КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2021 Т. 13 № 1 С. 33–45
DOI: 10.20537/2076-7633-2021-13-1-33-45

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

УДК: 519.7

Алгоритм распознавания простых графов коллективом агентов

А. В. Стёпкин[a], А. С. Стёпкина[b]

ГВУЗ «Донбасский государственный педагогический университет»,
Украина, 84116, Донецкая область, г. Славянск, ул. Г. Батюка, д. 19

E-mail: [a] stepkin.andrey@rambler.ru, [b] brusal@ukr.net

Получено 06.07.2020, после доработки — 08.11.2020.
Принято к публикации 16.11.2020.

Исследование, представленное в работе, посвящено проблеме распознавания конечных графов с помощью коллектива агентов. В работе рассматриваются конечные неориентированных графы без петель и кратных ребер. Коллектив агентов состоит из двух агентов-исследователей, которые имеют конечную память, независимую от числа вершин исследуемого ими графа, и используют по две краски каждый (в общей сложности используется три различные краски, так как цвет одной из красок у агентов совпадает), и одного агента-экспериментатора, который обладает конечной, неограниченно растущей внутренней памятью. Агенты-исследователи могут одновременно передвигаться по графу, считывать и изменять метки элементов графа, а также передавать необходимую информацию третьему агенту — агенту-экспериментатору. Агент-экспериментатор — это неподвижный агент, в памяти которого фиксируется результат функционирования агентов-исследователей на каждом шаге и, кроме того, постепенно выстраивается представление исследуемого графа (изначально неизвестного агентам) списком ребер и списком вершин.

В работе подробно описаны режимы работы агентов-исследователей с указанием приоритетности их активации, рассмотрены команды, которыми обмениваются агенты-исследователи с агентом-экспериментатором во время выполнения тех или иных процедур. Также подробно рассмотрены проблемные ситуации, возникающие в работе агентов-исследователей, например окрашивание белой вершины при одновременном попадании двух агентов в одну и ту же вершину или пометка и распознавание ребер перешейков (ребра, соединяющие подграфы, распознаваемые различными агентами-исследователями) и так далее. Представлен полный алгоритм работы агента-экспериментатора с подробным описанием процедур обработки полученных от агентов-исследователей сообщений, на основании которых и происходит построение представления исследуемого агентами графа. Также в работе проведен полный анализ временной, емкостной и коммуникационной сложностей построенного алгоритма.

Представленный алгоритм распознавания графов имеет квадратичную (от числа вершин исследуемого графа) временную сложность, квадратичную емкостную сложность и квадратичную коммуникационную сложность. Работа алгоритма распознавания основывается на методе обхода графа в глубину.

Ключевые слова: коллектив агентов, распознавание графов, метод в глубину

COMPUTER RESEARCH AND MODELING
2021 VOL. 13 NO. 1 P. 33–45
DOI: 10.20537/2076-7633-2021-13-1-33-45

Ки&М

MATHEMATICAL MODELING AND NUMERICAL SIMULATION

UDC: 519.7

Algorithm of simple graph exploration by a collective of agents

A. V. Stepkin[a], A. S. Stepkina[b]

State Higher Educational Institution "Donbass State Pedagogical University",
19 G. Batuka st., Donetsk region, Slavyansk, 84116, Ukraine

E-mail: [a] stepkin.andrey@rambler.ru, [b] brusal@ukr.net

Received 06.07.2020, after completion — 08.11.2020.
Accepted for publication 16.11.2020.

The study presented in the paper is devoted to the problem of finite graph exploration using a collective of agents. Finite non-oriented graphs without loops and multiple edges are considered in this paper. The collective of agents consists of two agents-researchers, who have a finite memory independent of the number of nodes of the graph studied by them and use two colors each (three colors are used in the aggregate) and one agent-experimental, who has a finite, unlimitedly growing internal memory. Agents-researches can simultaneously traverse the graph, read and change labels of graph elements, and also transmit the necessary information to a third agent — the agent-experimenter. An agent-experimenter is a non-moving agent in whose memory the result of the functioning of agents-researchers at each step is recorded and, also, a representation of the investigated graph (initially unknown to agents) is gradually built up with a list of edges and a list of nodes.

The work includes detail describes of the operating modes of agents-researchers with an indication of the priority of their activation. The commands exchanged between agents-researchers and an agent-experimenter during the execution of procedures are considered. Problematic situations arising in the work of agents-researchers are also studied in detail, for example, staining a white vertex, when two agents simultaneously fall into the same node, or marking and examining the isthmus (edges connecting subgraphs examined by different agents-researchers), etc. The full algorithm of the agent-experimenter is presented with a detailed description of the processing of messages received from agents-researchers, on the basis of which a representation of the studied graph is built. In addition, a complete analysis of the time, space, and communication complexities of the constructed algorithm was performed.

The presented graph exploration algorithm has a quadratic (with respect to the number of nodes of the studied graph) time complexity, quadratic space complexity, and quadratic communication complexity. The graph exploration algorithm is based on the depth-first traversal method.

Keywords: collective of agents, exploration of graphs, graph traversal

Citation: *Computer Research and Modeling*, 2021, vol. 13, no. 1, pp. 33–45 (Russian).

