УДК 519.17

А.В. СТЁПКИН

ИСПОЛЬЗОВАНИЕ КОЛЛЕКТИВА АГЕНТОВ ДЛЯ РАСПОЗНАВАНИЯ НЕОРИЕНТИРОВАННЫХ ГРАФОВ

Аннотация. Рассматривается задача распознавания конечных неориентированных графов коллективом агентов. Два агента-исследователя одновременно передвигаются по графу, считывают и изменяют метки элементов графа, передают необходимую информацию агенту-экспериментатору, который строит представление исследуемого графа. Построен алгоритм распознавания линейной (от числа вершин графа) временной сложности и квадратической емкостной сложности. Разработана процедура оптимизации разбиения графа на части, распознаваемые различными агентами. Алгоритм основан на методе обхода графа в глубину.

Ключевые слова: распознавание графа, коллектив агентов, обход графа.

ВВЕДЕНИЕ

В настоящее время существует много различного рода сред, требующих досконального изучения [1]. Это является одной из причин активного развития такого направления математической кибернетики, как теория дискретных динамических систем. Исследуемая среда — это своего рода дискретная система, представленная как модель взаимодействия управляющей и управляемой систем, часто представляемого как процесс перемещения управляющего автомата по графу управляемой системы [2]. Это и привело к обширному и интенсивно развивающемуся исследованию поведения автоматов в лабиринтах [3–5]. Исследованию графа с помощью одного агента посвящено много публикаций, при этом остается малоисследованным распознавание графа несколькими блуждающими по нему агентами. Поэтому задача проведения систематического исследования экспериментов по распознаванию графа несколькими блуждающими по нему агентами является актуальной. При таком распознавании основной является проблема эффективности взаимодействия агентов с целью уменьшения затрат времени и памяти на распознавание. Требуется разработать такие алгоритмы движения, при которых блуждающие агенты не будут препятствовать один другому в работе и не дублировать ее.

В настоящей статье рассматривается коллектив, состоящий из трех агентов: два агента-исследователя (АИ) блуждают по графу и перекрашивают его элементы, передавая полученную информацию агенту-экспериментатору (АЭ). Разработана процедура, позволяющая агентам после завершения распознавания своей территории искать новые территории для распознавания. Этим была решена проблема простоя агента в случае, когда начальное расположение агентов не позволяло распознавать граф в равных частях и одному из агентов приходилось простаивать, пока второй агент не заканчивал работу над распознаванием оставшейся части, которая могла намного превышать распознанную простаивающим агентом часть территории. Поэтому начальное расположение агентов существенно влияло на конечную временную сложность алгоритма, а в некоторых случаях приводило к тому, что весь граф (кроме вершины, в которой находился второй агент) распознавался одним агентом.

Взаимодействие агентов-исследователей осуществляется а результате перекрашивания элементов графа. Ранее полученные алгоритмы [6, 7] решения рас-

сматриваемой задачи имеют кубическую (от числа вершин графа) и квадратическую временные сложности соответственно при неизменных квадратических емкостной сложности и коммуникационной сложности. Рассмотрим алгоритм решения данной проблемы в случае, когда два агента-исследователя, A и B, одновременно передвигаются по неизвестному конечному неориентированному графу без петель и кратных ребер, обмениваются необходимой информацией с агентом-экспериментатором, который восстанавливает исследуемый граф. При этом предложен алгоритм построения по графу маршрутов АИ, позволяющих АЭ точно восстановить граф среды. Каждому АИ для работы требуется две краски: для A — это r и b, а для B — y и b. Алгоритм основан на методе обхода графа в глубину [8], имеет линейную временную сложность, квадратическую емкостную сложность и коммуникационную сложность, равную $O(n^2 \cdot \log(n))$. При описании алгоритма используются результаты и обозначения из работ [6, 7].

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

Пусть $G = (V, E)$ — связный неориентированный конечный граф без петель и кратных ребер, где V — множество вершин, E — множество ребер (двухэлементных подмножеств (u, v), где $u, v \in V$). Для уточнения принадлежности множеств V и E некоторому графу G приняты обозначения V_G и E_G. Тройку $((u, v), v)$ будем называть инцидентором (точкой соприкосновения) ребра (u, v) и вершины v. Обозначим I множество всех инцидентров в графе. Множество $L = V \cup E \cup I$ назовем множеством элементов графа G. 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, назовем функцией раскраски графа G. Пара (G, μ) называется раскрашенным графом. Последовательность $u_1, u_2, \ldots, u_k$ попарно смежных вершин графа G называется путем, имеющем длину k. 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 n и m соответственно. Очевидно, что $m \le n(n-1)/2$. Восстановление графа G происходит на основе созданной АИ нумерации путем построения графа H, изомофного G. Изоморфизмом графа G и графа H назовем такую биекцию $\varphi: V_G \to V_H$, что $(v, u) \in E_G$ именно тогда, когда $(\varphi(v), \varphi(u)) \in E_H$. Таким образом, изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов. Коллектив агентов, распознающий ранее им неизвестный граф G, состоит из трех агентов: двух АИ (A и B) и одного АЭ. Агенты-исследователи перемещаются по графу и могут изменять окраску элементов графа. АЭ на основании сообщений от АИ создает представление графа. Мобильные агенты A и B имеют конечную, но растущую на каждом шаге память. В начале работы агенты-исследователи A и B помещаются в произвольные несовпадающие вершины графа G, при этом нумеруют их и передают номера АЭ, который помещает их в множество вершин V_H. Агенты передвигаются по графу из вершины v в вершину u по ребру (v, u), при этом могут изменять окраску вершин v, u, ребер (v, u), инцидентров $((v, u), v)$, $((v, u), u)$, а также записывать в вершины номера. АИ, находясь в вершине v, считывают метки всех элементов окрестности $Q(v)$ и номера смежных с ней вершин. На основании этой информации АИ определяет, по какому ребру будет дальше перемещаться и как будет окрашивать элементы графа. Агент-экспериментатор может передавать сообщения для АИ, а также принимать и идентифицировать его сообщения. Он обладает конечной, неограниченно растущей внутренней памятью, в которой

фиксируется результат функционирования АИ на каждом шаге и строится представление графа G, вначале неизвестного агентам, списками ребер и вершин. Отметим, что АИ явно обмениваются сообщениями только с АЭ; между собой АИ обмениваются сообщениями через АЭ.

АЛГОРИТМ РАБОТЫ АГЕНТОВ-ИССЛЕДОВАТЕЛЕЙ

В начале работы алгоритма все элементы графа окрашены в белый цвет. При работе алгоритма элементы графа могут быть окрашены агентами-исследователями. Для краткости элементы графа в зависимости от их окраски будем называть белыми, красными, желтыми и т.д.

