ПРОГРАММИРОВАНИЕ, 2015, № 6, с. 3–8

ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ ПРОГРАММИРОВАНИЯ

УДК 681.3

ИССЛЕДОВАНИЕ ГРАФА НАБОРОМ АВТОМАТОВ

© 2015 г. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин

Институт системного программирования РАН

109004 г. Москва,ул. А. Солженицына, 25

E-mail: igor@ispras.ru, kos@ispras.ru, kuliamin@ispras.ru

Поступила в редакцию 12.05.2015

В данной статье описан алгоритм обхода (извлечения полной информации о структуре) заранее неизвестного ориентированного графа при помощи неограниченного набора конечных автоматов, взаимодействующих при помощи обмена сообщениями и способных перемещаться вдоль дуг графа в соответствии с их ориентацией. В предположении об ограниченности времени работы базовых операций и времени пересылок отдельных сообщений общее время работы алгоритма в худшем случае ограничено $O(m+nd)$, где n — число вершин графа, m — число его дуг, d — диаметр графа, причем эту оценку нельзя улучшить. Полные доказательства приводимых в статье утверждений опубликованы в [6].

1. ВВЕДЕНИЕ

Задача извлечения структуры неизвестного графа за счет последовательных проходов по его ребрам, начиная от некоторой вершины, возникает в различных областях. Примерами являются анализ структуры компьютерных сетей, интерфейса Web-приложений, потоков управления и данных в коде, а также построение иных автоматных моделей систем. В данной работе рассматривается такая задача для случая ориентированного графа, исследуемого набором агентов, каждый из которых не может вместить полную информацию о графе, но все вместе они способны это сделать.

Каждый агент может быть представлен как конечный автомат с ограниченной памятью и конечным набором возможных действий. Эти действия включают извлечение идентификатора текущей вершины, итерацию дуг графа, выходящих из текущей вершины, продвижение по очередной дуге графа, а также обмен сообщениями с другими агентами, например, для выяснения того, был ли кто-нибудь из них в текущей вершине данного агента, и использовалась ли выходящая из нее дуга. Поскольку граф заранее не ограничен, есть возможность создания новых агентов, размещающихся в некоторой начальной вершине графа. Граф считается исследованным, если по информации, имеющейся у агентов, можно установить, что каждая его дуга была пройдена хотя бы один раз, и, таким образом, все его вершины и дуги известны текущему набору агентов. При этом не предполагается, что информация о структуре графа представлена явно, достаточно возможности для ее дальнейшего извлечения уже без выполнения проходов по дугам.

Подобная постановка задачи имеет практические приложения в тестировании сложных систем, моделируемых автоматами с большим количеством состояний. В работе [1] рассматривается тестирование моделей микропроцессоров, в которых возникают миллионы состояний и десятки миллионов переходов. В [1, 2] тестирование выполняется ограниченным набором агентов, каждый из которых имеет доступ к полной информации о структуре уже исследованной части графа. В данной работе предлагается алгоритм исследования графа набором агентов, где каждый агент работает уже только с локально доступной ему информацией. Ранее в работах [3–5] были предложены алгоритмы для исследования заранее неизвестного графа одним агентом, обладающим как неограниченной, так и ограниченной памятью. Там же доказано, что

для графов с n вершинами и m дугами оценка времени работы такого агента в худшем случае $O(nm)$. В данной работе показывается, что при использовании многих агентов эту оценку можно снизить до $O(m + nd)$, где d — диаметр графа.

Дальнейшая структура статьи такова: во втором разделе дается формальная постановка задачи исследования графа набором агентов, далее приводится предлагаемый алгоритм работы такого набора агентов и доказывается оценка времени его работы в худшем случае, в заключении суммируется содержание работы и перечисляются возможные направления развития предложенного алгоритма.

2. ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ

Далее мы считаем, что среда выполнения агентов поддерживает операции создания и уничтожения агента. Каждый агент в любой момент времени находится в некоторой вершине графа (несколько агентов могут находится одновременно в одной вершине) и имеет уникальный идентификатор (идентификатор агента возвращается операцией, создающей его). Есть две возможности создания агента: в одном случае после создания агент находится в выделенной начальной вершине; в другом агент создается другим агентом как находящийся в той же вершине, где был этот второй, после чего агент-создатель теряет возможность взаимодействовать с графом. Вершины графа также имеют уникальные идентификаторы. Агент может получить команду на проход по определенной дуге, выходящей из вершины, в которой он находится. Дуга при этом задается своим номером — все дуги, выходящие из одной вершины, пронумерованы. Операция прохода по дуге возвращает идентификатор вершины, в которой агент оказался, и количество выходящих из нее дуг. Кроме того, агенты могут обмениваться сообщениями — агент может послать сообщение либо всем агентам, либо какому-то конкретному, указав его идентификатор, а также принять сообщение от любого отправителя. Сообщения при передаче не теряются и не искажаются, не обгоняют друг друга при передаче между двумя агентами. Помимо информации о типе сообщения (есть конечное число типов сообщений), они также могут нести конечное число слотов, хранящих идентификаторы состояний или агентов.