Введение

Все более актуальным и стремительно развивающимся направлением кибернетики становится робототехника. В истоках робототехники стоит желание помочь человеку при выполнении тяжелых и даже опасных работ, а то и вовсе заменить человека при выполнении таких работ. Если взять во внимание, что в мире до сих пор остаются неисследованные среды [Albers, Henzinger, 2000; Goth, 2009] по причине опасности или недоступности для человека, то становится очевидным, что решение задачи исследования сред роботами является актуальной проблемой [Грунский, Тихончев, 2001; Wang et al., 2008; Sapunov, Senchenko, 2019], причем при использовании не одного робота, а сразу нескольких [Wang et al., 2009; Zhang, 2010; Kilibarda et al., 2003] с более высокой вероятностью (если учесть возможность выхода из строя агентов в процессе работы в среде) будет решена поставленная задача, а возможно, даже будет улучшена временная оценка работы алгоритма.

Началом исследований в этом научном направлении принято считать работу К. Шеннона [Shannon, 1951], в которой рассматривалась задача поиска автоматом-мышью заданной цели в лабиринте.

Активное изучение поведения автоматов в лабиринтах начинается в 1971 году после появления работы К. Деппа [Dopp, 1971], описывающей обход шахматных лабиринтов конечными автоматами. Со временем условия задачи расширялись, и автомату уже необходимо было посетить все вершины и/или ребра исследуемого графа (лабиринта). В дальнейшем в работе [Dudek, Jenkin, 2000] проводились исследования в области анализа свойств неизвестной среды при различных способах взаимодействия автомата с операционной средой, а также при различной априорной информации о ней.

Анализ графа включает в себя ряд частных задач. Рассмотрим основные: проблема самолокализации (определения вершины графа, в которой изначально расположен автомат), проблема контроля карты (проверки изоморфизма исследуемого графа и графа-эталона (карты)) и проблема полного распознавания графа (построения графа-эталона (карты) исследуемого графа). Определен ряд подходов к решению последней проблемы, предложен ряд алгоритмов блуждания агента по графу и способов отметки элементов графа красками или камнями, позволяющих распознать исследуемый граф с точностью до изоморфизма. Так, в работе [Dudek et al., 1993] предлагается метод, в котором агент при передвижении метит только инциденторы (точка соединения ребер с вершинами) пройденных ребер. Для распознавания ребер агент переходит по ним, затем кратчайшим путем возвращается в начальную вершину. При этом запоминает метки пройденных инциденторов, которые однозначно определяют вершину, в которую попал агент, переходя по исследуемому ребру. В работе [Грунский, Тихончев, 2001] предложен метод распознавания графа, в котором автомату для определения вершины, инцидентной неисследованному ребру, не надо возвращаться в начальную вершину. Но для проведения эксперимента по распознаванию графа автомату требуется число различных красок, равное высоте дерева обхода графа в глубину.

Начиная с 1993 года публикуются работы по исследованию графов роем агентов [Dudek et al., 1993; Dudek et al., 1998]. Алгоритм работы роя заключается в следующем. Все агенты (память каждого агента может вместить полную карту исследуемого графа) начинают работу с одной вершины. Договорившись заранее о времени следующей встречи, агенты делят доступные ребра между собой и расходятся по ним. После обхода определенной части графа агенты возвращаются в исходную вершину для объединения полученных карт. Далее они договариваются о следующей встрече и снова расходятся, и так продолжается до полного распознавания графа. Аналогичные алгоритмы с использованием роя агентов рассматривались в работах [Wang et al., 2008; Wang et al., 2009; Zhang, 2010]. А вот, например, в работе [Das et al., 2007] используется коллектив агентов (память каждого агента может вместить полную карту исследуемого графа), которые начинают работу с разных вершин графа и взаимодействуют между собой посредством токенов, записываемых агентами в вершины графа.

При попадании нескольких агентов в одну вершину каждый из них знает, сколько агентов в вершине, и может обмениваться информацией с любым из них, таким образом объединяя необходимые части карт.

В работе [Стёпкин, 2013] был рассмотрен аналогичный представленному в этой статье коллектив агентов, но с дополнительными ограничениями, накладываемыми на исследуемый граф, используемых агентов и на канал связи между агентами. Например, в работе 2013 года каждая вершина исследуемого графа должна иметь память, позволяющую записать число, равное количеству вершин этого графа, в то время как представленная работа позволяет не накладывать это ограничение и каждая вершина должна иметь одинаковую для графа любой размерности память, достаточную для хранения нанесенных красок (не больше двух красок одновременно). Также в работе 2013 года была возможность отправлять номера всех вершин из окрестности вершины, в которой находился агент-исследователь, что значительно упрощало работу с перешейками и обратными ребрами, но значительно увеличивало ширину канала связи, в то время как сейчас мы отправляем только количество шагов, сделанных агентами при выполнении аналогичных процедур. Стоит также отметить, что в работе 2013 года была возможность считывать номера с дальних вершин, что влекло за собой зависимость памяти агента-исследователя от размерности исследуемого графа, и один коллектив агентов мог распознавать только те графы, на которые у него хватало памяти. В этой же работе представлен алгоритм, который позволяет одному коллективу распознавать граф любой размерности.

В работе [Nagavarapu et al., 2016] предлагается децентрализованный подход к исследованию графов коллективом агентов, в котором агенты могут избегать столкновений, не имея между собой прямой связи (обмен информацией между роботами происходит посредством маяков, установленных в ранее посещенных вершинах графа). Стоит отметить, что в этом алгоритме принятие решений лежит непосредственно на каждом агенте и не зависит от действий других агентов. Предлагаемый децентрализованный метод гарантирует завершение исследования неизвестной среды за конечное число шагов. Структура исследуемой среды строится постепенно во время работы агентов.

