vychisleniy [Theory and practice of parallel computing] Moscow, Internet-Un-t Inform. Tekhnologii, BINOM, Lab. znanii, 2007. 423 c.

6. Ahdreichenko D. K., Andreichenko K. P., Kononov V. V. On the Stability of the Angular Stabilization System of the Rotating Rod under Longitudinal Acceleration. *Journal of Computer and System Sciences International*, 2013, vol. 52, no 5, pp. 686–699. DOI: 10.1134/S1064230713030027.
7. Andreichenko D. K. An Efficient Algorithm for Numerical Inversion of the Laplace Transform. *Comput. Math. Math. Phys.*, 2000, vol. 40, no 7, pp. 1030–1044.

УДК 519.7

ВОССТАНОВЛЕНИЕ ГРАФА С ПОМЕЧЕННЫМИ ВЕРШИНАМИ ПЕРЕМЕЩАЮЩИМСЯ ПО НЕМУ МОБИЛЬНЫМ АГЕНТОМ

С. В. Сапунов

Кандидат физико-математических наук, научный сотрудник, Институт прикладной математики и механики НАН Украины, Донецк, sapunov_sv@yahoo.com

Рассматривается задача построения автономным мобильным агентом топологической модели своей операционной среды. Модель среды представляет собой связный неориентированный граф с помеченными вершинами. В работе предложен полиномиальный алгоритм восстановления и разметки графа среды для коллектива из агента-исполнителя и агента-вычислителя.

Ключевые слова: графы с помеченными вершинами, мобильный агент, восстановление графа.

ВВЕДЕНИЕ

Помеченные графы широко применяются в информатике для описания и моделирования разнообразных вычислительных процессов. В этом контексте наиболее изучены конечные орграфы с помеченными дугами (LTS [1], взвешенные автоматы [2], конечные автоматы). Тем не менее существует множество вычислительных процессов, естественным образом представляемых графами с помеченными вершинами (в программировании, робототехнике [3], верификации моделей [4]). В данной работе исследуется модель информационной системы как системы взаимодействующих объектов: агента и его операционной среды [5]. Операционная среда представляется в виде топологической модели, т. е. графа с помеченными вершинами.

Одной из центральных задач анализа операционной среды является построение ее карты, т. е. восстановление графа среды [3]. Причем построенная карта должна быть пригодной для дальнейшей навигации агентов. В данной работе под навигацией понимается перемещение агента (робота, поисковой программы и т. п.) из текущей области его операционной среды в заданную ее область. Будем считать карту пригодной для навигации, если на основании описания пути по карте агент, получающий минимум локальной информации о среде, может пройти этот путь в среде.

1. ПОСТАНОВКА ЗАДАЧИ

Мобильный агент установлен в произвольную вершину неизвестного ему графа, все вершины которого не помечены или, что то же самое, помечены одной и той же меткой. Требуется разработать алгоритм перемещений агента по графу и разметки пройденных вершин, который позволяет восстановить граф. Причем полученная разметка должна способствовать навигации агента по графу.

К настоящему времени известно довольно много методов восстановления графов [6]. Однако эти методы, как правило, предполагают разметку всех вершин графа одной и той же меткой, что неприемлемо для последующей навигации. Тривиальным методом решения поставленной задачи является восстановление графа с разметкой каждой его вершины уникальной меткой. В работе предложен алгоритм, в котором на основе анализа структуры графа уменьшается число различных меток.

2. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Будем полагать, что агент воспринимает окружающую его среду как простой конечный связный неориентированный граф, вершины которого могут быть помечены метками из некоторого конечного их множества. Такие графы назовем помеченными и формально определим их следующим образом. Помеченным графом называется простой конечный связный неориентированный граф с помеченными вершинами $G = (V, E, M, \mu)$, где V — множество вершин, $|V| = n$, E — множество ребер (т.е. неупорядоченных пар вершин), M — множество меток, $|M| = m$, $\mu : V \to M$ — сюръективная функция разметки. Положим, что среда является пассивной и не может самостоятельно изменять граф и его разметку.

Пусть $G = (V_G, E_G, M, \mu_G)$ и $H = (V_H, E_H, M, \mu_H)$ — помеченные графы. Изоморфизмом графов G и H называется такая биекция $\varphi : V_G \to V_H$, для которой $\mu_G(v) = \mu_H(\varphi(v))$ и для любого ребра $(v', v'') \in E_G$ существует ребро $(\varphi(v'), \varphi(v'')) \in E_H$, и наоборот.