Рассмотрим режимы работы АИ. При описании режимов в скобках указываются сообщения, которые АИ отправляют агенту-экспериментатору, попадая в рассматриваемую ситуацию («СООБЩ_АИ_А»; «СООБЩ_АИ_В»). АЭ, в свою очередь, обрабатывает полученное сообщение и отправляет АИ данные, необходимые для завершения хода.

Обычный режим работы (ОРР). АИ сканирует окрестность вершины, в которой он находится, на наличие белых вершин, выбирает произвольную белую вершину и переходит в нее («ВПЕРЕД_А($x_1, x_2, \ldots, x_q$)»; «ВПЕРЕД_В($k_1, k_2, \ldots, k_g$)», где x_i, k_i — номер вершины, из которой был произведен переход, q, g — количество оставшихся в ее окрестности белых вершин после ухода из нее агента). Таким образом, АИ продвигается вперед по белым вершинам, окрашивая вершины, соединяющие их ребра и дальние инциденторы, в «свой» цвет, входит в новые вершины, записывает в них номера, полученные от АЭ.

Если в окрестности вершины, в которой находится АИ, не обнаружено белой вершины, использованной для дальнейшего продвижения вперед, то АИ A отправляет АЭ сообщение «ПРИСВ_Amr», задействовав один ход, и затем возвращается назад, окрашивая пройденные вершины, ребра и ближние инциденторы в черный цвет («НАЗАД_А»). АИ B, не обнаружив белой вершины для дальнейшего продвижения вперед, сразу начинает двигаться назад, окрашивая пройденные вершины, ребра и ближние инциденторы в черный цвет («НАЗАД_В»). Возвращаясь назад своим путем, АИ на каждом шаге проверяют наличие белых ребер с дальней вершиной, окрашенной в «чужой» цвет (т.е. перешейков — ребер, соединяющих области работы разных агентов-исследователей). Если такая вершина существует, но второй АИ еще не пометил первый перешеек, то первый АИ будет пропускать ход, ожидая, пока второй АИ не пометит свой первый перешеек. Благодаря этой процедуре предотвращаются потери вторым АИ такого перешейка, который будет отмечен как ребро для перехода в чужую область (поскольку только через чужую область возможен переход в новую часть графа, требующую распознавания). Возвратившись в начальную вершину, АИ завершает работу на данной территории («СТОП_А»; «СТОП_В»). Таким образом, строится несколько (не менее двух) деревьев методом обхода в глубину.

Окончив работу на своей территории (все вершины этой территории окрашены в черный цвет), АИ будет искать новую территорию для распознавания. Поэтому в процессе распознавания ему необходимо каким-то образом отметить путь к вершине (назовем ее вершиной перехода (ВП)), инцидентной перешейку, через который будет совершен переход. Путь к ВП в итоге будет состоять из ребер (v, u), окрашенных как $(\mu((v, u), v) = w)$ and $(\mu(v, u) = b)$ and $(\mu((v, u), u) = r)$ and $(\mu(u) = b)$ для АИ A и $(\mu((s, z), s) = w)$ and $(\mu(s, z) = b)$ and $(\mu((s, z), z) = y)$ and $(\mu(z) = b)$ для АИ B.

Рассмотрим возможные ситуации и особенности функционирования АИ для каждого состояния. Отметим, что первым агентом (им может быть любой из АИ)

считается тот, через территорию которого совершается переход, а вторым агентом совершается переход через чужую область.

1. Построение агентом-исследователем пути к первому перешейку, обнаруженному на своем пути. В этом случае изменения в обычном режиме работы АИ, которые коснутся только способа покраски, произойдут только тогда, кода АИ будет переходить из вершины с номером, меньшим (или равным) номера ВП данного АИ, в вершину с меньшим номером. В таком случае при движении назад по своему пути АИ окрашивает пройденные вершины и ребра в черный цвет («НАЗАД_А»; «НАЗАД_В»). Ближний инцидентор при этом не окрашивается и сохраняет «свой» для данного АИ цвет. В дальнейшем по таким инцидентором АИ будет ориентироваться при движении к перешейку для перехода в новую область распознавания.

При переходе одного из АИ в новую часть графа, требующую распознавания, у другого АИ изменяется перешеек (или фиксируется, если ранее не обнаружено ни одного перешейка) для перехода в чужую область распознавания (отметим, что его ВП не изменится только в том случае, если новый и старый перешейки инцидентны одной вершине). Это потребует модификации старого пути либо построения нового пути.

2. Построение агентом-исследователем пути к новому перешейку, полученному после перехода другого АИ в новую часть графа, требующую распознавания.

2.1. Если до назначения новой вершины перехода (НВП) АИ не обнаружил ни одного перешейка, то построение пути происходит аналогично построению пути к первому перешейку, обнаруженному на своем пути (было рассмотрено выше).

2.2. Если НВП находится в той же ветке, что и ВП, только глубже, то изменения в ОРР агента-исследователя будут такими же, как и при построении пути к первому перешейку, обнаруженному на своем пути; при этом изменения наступят, начиная с момента попадания АИ в НВП.

2.3. Если НВП при изменении перешейка для перехода совпадает с ВП, то изменения в работе АИ будут такими же, как и при построении пути к первому перешейку, обнаруженному на своем пути.

2.4. Если НВП находится в той же ветке, что и ВП, только ближе к исходной вершине (номер НВП меньше номера ВП), то АИ, попав в НВП, окрашивает ближний инцидентор ребра, продолжающего старый путь, в «свой» цвет («УК_ПУТИ_А», «УК_ПУТИ_В»). Этим укорачивается старый путь и мы получаем путь к НВП. Далее АИ функционирует так же, как и при построении пути к первому перешейку, обнаруженному на своем пути.

2.5. Если НВП находится в ветке, отличной от тех, в которых пребывают ВП и/или предыдущие НВП (если таковые имеются), то АИ, попав в НВП, начинает функционировать как и при построении пути к первому перешейку, обнаруженному на своем пути. Это происходит до момента, пока АИ не попадет в вершину, из которой берут начало ветки, содержащие ВП (либо предыдущую НВП) и НВП. В этой вершине АИ отправляет АЭ номера дальних вершин ребер, окрашенных в черный цвет, с дальним инцидентором, имеющим «свой» цвет («ОТПР_НОМ_А(x_1, x_2)»; «ОТПР_НОМ_В(k_1, k_2))»). АЭ, в свою очередь, определяет меньший из номеров и запоминает его. Далее АИ выбирает произвольно одно из двух ребер, окрашивает ближний инцидентор в «свой» цвет и отправляет АЭ номер дальней вершины выбранного ребра («ВЫБОР_ВЕРШ_А(x)», «ВЫБОР_ВЕРШ_В(k)»). АЭ, в свою очередь, сравнивает номер полученной вершины с минимальным номером, сохраненным ранее. Если номера совпали, то

инцидентор окрашен правильно, старый путь разорван («ОБНУЛ_ПЕР_А», «ОБНУЛ_ПЕР_В»). Если номера не совпали, то окрашенный на предыдущем шаге инцидентор окрашивается в белый цвет, а ближний инцидентор второго такого ребра окрашивается в «свой» цвет, тем самым разрывая старый путь. АИ действует подобным образом, пока не будет разорван последний старый путь, который ведет к ВП или предыдущим НВП. Далее АИ функционирует так же, как и при построении пути к первому перешейку, обнаруженному на своем пути.