А вот, например, в работе [Banfi et al., 2018] агенты обмениваются информацией о проделанной работе с базовой станцией через специальную сеть. Эта сеть имеет некоторые ограничения по обмену данными. То есть для координации действий между агентами нужно сначала сформировать сеть, удовлетворяющую этим условиям. Иными словами, обмен данными с базовой станцией происходит в заранее назначенных местах. Особенность статьи — асинхронные стратегии, работающие с произвольными моделями коммуникации. Под асинхронностью понимается возможность выдавать инструкции подгруппам агентов (в тот момент, когда они будут готовы их принять).

В статье [Nagavarapu et al., 2020] сделана попытка разработки общих концепций и требований для исследования графов мультиагентными системами. Предлагаемое представление для мультиагентной системы направлено на предоставление общих условий для объявления завершения исследования графа и завершения такого исследования за конечное время. В работе предложены модификации матрицы инцидентности для организованного обмена информацией между агентами. Также представлен общий алгоритм работы коллектива агентов, исследующих граф. Алгоритм основан на предложенной обобщенной структуре и модифицированной матрице инцидентности.

Исследованию графа при помощи одного агента посвящено много работ, получено множество результатов о возможности и сложности такого распознавания. При этом остается малоисследованным распознавание графа коллективом блуждающих по нему агентов. Это делает актуальными задачи проведения систематического исследования экспериментов по распознаванию графа несколькими блуждающими по нему агентами, то есть создания маршрутов движения агентов по неизвестному графу, разметки его элементов, сбора и обработки локальной информации о графе и способов построения графа по этой информации, с точностью до отметок на элементах графа. А также задачи, направленные на поиск методов оптимизации расхода ресурсов, затрат времени, нагрузки на канал связи и т. д. при распознавании графов.

В известных работах мало внимания уделено исследованию обмена информацией между агентами, а также ее сложности и возможного влияния на временную и емкостную сложность.

Цель работы — создание эффективного метода и построение соответствующего алгоритма распознавания неизвестных графов при помощи коллектива агентов [Стёпкин, 2013; Stepkin, 2015], а также исследование временной, емкостной и коммуникационной сложности построенного алгоритма.

Основные понятия и определения

В работе рассматриваются конечные неориентированные графы без петель и кратных ребер. Понятия, которые не рассмотрены ниже, общеизвестны, и с ними можно ознакомиться, например, в [Кормен и др., 2001; Касьянов, Евстигнеев, 2003]. Пусть $G=(V,E)$ — граф, у которого V — множество вершин, E — множество ребер (двухэлементных подмножеств (v,u), где $u,v\in V$). Тройку $((v,u),u)$ будем называть инцидентором (точкой прикосновения) ребра (v,u) и вершины u. Множество таких троек обозначим как 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, а b — черный. Пара (G,μ) называется раскрашенным графом. Последовательность $u_1,u_2,\ldots,u_k$ попарно смежных вершин называется путем в графе G, а k — длиной 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 инцидентов $((v,u),v)$, $((v,u),u)$. Мощность множеств вершин V и ребер E обозначим через n и m соответственно. Ясно, что $m\le\frac{n(n-1)}{2}$. Изоморфизмом графа G и графа H назовем такую биекцию $\phi: V_G\to V_H$, что $(v,u)\in E_G$ тогда и только тогда, когда $(\phi(v),\phi(u))\in E_H$. Таким образом, изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов.

Коллектив агентов состоит из двух типов агентов:

- агент-исследователь (АИ) — это агент, который может перемещаться по конечному графу, в процессе работы может окрашивать вершины, ребра и инциденторы графа, воспринимать эти отметки (на основании этих отметок и осуществляется перемещение АИ по графу), также АИ может отправлять сообщения агенту-экспериментатору;
- агент-экспериментатор (АЭ) — это неподвижный агент, который в процессе работы может передавать, принимать и идентифицировать сообщения АИ, обладает конечной, неограниченно растущей внутренней памятью, в которой фиксируется результат функционирования АИ на каждом шаге и, кроме того, постепенно выстраивается представление графа G (изначально неизвестного агентам) списками ребер и вершин.

В работе предложен алгоритм, в котором используются два АИ (назовем их A и B) и один АЭ. Для работы каждому АИ требуется две краски: у агента A это r и b, у агента B — y и b, то есть всего в алгоритме используется три краски: r, y, b.

Стратегия решения задачи

При описании алгоритма используются результаты и обозначения из [Стёпкин, 2011; Грунский, Стёпкин, 2009]. Алгоритм основан на методе обхода графа в глубину [Кормен и др., 2001].

В начале работы АИ *A* и *B* помещаются в произвольные несовпадающие вершины графа *G*. Агенты идут «в глубину», пока это возможно, возвращаются назад, ищут другой путь с еще не посещенными вершинами и не пройденными ребрами. В случае обнаружения смежной вершины, окрашенной в «чужой» цвет, агент метит все перешейки (ребра, соединяющие подграфы, распознаваемые различными агентами-исследователями) из текущей вершины в чужую область и сообщает второму АИ (через АЭ) о необходимости распознавания помеченных перешейков. Пока второй АИ распознает перешейки, первый АИ не может метить новые перешейки. В случае отсутствия других возможных вариантов перемещения, кроме как пометить новый перешеек, первый АИ останавливается до того момента, пока второй АИ не распознает все помеченные перешейки. При движении в белые вершины графа агенты-исследователи *A* и *B* окрашивают его элементы в красный и желтый цвета соответственно. При движении «в глубину» красный (желтый) путь удлиняется, при движении назад — укорачивается. При движении АИ назад (для распознавания обратных ребер или перешейков) длина пути не изменяется. Вершина, из которой возможно лишь движение назад по своему пути (либо вовсе отсутствуют варианты перемещения), окрашивается в черный цвет. Таким образом формируются красный и желтый пути каждого из агентов. Алгоритм заканчивает работу, когда красный и желтый пути становятся пустыми, а все вершины — черными. Выполняя обход графа *G*, агенты создают неявную нумерацию посещенных вершин. Первый раз посетив вершину, агент *A* окрашивает ее в красный цвет (агент *B* — в желтый), и АЭ ставит ей в соответствие номер, равный значению переменной *Ct_A* (*Ct_B* — для агента *B*). Отметим, что переменные *Ct_A* и *Ct_B* хранятся и обрабатываются агентом-экспериментатором и могут принимать только нечетные и четные значения соответственно. Восстановление графа *G* происходит на основе созданной агентами нумерации [Грунский, Стёпкин, 2009] путем построения графа *H*, изоморфного *G*.