Чтобы уметь работать с графами произвольного размера, мы не накладываем ограничений на количество используемых агентов. Для возможности оперировать с неограниченными идентификаторами агентов или состояний мы считаем, что у агента есть конечное число ячеек памяти, способных хранить идентификатор состояния или агента. Над идентификаторами производятся только операции копирования между этими ячейками, между ячейкой памяти и слотом сообщения, также возможно сопоставление идентификаторов состояний на равенство или неравенство. Таким образом, агенты являются расширенными конечными автоматами, оперирующими с переменными-идентификаторами. Другое возможное препятствие для конечности агентов — необходимость оперировать номерами неограниченного числа дуг, выходящих из одной вершины. Чтобы избежать этого, произвольный граф перестраивается в такой, в котором число выходящих из вершин дуг ограничено некоторой константой R, для этого достаточно каждую вершину разбить на несколько вершин. Из каждой новообразованной вершины будут выходить не более $R - 1$ дуг, плюс еще одна дуга, соединяющая каждую вершину из полученной связки со следующей. Все вершины в этой связке, кроме первой, могут иметь пустой идентификатор, поскольку в них входит только одна дуга, и искать их среди известных вершин после прохода по этой дуге не нужно, поэтому расширения множества идентификаторов можно избежать.

Итак, исходно задан граф неизвестной структуры с фиксированной начальной вершиной, имеется возможность выполнять описанные выше операции. Требуется задать программу действий агентов и среды таким образом, чтобы в итоге множеством агентов была получена полная информация о структуре графа.

Используемые далее обзначения таковы: n — число вершин графа, m — количество дуг в нем, d — длина самого длинного пути от начальной до другой вершины (диаметр графа).

3. ПРЕДЛАГАЕМЫЙ АЛГОРИТМ

Предлагается следующий алгоритм работы агентов для решения поставленной задачи.

- Каждый агент может работать в двух режимах — менеджера вершины и ползунка. Ползунок создается в некоторой вершине, проходит определенный путь и в итоге либо становится менеджером вершины, либо уничтожается. Менеджер вершины отвечает за хранение данных о выходящих из вершины дугах, в том числе еще не пройденных, и определяет, куда будет двигаться очередной ползунок, пришедший в данную вершину. Если число выходящих дуг из вершины графа не ограничено, создается несколько менеджеров вершины, каждый из которых хранит информацию лишь об ограниченном числе выходящих дуг и адрес следующего. Менеджером, ассоциированным с идентификатором вершины, является лишь первый из них.

- Для каждой выходящей дуги менеджер вершины хранит ее статус и идентификатор менеджера концевой вершины, если он известен. Статус дуги может быть активным, пассивным или завершенным. Активной считается еще не пройденная дуга, или дуга, по которой нужно отправить ползунок, потому что от менеджера ее концевой вершины пришел запрос на такое отправление. Пассивной считается дуга, по которой пока ползунки отправлять не нужно (в ответ на запрос на ползунок, один уже был отправлен). Завершенной считается дуга, по которой уже не нужно отправлять ползунки, потому что от менеджера ее концевой вершины пришло сообщение, что все ведущие из нее пути приводят в уже пройденные вершины. Сразу после создания менеджера все выходящие дуги активны.

  Менеджер вершины с непустым идентификатором также хранит ссылку на следующий такой же менеджер. Такие ссылки используются для обработки запросов поиска менеджера вершины по идентификатору вершины путем прохода по образованному этими ссылками списку.

  Менеджер вершины хранит номер используемой дуги, по которой ползунок был отправлен в последний раз. Сразу после создания менеджера этот номер равен 0, что значит, что такой дуги нет.

  Кроме того, менеджер вершины хранит номер дуги, входящей в его вершину, при проходе по которой эта вершина была обнаружена первый раз. Такие дуги вместе с известными на данный момент вершинами образуют дерево — остов уже пройденной всеми агентами части графа.

- Набор сообщений, которым обмениваются агенты, и операций, которые они могут выполнять, таков.