Отметим, что АИ по результатам сканирования не могут точно определить, находится ли он в другой ветке или нет. Определение происходит только при отслеживании номеров вершин. В результате может возникнуть ситуация, когда АИ будет искать вершину, из которой берут начало ветки, содержащие ВП (или предыдущую НВП) и НВП, которых просто не существует. Чтобы исключить подобную ситуацию, АЭ отслеживает номера вершин, которые посещает АИ. Когда номер вершины, в которой находится АИ, больше номера ВП, то возможны два варианта.

Первый вариант. АИ находится в той же ветке, что и ВП, только глубже. Тогда при возврате назад по своему пути АИ рано или поздно попадет в ВП; переменная *odna_vetka_A* (*odna_vetka_B* — для агента *B*), хранящаяся в АЭ, принимает значение 1. АИ, обнаружив смену значений переменной, отправляет АЭ сообщение («ОТМЕН_УКОР_А», «ОТМЕН_УКОР_В»). Исходя из этого, АИ не будет искать вершины, из которой берут начало ветки, содержащие ВП и НВП, до попадания в следующую НВП (если таковая появится).

Второй вариант. АИ находится в другой ветке. Тогда рано или поздно АИ попадет в вершину, из которой берут начало ветки, содержащие ВП и НВП. В этом случае переменная *konec_drug_vetki_A* (*konec_drug_vetki_B* для агента *B*), хранящаяся в АЭ, принимает значение 1; АИ, обнаружив смену переменной, отправляет АЭ сообщение («ОТМЕН_УКОР_А», «ОТМЕН_УКОР_В») и далее не будет искать вершины, из которой берут начало ветки, содержащие ВП и НВП, до попадания в следующую НВП (если таковая появится).

3. Построение первым АИ пути к новому перешейку, полученному после нескольких переходов второго АИ в новую часть графа, требующую распознавания, при неизменной области функционирования первого АИ. В этом случае происходит смена перешейка для перехода в чужую область и соответственно получаем НВП. Так как это возможно только относительно вершин, находящихся в той же ветке, в которой в данный момент работает первый АИ, и расположенных ближе или на таком же расстоянии к начальной вершине, как первый АИ, то при построении нового пути используются принципы работы, изложенные в пп. 2.3–2.5.

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

Режим распознавания перешейков (РРП). Этот режим незначительно отличается от любого режима для АИ. Предположим, что при движении вперед агент A обнаружил в вершине перешейки (идентификация перешейков происходит по дальней вершине, окрашенной в «чужой» цвет, и по ближнему инцидентору, окрашенному в белый цвет), при этом 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 им перешейки в эту вершину, или B выполняет возврат назад

по своему пути). Тогда агент A отправляет агенту-экспериментатору все номера дальних вершин перешейков («ПЕР_А($x_1, x_2, \ldots, x_l$)», где x_i — номера, записанные агентом B в вершинах своего пути). Если агент B находится в одной из дальних вершин перешейков и еще не распознал 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. Заметим, что агент A узнает о нахождении B в смежной вершине в результате сканирования окрестности на наличие перешейков. О возможности распознавания перешейка, в дальней вершине которого находится B, агент A узнает, исходя из значения переменной mr_A, запрашиваемой у АЭ. Если при движении вперед перешейки обнаружил агент B и ни в одной из дальних вершин этих перешейков нет агента A (или 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($k_1, k_2, \ldots, k_l$)», где k_i — номера, записанные агентом A, в вершинах своего пути). Если 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. Агент B узнает о нахождении A в смежной вершине в результате сканирования окрестности на наличие перешейков. Но о возможности распознавания перешейков при наличии в дальней вершине одного из них агента A агент B узнает по значениям переменной mr_B, запрашиваемой у АЭ.

Следует отметить, что если АИ впервые попадает в вершину, содержащую перешейки, то для возможности в дальнейшем перейти на чужую территорию в поисках новых частей графа для распознавания агент-исследователь должен отметить один из перешейков, по которому может быть совершен переход. Этот перешеек и является первым обнаруженным перешейком. Теперь рассмотрим, каким образом это происходит. После отправки номеров дальних вершин обнаруженных перешейков АЭ определяет минимальный номер и запоминает его. Далее АИ начинает перебирать все перешейки, которые он распознал в текущей вершине. Он выбирает произвольное белое ребро с белыми ближним и дальним инциденторами, дальняя вершина которого окрашена в «чужой» цвет (при этом если в дальней вершине перешейка находится второй АИ, то о возможности проверки этого перешейка агент узнает, исходя из значения переменных mr_A, mr_B для агентов A и B соответственно), окрашивает ближний инцидентор в «свой» цвет и отправляет номер дальней вершины АЭ («ВЫБОР_ВЕРШ_А(x)», «ВЫБОР_ВЕРШ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути). АЭ, в свою очередь, сравнивает номер полученной вершины с минимальным номером, сохраненным ранее. Если номера не совпали, то ближний инцидентор рассматриваемого перешейка окрашивается в черный цвет. Если номера совпали, то АИ непосредственно отправляет для АЭ сообщение («ПЕРВ_ПЕР_А», «ПЕРВ_ПЕР_В»).

В случае, когда АИ продвигается назад по своему пути и у него нет пропущенных белых ребер (ПБР) и при этом не помечен первый перешеек, он начинает искать первый обнаруженный перешеек другого агента. Идентификация такого перешейка происходит по дальней вершине, окрашенной в «чужой» (или черный) цвет, и по дальнему инциденту, окрашенному в «чужой» цвет. Обнаружив этот перешеек, АИ окрашивает его ближний инцидентор в «свой» цвет и отправляет АЭ сообщение («ПЕРВ_ПЕР_А», «ПЕРВ_ПЕР_В»). Далее построение пути к этому перешейку проводится так же, как и построение пути к первому обнаруженному перешейку. Если обнаружен перешеек с дальней вершиной, окрашенной в «чужой» цвет, и дальним инцидентором, окрашенным

в «чужой» (или белый) цвет, при этом другим АИ еще не помечен первый перешеек (это может произойти, если он производит перебор обнаруженных перешейков для определения первого), то АИ останавливается до пометки, сделанной другим АИ первого перешейка. Отметим, что в новой области распознавания первым перешейком для перехода в новой области АИ выбирает ребро, по которому был совершен переход в новую область.