Путем в графе G будем называть последовательность вершин $p = v_1 \dots v_k$ такую, что $(v_i, v_{i+1}) \in E$, $i = 1, \dots, k-1$. Число $k \in \mathbb{N}$ будем называть длиной пути p. Меткой $\mu(p)$ пути p назовем слово $w = \mu(v_1) \dots \mu(v_k)$ в алфавите меток M. Будем говорить, что слово w определяется вершиной v_1. Длину слова w будем обозначать через $d(w)$. Начальный отрезок или префикс длины k слова w будем обозначать через $\mathrm{pre}_k(w)$. Конечный отрезок или суффикс длины k слова w будем обозначать через $\mathrm{suf}_k(w)$. Инверсией слова $w = \mu(v_1) \dots \mu(v_k)$ назовем слово $w^{-1} = \mu(v_k) \dots \mu(v_1)$. Через M^* обозначим множество всех конечных слов в алфавите M, включая пустое слово e длины 0, а через M^+ обозначим множество $M^* \setminus \{e\}$. Множество L_v всех слов $w \in M^+$, определяемых вершиной $v \in V$, будем называть языком, определяемым этой вершины. Граф G будем называть приведенным, если для любых вершин $v, s \in V$ из $v \neq s$ следует $L_v \neq L_s$. Определим на M^+ частичную операцию $\circ$ композиции слов. Пусть $a, b \in M$, $w, u \in M^*$, тогда $wa \circ au = wau$ и $wa \circ bu$ не определено, если $a \neq b$. Введем операцию $* : V \times M^+ \to 2^V$ соотношением: для любой вершины $v \in V$ и любого слова $w \in M^+$ через $v \star w$ обозначим множество всех вершин $s \in V$ таких, что существует путь p из v в s, и $\mu(p) = w$. Ясно, что если слово $w \in L_v$, то $|v \star w| > 0$ и $|v \star w| = 0$ в противном случае.

Под k-окрестностью $\Gamma_v^{(k)}$ вершины $v \in V$ будем понимать множество всех вершин, находящихся от v на расстоянии не превосходящем $k \in \mathbb{N}$. Таким образом, имеем: $\Gamma_v^{(1)} = \{v\}$, $\Gamma_v^{(2)} = \Gamma_v^{(1)} \cup \{s \mid (v, s) \in E\}$, $\Gamma_v^{(3)} = \Gamma_v^{(2)} \cup \left(\bigcup_{s \in \Gamma_v^{(2)}} \Gamma_s^{(2)}\right)$ и т. д.

Под восстановлением графа будем понимать получение агентом такого его представления, опираясь на которое агент мог бы переместиться из текущей вершины графа в любую его вершину.

С целью уточнения сложности восстановления графа среды разделим сложность перемещений, связанных с размечиванием вершин, и сложность вычислений, основанных на анализе получаемой разметки. Для этого будем использовать следующее представление агента в виде пары агентов, взаимодействующих через двусторонний канал связи: агента исполнителя (АИ), функционирующего непосредственно в среде, и агента вычислителя (АВ), функционирующего вне среды. В каждый момент времени агент АИ находится в одной из вершин графа среды. На свой вход он получает разметку локальной окрестности текущей вершины и сообщения от агента АВ. Сообщения содержат команды на изменение метки текущей вершины, перемещение в другую вершину и передачу информации о разметке. Команда на перемещение содержит метку целевой вершины, смежной с вершиной, в которой находится АИ. Агент АИ не различает одинаково помеченные вершины. Поэтому если заданной меткой помечены несколько вершин, то он случайным образом выбирает среди них вершину для перехода. Полагаем, что агент АИ обладает достаточным запасом и ассортиментом меток для осуществления разметки исследуемого графа. Агент АВ расположен вне среды и получает информацию о ней только от агента АИ. На основе этой информации агент АВ строит модель среды в виде помеченного графа. Полагаем, что этот агент обладает достаточным объемом памяти для хранения модели исследуемого графа.

3. О РАЗМЕТКЕ СРЕДЫ И ПОЛЕ ЗРЕНИЯ АГЕНТА-ИССЛЕДОВАТЕЛЯ

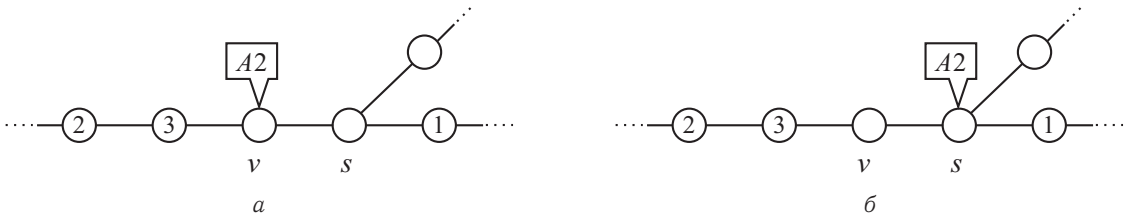
Функцию разметки $\mu : V \to M$ будем назвать детерминированной или Д-разметкой, если для любой вершины $v \in V$ и любых вершин $s, t \in \Gamma_v^{(2)}$ из $s \neq t$ следует $\mu(s) \neq \mu(t)$. Помеченный граф с детерминированной функцией разметки будем называть детерминированным или Д-графом. В работе [7] показано, что для любой вершины v Д-графа путь с меткой $w \in L_v$ определен однозначно. Там же показано, что расстояние между двумя одинаково помеченными вершинами Д-графа не меньше 4. Одной из центральных проблем, возникающих при навигации мобильных агентов, является проблема самостоятельного определения агентом своего положения в среде (задача самолокализации агента) [3]. В работах [8,9] предложено решение задачи самолокализации для приведенных Д-графов. В работе [7] показано, что если в Д-графе существует вершина с уникальной меткой, то этот граф является приведенным. Следовательно, на вершинах любого конечного связного простого неорграфа может быть построена такая Д-разметка, что полученный в результате Д-граф является приведенным, т. е. для него разрешима задача самолокализации.