  - Создание агента в начальной вершине. Он создается как ползунок.
  - Создание ползунка другим агентом в той вершине, в которой находится этот другой. Второй агент при этом теряет возможность взаимодействовать с графом. Несмотря на кажущуюся искусственность, такая операция легко реализуется, если агенты представляют собой процессы — она соответствует созданию клона процесса в текущем состоянии и передаче ему прав на работу с графом.
  - Выполнение перехода по дуге с номером i в текущей вершине графа, в результате которого ползунок оказывается в конечной вершине данной дуги и получает ее идентификатор и количество дуг, выходящих из нее.
  - Запрос номера дуги для дальнейшего движения от ползунка менеджеру текущей вершины, содержащий идентификатор отправителя. Ответ на него содержит номер дуги для выполнения перехода и идентификатор менеджера конечной вершины дуги (возможно, пустой, если у конечной вершины нет менеджера).
  - Сообщение о поиске менеджера вершины с заданным идентификатором вершины содержит этот идентификатор и идентификатор пославшего его агента. В ответ на него может прийти сообщение об имеющимся менеджере с идентификатором этого менеджера.

- Запрос нового ползунка для дальнейшего прохода по графу. Такой запрос содержит номер дуги, по которой нужно послать ползунок и идентификатор менеджера ее концевой вершины.
- Сообщение о завершенности дуги, обозначающее, что ходить по ней дальше не имеет смысла, потому что либо она заканчивается в вершине без выходящих дуг, либо все выходящие дуги приводят в уже известные вершины, проход в которые может быть организован другими путями.

1. В начале работы первый созданный агент становится менеджером начальной вершины. При этом ему становится известно количество выходящих из нее дуг. Как менеджер вершины, только что оказавшийся в ней, он создает ползунок, находящийся в той же вершине, и делегирует ему возможность прохода по дугам графа, а также сообщает свой идентификатор для обмена сообщениями.

2. Только что созданный или только что узнавший идентификатор менеджера текущей вершины ползунок посылает этому менеджеру запрос номера дуги, по которой двигаться дальше.

3. Получив запрос номера дуги для движения, менеджер вершины ищет активную дугу, следующую за используемой. Это либо активная дуга с минимальным номером, превосходящим номер используемой дуги, либо, если таковой не окажется, просто активная дуга с минимальным номером. Ее номер записывается как номер используемой дуги. Если активных дуг в данный момент нет вообще, то номер используемой дуги становится равным 0.

   Если полученный номер используемой дуги не равен 0, то пославший запрос ползунок получает в ответ этот номер и идентификатор конца используемой дуги, если он известен. Используемая дуга при этом переводится в статус пассивной. Кроме того, менеджер начальной вершины после этого создает новый ползунок, а менеджер другой вершины посылает менеджеру начальной вершины входящей дуги запрос на новый ползунок.

   Если номер используемой дуги равен 0, менеджер вершины запоминает приславший запрос ползунок как ждущий.

4. Ползунок, получивший номер дуги для прохода по ней, запоминает идентификатор менеджера текущей вершины и выполняет проход по указанной дуге. В результате он получает идентификатор концевой вершины и число выходящих из нее дуг.

   Если в сообщении с номером дуги он получил также и непустой идентификатор менеджера концевой вершины, значит он проходит по уже известной дуге. Далее он выполняет п. 2.

   Если идентификатор менеджера концевой вершины пуст, то его нужно определить.

   В том случае, когда у вершины нет выходящих дуг, вершина терминальная и не должна иметь менеджера. Ползунок в этом случае посылает менеджеру начальной вершины дуги, по которой он только что прошел, сообщение о завершенности дуги (больше по ней нет смысла ходить) и уничтожается.

   Если у вершины есть выходящие дуги, но ее идентификатор пуст — это вершина с единственной входящей дугой. Ползунок при этом становится менеджером этой вершины, инициализирует данные менеджера и создает новый ползунок в текущей вершине, делегируя ему взаимодействие с графом.

   Если у вершины есть выходящие дуги и идентификатор не пуст, ползунок запускает поиск менеджера вершины, отправляя менеджеру начальной вершины сообщение о поиске менеджера с идентификатором вершины и своим идентификатором в качестве данных.

5. Получив сообщение о поиске вершины, менеджер сравнивает идентификатор вершины из сообщения с идентификатором своей вершины.

Если идентификаторы не совпадают, менеджер проверяет, есть ли следующий менеджер в списке для поиска. Если такого нет, значит искомого менеджера нет, и данный менеджер ставит ссылку на агент с идентификатором, находящимся в сообщении (пославший его ползунок), как на следующий и переправляет сообщение ему. Если сообщение о поиске вершины получает ползунок, значит менеджер этой вершины не найден, и он сам должен стать ее менеджером. При этом он инициализирует данные менеджера и создает новый ползунок в данной вершине, делегируя ему взаимодействие с графом.

Если идентификаторы не совпадают, и есть следующий менеджер в списке для поиска, сообщение о поиске переправляется ему.

Если идентификаторы совпадают, то агенту с идентификатором, находящимся в сообщении, посылается сообщение о найденном менеджере с идентификатором данного менеджера. Если ползунок получает сообщение о найденном менеджере вершины с его идентификатором, он выполняет п. 2.