При одновременном попадании двух АИ в одну белую вершину каждый из них окрашивает вершину наполовину, и она становится красно-желтой. Агент B на следующем шаге отступает назад по своему пути, удаляя метки, оставленные им на предыдущем шаге (удаляется краска с ребра и ближнего инцидентора), и переключается в ОРР. Агент A рассматривает разноцветную вершину как свою, но при распознавании окрашивает в черный цвет всю вершину.

При попадании АИ в ситуацию, когда в вершине возможен выбор сразу нескольких режимов работы, первым будет выбран РРП, за ним РРОР и, наконец, ОРР. Попадание обоих АИ в одну белую вершину в этом списке не рассматривается, поскольку такая ситуация приведет к изменениям в работе исключительно агента B и в этот момент другие режимы работы для него будут недоступны.

Теперь рассмотрим режим работы АИ, когда он закончил распознавание своей территории, но у второго агента остались пропущенные белые ребра (за которыми могут быть большие территории, требующие распознавания). АИ продвигается по черным ребрам с ближним белым инцидентором, дальним инцидентором «своего» цвета и черной дальней вершиной до тех пор, пока это возможно. Необнаружение такого ребра будет означать, что АИ попал в вершину, из которой возможен переход в чужую область распознавания (ВП или последняя НВП). Попав в такую вершину, агент переходит в чужую область распознавания через белый перешеек с ближним инцидентором, окрашенным в «свой» цвет (если переход выполняется через ВП), либо через белый перешеек с дальним инцидентором и дальней вершиной, окрашенной в «чужой» цвет (если переход выполняется через последнюю НВП). Тогда возможны два варианта.

1. При переходе агент попал в черную вершину. В этом случае АИ передвигается по черным вершинам, выбирая ребро, у которого ближний инцидентор окрашен в черный или «чужой» цвет. При этом если ближние инциденторы черные, то при переходе агент ничего не окрашивает, но если инциденторы окрашены в «чужой» цвет, то, выполняя переход, агент окрашивает их в черный цвет. Как только дальняя вершина такого ребра будет окрашена в «чужой» цвет, АИ проверяет наличие в этой вершине первого АИ (если первый АИ находится в этой вершине, то второй АИ не выполняет никаких действий до ухода первого АИ из вершины) и переходит в эту вершину. Если эта вершина остается окрашенной в «чужой» цвет, то далее второй АИ работает по алгоритму, описанному ниже (второй вариант) для случая попадания агента в вершину, окрашенную в «чужой» цвет. Иначе АИ продолжает движение по черным вершинам. Отметим, что, передвигаясь по черным вершинам чужого пути, второй АИ на каждом шаге проверяет через АЭ условие окончания работы первого АИ (количество пропущенных белых ребер равно нулю, и при этом отправлена команда об окончании распознавания территории $usl\_ost\_A = 1$ (территория $usl\_ost\_B = 1$ для агента B)). Если условие выполнено, то алгоритм заканчивает работу.

2. При переходе агент попал в вершину, окрашенную в «чужой» цвет. В этом случае сразу проверяется наличие пропущенных белых ребер (ПБР) у первого агента. Если их нет, то проверяется работа первого агента на предмет ее окончания. Если АИ не закончил работу, то второй АИ не проводит никаких действий до появления ПБР или до завершения работы первым АИ. Если агент

закончил работу, то алгоритм завершает работу. Если имеются пропущенные белые ребра, то агенту нужно определиться относительно дальнейшего направления движения.

Если ПБР имеются в вершине, в которой находится АИ, то по чужому пути АИ не передвигается, а сразу переходит в новую область распознавания, окрашивая дальние инцидентор и вершину в «свой» цвет, а также отправляет АЭ номер вершины, из которой совершает переход («ПЕРЕХОД_Н_ОБЛ_А(x)», «ПЕРЕХОД_Н_ОБЛ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути). В случае перехода второго АИ в чужую область распознавания и одновременного попадания с первым АИ в одну вершину, первый агент пропускает ход, во время которого второй АИ переходит в новую область распознавания, окрашивая дальние инцидентор и вершину в «свой» цвет («ПЕРЕХОД_Н_ОБЛ_А(x)», «ПЕРЕХОД_Н_ОБЛ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути).

Если в вершине, в которой находится АИ, нет ПБР, то агент отправляет агенту-экспериментатору номер этой вершины («ИНИЦ_ПОЛОЖ_А(x)», «ИНИЦ_ПОЛОЖ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути). АЭ, в свою очередь, проверяет наличие в дереве вершин с ПБР, расположенные ближе к исходной вершине первого АИ, чем вершина, в которой находится второй АИ. Если такие вершины есть, то АИ начинает движение назад по чужому пути, проверяя на каждом шаге наличие ПБР. Обнаружив такие ребра, АИ переходит по любому из них в новую область распознавания, окрашивая дальние инцидентор и вершину в «свой» цвет («ПЕРЕХОД_Н_ОБЛ_А(x)», «ПЕРЕХОД_Н_ОБЛ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути.

Если вершины с ПБР имеются, но расположены глубже по дереву, то АИ начнет двигаться вперед по чужому пути к ближайшей вершине с ПБР, на каждом шаге проверяя наличие пропущенных первым АИ белых ребер. Если таких ребер нет, то АИ останавливает движение до их появления либо до полного завершения работы алгоритма. Если при движении вперед по чужому пути второй АИ попадает с первым АИ в одну вершину, в которой есть ПБР, то первый АИ останавливается в ожидании перехода второго АИ в новую область распознавания. Достигнув нужной вершины, АИ переходит в новую область распознавания, окрашивая дальние инцидентор и вершину в «свой» цвет («ПЕРЕХОД_Н_ОБЛ_А(x)», «ПЕРЕХОД_Н_ОБЛ_В(k)», где x, k — номера, записанные агентами B и A соответственно в вершинах своего пути. После перехода в новую область распознавания АИ заново начинает выполнение алгоритма обхода, но счетчик номеров вершин не обнуляется, т.е. продолжается дальнейшая нумерация. Отметим, что после перехода АИ вершиной перехода новой области распознавания будет отмечена вершина, в которую был совершен переход.

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, а при движении назад по своему направлению путь укорачивается. Если АИ возвратился в вершину, из которой начал обход графа, а в ее окрестности не оказалось белых вершин, то АИ окрашивает эту вершину 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. Посетив вершину первый раз, агент A окрашивает ее в красный цвет (агент B окрашивает в желтый цвет), записывает в память вершины соответствующий номер (полученный от АЭ), равный значению переменной *Сч_А*

(*Сч_B* для агента *B*). Восстановление графа *G* происходит на основе созданной агентами-исследователями нумерации путем построения графа *H*, изоморфного *G*.