В качестве меры сложности агента будем рассматривать его поле зрения или размер наблюдаемой им локальной окрестности. Обозначим через $A^{(k)}$ агента, наблюдающего метки всех вершин из k-окрестности текущей вершины. В работе [7] показано, что агент $A^{(k)}$ может осуществлять навигацию на Д-графе с числом вершин $n \geqslant 3$ тогда и только тогда, когда $k \geqslant 2$. Таким образом, агент $A^{(2)}$ является простейшим агентом, который может осуществлять навигацию на Д-графе, а Д-разметка графа является достаточным условием для того, чтобы простейший агент мог осуществлять навигацию на этом графе. Там же показано, что для Д-разметки произвольного графа агентом $A^{(k)}$ достаточно, чтобы k превосходило 2. В той же работе предложен метод восстановления графа коллективом из агентов АИ и АВ, где в качестве агента АИ используется агент $A^{(3)}$.

Увеличение объема информации, поступающей на вход агента (т.е. его поля зрения), приводит к увеличению сложности робота, который реализует функции агента. Поэтому целесообразно рассматривать агентов с ограничениями на поле зрения.

Определим средства, с помощью которых агент $A^{(2)}$ смог бы выполнять функции агента АИ при Д-разметке графа. Будем считать, что в качестве меток используются натуральные числа, и для разметки вершины агент АВ выбирает наименьшую метку, отсутствующую в поле зрения агента АИ.

В примере на рис. 1, *а* в наблюдаемой агентом $A^{(2)}$ окрестности $\Gamma_v^{(2)}$ нет вершины с меткой 1. Следовательно, 1 является наименьшей из возможных меток для v. Но если $A^{(2)}$ пометит v этой меткой, то разметка уже не будет детерминированной. Для того, чтобы выбрать правильную метку для v, $A^{(2)}$, надо перейти в вершину s, смежную с v, и вернуться обратно. Пример на рис. 1, *б* показывает, что $A^{(2)}$ без дополнительных средств не может определить вершину, где он находился ранее, если она помечена той же меткой, что и текущая. Действительно, в окрестности $\Gamma_s^{(2)}$ есть две одинаково помеченные вершины, и $A^{(2)}$ не может определить, какая из них v.



а *б*

Рис. 1. Примеры разметки среды и поля зрения агента $A^{(2)}$

Для того чтобы избежать такой ситуации, наделим агента $A^{(2)}$ возможностью устанавливать и подбирать в вершинах графа камни одного и того же типа p. При этом положим, что метка вершины заменяется типом установленного в ней камня. Рассмотрим пример на рис. 2.

Пусть $A^{(2)}$ изначально находится в вершине v и ему надо определить, какой меткой ее пометить. $A^{(2)}$ устанавливает в v камень p и перемещается в вершину s (s выбирается произвольно среди непомеченных вершин из окрестности $\Gamma_v^{(2)}$). Далее $A^{(2)}$ после наблюдения разметки окрестности $\Gamma_s^{(2)}$ устанавливает в s другой экземпляр камня p и возвращается в v по пути с меткой pp. Затем $A^{(2)}$ перемещается в вершину t и наблюдает ее окрестность (t также выбирается произвольно). Находясь в t, $A^{(2)}$ «видит» две вершины с меткой p и не может определить, какая из них v. Следовательно, вершину, подлежащую разметке, требуется метить камнем, отличающимся от камней, предназначенных для временной разметки ее 2-окрестности.

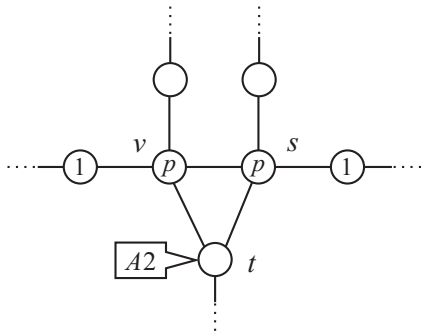


Рис. 2. Разметка среды и поле зрения агента $A^{(2)}$ с двумя одинаковыми камнями

Предоставим агенту $A^{(2)}$ возможность метить текущую вершину графа переносными камнями типов p_1 и p_2. Следующий параграф демонстрирует, что агент $A^{(2)}$ с камнями обладает достаточными средствами для выполнения функций агента АИ при Д-разметке графа.

4. ВОССТАНОВЛЕНИЕ ГРАФА СРЕДЫ

Построение минимальной Д-разметки на основе только лишь локальной информации о вершинах (т.е без знания всего графа) представляется невозможным в общем случае. Поэтому будем рассматривать «жадный» алгоритм решения этой задачи коллективом из агентов АИ и АВ. В основу предлагаемого в работе метода восстановления графа положен метод обхода графа в ширину [10]. При этом для каждой вершины $v \in V$ в качестве имени используется метка пути в нее из начальной вершины по дереву обхода. (Этим объясняется выбор в пользу метода обхода в ширину, так как путь в любую вершину из начальной по дереву обхода имеет кратчайшую длину среди всех таких путей, то и длина имени этой вершины также кратчайшая.)

