Журнал обчислювальної та прикладної математики **2013, №2(112)** Journal of Computational & Applied Mathematics

УДК 519.17

РАСПОЗНАВАНИЕ КОНЕЧНЫХ НЕОРИЕНТИРОВАННЫХ ГРАФОВ КОЛЛЕКТИВОМ АГЕНТОВ

А. В. Стёпкин

Резюме. В работе рассматривается проблема распознавания конечных неориентированных графов тремя агентами. Построен алгоритм распознавания линейной (от числа вершин графа) временной сложности, квадратичной емкостной сложности и коммуникационной сложности равной $O\left(n^2 \cdot log(n)\right)$. Для распознавания два, передвигающиеся по графу, агента используют по две различные краски (всего три краски). Алгоритм основан на методе обхода графа в глубину.

Введение

В настоящее время одним из центральных направлений кибернетики является распознавание неизвестной среды, заданной графом [1, 2, 3]. Данная работа посвящена решению проблемы распознавания несколькими агентами среды, заданной конечным неориентированным графом [4]. Предыдущие попытки решения нашей задачи [5, 6] позволили получить алгоритмы кубической (от числа вершин графа) и квадратической временных сложностей, соответственно, при неизменной квадратической емкостной сложности.

Рассмотрим решение нашей проблемы в случае, когда два агента-исследователя (АИ) A и B одновременно передвигаются по неизвестной среде, заданной конечным неориентированным графом без петель и кратных ребер, и обмениваются необходимой информацией с агентом-экспериментатором (АЭ), который постепенно восстанавливает исследуемый граф. В работе предложен алгоритм построения маршрутов АИ по графу, позволяющих АЭ точно восстановить граф среды. Каждому АИ для работы требуется две краски: у A это r и b, у B — y и b. Алгоритм основан на методе обхода графа в глубину [7], имеет линейную (от числа вершин графа) временную сложность, что на порядок меньше, чем в ранее рассмотренном алгоритме [6], и квадратичную емкостную сложность, которая не изменилась в сравнении с [6]. В работе [6] легко увидеть, что коммуникационная сложность алгоритма равна $O(n^2)$, а в предлагаемом алгоритме эта сложность в $log(n)$ раз больше и равна $O\left(n^2 \cdot log(n)\right)$. То есть, за счет увеличения коммуникационной сложности в $log(n)$ раз, достигнуто снижение временной сложности в n раз. При описании алгоритма используются результаты и обозначения из [5, 6].

1. Основные определения.

Понятия, которые не рассмотрены ниже, общеизвестны и с ними можно ознакомиться, например, в [7, 8].

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 инциденторов $((v, u), v), ((v, u), u)$. 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 имеют конечную на каждом шаге, но растущую память. В начале работы АИ 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), инциденторов $((v, u), v), ((v, u), u)$, а так же записывают в вершинах номера, используя двоичный код. Находясь в вершине v, АИ воспринимает метки всех элементов окрестности $Q(v)$ и номера смежных вершин, на основании этой информации 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 агенты-исследователи агенту-экспериментатору, попав в рассматриваемую ситуацию («СООБЩЕНИЕ_АИ_А»; «СООБЩЕНИЕ_АИ_В»).

1. *Обычный режим работы* (ОРР). АИ движется вперед по белым вершинам, окрашивая вершины, соединяющие их ребра и дальние инциденторы в «свой» цвет, записывает в вершины номера, получаемые на каждом шаге от АЭ в двоичном коде («ВПЕРЕД_А»; «ВПЕРЕД_В»). Если нет возможных путей перемещения, то АИ A отправляет АЭ сообщение «ПРИСВ_Amr», на что затрачивает один ход, далее возвращается назад, окрашивая пройденные вершины, ребра и ближние инциденторы в черный цвет («НАЗАД_А»). АИ B не обнаружив возможных путей перемещения сразу начинает движение назад, окрашивая пройденные вершины, ребра и ближние инциденторы в черный цвет («НАЗАД_В»).

Вернувшись в начальную вершину, АИ завершает работу («СТОП_А»; «СТОП_В»).