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 агентами *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 значение 1 или 0 (значение 1 позволяет агенту *A* распознавать и проверять (при определении первого перешейка) перешейки, в том числе и те, в дальней вершине которых находится агент *B*; значение 0 позволяет агенту *A* распознавать и проверять только те перешейки (если они существуют), в дальней вершине которых нет агента *B*); mr_B — переменная, которая принимает значения 1 или 0 (значение 1 позволяет агенту *B* распознавать и проверять все перешейки даже при наличии в дальней вершине одного из перешейков агента *A*; значение 0 запрещает агенту *B* распознавание и проверку тех перешейков, в дальней вершине которых находится агент *A*); $r(1), r(2), \ldots, r(t)$ — список номеров вершин красного пути, где t — длина этого списка; $y(1), y(2), \ldots, y(p)$ — список номеров вершин желтого пути, где p — длина этого списка; *Mes* — текущее сообщение; $perv\_per\_A$ и $perv\_per\_B$ — переменные, используемые агентами *A* и *B* соответственно для сигнализации АЭ о том, что АИ пометил первый перешеек; $sovp\_\min\_A$ и $sovp\_\min\_B$ — переменные, используемые агентами *A* и *B* соответственно для определения перешейка, инцидентного вершине с наименьшим номером. После попадания в вершину с перешейками первый раз агент для определения минимального номера отправляет АЭ номера всех дальних вершин, инцидентных перешейкам. Далее АИ перебирает перешейки, отправляя номер дальней вершины выбранного перешейка АЭ. Как только отправленный номер совпадет с минимальным, соответствующая переменная принимает значение 1. После этого АИ помечает необходимый перешеек. Переменные $ukor\_put\_A$, $ukor\_put\_B$ сигнализируют агентам *A* и *B* соответственно о необходимости перестроить путь к НВП. Рассматриваемая переменная используется в том случае, когда НВП находится с ВП в одной ветке, но выше по дереву (номер НВП меньше номера ВП). Переменные $ukor\_put\_AA$ и $ukor\_put\_BB$ сигнализируют агентам *A* и *B* соответственно о необходимости перестроить путь к НВП. Рассматриваемая переменная используется в том случае, если номер НВП больше номера ВП. Переменные q_A, q_B используются агентами *A* и *B* соответственно для подсчета количества ПБР на своем пути. Значения $nazad\_kras\_A = 1$, $nazad\_kras\_B = 1$ сигнализируют АИ о том, что при движении назад ближний инцидентор не окрашивается и продолжает быть окрашенным в «свой» для данного агента цвет. В то же время при значении 0, делая шаг назад, ближний инцидентор должен окрашиваться в черный цвет, переменные usl_ost_A и usl_ost_B сигнализируют агентам *A* и *B* о том, что второй АИ закончил распознавание своей территории и находится в поиске новой территории для распознавания; $naprav\_A$, $naprav\_B$ — переменные, определяющие направление движения агентам *A* и *B* соответственно по чужому пути (значение 1 — направление назад, 0 — вперед); $nov\_versh\_per\_A$, $nov\_versh\_per\_B$ — переменные, в которых хранится номер НВП агентов *A* и *B* соответственно; $versh\_per\_A$, $versh\_per\_B$ — переменные, в которых хранится номер ВП агентов *A* и *B* соответственно; $proveroch\_versh\_A$ и

proveroch_ versh_ B — переменные, используемые агентами *A* и *B* соответственно при движении назад по своему пути для хранения номера вершины, принадлежащей пути, к ВП, из которой ведет белый путь. Иными словами, когда АИ делает шаг вперед из такой вершины, рассматриваемая переменная принимает значение номера вершины, из которой началось движение вперед. На основе этих переменных формируются значения переменных *nazad_ kras_ A* и *nazad_ kras_ B* соответственно; *men_ prov_ versh_ A* и *men_ prov_ versh_ B* — переменные, используемые агентами *A* и *B* соответственно для управления процессом изменения значений переменных *proveroch_ versh_ A*, *proveroch_ versh_ B*. Переменные *odna_ vetka_ A* и *odna_ vetka_ B* используются агентами *A* и *B* соответственно для определения местоположения НВП относительно ВП. Если эти вершины находятся в одной ветке (что определяется попаданием АИ в ВП после движения назад по своему пути из НВП), то переменной присваивается значение 1, в противном случае — значение 0. Переменные *konec_ drug_ vetki_ A* и *konec_ drug_ vetki_ B* используются агентами *A* и *B* соответственно для определения местоположения НВП относительно ВП. Если эти вершины находятся в разных ветках (что определяется попаданием АИ в вершину с меньшим номером, чем у ВП (без прохода через ВП), после движения назад по своему пути из НВП), то переменной присваивается значение 1, в противном случае — значение 0. Множества *N_ v_ A*, *N_ v_ B* используются агентами *A* и *B* соответственно для хранения номеров вершин, в которых есть ПБР. Количество одинаковых номеров во множестве равно количеству ПБР в вершинах с этим номером. Переменные min_ *n_ A* и min_ *n_ B* используются агентами *A* и *B* соответственно для хранения минимального номера вершины, необходимого при окрашивании первого перешейка и отрезании пути к ВП или к предыдущим НВП. Переменные *pred_ nazad_ A* и *pred_ nazad_ B* используются агентами *A* и *B* соответственно для определения того, двигался ли АИ назад по своему пути на предыдущем шаге.

1. $Сч\_A := 1$; $Сч\_B := 1$; $M := \varnothing$; $N := \varnothing$; $E_H := \varnothing$; $STOP\_A := 0$; $STOP\_B := 0$; $mr\_A := 0$; $mr\_B := 0$; $t := 1$; $p := 1$; $r(t) := C \div\_ A$; $y(p) := Сч\_B$; $V_H := \{A[1], B[1]\}$; $perv\_per\_A := 0$; $perv\_per\_B := 0$; $sovp\_\min\_A := 0$; $sovp\_\min\_B := 0$; $ukor\_put\_A := 0$; $ukor\_put\_B := 0$; $ukor\_put\_AA := 0$; $ukor\_put\_BB := 0$; $q\_A := 0$; $q\_B := 0$; $nazad\_kras\_A := 0$; $nazad\_kras\_B := 0$; $usl\_ost\_A := 0$; $usl\_ost\_B := 0$; $naprav\_A := 0$; $naprav\_B := 0$; $nov\_versh\_per\_A := 0$; $nov\_versh\_per\_B := 0$; $versh\_per\_A := 0$; $versh\_per\_B := 0$; $proveroch\_versh\_A := 0$; $proveroch\_versh\_B := 0$; $men\_prov\_versh\_A := 0$; $men\_prov\_versh\_B := 0$; $odna\_vetka\_A := 0$; $odna\_vetka\_B := 0$; $konec\_drug\_vetki\_A := 0$; $konec\_drug\_vetki\_B := 0$; $N\_v\_A := \varnothing$; $N\_v\_B := \varnothing$; $\min\_n\_A := 0$; $\min\_n\_B := 0$; $pred\_nazad\_A := 0$; $pred\_nazad\_B := 0$;
2. *while* $(STOP\_A = 0)$ *or* $(STOP\_B = 0)$ *do*
3. *if* $M \neq \varnothing$ *then do*
4. прочитать в *Mes* сообщение и удалить его из списка M;
5. *ОБР_СП_А*(); *end do*;
6. *if* $N \neq \varnothing$ *then do*
7. прочитать в *Mes* сообщение и удалить его из списка N;
8. *ОБР_СП_В*(); *end do*;
9. *if* $(q\_B = 0)$ *and* $(STOP\_B = 1)$ *then* $usl\_ost\_A := 1$;
10. *if* $(q\_A = 0)$ *and* $(STOP\_A = 1)$ *then* $usl\_ost\_B := 1$;
11. *if* $(versh\_per\_A = r(t))$ *and* $(nov\_versh\_per\_A = 0)$ then $proveroch\_versh\_A := r(t)$;
12. *if* $(versh\_per\_B = y(p))$ *and* $(nov\_versh\_per\_B = 0)$ then $proveroch\_versh\_B := y(p)$;
13. *if* $(nov\_versh\_per\_A = r(t))$ *and* $(nov\_versh\_per\_A < versh\_per\_A)$ *then* $ukor\_put\_A := 1$, $proveroch\_versh\_A := r(t)$;
14. *if* $(nov\_versh\_per\_B = y(p))$ *and* $(nov\_versh\_per\_B < versh\_per\_B)$ *then*
15. *if* $(nov\_versh\_per\_A = r(t))$ *and* $(nov\_versh\_per\_A = versh\_per\_A)$ *then* $ukor\_put\_A := 0$, $ukor\_put\_AA := 0$, $proveroch\_versh\_A := r(t)$;
16. *if* $(nov\_versh\_per\_B = y(p))$ *and* $(nov\_versh\_per\_B = versh\_per\_B)$ *then* $ukor\_put\_B := 0$,