Зададим линейный порядок на множестве M. Без потери общности можно предположить, что $M \subset \mathbb{N}$ и, что перед началом работы алгоритма все вершины исследуемого графа помечены меткой 0.

Граф H модели среды хранится в виде списка *Модель*, где вместе с каждой вершиной v графа H хранится также список *Окрестность* (v) смежных с ней вершин. Для хранения множества вершин, окрестности которых еще не обработаны, используется список *Очередь* типа FIFO.

Алгоритм 1 описывает функционирование агента АВ. Первоначально агент АИ устанавливается в произвольную вершину неизвестного агентам графа. Так как по предположению все вершины графа помечены одной и той же меткой $0 \in M$, то агентам для выбора метки начальной вершины не требуется изучать разметку ее 3-окрестности. Агент АВ выбирает минимальную метку из $M \setminus \{0\}$ и передает агенту АИ указание пометить этой меткой начальную вершину. В дальнейшем эта метка и метка 0 не используются для разметки. Метка начальной вершины, также являющаяся ее именем, помещается в *Очередь* и *Модель*. Далее, до тех пор, пока *Очередь* не опустеет, последовательно выполняются строки 8–12. Вызываемые в этих строках процедуры представлены алгоритмами 2–6.

Алгоритм 1. ВОССТАНОВЛЕНИЕ ГРАФА СРЕДЫ (агент АВ)

1: $\mu(v_0) := \min(M \setminus \{0\})$
2: $A := M \setminus \{0, l\}$
3: Передать «пометить меткой $\mu(v_0)$» и ждать ответ.
4: $\mu(v_0) \to$ *Очередь*
5: $\mu(v_0) \to$ *Модель*
6: $x := \mu(v_0)$

7: **while** *Очередь* $\neq \varnothing$ **do**
8: $y := \mathrm{top}\,(\textit{Очередь})$
9: ПЕРЕХОД ИЗ x В y
10: $x := y$
11: РАЗМЕТКА $\Gamma_x^{(2)}$
12: ПОПЕРЕЧНЫЕ РЕБРА ИЗ x
13: **end while**
14: Передать «стоп».

Алгоритм 2 описывает вычисление агентом АВ слова w, являющегося меткой пути из вершины с именем x в вершину с именем y, и передачу агенту АИ команд по перемещению по этому пути. Вычисленный путь является путем по дереву обхода в ширину известной к этому моменту части графа. Здесь $w[i]$ обозначает i-й символ слова w.

Алгоритм 2. ПЕРЕХОД ИЗ x В y (агент АВ)

1: $k := 1$
2: **while** $\mathrm{pref}_k(x) = \mathrm{pref}_k(y)$ **do**
3: $k := k + 1$
4: **end while**
5: $w := \left(\mathrm{suf}_{d(x)-k+2}(x)\right)^{-1} \circ \mathrm{suf}_{d(y)-k+2}(y)$
6: **for** $i = 1$ **to** $d(w) - 1$
7: **if** агент АИ готов **then**
8: Передать «перейти $w[i]w[i+1]$» и ждать ответ.
9: **end if**
10: **end for**

Алгоритм 3 описывает управление агентом АВ процессом Д-разметки 2-окрестности вершины, в которой находится агент АИ. До тех пор, пока в этой области есть вершины с меткой 0, агент АВ отправляет в них агента АИ с целью исследования разметки их 3-окрестностей. В строке 4 вызывается процедура управления разметкой текущей вершины с меткой 0. Здесь S обозначает множество меток всех вершин из 2-окрестности текущей вершины.

Алгоритм 3. РАЗМЕТКА $\Gamma_x^{(2)}$ (агент АВ)

1: Передать «вычислить S» и ждать ответ.
2: **while** $0 \in S$ **do**
3: Передать «перейти $\mathrm{suf}_1(x)0$» и ждать ответ
4: РАЗМЕТКА ВЕРШИНЫ z
5: Передать «перейти $\mathrm{suf}_1(z)\mathrm{suf}_1(x)$» и ждать ответ.
6: Передать «вычислить S» и ждать ответ.
7: **end while**

Алгоритм 4 вызывается из алгоритма 3 и описывает управление агентом АВ процессом Д-разметки текущей вершины и добавления этой вершины в построенную к этому времени модель среды. Агент АВ дает указание агенту АИ осмотреть 3-окрестность текущей вершины. На основе полученных от АИ сообщений о наблюдаемой им разметке АВ корректирует множество допустимых для Д-разметки

меток. Как только исследование 3-окрестности завершено АВ выбирает наименьшую доступную метку и дает АИ указание пометить текущую вершину этой меткой. Далее АВ вычисляет имя этой вершины, помещает его в *Очередь* и *Модель*. Тем самым новая вершина добавляется в граф модели.