2. *Режим распознавания обратных ребер* (РРОР). Обратное ребро — это белое ребро, дальняя вершина которого окрашена в «свой» цвет. Если при движении вперед было обнаружено обратное ребро, то АИ сканирует окрестность вершины, в которой находится, считывает номера всех смежных вершин, инцидентных обратным ребрам, и отправляет список номеров АЭ («ОБР_А($x_1, x_2, ..., x_l$)»; «ОБР_В($k_1, k_2, ..., k_l$)», где x_i, k_i — номера, записанные агентами A и B соответственно, в вершинах своего пути).

3. *Режим распознавания перешейков* (РРП). Перешеек — это ребро, соединяющее вершины, принадлежащие областям работы разных агентов. Этот режим немного отличается для каждого из АИ. Если агент A обнаружил в вершине перешейки, и ни в одной из дальних вершин этих перешейков нет агента B (или агент B находится в одной из этих вершин, но уже распознал все, ранее обнаруженные ним, перешейки в эту вершину, или же B выполняет возврат назад по своему пути), то агент A отправляет все номера дальних вершин перешейков АЭ («ПЕР_А($x_1, x_2, ..., x_l$)», где x_i — номера, записанные агентом B, в вершинах своего пути). Если же агент B находится в одной из дальних вершин перешейков и ещё не распознавал инцидентные ей перешейки или не возвращается назад по своему пути, то агент A отправляет АЭ номера всех дальних вершин, обнаруженных перешейков, кроме номера вершины, в которой находится агент B.

Если перешейки обнаружил агент B и ни в одной из дальних вершин этих перешейков нет агента A (или же A находится в одной из этих вершин, но выполняет возврат назад по своему пути), то B передает АЭ номера всех дальних вершин обнаруженных перешейков («ПЕР_В($k_1, k_2, ..., k_l$)», где k_i — номера, записанные агентом A, в вершинах своего пути). Если же агент A находится в одной из дальних вершин перешейков и не выполняет возврат назад по своему пути, то B не выполняет никаких действий до ухода A из окрестности вершины, в которой находится B.

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, удаляя метки, оставленные на предыдущем шаге (удаляется краска с ребра и ближнего инцидентора), и переключается в обычный режим работы. 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, 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 построения агентами пути «своего» цвета. При движении в белую вершину красный (желтый) путь удлиняется, при движении назад по своему пути — укорачивается. 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 нумерацию посещенных вершин. Первый раз посетив вершину агент A окрашивает её в красный цвет (агент B — в желтый цвет), записывает в память вершины соответствующий номер (полученный от АЭ), равный значению переменной $Сч\_A$ ($Сч\_B$ для агента B). Восстановление графа G происходит на основе созданной агентами-исследователями нумерации, путем построения графа H изоморфного G.

Рассмотрим алгоритм работы агента-экспериментатора:

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. M, N — списки сообщений от агентов A и B соответственно. $STOP\_A, STOP\_B$ — переменные, используемые агентами A и B соответственно, для сигнализации АЭ о завершении распознавания своей области. mr_A — переменная, которая принимает значения «1» или «0». Значение «1» позволяет агенту A распознавать перешейки, в том числе и те, во второй вершине которых стоит агент B (на предыдущем шаге агентом B были распознаны все, обнаруженные ним перешейки, либо же агент B двигается назад по своему пути и не проверяет наличие перешейков). Значение «0» позволяет агенту A распознавать только те перешейки (если они существуют), во второй вершине которых нет агента B. mr_B — переменная, которая принимает значения «1» или «0». Значение «1» позволяет агенту B распознавать перешейки, даже при наличии в дальней вершине одного из перешейков агента A (агент A двигается назад по своему пути и не проверяет наличие перешейков). Значение «0» запрещает агенту B распознавание перешейков если в дальней вершине одного из перешейков находится агент A. $r(1), r(2), ..., r(t)$ — 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. $Сч\_B := 1$;
3. $M := \varnothing, N := \varnothing, E_H := \varnothing, STOP\_A := 0, STOP\_B := 0, mr\_A := 0,$ $mr\_B := 0$;
4. $t := 1$;
5. $p := 1$;
6. $r(t) := Сч\_A$; (номера вершин красного пути)
7. $y(p) := Сч\_B$; (номера вершин желтого пути)
8. $V_H := \{A[1], B[1]\}$;
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* $Mes =$ «ПЕР_A$(x_1, x_2, ..., x_l)$» *then* ПЕР_A$(x_1, x_2, ..., x_l)$;
2. *if* $Mes =$ «ОБР_A$(x_1, x_2, ..., x_l)$» *then* ОБР_A$(x_1, x_2, ..., x_l)$;
3. *if* $Mes =$ «ВПЕРЕД_A» *then* ВПЕРЕД_A();
4. *if* $Mes =$ «ПРИСВ_Amr» *then* ПРИСВ_Amr();
5. *if* $Mes =$ «НАЗАД_A» *then* НАЗАД_A();
6. *if* $Mes =$ «СТОП_A» *then* СТОП_A().