ukor_put_BB:= 0,*proveroch_versh_B*:= $y(p)$;

17. *if* $(nov\_versh\_per\_A = r(t))$ *and* $(nov\_versh\_per\_A > versh\_per\_A)$ *then ukor_put_AA*:= 1, *proveroch_versh_A*:= $r(t)$;
18. *if* $(nov\_versh\_per\_B = y(p))$ *and* $(nov\_versh\_per\_B > versh\_per\_B)$ *then ukor_put_BB*:= 1, *proveroch_versh_B*:= $y(p)$;
19. *if* $proveroch\_versh\_A \geq r(t)$ *then nazad_kras_A*:= 1;
20. *else nazad_kras_A*:= 0;
21. *if* $proveroch\_versh\_B \geq y(p)$ *then nazad_kras_B*:= 1;
22. else *nazad_kras_B*:= 0;
23. *if* $proveroch\_versh\_A > r(t)$ *then men_prov_versh_A*:= 1;
24. *else men_prov_versh_A*:= 0;
25. *if* $proveroch\_versh\_B > y(p)$ *then men_prov_versh_B*:= 1;
26. *else men_prov_versh_B*:= 0;
27. *if* $(versh\_per\_A = r(t))$ *and* $(nov\_versh\_per\_A \neq 0)$ *then odna_vetka_A*:= 1;
28. *if* $(versh\_per\_B = y(p))$ *and* $(nov\_versh\_per\_B \neq 0)$ *then odna_vetka_B*:= 1;
29. *if* $(versh\_per\_A > r(t))$ *and* $(nov\_versh\_per\_A \neq 0)$ *then konec_drug_vetki_A*:= 1;
30. *if* $(versh\_per\_B > y(p))$ *and* $(nov\_versh\_per\_B \neq 0)$ *then konec_drug_vetki_B*:= 1;
31. *if* $(q\_B = 0)$ *and* $(STOP\_B = 1)$ *then usl_ost_A*:= 1;
32. *else usl_ost_A*:= 0;
33. *if* $(q\_A = 0)$ *and* $(STOP\_A = 1)$ *then usl_ost_B*:= 1;
34. *else usl_ost_B*:= 0;
35. *end do*;
36. печать V_H, E_H.

Рассмотрим процедуры, используемые в алгоритме.

ОБР_СП_А():

1. *if* $Mes = "ПЕР\_A(x_1, x_2, \ldots, x_l)"$ *then* $ПЕР\_A(x_1, x_2, \ldots, x_l)$;
2. *if* $Mes = "ОБР\_A(x_1, x_2, \ldots, x_l)"$ *then* $ОБР\_A(x_1, x_2, \ldots, x_l)$;
3. *if* $Mes = "ВПЕРЕД\_A(x_1, x_2, \ldots, x_q)"$ *then* $ВПЕРЕД\_A(x_1, x_2, \ldots, x_q)$;
4. *if* $Mes = "ПРИСВ\_Amr"$ *then ПРИСВ_Amr*();
5. *if* $Mes = "НАЗАД\_A"$ *then НАЗАД_А*();
6. *if* $Mes = "УК\_ПУТИ\_A"$ *then УК_ПУТИ_А*();
7. *if* $Mes = "ОТПР\_НОМ\_A(x)"$ *then* $ОТПР\_НОМ\_A(x)$;
8. *if* $Mes = "ВЫБОР\_ВЕРШ\_A(x)"$ *then* $ВЫБОР\_ВЕРШ\_A(x)$;
9. *if* $Mes = "ОБНУЛ\_ПЕР\_A"$ *then ОБНУЛ_ПЕР_А*();
10. *if* $Mes = "ОТМЕН\_УКОР\_A"$ *then ОТМЕН_УКОР_А*();
11. *if* $Mes = "ПЕРВ\_ПЕР\_A"$ *then ПЕРВ_ПЕР_А*();
12. *if* $Mes = "ИНИЦ\_ПОЛОЖ\_A(x)"$ *then* $ИНИЦ\_ПОЛОЖ\_A(x)$;
13. *if* $Mes = "ПЕРЕХОД\_H\_ОБЛ\_A(x)"$ *then* $ПЕРЕХОД\_H\_ОБЛ\_A(x)$;
14. *if* $Mes = "СТОП\_A"$ *then СТОП_А*().

ПЕР_А($x_1, x_2, \ldots, x_l$):

1. $E_H := E_H \cup \{(A[r(t)], B[x_1]); (A[r(t)], B[x_2]); \ldots; (A[r(t)], B[x_l])\}$;
   $q\_B := q\_B - l$; $N\_v\_B := N\_v\_B \setminus \{x_1, x_2, \ldots, x_l\}$.
2. *if perv_per_A* = 0 *then* $\min\_n\_A := \min x_1, x_2, \ldots, x_l$.
   $ОБР\_A(x_1, x_2, \ldots, x_l)$: $E_H :=$
   $= E_H \cup \{(A[r(t)], A[x_1]); (A[r(t)], A[x_2]); \ldots; (A[r(t)], A[x_l])\}$;
   $q\_A := q\_A - l$; $N\_v\_A := N\_v\_A \setminus \{x_1, x_2, \ldots, x_l\}$.