Алгоритм 4. РАЗМЕТКА ВЕРШИНЫ z (агент АВ)

1: $A' := A$
2: Передать «осмотреть 3-окрестность» и ждать ответ.
3: **while** агент A_1 передал S **do**
4: $A' := A' \setminus S$
5: Принять сообщение от агента A_1.
6: **end while**
7: $l := \min(A')$
8: Передать сообщение «пометить меткой l» и ждать ответ.
9: $z := xl$
10: $z \to$ *Очередь*
11: $z \to$ *Модель*
12: $z \to$ *Окрестность*(x)
13: $x \to$ *Окрестность*(z)

Окончив разметку 2-окрестности текущей вершины, коллектив агентов приступает к проверке инцидентных ей ребер, поперечных дереву обхода. Алгоритм 5 описывает управление агентом АВ процессом этой проверки. Если в окрестности текущей вершины присутствует вершина, помеченная до начала разметки этой окрестности, то АИ получает указание перейти в эту вершину, положить в ней камень p_1 и вернуться назад. Затем вызывается процедура поиска этого камня в известной части графа среды. Здесь *Разметка*(x) обозначает множество меток всех вершин из 2-окрестности вершины x. Вычисляется *Разметка*(x) как объединение $\mathrm{suf}_1(y)$ для всех y из *Окрестность*(x).

Алгоритм 5. ПОПЕРЕЧНЫЕ РЕБРА ИЗ x (агент АВ)

1: Предать «вычислить S» и ждать ответ.
2: **while** $S \setminus$ *Разметка*$(x) \neq \varnothing$ **do**
3: ВЫБРАТЬ $l \in S \setminus$ *Разметка*(x)
4: Передать «перейти $\mathrm{suf}_1(x)l$» и ждать ответ.
5: Передать «положить p_1» и ждать ответ.
6: Передать «перейти $p_1\mathrm{suf}_1(x)$» и ждать ответ.
7: ПОИСК p_1
8: Предать «вычислить S» и ждать ответ.
9: **end while**

Алгоритм 6 описывает управление агентом АВ процедурой поиска агентом АИ камня p_1, установленного ранее в одной из вершин графа среды. Пусть текущей вершиной является вершина x, а вершина, в которой установлен камень p_1 помечена меткой l. Агент АВ выбирает из списка *Модель* все вершины, длина имени которых не меньше длины x и метка которых равна l. Агент АИ получает указание последовательно переходить в эти вершины до тех пор, пока не будет найден камень p_1. После обнаружения камня АВ добавляет к графу модели поперечное ребро и направляет АИ обратно в вершину x.

Алгоритм 6. ПОИСК p_1 (агент АВ)

```
 1: for all w ∈ Множество do
 2:   if d(w) ⩽ d(x) and suf_1(w) = l then
 3:     w → Выборка
 4:   end if
 5: end for
 6: y := x
 7: while false do
 8:   w := top(Выборка)
 9:   z := pre_{d(w)−1}(w)
10:   ПЕРЕХОД ИЗ y В z
11:   Предать «вычислить S» и ждать ответ.
12:   if p_1 ∈ S then
13:     return true
14:   else
15:     y := z
16:   end if
17: end while
18: w → Окрестность(x)
19: x → Окрестность(w)
20: Передать «подобрать p_1» и ждать ответ.
21: Передать «перейти suf_1(w)suf_1(x)» и ждать ответ.
```

Алгоритм 7 описывает функционирование агента АИ. Первоначально АИ помещается в произвольную вершину неизвестного графа среды. Далее АИ ожидает получения от АВ одной из семи команд, реакция на которые описывается в соответствующих строках алгоритма. Наиболее сложным набором действий агента АИ является обход 2-окрестности текущей вершины. Для этого в строке 10 вызывается соответствующая процедура.

Алгоритм 7. РАЗМЕТКА ГРАФА СРЕДЫ (агент АИ)

```
 1: Получить сообщение от агента A_2
 2: while сообщение не «стоп» do
 3:   if сообщение «пометить меткой l» then
 4:     Заменить метку текущей вершины меткой l
 5:   else if сообщение «перейти l_i l_j» then
 6:     Перейти из вершины с меткой l_i в смежную вершину с меткой l_j
 7:   else if сообщение «вычислить S» then
 8:     Вычислить множество S меток окрестности текущей вершины
 9:   else if сообщение «осмотреть 3-окрестность» then
10:     ОБХОД ОКРЕСТНОСТИ ТЕКУЩЕЙ ВЕРШИНЫ
11:   else if сообщение «положить p_1» then
12:     Положить камень p_1 в текущей вершине
13:   else if сообщение «подобрать p_1» then
14:     Подобрать камень p_1 в текущей вершине
15:   end if
16:   Передать «готов» и ждать ответ.
17: end while
18: if сообщение «стоп» then
19:   Конец работы
20: end if
```

Алгоритм 8 описывает действия агента АИ при получении от агента АВ команды «осмотреть 3-окрестность». Эти действия АИ выполняет самостоятельно и заключаются они в последовательном посещении всех вершин из окрестности текущей. Попадая в вершину с меткой 0, агент оставляет в ней камень p_2 и передает АВ разметку ее 2-окрестности. Обход вершин с метками, отличными от 0, и передача разметки их окрестностей производятся без использования камней p_2. Алгоритмы 9 и 10 описывают соответствующие процедуры.