$ПЕР\_A(x_1, x_2, ..., x_l)$: выполняется операция:
$E_H := E_H \cup \{(A[r(t)], B[x_1]); (A[r(t)], B[x_2]); ...; (A[r(t)], B[x_l])\}$.
$ОБР\_A(x_1, x_2, ..., x_l)$: выполняется операция:
$E_H := E_H \cup \{(A[r(t)], A[x_1]); (A[r(t)], A[x_2]); ...; (A[r(t)], A[x_l])\}$.
$ВПЕРЕД\_A()$: выполняются операции: $Сч\_A := Сч\_A + 1$; $t := t + 1$.
$r(t) := Сч\_A$; $V_H := V_H \cup \{A[Сч\_A]\}$; $E_H := E_H \cup \{(A[r(t-1)], A[r(t)])\}$;
$mr\_B := 0$.
$ПРИСВ\_Amr()$: $mr\_B := 1$.
$НАЗАД\_A()$: из списка $r(1), ..., r(t)$ удаляется элемент $r(t)$; $t := t - 1$.
$СТОП\_A()$: $STOP\_A := 1$.

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

$ОБР\_СП\_B()$:
1. *if* $Mes =$ «ВОЗВРАТ_B» *then* ВОЗВРАТ_B();
2. *if* $Mes =$ «ПЕР_B$(k_1, k_2, ..., k_l)$» *then* ПЕР_B$(k_1, k_2, ..., k_l)$;
3. *if* $Mes =$ «ОБР_B$(k_1, k_2, ..., k_l)$» *then* ОБР_B$(k_1, k_2, ..., k_l)$;
4. *if* $Mes =$ «ВПЕРЕД_B» *then* ВПЕРЕД_B();

5. *if* $Mes =$ «НАЗАД_B» *then* НАЗАД_B();
6. *if* $Mes =$ «СТОП_B» *then* СТОП_B().

ВОЗВРАТ_B(): $E_H := E_H \backslash \{(y(p-1), y(p))\}$; $V_H := V_H \backslash \{B[Сч\_B]\}$;
$Сч\_B := Сч\_B - 1$; $p := p - 1$; $y(p) := Сч\_B$.

ПЕР_B$(k_1, k_2, ..., k_l)$: выполняются операции:
$E_H := E_H \cup \{(B[y(p)], A[k_1]); (B[y(p)], A[k_2]); ...; (B[y(p)], A[k_l])\}$;
$mr\_A := 1$.

ВПЕРЕД_B(): выполняются операции: $Сч\_B := Сч\_B + 1$; $p := p + 1$;
$y(p) := Сч\_B$; $V_H := V_H \cup \{B[Сч\_B]\}$; $E_H := E_H \cup \{(B[y(p-1)], B[y(p)])\}$;
$mr\_A := 0$.

НАЗАД_B(): из списка $y(1), ..., y(p)$ удаляется элемент $y(p)$; $p := p - 1$;
$mr\_A := 1$.

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 процедуры выполняются агентом-экспериментатором, после посещения агентами-исследователями белых вершин исследуемого графа G. Процедурами АЭ ВПЕРЕД_A() и ВПЕРЕД_B() создается по одной новой вершине (для каждой из процедур) графа H.

При одновременном попадании агентов A и B в одну белую вершину процедурами ВПЕРЕД_A() и ВПЕРЕД_B() будет создано две новые вершины графа H. Для того, чтобы вершина созданная агентом B не дублировала вершину, созданную агентом A, на следующем шаге она будет удалена процедурой ВОЗВРАТ_B(). Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\varphi : V_G \to V_H$. 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 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.

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 *ОБР_A*() или *ОБР_B*() АЭ распознает обратные ребра (v, u) графа G и ставит им в однозначное соответствие ребра $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур *ПЕР_A*() или *ПЕР_B*() АЭ распознает перешейки (v, u) графа G и ставит им в однозначное соответствие ребра $(\varphi(v), \varphi(u))$ графа H. Следовательно, φ является изоморфизмом графа G на граф H.