6. Менеджер вершины может получить запрос на новый ползунок с номером одной из выходящих из его вершины дуг. При этом эта дуга объявляется активной.

   Если при этом у него есть ждущий ползунок, этому ползунку отправляется сообщение с указанием идти по этой дуге (с ее номером и идентификатором менеджера конечной вершины). Дуга остается пассивной, ссылка на ждущий ползунок обнуляется. Если данный менеджер связан с начальной вершиной, он создает новый ползунок, если нет — менеджеру начальной вершины, входящей в вершину данного менеджера дуги, посылается запрос на новый ползунок с указанием номера входящей дуги. Если ждущего ползунка нет, больше ничего не делается.

7. Менеджер вершины может получить сообщение о завершенности одной из выходящих из этой вершины дуг от менеджера ее концевой вершины. В этом случае дуга помечается как завершенная.

   Если при этом у него есть ждущий ползунок и незавершенных дуг больше нет, этот ползунок уничтожается. Если данный менеджер связан с начальной вершиной, то работа всего алгоритма завершается — граф полностью обойден, если нет — менеджеру начальной вершины, входящей в вершину данного менеджера дуги, посылается сообщение о завершенности входящей дуги.

   Если ждущего ползунка нет или есть незавершенные выходящие дуги, больше ничего не делается.

В работе [6] доказано, что, если считать время выполнения всех перечисленных выше операций агентов, включая выполнение прохода по дуге, а также время пересылки сообщений между агентами, ограниченным некоторой константой, то время работы описанного алгоритма в худшем случае ограничено $O(m + nd)$. Там же приведен пример графа, на котором эта оценка достигается, т.е., описанный алгоритм затрачивает на обход графа время $\Omega(m + nd)$.

4. ЗАКЛЮЧЕНИЕ

В статье представлен алгоритм обхода (или извлечения полной информации о структуре) заранее неизвестного графа при помощи неограниченного набора параллельно действующих агентов, являющихся конечными автоматами и взаимодействующих друг с другом с помощью посылки сообщений. Время работы предложенного алгоритма в худшем случае в предположении об ограниченности времени работы некоторых базовых операций и времени пересылок отдельных сообщений между агентами ограничено $O(m+nd)$, где n — число вершин графа, m — число его дуг, d — диаметр графа. Найдено семейство графов, на котором эта оценка достигается. В данной работе для облегчения понимания алгоритм представлен не совсем в формальном виде, доказательство приводимых оценок эффективности опущено. Формальное изложение с полными доказательствами читатель может найти в [6].

Дальнейшее развитие представленного результата — разработка варианта алгоритма, адаптированного для выполнения в многопроцессорной

системе, когда на каждом процессоре (или ядре) работает один (или несколько, но небольшое число) агентов-ползунков и один процесс-менеджер, выполняющие функцию хранения информации о некоторой части графа. При этом в рамках процесса-менеджера стоит хранить информацию не об одной вершине, а о нескольких, иначе затраты на поддержание работы многих менеджеров существенно снизят общую эффективность. Кроме того, обмен сообщениями между процессами в этом случае стоит оптимизировать так, чтобы он происходил большей частью внутри одной машины. Детали подобного алгоритма пока не проработаны.

Другие возможные направления развития алгоритма — его модификация для возможности работы с недетерминированными графами и с внешней памятью. В недетерминированных графах одному номеру выходящей из вершины дуги может соответствовать несколько реальных дуг, и выполнение прохода по такому номеру может приводить в разное время в различные концевые вершины. Такие графы часто встречаются при моделировании сложных программных или аппаратных систем на высоком уровне абстракции. Использование внешней памяти может снять требование, чтобы вся информация о графе могла поместиться в память всего набора используемых агентов.

СПИСОК ЛИТЕРАТУРЫ

1. *Demakov A., Kamkin A., Sortov A.* High-Performance Testing: Parallelizing Functional Tests for Computer Systems Using Distributed Graph Exploration. Open Cirrus Summit 2011, Moscow.
2. *Бурдонов И.Б., Грошев С.Г., Демаков А.В., Камкин А.С., Косачев А.С., Сортов А.А.* Параллельное тестирование больших автоматных моделей // Вестник ННГУ. 2011. № 3. Р. 187–193.
3. *Бурдонов И.Б., Косачев А.С., Кулямин В.В.* Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай // Программирование. 2003. № 5. С. 59–69.
4. *Бурдонов И.Б., Косачев А.С., Кулямин В.В.* Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированный случай // Программирование. 2004. № 1. С. 2–17.
5. *Бурдонов И.Б.* Обход неизвестного ориентированного графа конечным роботом // Программирование. 2004. № 4. С. 11–34.
6. *Бурдонов И.Б., Косачев А.С.* Обход неизвестного графа коллективом автоматов // Труды ИСП РАН. 2014. Т. 26. Вып. 2. С. 43–86.