Алгоритм 8. ОБХОД ОКРЕСТНОСТИ ТЕКУЩЕЙ ВЕРШИНЫ (агент АИ)

Вычислить множество S меток окрестности текущей вершины
Передать «S».
Положить камень p_1 в текущей вершине
ОБХОД НЕ ПОМЕЧЕННЫХ ВЕРШИН
ОБХОД ПОМЕЧЕННЫХ ВЕРШИН
Подобрать камень p_1 в текущей вершине
Передать «Обход окончил»

Алгоритм 9. ОБХОД НЕ ПОМЕЧЕННЫХ ВЕРШИН (агент АИ)

while $0 \in S$ **do**
  Перейти из вершины с меткой p_1 в смежную вершину с меткой 0
  Вычислить множество S меток окрестности текущей вершины
  Передать «S».
  Положить камень p_2 в текущей вершине
  Перейти из вершины с меткой p_2 в смежную вершину с меткой p_1
end while
Вычислить множество S меток окрестности текущей вершины
while $p_2 \in S$ **do**
  Перейти из вершины с меткой p_1 в смежную вершину с меткой p_2
  Подобрать камень p_2 в текущей вершине
  Перейти из вершины с меткой 0 в смежную вершину с меткой p_1
  Вычислить множество S меток окрестности текущей вершины
end while

Алгоритм 10. ОБХОД ПОМЕЧЕННЫХ ВЕРШИН (агент АИ)

Вычислить множество S меток окрестности текущей вершины
for all $l \in S$ **and** $l \neq 0$ **do**
  Перейти из вершины с меткой p_1 в смежную вершину с меткой l
  Вычислить множество S меток окрестности текущей вершины
  Передать «S».
  Перейти из вершины с меткой l в смежную вершину с меткой p_1
end for

5. АНАЛИЗ АЛГОРИТМА

Пусть $G_i = (V_{G_i}, E_{G_i}, M, \mu_{G_i})$ обозначает граф среды и $H_i = (V_{H_i}, E_{H_i}, M, \mu_{H_i})$ — граф модели после i-й итерации основного цикла алгоритма 1. Так как на каждой итерации этого цикла хотя бы одна вершина графа среды получает метку, отличную от 0, то после n итераций не останется ни одной вершины с меткой 0.

Пусть V'_{G_i} — множество вершин графа G_i, метки которых отличны от 0. Обозначим через G'_i подграф графа G_i, порожденный множеством V'_{G_i}.

Теорема 1. *Граф G'_i является Д-графом для любого $i \leqslant n$.*

Доказательство. Пусть существуют вершины v', $v'' \in V'_{G_i}$ такие, что $\mu_{G_i}(v') = \mu_{G_i}(v'')$. Расстояние между этими вершинами не может равняться 2 или 3, так как при выполнении алгоритма агент A_1 передает метки всех вершин из 3-окрестности каждой размечаемой вершины агенту A_2 и тот не использует эти метки при разметке. Следовательно, расстояние между этими вершинами больше или равно 4. Тогда в 2-окрестности каждой вершины графа G'_i нет вершин с одинаковыми метками, и он является Д-графом. □

Теорема 1 показывает, что коллектив агентов действительно строит Д-разметку графа среды.

Теорема 2. *Граф H_i изоморфно вкладывается в граф G_i для любого $i < n$.*

Доказательство. Новая вершина добавляется в граф модели тогда и только тогда, когда очередная вершина графа среды получает метку, отличную от 0. Следовательно, $\left|V'_{G_i}\right| = |V_{H_i}|$. Напомним, что имя всякой вершины графа модели является меткой простого пути из начальной вершины графа среды по дереву его обхода в ширину. Зададим соответствие $\varphi : V_{H_i} \to V'_{G_i}$ следующим образом. Пусть $x \in V_{H_i}$, тогда вершиной $\varphi(x)$ объявим вершину $v_0 \star x \in V'_{G_i}$. Очевидно, что $\mu_{G_i}(v_0 \star x) = \mathrm{suf}_1(x) = \mu_{H_i}(x)$. Следовательно, φ устанавливает взаимно однозначное соответствие между V_{H_i} и V'_{G_i}, сохраняющее разметку вершин. Пусть $(x, y) \in E_{H_i}$. Если x является начальным отрезком y, то $(\varphi(x), \varphi(y)) \in E_{G'_i}$, так как вершина $v_0 \star x$ является родителем вершины $v_0 \star y$ в дереве обхода в ширину. Если x не является начальным отрезком y и y не является начальным отрезком x, то ребро (x, y) было добавлено в модель после того как было обнаружено ребро $(v_0 \star x, v_0 \star y)$ поперечное к дереву обхода в ширину. Таким образом, φ устанавливает изоморфизм между графом H_i и подграфом графа G'_i. □

Из теорем 1 и 2 следует, что существование пути с меткой w из вершины $x \in V_{H_i}$ в вершину $y \in V_{H_i}$ означает существование пути с той же меткой из вершины $v_0 \star x \in V_{G_i}$ в вершину $v_0 \star y \in V_{G_i}$. Таким образом, модель H_i может быть использована для навигации в среде G_i.