Далее рассмотрим режимы, в которых могут функционировать АИ. При описании режимов в скобках указываются сообщения, отправляемые АИ в соответствующей ситуации, с указанием названия агента, который отправляет сообщение («СООБЩЕНИЕ_А», «СООБЩЕНИЕ_В»).

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

2. *Режим распознавания обратных ребер.* Если при движении вперед в вершине *v* было обнаружено белое ребро, у которого дальняя вершина окрашена в «свой» цвет (обратное ребро), то АИ переключается в режим распознавания обратных ребер и красит в «свой» цвет ближние инциденторы всех обратных ребер, инцидентных вершине *v* («МЕТКА_ОР_А», «МЕТКА_ОР_В»). Завершив покраску инциденторов, АИ двигается назад по своему пути, на каждом шаге отправляя сообщение («ОТСТУП_А», «ОТСТУП_В»), до обнаружения вершины, инцидентной помеченному ранее обратному ребру, переходит по этому ребру, окрашивая его в черный цвет («ВПЕРЕД_ОР_А», «ВПЕРЕД_ОР_В»). Если остались нераспознанные обратные ребра, то АИ возвращается назад по пройденному на предыдущем шаге ребру, окрашивая в черный цвет ближний инцидентор, и продолжает поиск обратных ребер. После обнаружения последнего обратного ребра АИ окрашивает ближний его инцидентор в черный цвет («ОР_РАСПОЗНАНЫ_А», «ОР_РАСПОЗНАНЫ_В») и завершает работу в режиме распознавания обратных ребер.

3. *Режим пометки перешейков.* Если в процессе обхода графа в вершине *v* был обнаружен перешеек, то при условии, что все ранее помеченные данным АИ перешейки были распознаны вторым АИ, агент переключается в режим пометки перешейков. Если же второй АИ еще не распознал все помеченные перешейки, то первый АИ не может метить новые перешейки, и в случае, когда у первого АИ отсутствуют другие возможные варианты перемещения, кроме

как пометить новый перешеек, он останавливается до того момента, пока второй АИ не распознает все помеченные перешейки. Работая в этом режиме, АИ окрашивает ближние инциденторы всех перешейков, инцидентных вершине *v*, в черный цвет («МЕТИМ_АВ», «МЕТИМ_ВА»). Когда все перешейки помечены, работа АИ в данном режиме завершается («ФИКС_А», «ФИКС_В»). По завершении режима пометки перешейков АЭ содержит информацию о количестве помеченных перешейков. В данном режиме работы агент *A* имеет приоритет над агентом *B*, поэтому в ситуации, когда оба АИ одновременно обнаружат один и тот же перешеек, он будет помечен агентом *A*.

4. *Режим распознавания перешейков.* Получив от АЭ команду о необходимости распознавания перешейков, АИ переключается в режим распознавания перешейков. Если в этот момент АИ работает в режиме распознавания обратных ребер, то он не переключится в режим распознавания перешейков до завершения текущего режима работы. При переключении в режим АИ проверяет наличие из вершины, в которой он находится, других возможных путей перемещения кроме как движение назад по своему пути. Если такие пути есть, то АИ возвращается назад по своему пути, ничего не окрашивая («ОТСТУП_А», «ОТСТУП_В»), до обнаружения ближайшей вершины, инцидентной белому ребру, у которого дальний инцидентор окрашен в черный цвет, а дальняя вершина окрашена в «чужой» цвет (ребро перешеек). Если же таких путей нет, то, возвращаясь назад по своему пути, АИ окрашивает его в черный цвет («НАЗАД_А», «НАЗАД_В») до тех пор, пока не попадет в вершину, инцидентную помеченному перешейку, либо же в вершину с другими возможными путями перемещения. Во втором случае АИ продолжает возвращаться назад по своему пути, ничего не окрашивая (каждый шаг назад, отправляя сообщение «ОТСТУП_А», «ОТСТУП_В»), до обнаружения ближайшей вершины, инцидентной помеченному перешейку.

Если в процессе пометки перешейков был помечен один перешеек, то АИ окрашивает ближний инцидентор помеченного перешейка в черный цвет («РАСП_АВ», «РАСП_ВА») и далее движется вперед в конец пути «своего» цвета.

