УДК 519.1

ВОЗМОЖНОСТЬ И СЛОЖНОСТЬ РАСПОЗНАВАНИЯ ГРАФОВ ТРЕМЯ АГЕНТАМИ

© А. В. Стёпкин

Институт прикладной математики и механики
Национальной академии наук Украины
ул. Розы Люксембург, 74, г. Донецк, Украина, 83114
e-mail: *stepkin.andrey@rambler.ru*

Abstract. Problem of exploration finite graphs by three agents is considered. Constructed an algorithm for exploration undirected graphs with $O\left(n^2\right)$ (n is amount of nodes of graph) time and space complexities. Two agents (which move on graph) needs two different colors (in total three colors) for graph exploration. An algorithm is based on depth-first traversal method.

Введение

В настоящее время важным и актуальным направлением кибернетики является распознавание неизвестной среды [1, 2]. В данной работе рассматривается проблема распознавания среды, заданной конечным графом [3, 4, 5], несколькими агентами. Ранее подобное распознавание рассматривалось в [6], в случае, когда два агента-исследователя (АИ) поочередно передвигаются по неизвестному конечному графу, обмениваются данными с агентом-экспериментатором (АЭ), который и восстанавливает исследуемый граф по данным, полученным от АИ.

Рассмотрим решение этой проблемы в случае, когда два агента-исследователя A и B одновременно передвигаются по неизвестной среде, заданной конечным графом (АИ могут окрашивать вершины, ребра и инциденторы графа, воспринимать эти отметки и на их основании осуществлять перемещение), которые обмениваются данными с АЭ (восстанавливает граф, по данным полученным от АИ, а также передает АИ информацию, необходимую им для дальнейшего функционирования). В работе предложен алгоритм построения маршрутов АИ по графу, позволяющих АЭ точно восстановить граф среды. Для этого каждому АИ требуется две краски: у A это r и b, у B — y и b. Алгоритм основан на методе обхода графа в глубину [7, 8]. При описании алгоритма используются результаты и обозначения из [6, 9].

Основным результатом данной работы является оптимизация алгоритма распознавания перешейков (ребра которые соединяют деревья, построенные различными АИ) и обратных ребер (ребра, которые соединяют несмежные вершины дерева, построенного АИ), что позволило, в итоге, улучшить временную сложность с $O\left(n^3\right)$ [6, 9] до $O\left(n^2\right)$, где n — число вершин графа.

1. 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 [7, 8, 10].

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 сюръективное 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 инцидентров $((v, u), v), ((v, u), u)$. 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.

Мобильные агенты A и B в начале работы помещаются в произвольные несовпадающие вершины графа G, которые сразу добавляются АЭ в множество вершин V_H. Агенты передвигаются по графу из вершины v в вершину u по ребру (v, u), могут изменять окраску вершин v, u, ребер (v, u), инцидентров $((v, u), v), ((v, u), u)$. Находясь в вершине v, АИ воспринимает метки всех элементов окрестности $Q(v)$ и на основании этих меток определяет, по какому ребру будет дальше 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, кроме того, строится представление графа G, вначале неизвестного агентам, списками ребер и вершин.

2. Стратегия решения

Принцип работы рассматриваемого алгоритма основан на методе поиска в глубину и заключается в том, что АИ идут «в глубину», пока это возможно, возвращаются назад, ищут другой путь с еще не посещенными вершинами и не пройденными ребрами.

Рассмотрим подробно режимы работы АИ. При описании режимов, в скобках указываются сообщения, которые отправят АИ, попав в рассматриваемую ситуацию («СООБЩЕНИЕ_АГЕНТА_А»; «СООБЩЕНИЕ_АГЕНТА_В»).

1. *Обычный режим работы.* (ОРР) Работая в этом режиме, АИ двигается вперед по белым вершинам, окрашивая эти вершины, соединяющие их ребра и дальние инциденторы в «свой» цвет («ВПЕРЕД_А»; «ВПЕРЕД_В»). Если белый путь и другие возможные пути перемещения отсутствуют, то АИ возвращается назад по своему пути, окрашивая пройденные вершины, соединяющие их ребра и ближние инциденторы в черный цвет («НАЗАД_А»; «НАЗАД_В»). АИ завершает работу тогда, когда его исходная вершина, вследствие отсутствия возможных путей перемещения, окрашивается в черный цвет («СТОП_А», «СТОП_В»).

