому все вершины посещаются агентами, т.е. красятся в красный и желтый цвета. Поскольку алгоритм действия агентов АИ основан на стратегии обхода графа вглубь, то все ребра графа G являются либо древесными (принадлежащими красному или желтому дереву и при первом прохождении красятся в красный или желтый цвета соответственно), либо обратными (и при единственном прохождении красятся в черный цвет), либо перешейками (которые красятся при первом прохождении в красный или желтый цвета, но отнести такие ребра к дереву одного из агентов АИ нельзя, т.к. пронумерованы инцидентные ему вершины разными агентами АИ). Из описания алгоритма следует, что агенты АИ проходят все ребра графа G, поскольку при окончании алгоритма все ребра становятся черными. При выполнении процедуры $ВПЕРЕД\_A()$ или $ВПЕРЕД\_B()$ АЭ распознает древесное ребро (v, u) и так нумерует вершину u, что ребру (v, u) однозначно соответствует ребро $(\varphi(v), \varphi(u))$ графа H. Выполняя процедуру $РАСП\_A()$ или $РАСП\_B()$, агент АЭ распознает обратное ребро (v, u) графа G и ставит ему

в однозначное соответствие ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении агентами процедур *ФИКС_ A*(), *ВПЕРЕД_ ABB*() или *ФИКС_ B*(), *ВПЕРЕД_ BAA*() 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. *Выполняя алгоритм распознавания, агенты распознают любой граф G с точностью до изоморфизма.*

Подсчитаем временную и емкостную сложность в равномерной шкале [9]. 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 (либо 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. Выполняя процедуры *ВПЕРЕД_ A*(v) (*ВПЕРЕД_ B*(s)) и *НАЗАД_ A*(v) (*НАЗАД_ B*(s)) агенты АИ проходят одно ребро, при выполнении процедуры *РАСП_ A*(v) (*РАСП_ B*(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 процедуры *РАСП_ A*(v) (*РАСП_ B*(s)) агенты АИ проходят фактически цикл, состоящий из обратного ребра и некоторого конечного отрезка красного (желтого) пути, соединяющего вершины инцидентные обратному ребру. Выполняя процедуры МЕТИМ_AB(v) (МЕТИМ_BA(s)) и РАСП_ABB(v) (РАСП_BAA(s)) оба агента АИ проходят один и тот же перешеек, сначала в одном направлении, потом в обратном. Выполняя процедуры ВПЕРЕД_AR(v) (ВПЕРЕД_BR(s)) и ОТСТУП_A(v) (ОТСТУП_B(s)) агенты АИ проходят одно красное (желтое) ребро. При выполнении ФИКС_A(v) (ФИКС_B(s)), 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 ребра, проход по нему агента АИ и обработка команды АЭ, полученной на данном этапе от агента АИ, осуществляется за 1 единицу времени. Тогда временная сложность алгоритма определяется 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) Общее время выполнения процедур *НАЗАД_ A*(v) и *НАЗАД_ B*(s) оценивается как $O(n)$; 4) На выполнение процедур *МЕТИМ_ A*(v) и *МЕТИМ_ B*(s) уходит время, оцениваемое как $2 \times O(m)$, т.е. как $O(n^2)$; 5) Выполнение *РАСП_ ABB*(v) и *РАСП_ BAA*(s) занимает время, оцениваемое как $O(n^2)$; 6) Процедуры *ВПЕРЕД_ AR*(v) и *ВПЕРЕД_ BR*(s) выполняются за время, оцениваемое как $O(n) \times m$, т.е. как $O(n^3)$; 7) Процедуры *ОТСТУП_ A*(v) и *ОТСТУП_ B*(s) выполняются за время, оцениваемое как $O(n) \times m$, т.е. как $O(n^3)$; 8) Время, затрачиваемое на *ФИКС_ A*(v) и *ФИКС_ B*(s), оценивается как $O(n)$; 9)

Время, затрачиваемое на *ОБН_ A(v)* и *ОБН_ B(s)*, оценивается как $O(n)$; 10) Выполнение процедур *РАСП_ A(v)* и *РАСП_ B(s)*, выполняется за время, которое оценивается как $O(n) \times m$, т.е. как $O(n^3)$; 11) Время выполнения *СТОИТ_ A(v)* и *СТОИТ_ B(s)* в общей сложности для всех трёх 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: $S(n) = O(n^2)$.

Теорема 2. *Временная сложность алгоритма распознавания равна* $O(n^3)$*, а емкостная –* $O(n^2)$*. При этом алгоритм использует 3 краски.*

Выводы. В работе предложен алгоритм распознавания графа тремя агентами, два из которых ходят по графу, а третий обрабатывает полученную от них информацию. Выполняя алгоритм распознавания, они строят граф H, изоморфный исследуемому. Каждому АИ требуется 2 краски, одна из которых совпадает по цвету для обоих агентов. Алгоритм останавливается через $O(n^3)$ шагов, и требует $O(n^2)$ единиц памяти.

1. *Капитонова Ю.В., Летичевский А.А.* Математическая теория проектирования вычислительных систем. – М.: Наука, 1988. – 296с.
2. *Кудрявцев В.Б., Алешин С.В., Подкозлин А.С.* Введение в теорию автоматов. – М.: Наука, 1985. – 320с.
3. *Kuipers B.* The spatial semantic hierarchy // Artificial Intelligence. – 2000. – V.119, №1. – 2. – P.191-233.
4. *Dudek G, Jenkin M.* Computational principles of mobile robotics. – Cambridge Univ. press, Cambridge, 2000. – 280p.
5. *Калибарда Г., Кудрявцев В.Б., Ущумлич Ш.* Независимые системы автоматов в лабиринтах // Дискретная математика. – 2003. – Т.15, вып.2. – С.3-39.
6. *Fraigniaud P., Jecincas D., Peer G., Pelc A., Peleg D.* Graph Exploration by a finite automaton // proc. 29 th Internac. Symp. On Math. Foundation of Computer Science (MFCS), LNCS 3153. – 2004. – P.451-462.
7. *Грунский И.С., Татаринов Е.А.* Распознавание конечного графа блуждающим по нему агентом // Вестник ДонНУ, сер. А естествен. науки 2009, вып.1. – С.492-497
8. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536с.
9. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МЦНМО, 2001. – 960с.
10. *Касьянов В.Н., Евстигнеев В.А.* Графы в программировании: обработка, визуализация и применение. – СПБ.: БХВ – Петербург, 2003. – 1104с.

Ин-т прикл. математики и механики НАН Украины, Донецк
Славянский гос. пед. ун-т
stepkin.andrey@rambler.ru

Получено 11.11.09