$ВПЕРЕД\_A(x_1, x_2, \ldots, x_q)$:

1. *if pred_nazad_A* = 1 *then do* $N\_v\_A := N\_v\_A \setminus \{r(t)\}$;
2. $q\_A := q\_A - 1$;
3. *pred_nazad_A*:= 0; *end do*;
4. *else do* $q\_A := q\_A + q$;
5. $N\_v\_A := N\_v\_A \cup \{x_1, x_2, \ldots, x_q\}$; *end do*;
6. *if men_prov_versh_A* = 1 *then proveroch_versh_A*:= $r(t)$;
7. $Сч\_A := Сч\_A + 1$; $t := t + 1$; $r(t) := Сч\_A$; $V_H := V_H \cup \{A\ [Сч\_A]\}$;
   $E_H := E_H \cup \{(A[r(t-1)], A[r(t)])\}$; $mr\_B := 0$.

ПРИСВ_Amr (): *mr_B*:= 1.

НАЗАД_А(): из $r(1), \ldots, r(t)$ удаляется элемент $r(t)$; $t := t - 1$; *pred_nazad_A*:= 1.

УК_ПУТИ_А(): *ukor_put_A*:= 0.

ОТПР_НОМ_А(x_1, x_2): min_ n_ $A := \min\{x_1, x_2\}$.
ВЫБОР_ВЕРШ_А(x): *if* $x = $ min_ n_ A then *sovp*_ min_ $A := 1$.
ОБНУЛ_ПЕР_А(): *sovp*_ min_ $A := 0$; min_ n_ $A := 0$.
ОТМЕН_УКОР_А(): *ukor*_ *put*_ *AA* := 0; *odna*_ *vetka*_ $A := 0$;
*konec*_ *drug*_ *vetki*_ $A := 0$.
ПЕРВ_ПЕР_А(): *versh*_ *per*_ $A := r(t)$; *perv*_ *per*_ $A := 1$; *sovp*_ min_ $A := 0$;
min_ n_ $A := 0$.
ИНИЦ_ПОЛОЖ_А(x): *if* $\exists\, n \in N\_v\_B$: $x - n > 0$ *then* *naprav*_ $A := 1$.
*ПЕРЕХОД_Н_ОБЛ_*А(x): *Сч_А* := *Сч_А* + 1; $t := 1$; $r(t) :=$ *Сч_А*;
$V_H := V_H \cup \{A\,[Сч\_A]\}$; $E_H := E_H \cup \{(A[x], A[r(t)])\}$; *mr*_ $B := 0$;
*STOP*_ $A := 0$;
*versh*_ *per*_ $A :=$ *Сч_А*; *perv*_ *per*_ $B := 1$; *naprav*_ $A := 0$; $q\_B := q\_B - 1$;
$N\_v\_B := N\_v\_B \setminus \{x\}$; *nov*_ *versh*_ *per*_ $B := x$.
СТОП_А(): $t := 0$; *mr*_ $A := 0$; *versh*_ *per*_ $A := 0$; *proveroch*_ *versh*_ $A := 0$;
*proveroch*_ *versh*_ $B := 0$; *STOP*_ $A := 1$; *pred*_ *nazad*_ $A := 0$; *ukor*_ *put*_ *BB* := 0.

Процедуры работы со списком сообщений от агента B (которые ниже не рассматриваются) аналогичны процедурам работы со списком сообщений от агента A.

ОБР_СП_В():
1. *if* *Mes* = "*ВОЗВРАТ_В*" *then* *ВОЗВРАТ_В*();
2. *if* *Mes* = "*ПЕР_В*($k_1, k_2, \ldots, k_l$)" *then* *ПЕР_В*($k_1, k_2, \ldots, k_l$);
3. *if* *Mes* = "*ОБР_В*($k_1, k_2, \ldots, k_l$)" *then* *ОБР_В*($k_1, k_2, \ldots, k_l$);
4. *if* *Mes* = "*ВПЕРЕД_В*($k_1, k_2, \ldots, k_g$)" *then* *ВПЕРЕД_В*($k_1, k_2, \ldots, k_g$);
5. *if* *Mes* = "*НАЗАД_В*" *then* *НАЗАД_В*();
6. *if* *Mes* = "*УК_ПУТИ_В*" *then* *УК_ПУТИ_В*();
7. *if* *Mes* = "*ОТПР_НОМ_В*(k)" *then* *ОТПР_НОМ_В*(k);
8. *if* *Mes* = "*ВЫБОР_ВЕРШ_В*(k)" *then* *ВЫБОР_ВЕРШ_В*(k);
9. *if* *Mes* = "*ОБНУЛ_ПЕР_В*" *then* *ОБНУЛ_ПЕР_В*();
10. *if* *Mes* = "*ОТМЕН_УКОР_В*" *then* *ОТМЕН_УКОР_В*();
11. *if* *Mes* = "*ПЕРВ_ПЕР_В*" then *ПЕРВ_ПЕР_В*();
12. *if* *Mes* = "*ИНИЦ_ПОЛОЖ_В*"(k) *then* *ИНИЦ_ПОЛОЖ_В*(k);
13. *if* *Mes* = "*ПЕРЕХОД_Н_ОБЛ_В*(k)" *then* *ПЕРЕХОД_Н_ОБЛ_В*(k);
14. *if* *Mes* = *СТОП_В* *then* СТОП_В().

ВОЗВРАТ_В(): $E_H := E_H \setminus \{y(p-1), y(p)\}$; $V_H := V_H \setminus \{B[Сч\_B]\}$;
Сч_В := *Сч_В* − 1; из $y(1), \ldots, y(p)$ удаляется элемент $y(p)$; $p := p - 1$;
*pred*_ *nazad*_ $B := 1$.
ПЕР_В($k_1, k_2, \ldots, k_l$):
1. $E_H := E_H \cup \{(B[y(p)], A[k_1]); (B[y(p)], A[k_2]); \ldots; (B[y(p)], A[k_l])\}$;
$q\_A := q\_A - l$; $N\_v\_A := N\_v\_A \setminus \{k_1, k_2, \ldots, k_l\}$; *mr*_ $A := 1$.
2. if *perv*_ *per*_ $B = 0$ *then* min_ n_ $B := \min k_1, k_2, \ldots, k_l$.

НАЗАД_В(): из $y(1), \ldots, y(p)$ удаляется элемент $y(p)$; $p := p - 1$; *mr*_ $A := 1$;
*pred*_ *nazad*_ $B := 1$.