2. *Режим распознавания обратных ребер.* (РРОР) Если, при движении вперед, в вершине v было обнаружено обратное ребро, то АИ переключается в режим распознавания обратных ребер и красит в «свой» цвет ближние инциденторы всех обратных ребер инцидентных вершине v («МЕТКА_ОР_А»; «МЕТКА_ОР_В»). Завершив покраску инциденторов, АИ передвигается назад по своему пути (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), 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 («ВПЕРЕД_ОР_А»; «ВПЕРЕД_ОР_В»). На этом этапе возможны два случая:

2.1. 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 (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), до обнаружения следующей вершины, инцидентной помеченному обратному ребру.

2.2. 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 («РЕБРА_РАСПОЗНАНЫ_А»; «РЕБРА_РАСПОЗНАНЫ_В») и завершает работу в режиме распознавания обратных ребер.

3. *Режим пометки перешейков.* (РПП) 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 (если второй АИ ещё не распознал ранее помеченные перешейки, то первый АИ не может

метить новые перешейки и в случае, когда у первого АИ отсутствуют другие возможные варианты перемещения, кроме как пометить новый перешеек, он останавливается до того момента, пока второй АИ не распознает все помеченные перешейки). В этом режиме АИ окрашивает ближние инциденторы всех перешейков, инцидентных вершине v, в черный цвет (на каждый окрашенный инцидентор отправляется одно сообщение «МЕТИМ_АВ»; «МЕТИМ_ВА»). Когда все перешейки помечены работа АИ в данном режиме завершается («ФИКС_А»; «ФИКС_В»). По завершению режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков. В данном режиме работы агент A имеет приоритет над агентом B, поэтому в ситуации, когда оба АИ одновременно обнаружат один и тот же перешеек, он будет помечен агентом A.

4. *Режим распознавания перешейков.* (РРП) Получив от АЭ команду о необходимости распознавания перешейков, АИ переключается в режим распознавания перешейков. Если в этот момент агент работает в РРОР, то АИ переключится в РРП лишь по завершению распознавания обратных ребер. 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 (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), 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 (каждый шаг назад, отправляется сообщение «НАЗАД_А»; «НАЗАД_В») до тех пор, пока не попадет в вершину инцидентную помеченному перешейку либо же в вершину с другими возможными путями перемещения. Во втором случае АИ продолжает возвращаться назад по своему пути ничего не окрашивая (каждый шаг назад, отправляя сообщение «ОТСТУП_А»; «ОТСТУП_В») до обнаружения ближайшей вершины, инцидентной помеченному перешейку.

При обнаружении помеченного перешейка возможны два случая:

4.1. Помечено один перешеек. АИ окрашивает ближний инцидентор помеченного перешейка в черный цвет («РАСП_АВ»; «РАСП_ВА»). Далее движется вперед в конец пути «своего» цвета.

4.2. Помечено не менее двух перешейков. АИ окрашивает ближний инцидентор помеченного перешейка в «свой» цвет («РАСП_АВ»; «РАСП_ВА»). Далее АИ

движется назад по своему пути (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»), пока не будет найден следующий помеченный перешеек. При обнаружении помеченного перешейка возможно два варианта:

4.2.1. Следующий помеченный перешеек не последний. АИ окрашивает ближний инцидентор в черный цвет («РАСП_АВ»; «РАСП_ВА»). На следующем шаге АИ снова возвращается назад по своему пути до следующего помеченного перешейка (каждый шаг назад, отправляется сообщение «ОТСТУП_А»; «ОТСТУП_В»).

4.2.2. Следующий помеченный перешеек последний. АИ окрашивает ближний инцидентор в «свой» цвет («РАСП_АВ»; «РАСП_ВА»). На следующем шаге АИ сообщает АЭ о распознавании всех помеченных другим АИ перешейков («ОБН_А»; «ОБН_В»). Далее 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 движется вперед в конец пути «своего» цвета.

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. При этом удаляет краску с ближнего инцидентора и ребра, окрашивает дальний инцидентор в черный цвет («ВОЗВРАТ_В») 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.

Если АИ попадает в ситуацию, когда в вершине возможен выбор сразу нескольких режимов работы, то первым будет выбран РРП, потом РПП, за ним РРОР и наконец ОРР. Режим работы при попадании двух АИ в одну белую вершину в этом списке не рассматривается потому, что такая ситуация приведет к изменениям в работе только агента B и, в этот момент, другие режимы работы для него будут недоступны.