Теорема 1. *Три агента, выполнив алгоритм распознавания на графе G, распознают рассматриваемый граф с точностью до изоморфизма.*

Подсчитаем временную, емкостную и коммуникационную сложности алгоритма в равномерной шкале [9]. 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 $\varphi(v) = 1\,(\varphi(s) = 1)$ 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 однократном выполнении процедур из ОРР АИ проходит одно ребро, либо же, не выполняя передвижений, отправляет одно сообщение, на что уходит один ход. При однократном выполнении процедур из РРОР АИ распознают не более $n-2$ обратных ребер, на что затрачивают один ход. При однократном выполнении процедуры из РРП АИ распознают не более $n-2$ перешейка, на что так же уходит один ход. При одновременном попадании двух АИ в одну белую вершину, агент A не меняет режим работы, а агент B затрачивает один ход на возврат в свою область работы. 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 $3 \cdot (n-2) + 2$ раз, общее время их выполнения оценивается как $O\,(n)$.
2. Процедуры из РРОР выполняются не более чем $n-2$ раз, то есть общее время их выполнения оценивается как $O\,(n)$.
3. Процедуры из РРП выполняются не более чем $n-2$ раз, то есть общее время их выполнения оценивается как $O\,(n)$.
4. Время, затрачиваемое агентом B, на возврат в свою область работы после попадания двух АИ в одну вершину, оценивается как $O\,(n)$.
5. Время простоя агентов в ожидании оценивается как $O\,(n)$.

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

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 $O\,(n \cdot log(n))\,, O\left(n^2\right), O\,(n \cdot log(n))\,, O\,(n \cdot log(n))$. Следовательно: $S\,(n) = O\left(n^2\right)$.

Коммуникационная сложность $K(n)$ алгоритма определяется объемом информации, которой необходимо обменяться агентам, для распознавания графа. При работе агентов в ОРР, объем передаваемой АИ информации оценивается как $O\,(n)$. АЭ при этом передаст объем информации, оцениваемой как $O\,(n \cdot log(n))$. При работе агентов в РРОР и в РРП объем переданной информации оценивается как $2 \times O\left(n^2 \cdot log(n)\right)$. Следовательно, суммарная коммуникационная сложность $K(n)$ алгоритма удовлетворяет соотношению $K\,(n) = O\left(n^2 \cdot log(n)\right)$.

Теорема 2. *Временная сложность алгоритма распознавания равна $O\,(n)$, емкостная — $O\left(n^2\right)$, а коммуникационная — $O\left(n^2 \cdot log(n)\right)$. При этом алгоритм использует 3 краски.*

4. Заключение.

В работе предложен новый алгоритм распознавания графа среды временной сложности $O(n)$, емкостной сложности $O(n^2)$ и коммуникационной сложности $O(n^2 \cdot log(n))$. АИ имеют конечную на каждом шаге, но растущую память, 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.

Литература

1. Albers S. Exploring unknown environments. / S. Albers, M. R. Henzinger // SIAM Journal on Computing. — 2000. — 29(4). — P. 1164–1188.
2. Das S. Distributed exploration of an unknown graph. / S. Das, P. Flocchini, A. Nayak, N. Santoro // In Proc. of 12th Structural Information and Communication Complexity (SIROCCO). — 2005. — P. 99–114.
3. Deng X. Exploring an unknown graph. / X. Deng, C. H. Papadimitriou // Journal of Graph Theory. — 1999. — 32(3). — P. 265–297.
4. 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.
5. Грунский И. С. Распознавание конечного графа коллективом агентов. / И. С. Грунский, А. В. Стёпкин // Труды ИПММ НАН Украины. — 2009. — Т. 19. — С. 43–52.
6. Стёпкин А.В. Возможность и сложность распознавания графов тремя агентами. / А. В. Стёпкин // Таврический вестник информатики и математики. — 2012. — №1 (20). — С. 88–98.
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. Ахо А. Построение и анализ вычислительных алгоритмов. / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М.: Мир, 1979. — 536 с.

Институт прикладной математики и механики Национальной академии наук Украины, ул. Розы Люксембург, 74, Донецк, 83114, Украина.

Поступила 01.11.2012