СВОЙСТВА АЛГОРИТМА РАСПОЗНАВАНИЯ

Процедуры ВПЕРЕД_А($x_1, x_2, \ldots, x_q$) и ВПЕРЕД_В($k_1, k_2, \ldots, k_g$) (при переходе в новую область распознавания — ПЕРЕХОД_Н_ОБЛ_А(x) и ПЕРЕХОД_Н_ОБЛ_В(k)) выполняются АЭ при посещении агентами-исследователями белых вершин исследуемого графа G. С помощью этих процедур АЭ создает по одной новой вершине графа H. При одновременном попадании агентов A и B в одну белую вершину процедурами ВПЕРЕД_А (x_1, $x_2, \ldots, x_q$) и ВПЕРЕД_В($k_1, k_2, \ldots, k_g$) будет создано две новые вершины графа H. Для того чтобы не допустить подобного дублирования вершин, на следующем шаге агент B процедурой ВОЗВРАТ_В() удалит вершину, созданную им на предыдущем шаге. Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\varphi: V_G \to V_H$, причем $\varphi(\upsilon) = Сч\_A$ (вершина υ окрашена в красный цвет) и $\varphi(s) = Сч\_B$ (вершина s окрашена в желтый

цвет). Указанное отображение φ является биекцией, поскольку АИ посещают все вершины и каждой вершине при первом ее посещении агентом присваивается единственный номер.

При выполнении процедуры ВПЕРЕД_А$(x_1, x_2, \ldots, x_q)$ или ВПЕРЕД_ В$(k_1, k_2, \ldots, k_g)$ АЭ распознает древесное ребро (v, u) и нумерует вершину u таким образом, чтобы ребру (v, u) однозначно соответствовало ребро $(\varphi(v), \varphi(u))$ графа H. При выполнении процедур ОБР_А$(x_1, x_2, \ldots, x_l)$ или ОБР_В$(k_1, k_2, \ldots, k_l)$ 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_А$(x_1, x_2, \ldots, x_l)$ или ПЕР_В$(k_1, k_2, \ldots, k_l)$ 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.

Определим временную, емкостную и коммуникационную сложности алгоритма по равномерной шкале [8]. Из описания алгоритма следует, что на каждом шаге алгоритма красный (желтый) путь — это простой путь, соединяющий начальную вершину v (s — в случае агента B) под номером $\varphi(v) = 1$ ($\varphi(s) = 1$) с вершиной u (z — в случае агента B) под номером $\varphi(u) = Сч\_A$ ($\varphi(z) = Сч\_B$). Следовательно, общая длина красного и желтого путей не превосходит n.

На однократное выполнение процедур из ОРР агент-исследователь затрачивает один ход. При однократном выполнении процедур из РРОР агенты 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 также затрачивается один ход. При одновременном попадании двух АИ в одну белую вершину агент A не меняет режима работы, а агент B затрачивает один ход на возврат в свою область работы. На выполнение каждой процедуры при поиске новой территории для распознавания АИ затрачивается один ход. При подсчете временной сложности алгоритма предположим, что инициализация алгоритма, анализ окрестности $Q(v)$ рабочей вершины и выбор одной из возможных процедур занимают некоторое постоянное число единиц времени. Также будем считать, что выбор ребер, проход по ним АИ и обработка сообщений, отправленных на данном этапе, осуществляется за одну единицу времени. Тогда временная сложность алгоритма определяется следующим образом.

Процедуры ОРР выполняются не более чем $2(n-1)+4n$ раз, общее время их выполнения оценивается как $O(n)$. Процедуры РРОР выполняются не более чем $n-2$ раза, т.е. общее время их выполнения оценивается как $O(n)$. Процедуры РРП выполняются не более чем $(n-2)+(n-1)$ раз, т.е. общее время их выполнения оценивается как $O(n)$. Время простоя АИ во всех режимах работы не превосходит $3(n-1)+2(n-2)+4n$, т.е. оценивается как $O(n)$. Процедуры АИ для перехода в новые области распознавания выполнятся не более чем $4(n-1)$ раз. 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, $r(1), \ldots, r(t)$, $y(1), \ldots, y(p)$, N_v_A и N_v_B, сложность которых соответственно определяется величинами $O(n \cdot \log(n))$, $O(n^2)$, $O(n \cdot \log(n))$, $O(n \cdot \log(n))$, $O(n^2)$, $O(n^2)$. Следовательно, $S(n) = O(n^2)$.

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 $O(n^2 \cdot \log(n))$. При этом АЭ передаст объем информации, оцениваемый как $O(n \cdot \log(n))$. При работе агентов в РРОР и в РРП объем переданной информации оценивается как $2 \cdot O(n^2 \cdot \log(n))$. Следовательно, суммарная коммуникационная сложность алгоритма удовлетворяет соотношению $K(n) = O(n^2 \cdot \log(n))$.

Теорема 2. Временная сложность алгоритма распознавания оценивается как $O(n)$, емкостная — $O(n^2)$, а коммуникационная — $O(n^2 \cdot \log(n))$. При этом алгоритм использует три краски.

ЗАКЛЮЧЕНИЕ

В настоящей статье предложен новый алгоритм распознавания конечных неориентированных графов временной сложности $O(n)$, емкостной сложности $O(n^2)$ и коммуникационной сложности $O(n^2 \cdot \log(n))$. Разработана процедура оптимизации разбиения графа на части, распознаваемые различными агентами. Агенты-исследователи имеют конечную, но растущую на каждом шаге память и используют по две краски каждый (всего три краски).

СПИСОК ЛИТЕРАТУРЫ

1. Albers S., Henzinger M.R. Exploring unknown environments // SIAM J. on Comput. — 2000. — **29,** N 4. — P. 1164–1188.
2. Килибарда Г., Кудрявцев В.Б., Ушчумлич Щ. Независимые системы автоматов в лабиринтах // Дискретная математика. — 2003. — **15**, вып. 2. — С. 3–39.
3. Dudek G., Jenkin M., Milios E., Wilkes D. Map validation in a graphlike world // Proc. of the 13th Intern. Joint Conf. on Artifical Intelligence (Chambery, France, August 1993). — San Fransisco: Morgan Kaufmann Publishers Inc., 1993. — P. 1648–1653.
4. Килибарда Г., Кудрявцев В Б., Ушчумлич Щ. Коллективы автоматов в лабиринтах // Дискретная математика. — 2003. — **15**, вып. 3. — С. 3–40.
5. Грунский И.С., Сапунов С.В. Контроль графов с отмеченными вершинами // Тр. Донецк. гос. техн. ун-та. Сер.: Выч. техника и автоматика. — 2002. — Вып. 38. — С. 226–232.
6. Грунский И.С., Стёпкин А.В. Распознавание конечного графа коллективом агентов // Тр. ИПММ НАН Украины. — 2009. — **19.** — С. 43–52.
7. Стёпкин А.В. Возможность и сложность распознавания графов тремя агентами // Таврический вестник информатики и математики. — 2012. — № 1 (20). — С. 88–98.
8. Кормен Т., Лейзерсон Ч. Алгоритмы: построение и анализ. — М.: МЦНМО, 2001. — 960 с.

Поступила 03.01.2014