Выполняя обход графа, агенты A и B создают соответственно красный и желтый пути. Рассмотрим принцип, по которому АИ строит путь «своего» цвета. При движении в белую вершину красный (желтый) путь удлиняется, при движении назад по своему пути — укорачивается. При движении АИ назад, для распознавания обратных ребер или перешейков, длина пути не изменяется. Вершина, из которой

возможно лишь движение назад по своему пути, либо вовсе отсутствуют варианты перемещения, окрашивается в черный цвет. Алгоритм заканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины черными.

Выполняя обход графа G агенты создают неявную нумерацию посещенных вершин. Первый раз посетив вершину агент A окрашивает её в красный цвет (агент B - в желтый цвет), и АЭ ставит ей в соответствие номер, равный значению переменной $Сч\_A$ ($Сч\_B$ для агента B). Отметим что переменные $Сч\_A$ и $Сч\_B$ принимают соответственно нечетные и четные значения. Восстановление графа G происходит на основе созданной агентами нумерации путем построения графа H изоморфного G.

Рассмотрим алгоритм работы АЭ:

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 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, 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 инцидентов, 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. $AN := 0, BN := 0, N\_A := 0, N\_B := 0, F := 0, K := 0, M := \varnothing, N := \varnothing, E := 0,$
   $L := 0, i := 0, j := 0, E_H := \varnothing, STOP\_A := 0, STOP\_B := 0,$
   $UDP\_A := FALSE, UDP\_B := FALSE, UDOBR\_A := FALSE,$
   $UDOBR\_B := FALSE, KOBR\_A := 0, KOBR\_B := 0$;
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*();
13. *end do*;
14. *if* $N \neq \varnothing$ *then do*
15. прочитать в Mes сообщение и удалить его из очереди N;
16. *ОБР_СП_B*();
17. *end do*;
18. *end do*;
19. печать V_H, E_H.

ОБР_СП_A():

1. *if* $Mes =$ "ВПЕРЕД_А" *then ВПЕРЕД_A*();
2. *if* $Mes =$ "МЕТИМ_АВ" *then МЕТИМ_AB*();
3. *if* $Mes =$ "НАЗАД_А" *then НАЗАД_A*();
4. *if* $Mes =$ "РАСП_АВ" *then РАСП_AB*();
5. *if* $Mes =$ "ФИКС_А" *then ФИКС_A*();
6. *if* $Mes =$ "ОБН_А" *then ОБН_A*();
7. *if* $Mes =$ "РЕБРА_РАСПОЗНАНЫ_А" *then РЕБРА_РАСПОЗНАНЫ_A*();
8. *if* $Mes =$ "ОТСТУП_А" *then ОТСТУП_A*();
9. *if* $Mes =$ "МЕТКА_ОР_А" *then МЕТКА_ОР_A*();

10. *if* $Mes =$ "ВПЕРЕД_ОР_А" *then* *ВПЕРЕД_ОР_А*();
11. *if* $Mes =$ "СТОП_А" *then* *СТОП_А*().

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

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(): $N\_A :=Сч\_A$; $BN := 1$; $E := 0$; $Q := F$;
$UDP\_A := (((F = Q)\, or\, (F = 1))\, and\, (Q \neq 1))$;

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():$KOBR\_A := KOBR\_A - 1$; $UDOBR\_A := (KOBR\_A = 0)$;
$E_H := E_H \cup \{(r(t), r(t-i))\}$;

СТОП_А(): $STOP\_A := 1$;

Процедуры работы со списком сообщений от агента B, которые не рассмотрены ниже, аналогичны процедурам работы со списком сообщений 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(): $E_H := E_H \backslash \{(y(p-1), y(p))\}$; $V_H := V_H \backslash \{Сч\_B\}$;
$Сч\_B{:=}\ Сч\_B{-}2$; $p := p-1$; $y(p) := Сч\_B$; $L := 1$; $.K := K+1$;

3. Свойства алгоритма распознавания

В начале работы алгоритма распознавания, при $n \geq 3$, как минимум, по одному разу выполняются процедуры: ВПЕРЕД_A() и ВПЕРЕД_B(). Эти сообщения передаются агенту-экспериментатору, когда АИ посещают белые вершины исследуемого графа G. Процедурами агента АЭ ВПЕРЕД_A() и ВПЕРЕД_B() создаются две новые вершины (по одной вершине для каждой из процедур) графа H.