Если же было помечено не менее двух перешейков, то АИ окрашивает ближний инцидентор помеченного перешейка в «свой» цвет («РАСП_АВ», «РАСП_ВА»). Далее АИ движется назад по своему пути («ОТСТУП_А», «ОТСТУП_В»), пока не будет найден следующий помеченный перешеек. В этом случае, если помеченный перешеек оказался не последним, АИ окрашивает ближний инцидентор в черный цвет («РАСП_АВ», «РАСП_ВА»), и на следующем шаге АИ снова возвращается назад по своему пути до следующего помеченного перешейка («ОТСТУП_А», «ОТСТУП_В»). Если же помеченный перешеек оказался последним, то АИ окрашивает ближний инцидентор в «свой» цвет («РАСП_АВ», «РАСП_ВА»). На следующем шаге АИ сообщает АЭ о распознавании всех помеченных другим АИ перешейков («ОБН_А», «ОБН_В»). Далее переходит по последнему перешейку в чужую область, окрашивая ближний инцидентор в черный цвет. На следующем шаге АИ переходит по первому распознанному перешейку в свою область, окрашивая дальний инцидентор в черный цвет. Далее АИ движется вперед в конец пути «своего» цвета.

5. *Одновременное попадание двух АИ в одну белую вершину.* В случае одновременного попадания обоих АИ в одну и ту же белую вершину каждый из них окрашивает вершину наполовину, и она становится красно-желтой. На следующем шаге агент *B* делает шаг назад по своему пути, удаляет краску, нанесенную на предыдущем шаге, с ближнего инцидентора и ребра, окрашивает дальний инцидентор в черный цвет («ВОЗВРАТ_В») и переходит в режим пометки перешейков, причем ребро, по которому он совершил переход, за счет обработки АЭ сообщения «ВОЗВРАТ_В» уже посчитано как первый перешеек, а длина желтого пути уменьшена на одну вершину. Обратим внимание, что агент *A* во время работы на своем пути обрабатывает красно-желтую вершину как вершину красного цвета.

Приоритетность переключения АИ в один из возможных режимов работы распределяется следующим образом: первым делом мы должны проверить наличие помеченных перешейков, поэтому наивысший приоритет имеет режим распознавания перешейков; далее мы проверяем,

нет ли из текущей вершины перешейков, которые можно пометить для распознавания агентом A, поэтому вторым по приоритету переключения будет режим пометки перешейков. Далее мы проверяем наличие обратных ребер из текущей вершины. Соответственно, третьим по приоритету у нас будет режим распознавания обратных ребер. Последним по приоритетности является обычный режим работы. Режим работы АИ в случае одновременного попадания их в одну белую вершину не рассматривается, так как агент A в этом случае продолжает работу в обычном режиме, а для агента B в этот момент выбор иного режима работы будет недоступен.

Алгоритм работы АЭ

Вход: списки сообщений SS_A и SS_B 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. Ct_A, Ct_B — счетчики числа посещенных вершин графа G агентами A и B соответственно. AN, BN — переменные, в которых значение 1 дает агентам A, B соответственно сигнал для возврата и распознания перешейков; значение же 0 сообщает агентам, что перешейки, помеченные для конкретного агента на распознавание, отсутствуют. N_A, N_B — переменные, в которых хранятся номера вершин, из которых агенты A и B соответственно в последний раз помечали перешейки. F, K — количество перешейков из вершин N_A, N_B соответственно, помеченных для распознания. Q, Z — переменные, используемые в некоторых подпрограммах работы с перешейками, для хранения значений переменных F, K соответственно, которые эти переменные имели на начало работы подпрограмм. SP_A, SP_B — списки сообщений от агентов A и B соответственно. E, L — переменные, в которых делается отметка о том, был ли на предыдущем шаге агентом A или B соответственно помечен перешеек (значение 1) или нет (значение 0). i, j — счетчики, используемые агентами A и B соответственно при определении номеров вторых вершин помеченных перешейков или номеров вторых вершин помеченных обратных ребер. $STOP\_A$, $STOP\_B$ — переменные, используемые агентами A и B соответственно для сигнализации АЭ о завершении распознавания своего подграфа. UDP_A, UDP_B — логические переменные, используемые агентами A и B соответственно для определения способа окраски инцидентов рассматриваемого в определенный момент перешейка. $UDOBR\_A$, $UDOBR\_B$ — логические переменные, используемые агентами A и B соответственно для определения, является ли рассматриваемое обратное ребро последним из помеченных. $KOBR\_A$, $KOBR\_B$ — переменные, в которые агенты A и B соответственно записывают количество помеченных обратных ребер. $r(1), r(2), \ldots, r(t)$ — 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 — обрабатываемое сообщение.

1. $AN := 0; BN := 0; N\_A := 0; N\_B := 0; F := 0; K := 0; SS\_A := \varnothing; SS\_B := \varnothing; E := 0; L := 0;$
   $i := 0; j := 0; E_H := \varnothing; Ct\_A := 1; Ct\_B := 2; V_H := \{Ct\_A, Ct\_B\}; t := 1; p := 1;$
   $r(t) := Ct\_A; y(p) := Ct\_B; UDP\_A := False; UDP\_B := False;$
   $UDOBR\_A := False; UDOBR\_B := False; KOBR\_A := 0; KOBR\_B := 0;$
   $STOP\_A := 0; STOP\_B := 0;$
2. *while* $(STOP\_A = 0)$ *or* $(STOP\_B = 0)$ *do*
3. *if* $SS\_A \neq \varnothing$ *then do*
4. *прочитать сообщение в* Mes и удалить его из SS_A;
5. *ОБР_СП_А*();
6. e*nd do*;

7. *if* $SS\_B \neq \varnothing$ *then do*
8. *прочитать сообщение в* *Mes* и удалить его из SS_B;
9. *ОБР_СП_В*();
10. *end do*;
11. e*nd do*;
12. печать V_H, E_H.