Теорема 3. *Графы H_n и G_n изоморфны.*

Доказательство. Пусть на i-й итерации основного цикла алгоритма 1 обработана вершина, имя которой имеет длину r. Так как вершины добавляются в список *Очередь* в порядке их удаленности от начальной вершины, то все вершины, длина имени которых меньше r, к этому времени уже обработаны и удалены из *Очереди*. Обозначим через H_i^{r-1} подграф графа H_i, порожденный всеми его вершинами, длина имени которых меньше r, и через G_i^{r-1} подграф графа G'_i, порожденный всеми его вершинами, расстояние до которых от вершины v_0 меньше r. По теореме 2 граф H_i^{r-1} изоморфно вкладывается в граф G_i^{r-1}. Пусть $v', v'' \in V_{G_i^{r-1}}$ и $(v', v'') \in E_{G_i^{r-1}}$. Обозначим через w и u метки простых путей по дереву обхода в ширину из v_0 в v' и v'' соответственно. Очевидно, что $w, u \in V_{H_i^{r-1}}$. Если v' является родителем v'' в дереве обхода в ширину, то w является начальным отрезком u и $(w, u) \in E_{H_i^{r-1}}$. Пусть ребро (v', v'') является поперечным к дереву обхода в ширину. По предположению все вершины графа H_i, длина имени которых меньше r, уже обработаны. Следовательно, все поперечные ребра, соединяющие вершины, расстояние до которых от вершины v_0 меньше r, были уже обнаружены и $(w, u) \in E_{H_i^{r-1}}$. Таким образом, графы H_i^{r-1} и G_i^{r-1} изоморфны.

Пусть обработана и удалена последняя вершина из списка *Очередь*, и длина имени этой вершины равна q. Покажем, что к этому моменту все вершины графа G_n имеют метки, отличные от 0. Пусть вершина $v' \in V_{G_n}$ и по окончании работы алгоритма $\mu(v') = 0$. Так как граф G_n связный, то существует вершина $v'' \in \Gamma^{(2)}_{v'}$ такая, что $\mu(v'') \neq 0$. Обозначим через w метку простого пути по дереву обхода из вершины v_0 в вершину v''. По определению w также является именем соответствующей v'' вершины графа H_n. Предположим, что вершина w была удалена из списка *Очередь* на i-й итерации основного цикла алгоритма 1. К этому моменту все вершины из 2-окрестности вершины v'' уже получили метки, не равные 0, что противоречит предположению. Таким образом, в графе G_n нет вершин с меткой 0 и $|V_{G_n}| = |V_{H_n}|$. Из теоремы 2 следует, что граф H_n изоморфно вкладывается в

граф G_n. Ранее уже доказано, что графы H_n^{q-1} и G_n^{q-1} изоморфны. Пусть $v', v'' \in V_{G_n}$, $(v', v'') \in E_{G_n}$ и расстояние от вершины v_0 до каждой из этих вершин равно q. Ясно, что в этом случае ребро (v', v'') является поперечным к дереву обхода в ширину. Обозначим через w и u метки простых путей по дереву обхода в ширину из v_0 в v' и v'' соответственно. Так как все вершины графа к этому времени уже обработаны, то все поперечные ребра уже обнаружены и $(w, u) \in E_{H_n}$. Следовательно, графы G_n и H_n изоморфны. □

Из теоремы 3 следует, что коллектив агентов корректно восстанавливает граф среды.

Для оценки сложности восстановления графа введем дополнительные обозначения. Пусть a обозначает максимальную из степеней вершин графа G, а b — максимальное число вершин с одной и той же меткой в графе G_n. Ясно, что $a = O(n)$ и $b = O(n)$.

Теорема 4. *Коллектив из агентов АИ и АВ восстанавливает произвольный связный неорграф G путем Д-разметки его вершин за время $O(n^4)$.*

Доказательство. Каждая вершина графа H один раз помещается в список *Очередь* и один раз удаляется из него. Следовательно, основной цикл алгоритма 1 выполняется n раз. В ходе выполнения этого цикла вызываются алгоритмы 2, 3 и 5. Алгоритм 2 описывает управление агентом АВ процессом перехода агента АИ из одной вершины графа в другую. Так как расстояние между любыми двумя вершинами не превышает n, то для выполнения алгоритма 2 достаточно времени $O(n)$. Алгоритм 3 описывает управление агентом АВ процессом Д-разметки 2-окрестности вершины, в которой находится агент АИ. Для выполнения основного цикла этого алгоритма достаточно времени $O(a)$. Для того, чтобы агент АВ смог выбрать метку вершины, агент АИ посещает все вершины из ее окрестности. Для этого достаточно времени $O(a)$. Таким образом, для выполнения алгоритма 3 достаточно времени $O(a^2)$. Алгоритм 5 описывает управление агентом АВ процессом проверки ребер, инцидентных текущей вершине и поперечных к дереву обхода. Для выполнения основного цикла алгоритма 5 достаточно времени $O(a)$. В ходе его выполнения за время $O(n)$ устанавливаются вершины, которые могут быть концом исследуемого ребра, и за время $O(nb)$ агент АИ посещает эти вершины. Таким образом, для выполнения алгоритма 5 достаточно времени $O(abn)$. Из всего сказанного следует, что для выполнения алгоритма 1 достаточно времени $O(n^2 + na^2 + abn^2) = O(n^4)$. □

Под коммуникационной сложностью функционирования коллектива агентов будем понимать общий объем сообщений, которыми они обмениваются.