При одновременном попадании двух АИ в одну белую вершину процедурами ВПЕРЕД_A() и ВПЕРЕД_B() будет создано две новые вершины графа H. Вершина созданная агентом B, на следующем шаге будет удалена процедурой ВОЗВРАТ_B(), 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_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. При выполнении процедур *ВПЕРЕД_ОР_A*() или *ВПЕРЕД_ОР_B*() 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_AB*() или *РАСП_BA*() 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. *Три агента, выполнив алгоритм распознавания на графе G, распознают этот граф с точностью до изоморфизма.*

Подсчитаем временную и емкостную сложности алгоритма в равномерной шкале [8]. 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.

Каждый раз при выполнении процедур из ОРР каждый АИ проходит одно ребро. При однократном выполнении процедуры из РРОР АИ метят не более $n-2$ (изначально одна вершина уже окрашена в «чужой» цвет, поэтому обратного ребра в неё быть не может) обратных ребер, по одному разу проходит не более $n-2$ ребер красного (желтого) пути, а так же по два раза проходится не более $n-2$ обратных ребер. При выполнении процедур из РПП АИ не совершают перехода по перешейкам, а просто окрашивают их ближние инциденторы, после чего, не делая передвижений, отправляют сообщение АЭ о завершении РПП, на что так же уходит один ход. При однократном выполнении процедуры из РРП АИ проходят не более $n-2$ ребер красного (желтого) пути, после чего помечая перешейки как распознанные АИ не совершают перехода по перешейку, а просто окрашивают его ближний инцидентор. Завершив покраску всех помеченных перешейков АИ уведомляет об этом АЭ, на что так же уходит один ход. Возвращаясь после распознавания всех перешейков АИ может пройти не более чем по одному разу два перешейка (либо не пройти ни одного перешейка в случае, когда был только один помеченный перешеек), а так же не более чем $n-2$ ребер красного (желтого) пути.

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 сообщений АЭ полученных на данном этапе от АИ осуществляется за 1 единицу времени. Тогда временная сложность алгоритма определяется следующими соотношениями:

1. Процедуры из ОРР выполняются не более чем $2\times(n-1)$ раз, общее время их выполнения оценивается как $O(n)$.
2. Время, затрачиваемое на работу в РРОР оценивается как $n\times 4\times O(n)$, то есть как $O(n^2)$.
3. На работу в РПП уходит время, оцениваемое как $n\times O(n)+O(n)=O(n^2)$.
4. Время выполнения РРП оценивается как
   $O(n^2)+O(n^2)+O(n)+O(n)+O(n^2)=O(n^2)$
5. Время простоя агентов при ожидании в общей сложности оценивается как
   $O(n)+O(n^2)=O(n^2)$.

Следовательно, суммарная временная сложность $T(n)$ алгоритма удовлетворяет соотношению: $T(n)=O(n^2)$.

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 и емкостная сложности алгоритма равны* $O(n^2)$*, где* n *— число вершин графа, при этом алгоритм использует 3 краски.*

Заключение

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

Автор выражает глубокую признательность своему научному руководителю к.ф.-м.н., с.н.с ИПММ НАНУ Грунскому И. С. за оказанную помощь в работе.

Список литературы

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. Albers S. Exploring unknown environments / S. Albers, M. R. Henzinger // SIAM Journal on Computing. — 2000. — 29(4). — P. 1164–1188.
3. Кудрявцев В. Б. Введение в теорию автоматов / В. Б. Кудрявцев, С. В. Алешин, А. С. Подколзин. — М.: Наука, 1985. — 320 с.
4. Грунский И. С. Эксперименты с помеченными графами / И. С. Грунский, С. В. Сапунов, Е. А. Татаринов // Дискретные модели в теории управляющих систем: электронный сборник материалов VIII международной конференции (г. Москва МГУ, 2009 р.). — Москва, 2009. — С. 43–44.
5. 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.
6. Грунский И. С. Распознавание конечного графа коллективом агентов. / И. С. Грунский, А. В. Стёпкин // Труды ИПММ НАН Украины. — 2009. — Т.19. — С. 43–52.
7. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М.: МЦНМО, 2001. — 960 с.
8. Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М.: Мир, 1979. — 536 с.
9. Стёпкин А. В. Распознавание конечных графов тремя агентами / А. В. Стёпкин // Искусственный интеллект. — 2011. — № 2. — С. 84–93.
10. Касьянов В. Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. — СПБ.: БХВ — Петербург, 2003. — 1104 с.

Статья поступила в редакцию 10.01.2012