ОБР_СП_А():
1. *if Mes* = ”ВПЕРЕД_А” *then ВПЕРЕД_А*();
2. *if Mes* = ”НАЗАД_А” *then НАЗАД_А*();
3. *if Mes* = ”СТОП_А” *then СТОП_А*();
4. *if Mes* = ”МЕТКА_ОР_А” *then МЕТКА_ОР_А*();
5. *if Mes* = ”ОТСТУП_А” *then ОТСТУП_А*();
6. *if Mes* = ”ВПЕРЕД_ОР_А” *then ВПЕРЕД_ОР_А*();
7. *if Mes* = ”ОР_РАСПОЗНАНЫ_А” *then ОР_РАСПОЗНАНЫ_А*();
8. *if Mes* = ”МЕТИМ_АВ” *then МЕТИМ_АВ*();
9. *if Mes* = ”ФИКС_А” *then ФИКС_А*();
10. *if Mes* = ”РАСП_АВ” *then РАСП_АВ*();
11. *if Mes* = ”ОБН_А” *then ОБН_А*();

ВПЕРЕД_А(): $Ct\_A := Ct\_A + 2; t := t + 1; r(t) := Ct\_A; V_H := V_H \cup \{Ct\_A\};$

$$E_H := E_H \cup \{(r(t-1), r(t))\};$$

НАЗАД_А(): из списка $r(1), r(2), \ldots, r(t)$ удаляется элемент $r(t)$; $t := t - 1;$

СТОП_А(): $STOP\_A := 1;$

МЕТКА_ОР_А(): $KOBR\_A := KOBR\_A + 1;$

ОТСТУП_А(): $i := i + 1;$

ВПЕРЕД_ОР_А(): $KOBR\_A := KOBR\_A - 1; UDOBR := (KOBR\_A = 0);$

$$E_H := E_H \cup \{(r(t), r(t-i))\};$$

ОР_РАСПОЗНАНЫ_А(): $i := 0;$

МЕТИМ_АВ(): $F := F + 1;\ E := 1;$

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

РАСП_АВ(): $E_H := E_H \cup \{(N\_B, r(t-i))\}; K := K - 1;$

$UDP\_B := (((K = Z)\ or\ (K = 1))\ and\ (Z \neq 1));$

ОБН_А(): $AN := 0;\ i := 0;$

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

Процедура *ОБР_СП_В*() такая же, как процедура *ОБР_СП_А*(), за исключением дополнительной проверки условия и вызова подпрограммы: «*if Mes* = ”ВОЗВРАТ_В” *then ВОЗВРАТ_В*();», которая касается только агента *B* и относится к режиму одновременного попадания двух АИ в одну белую вершину.

ВОЗВРАТ_В(): $E_H := E_H \setminus \{(y(p-1), y(p))\}; V_H := V_H \setminus \{Ct\_B\}; Ct\_B := Ct\_B - 2;$

$p := p - 1; y(p) := Ct\_B; L := 1; K := K + 1;$

Свойства алгоритма распознавания

При условии, что $n \geq 3$, как минимум по одному разу выполняются процедуры *ВПЕРЕД_А*() и *ВПЕРЕД_В*(). В каждой из процедур создается по одной вершине графа *H*. При одновременном попадании двух АИ в одну белую вершину этими процедурами будут созданы две новые вершины графа *H*. Вершина, созданная агентом *B*, на следующем шаге будет удалена процедурой *ВОЗВРАТ_В*(), так как она дублирует вершину, созданную агентом *A*. Таким образом, процесс выполнения описанного алгоритма индуцирует отображение $\phi: V_G \to V_H$ вершин графа *G* в вершины графа *H*. Причем $\phi(v) = t$ (когда вершина *v* окрашена в красный цвет и $t = Ct\_A$) и $\phi(s) = p$ (когда вершина *s* окрашена в желтый цвет и $p = Ct\_B$). Указанное отображение естественным образом устанавливает неявную нумерацию вершин графа *G*. Более того, отображение ϕ является биекцией, поскольку в связном графе *G* все вершины достижимы из начальных вершин. А это означает, что все вершины посещаются агентами, то есть окрашиваются в красный и желтый цвета. При выполнении процедуры *ВПЕРЕД_А*() или *ВПЕРЕД_В*() АЭ распознает древесное ребро (v,u) и так нумерует вершину *u*, что ребру (v,u) однозначно соответствует ребро $(\phi(v),\phi(u))$ графа *H*. При выполнении процедур *ВПЕРЕД_ОР_А*() или *ВПЕРЕД_ОР_В*() АЭ распознает обратное ребро (v,u) графа *G* и ставит ему в однозначное соответствие ребро $(\phi(v),\phi(u))$ графа *H*. При выполнении процедур *РАСП_АВ*() или *РАСП_ВА*() АЭ распознает перешеек (v,u) графа *G* и ставит ему в однозначное соответствие ребро $(\phi(v),\phi(u))$ графа *H*. Следовательно, ϕ является изоморфизмом графа *G* на граф *H*.

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

Подсчитаем временную, емкостную и коммуникационные сложности алгоритма в равномерной шкале. При подсчете временной сложности алгоритма будем считать, что инициализация алгоритма, анализ окрестности 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 АЭ, полученных на данном этапе от АИ, осуществляются за единицу времени.

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 $\phi(v) = 1$ ($\phi(s) = 2$) с вершиной *u*(*z*) с номером $\phi(u) = Ct\_A$ ($\phi(z) = Ct\_B$). Следовательно, общая длина красного и желтого пути не превосходит *n*.

Каждый раз при выполнении процедур обычного режима работы каждый АИ проходит одно ребро. Это означает, что процедуры обычного режима работы выполняются не более чем $2 \times (n-1)$ раз (учитывая, что АИ в этом режиме пройдет по каждому ребру своего пути два раза), а значит, общее время их выполнения оценивается как *O*(*n*).