Теорема 5. *Коммуникационная сложность восстановления графа коллективом из агентов АИ и АВ равна $O(n^3 \log m)$ бит.*

Доказательство. Сообщение максимальной длины передает агент АИ агенту АВ при выполнении алгоритма 8. В этом сообщении содержится разметка 2-окрестности текущей вершины. Так как для передачи одной метки требуется $\log m$ бит, то для передачи разметки достаточно $O(a \log m)$ бит. При выполнении алгоритма 8 передача разметки выполняется $O(a)$ раз. Следовательно, один вызов алгоритма 8 влечет передачу $O(a^2 \log m)$ бит информации. Так как в процессе восстановления графа алгоритм 8 вызывается n раз, то коммуникационная сложность равна $O(na^2 \log m) = O(n^3 \log m)$ бит. Все другие передачи, осуществляемые в ходе восстановления графа, дают в сумме слагаемое линейного порядка и могут не учитываться. Следовательно, коммуникационная сложность восстановления графа составляет $O(n^3 \log m)$ бит. □

ЗАКЛЮЧЕНИЕ

В работе решена задача восстановления связного неорграфа мобильным агентом путем детерминированной разметки его вершин. Разработан алгоритм восстановления графа агентом, наблюдающим 2-окрестности посещаемых им вершин. Доказана корректность алгоритма и найдены оценки его временной и коммуникационной сложности.

Библиографический список

1. *Letichevsky A.* Algebra of behavior transformation and its application // Structural Theory of Automata, Semigroups and Universal Algebra. Springer, 2005. P. 241–272.
2. *Droste M., Kuich W., Vogler H.* Handbook of Weighted Automata. Springer, 2009. 608 p.
3. *Dudek G., Jenkin M.* Computational Principles of Mobile Robotics. Cambridge : Cambridge Univ. Press, 2010. 406 p.
4. *Baier C., Katoen J.-P.* Principle of Model Checking. MIT Press, 2008. 984 p.
5. *Капитонова Ю. В., Летичевский А. А.* Математическая теория проектирования вычислительных систем. М. : Наука, 1988. 298 с.
6. *Голубев Д. В.* Об обходе графов автоматами с одной нестираемой краской // Интеллектуальные системы. 1999. Т. 4, вып. 1–2. С. 243–272.
7. *Грунский И. С., Сапунов С. В.* Восстановление графа операционной среды мобильного робота путем разметки вершин, пригодной для дальнейшей навигации // Искусственный интеллект. 2012. № 4. С. 420–428.
8. *Грунский И. С., Сапунов С. В.* Идентификация вершин помеченных графов // Труды ИПММ НАНУ. 2010. Т. 21. С. 86–97.
9. *Грунский И. С., Сапунов С. В.* Диагностика местоположения мобильного робота на основе топологической информации о среде // Искусственный интеллект. 2011. № 2. С. 15–25.
10. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы : построение и анализ. М. : МЦНМО, 2001. 960 с.

Reconstruction of a Labeled Graph by a Graph-walking Mobile Agent

S. V. Sapunov

Institute of Applied Mathematics and Mechanics, National Academy of Sciences of Ukraine, 74, R. Luxemburg st., 83114, Donetsk, Ukraine, sapunov_sv@yahoo.com

The problem of construction of graph-like operational environment by a mobile agent is considered. The model of environment is defined as a simple undirected vertex labeled graph. We propose a polynomial time algorithm of graph reconstruction and labeling for the collective consisting of agent-explorer and agent-supervisor.

Key words: vertex labeled graphs, mobile agent, graph reconstruction.

References

1. Letichevsky A. Algebra of behavior transformation and its application. *Structural Theory of Automata, Semigroups and Universal Algebra*, Springer, 2005, pp. 241–272.
2. Droste M., Kuich W., Vogler H. *Handbook of Weighted Automata*. Springer, 2009, 608 p.
3. Dudek G., Jenkin M. *Computational Principles of Mobile Robotics*. Cambridge, Cambridge Univ. Press, 2010, 406 p.
4. Baier C., Katoen J.-P. *Principle of Model Checking*. MIT Press, 2008, 984 p.
5. Kapitonova Yu. V., Letichevsky A. A. *Mathematical Theory of Computational Systems Design*. Moscow, Nauka, 1988, 298 p. (in Russian)
6. Golubev D. V. On Graph Traversal by Automata with Single Nonerasable Coloration. *Intelligent systems*, 1999, vol. 4, iss. 1–2, pp. 243–272 (in Russian).
7. Grunsky I. S., Sapunov S. V. Reconstruction of the graph of operating environment of mobile robot by vertex-labeling sufficient for further navigation. *Iskusstvennyj intellekt* [Artificial Intelligence], 2012, no. 4, pp. 420–428.
8. Grunsky I. S., Sapunov S. V. Vertex Identification on Vertex Labeled Graphs. *Trudy IPMM NANU*, 2010, vol. 21, pp. 86–97 (in Russian).
9. Grunsky I. S., Sapunov S. V., Mobile robot location diagnostics on the basis of topological information about environment. *Iskusstvennyj intellekt* [Artificial Intelligence], 2011, no. 2, pp. 15–25.
10. Cormen T., Leiserson Ch., Rivest R. *Algorithms: construction and analysis*. Moscow, MCNMO, 2001, 960 p. (in Russian).