При однократном выполнении процедуры режима распознавания обратных ребер АИ метят не более $n-2$ (так как граф простой и как минимум одна вершина окрашена в «чужой» цвет) обратных ребер, по одному разу проходит не более $n-2$ ребер красного (желтого) пути, а также по два раза проходится не более $n-2$ обратных ребер. Время, затрачиваемое на работу режима распознавания обратных ребер, оценивается как $4 \times n \times O(n)$, то есть как $O(n^2)$.

При выполнении процедур режима пометки перешейков АИ не совершают переходов по перешейкам, а просто окрашивают их ближние инциденторы, после чего отправляют сообщение АЭ о завершении режима пометки перешейков, на что также уходит один ход. То есть на работу в этом режиме уходит время, оцениваемое как $n \times O(n) + O(n)$, то есть как $O(n^2)$.

При однократном выполнении процедуры режима распознавания перешейков АИ проходят не более $n-2$ ребер красного (желтого) пути, после чего, помечая перешейки как распознанные, АИ не совершают перехода по перешейку, а просто окрашивают его ближний инцидентор. Завершив покраску всех помеченных перешейков, АИ уведомляет об этом АЭ, на что уходит один ход. Возвращаясь после распознавания всех перешейков, АИ может пройти максимум по одному разу два перешейка, а также не более чем $n-2$ ребер красного (желтого) пути. Время выполнения этого режима оценивается как $O\left(n^2\right)+O\left(n^2\right)+O(n)+O(n)+O\left(n^2\right)$, то есть как $O\left(n^2\right)$.

Время простоя агентов в ожидании в общей сложности оценивается как $O\left(n^2\right)$. Следовательно, суммарная временная сложность алгоритма удовлетворяет соотношению $T(n)=O\left(n^2\right)$. Емкостная сложность алгоритма определяется сложностью списков V_H, E_H, $r(1)\ldots r(t)$, $y(1)\ldots y(p)$, сложность которых определяется величинами $O(n)$, $O\left(n^2\right)$, $O(n)$, $O(n)$ соответственно. Следовательно, $S(n)=O\left(n^2\right)$.

На каждом шаге алгоритма АИ могут отправить АЭ не более четырех сообщений (в том числе и запросы значений, необходимых для выбора режима работы, переменных) порознь или одновременно и получить от него не более трех сообщений каждый, поэтому объем передаваемой агентами информации оценивается как $14\times O\left(n^2\right)$. Следовательно, $K(n)=O\left(n^2\right)$.

Учитывая описанные выше предположения о способе подсчета временной сложности, имеем следующую теорему.

Теорема 2. Временная, емкостная и коммуникационная сложности алгоритма равны $O\left(n^2\right)$, где n — число вершин графа, при этом в алгоритме используется 3 краски.

Заключение

В работе предложены новый метод и соответствующий алгоритм распознавания графа коллективом агентов, который состоит из двух агентов-исследователей, блуждающих по неизвестному графу, а также одного неподвижного агента-исследователя, распознающего граф по результатам работы двух блуждающих агентов.

Основным преимуществом построенного алгоритма является то, что удалось оптимизировать работу агентов и процесс обмена информацией между агентами таким образом, что стало возможным использование агентов-исследователей с конечной памятью, которая не зависит от размера графа. Это делает возможным использование одних и тех же агентов для работы в средах, о размерах которых нет никакой априорной информации. В большинстве же известных работ блуждающие агенты обладают памятью, которая позволит сохранить полную карту графа. То есть объем памяти агентов напрямую зависит от размерности исследуемого графа. А это значит, что при подборе блуждающих агентов для распознавания графов разной размерности необходимо это учитывать. Конечной, неограниченно растущей внутренней памятью в нашем алгоритме обладает только неподвижный агент-экспериментатор, в памяти которого и совершается построение карты графа.

Также нами были исследованы временная, емкостная и коммуникационная сложности полученного алгоритма, которые оцениваются сверху как $O\left(n^2\right)$ и не превышают (а в большинстве случаев — улучшают) сложности аналогичных алгоритмов.

На реализацию процесса распознавания графа в нашем алгоритме использовано всего 3 краски. Количество красок также не зависит от параметров графа и является постоянной величиной.

Список литературы (References)

Грунский И. С., Стёпкин А. В. Распознавание конечного графа коллективом агентов // Труды ИПММ НАН Украины. — 2009. — Т. 19. — С. 43–52.
Grunsky I. S., Stepkin A. V. Raspoznavanie konechnogo grafa kollektivom agentov [Finite graph exploration by collective of agents] // Proceedings of IPMM NAS of Ukraine. — 2009. — Vol. 19. — P. 43–52 (in Russian).

Грунский И. С., Тихончев М. Ю. О распознавании графов конечным автоматом // Вестник Донецкого ун-та. Сер. А. Естественные науки. — 2001. — № 2. — С. 351–356.
Grunsky I. S., Tikhonchev M. Yu. O raspoznavanii grafov konechnym avtomatom [About graph exploration by a finite automata] // Bulletin of Donetsk National University. Ser. A. Natural Sciences. — 2001. — No. 2. — P. 351–356 (in Russian).

Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. — М.: МЦНМО, 2001.
Cormen T., Leiserson Ch., Rivest R. Introduction to Algorithms. — MIT Press and McGraw-Hill, 2001. (Russ. ed.: *Kormen T., Leizerson Ch., Rivest R.* Algoritmy: postroenie i analiz. — Moscow: MCNMO, 2001.)

Касьянов В. Н., Евстигнеев В. А. Графы в программировании: обработка, визуализация и применение. — СПб.: БХВ–Петербург, 2003.
Kas'yanov V. N., Evstigneev V. A. Grafy v programmirovanii: obrabotka, vizualizatciya i primenenie [Graphs in programming: processing, visualization and application]. — Saint Petersburg: BXV–Peterburg, 2003 (in Russian).

Сапунов С. В., Сенченко А. С. Лингвистическое представление графов с помеченными вершинами // Доповіді Національної академії наук України. — 2019. — № 11. — С. 17–24.
Sapunov S. V., Senchenko A. S. Lingvisticheskoe predstavlenie grafov s pomechennymi vershinami [Linguistic representation of vertex-labeled graphs] // Reports of the National Academy of Sciences of Ukraine. — 2003 (in Russian). — DOI: https://doi.org/10.15407/dopovidi2019.11.017

Стёпкин А. В. Использование коллектива агентов для распознавания графов // Компьютерные исследования и моделирование. — 2013. — Т. 5, № 4. — С. 525–532.
Stepkin A. V. Ispol'zovanie kollektiva agentov dlya raspoznavaniia grafov [Using collective of agents for exploration of graph] // Computer Research and Modeling. — 2013. — Vol. 5, No. 4. — P. 525–532 (in Russian).

Стёпкин А. В. Использование коллектива агентов для распознавания неориентированных графов // Кібернетика і системний аналіз. — 2015. — Т. 51, № 2. — С. 75–88.
Stepkin A. Using a Collective of Agents for Exploration of Undirected Graphs // Cybernetics and Systems Analysis. — 2015. — Vol. 51, No. 2. — P. 223–233. (Original Russian paper: *Stepkin A. V.* Ispol'zovanie kollektiva agentov dlya raspoznavaniya neorientirovannyx grafov // Kibernetika i sistemnyi analiz. — 2015. — Vol. 51, No. 2. — P. 75–88.) — https://doi.org/10.1007/s10559-015-9715-z

Стёпкин А. В. Распознавание конечных графов тремя агентами // Искусственный интеллект. — 2011. — № 2. — С. 84–93.
Stepkin A. V. Raspoznavanie konechnyx grafov tremya agentami [Finite graphs exploration by three agents] // Artificial Intelligence. — 2013. — Vol. 5, No. 4. — P. 525–532 (in Russian).

Albers S., Henzinger M. R. Exploring unknown environments // SIAM Journal on Computing. — 2000. — No. 29 (4). — P. 1164–1188.

Banfi J., Quattrini Li A., Rekleitis I. et al. Strategies for coordinated multirobot exploration with recurrent connectivity constraints // Autonomous Robots. — 2018. — Vol. 42. — P. 875–894. — https://doi.org/10.1007/s10514-017-9652-y

Das S., Flocchini P., Kutten S., Nayak A., Santoro N. Map construction of unknown graphs by multiple agents // Theoretical Computer Science. — 2007. — Vol. 385, No. 1–3. — P. 34–48.

Dopp K. Automaten in labirinthen // Electronische Informationsverarbeitung und Kybernetik. — 1971. — Vol. 7, No. 2. — P. 79–94.

Dudek G., Jenkin M. Computational principles of mobile robotics // Cambridge Univ. press. — 2000. — 280 p.

Dudek G., Jenkin M., Milios E., Wilkes D. Map validation in a graphlike world // Proceedings of the 13th International Joint Conference on Artifical Intelligence (Chambery, France, August 1993). — San Fransisco: Morgan Kaufmann Publishers Inc., 1993. — P. 1648–1653.

Dudek G., Jenkin M., Milios E., Wilkes D. A taxonomy for swarm robots // In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 1993. — Yokohama, Japan. — P. 441–447.

Dudek G., Jenkin M., Milios E., Wilkes D. Topological exploration with multiple robots // Proc. 7th International Symposium on Robotics with Application (ISORA). — 1998. — Anchorage, Alaska, USA.

Goth G. Exploring new frontiers // Communications of the ACM. — 2009. — Vol. 52 (11). — P. 21–23.

Kilibarda G., Kudryavtsev V. B., Ushchumlich Sh. Collectives of automata in labyrints // Discrete Math. and Appl. — Vol. 13, Iss. 5. — P. 429–466. — https://doi.org/10.1515/156939203322694736

Nagavarapu S. C., Vachhani L., Sinha A. et al. Generalizing Multi-agent Graph Exploration Techniques // International Journal of Control, Automation and Systems. — 2020. — https://doi.org/10.1007/s12555-019-0067-8

Nagavarapu S. C., Vachhani L., Sinha A. Multi-Robot Graph Exploration and Map Building with Collision Avoidance: A Decentralized Approach // Journal of Intelligent & Robotic Systems. — 2016. — Vol. 83. — P. 503–523. — https://doi.org/10.1007/s10846-015-0309-9

Shannon C. E. Presentation of a maze-solving machine // Cybernetics Trans, of the 8th Conf. of the JosiahMacy Jr. Found / ed. H. Foerster. — 1951. — P. 173–180.

Wang H., Jenkin M., Dymond P. Enhancing exploration in graph-like worlds // In Proceedings of the Canadian Conference on Computer and Robot Vision (CRV). — 2008. — P. 53–60.

Wang H., Jenkin M., Dymond P. It can be beneficial to be ‘lazy’ when exploring graph-like worlds with multiple robots // Proceedings of the IASTED International Conference on Advances in Computer Science and Engineering (ACSE). — 2009. — P. 55–60.

Zhang C. Parallelizing Depth-First Search for Robotic Graph Exploration. — Cambridge, Massachusetts: Harvard College, 2010.