Мониторинг динамически меняющегося графа

Игорь Бурдонов <igor@ispras.ru>
Александр Косачев <kos@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Исследование ориентированных графов является корневой задачей во многих приложениях. Такое исследование имеет особую специфику тогда, когда граф моделирует сеть связи, в том числе сеть интернета и GRID. Узел сети имеет локальную информацию о сети: он «знает» только о дугах, выходящих из этой вершины, но «не знает», куда (в какие вершины) эти дуги ведут. Узлы сети обмениваются сообщениями, передаваемыми по сетевым связям, которые в графе изображаются как дуги и играют роль каналов передачи сообщений. Исследование графа базируется на его обходе, когда сообщение проходит по каждой дуге графа. Пока не пройдена какая-то дуга, нет уверенности, что она не ведёт в ещё не исследованную часть графа. Обычно рассматривается обход графа с помощью одного сообщения, циркулирующего в сети. Обход выполняется быстрее, если выполнять его параллельно: по сети одновременно циркулирует не одно, а множество сообщений. В данной работе рассматривается параллельное исследование сильно связного графа, целью которого является не просто обход графа, а сбор полной информации о графе в каждой его вершине. Вторая особенность работы – исследование динамически меняющегося графа: его дуги могут исчезать, появляться или менять свои конечные вершины. Предлагается алгоритм работы автоматов, который обеспечивает сбор полной информации о графе в каждой вершине графа.

Ключевые слова: ориентированные графы, исследование графа, обход графа, взаимодействующие автоматы, параллельная работа, динамически меняющиеся графы.

DOI: 10.15514/ISPRAS-2015-27(1)-5

Для цитирования: Бурдонов Игорь, Косачев Александр. Мониторинг динамически меняющегося графа. Труды ИСП РАН, том 27, вып. 1, 2015 г., стр. 69-96. DOI: 10.15514/ISPRAS-2015-27(1)-5.

1. Введение

Исследование ориентированных графов является корневой задачей во многих приложениях. Такое исследование имеет особую специфику тогда, когда граф

моделирует сеть связи, в том числе сеть интернета и GRID. В этом случае узлы сети обычно «не знают» всего графа и имеют только локальную информацию о нём. В частности, узел сети, соответствующий вершине графа, может «знать» только о дугах, выходящих из этой вершины, и даже «не знать», куда (в какие вершины) эти дуги ведут. Исследование графа можно свести к задаче сбора полной информации о графе в каких-то или во всех узлах сети. Для решения этой задачи узлы сети могут обмениваться сообщениями, передаваемыми по сетевым связям, которые в графе изображаются как дуги.

Исследование графа базируется на его обходе, когда сообщение проходит по каждой дуге графа. Пока не пройдена какая-то дуга, нет уверенности, что она не ведёт в ещё не исследованную часть графа. Обход графа – это уже классическая задача обхода лабиринта. Эта задача нетривиальна, если граф ориентирован, то есть в лабиринте «улицы с односторонним движением».

Обход ориентированного сильно-связного графа требует времени порядка nm, где n – число вершин графа, а m – число дуг. Такое время обхода достигается многими хорошо известными алгоритмами: обход в глубину, обход в ширину, «жадный» алгоритм и др. [1,2,3].

В 1966 г. М.О. Рабин поставил задачу обхода ориентированного графа конечным автоматом [4]. Автомат на графе аналогичен машине Тьюринга: ячейке ленты соответствует вершина графа, а движение влево или вправо по ленте заменяется переходом по одной из дуг, выходящих из текущей вершины графа. Это эквивалентно движению по графу одного сообщения, которое пересылается по дугам графа автоматами, неподвижно «сидящими» в вершинах графа. Дуги графа играют роль каналов передачи сообщений. Автоматы в вершинах графа идентичны друг другу, но могут находиться в разных состояниях. Автомат, находящийся в вершине, посылает сообщение по одной из дуг, выходящих из этой вершины, и через какое-то время такое сообщение принимается автоматом в конце дуги, который, в свою очередь, может модифицировать это сообщение и послать его дальше.

На сегодняшний день наиболее быстрый алгоритм обхода графа с помощью одного сообщения предложен в [5], он имеет оценку $nm+n2loglogn$. При повторном обходе, когда автоматы в вершинах находятся не в начальном состоянии, а в конечных состояниях после первого обхода, оценка уменьшается до $nm+n2l(n)$, где $l(n)$ — число логарифмирований, при котором достигается соотношение $1 \le log(log...(n)...) < 2$ [6]. Отличие от нижней оценки nm объясняется тем, что сообщению бывает нужно «вернуться» в начало только что пройденной дуги.

Обход графа можно выполнить быстрее, если по дугам графа может параллельно пересылаться не одно, а много сообщений. Оценка времени работы алгоритма зависит от числа сообщений, которые могут одновременно передаваться по дуге. Такое число называется *ёмкостью дуги* и обозначается

k. В [7] предложен алгоритм такого обхода, имеющий оценку порядка *n/k+D*, где *D* – длина максимального пути (маршрута без самопересечений) в графе. Правда, число состояний автоматов в вершинах графа зависит от числа *n* вершин графа и ограничения *s* на число дуг, выходящих из вершины, и имеет порядок *nDlogs*. Этот алгоритм строит на графе с выделенной начальной вершиной (корнем) структуру из прямого и обратного остовов графа: прямой остов ориентирован от корня, а обратный – к корню. В дальнейшем эта структура может использоваться для параллельного вычисления любой требуемой функции от мультимножества значений, записанных в вершинах графа (в памяти автоматов, находящихся в вершинах графа).

В данной работе мы также рассматриваем исследование графа с помощью многих параллельно передаваемых сообщений, но с двумя существенными отличиями от предыдущих работ по исследованию графа. Во-первых, целью такого исследования является не просто обход графа или построение какой-то структуры на графе, а сбор полной информации о графе в каждой его вершине. Во-вторых, мы рассматриваем граф, который может динамически изменяться: какие-то его дуги могут исчезать, появляться или менять свою конечную вершину.

2. Постановка задачи

Рассматривается ориентированный граф с *n* вершинами и *m* дугами, в котором вершины не меняются, а дуги могут меняться со временем. Будем предполагать, что в каждый момент времени граф сильно связный. Кроме того, будем считать, что если изменения в графе прекращаются, то с этого момента времени длина максимального пути в графе не превышает *D*.

В вершинах графа находятся автоматы, которые могут получать специальные сигналы от графа и обмениваться между собой сообщениями, передаваемыми по дугам графа в направлении их ориентации. Каждая вершина имеет уникальный (среди вершин) *идентификатор вершины*. Будем считать, что существует *пустой* идентификатор, отличный от идентификатора любой вершины. Автомат может узнать идентификатор своей вершины с помощью примитива «*Дай идентификатор*». Для краткости везде, где это не приведёт к недоразумениям, мы будем вместо «автомат вершины» писать просто «вершина».

Все дуги, выходящие из вершины, перенумерованы, начиная с номера 1 до наибольшего номера, не превосходящего некоторого числа *s*. Очевидно, $s \le m$. Дуга однозначно идентифицируется парой (идентификатор начала дуги, номер дуги), которую мы будем называть *идентификатор дуги*. Число дуг *m* – это число различных идентификаторов дуг в графе.

Автомат, посылая сообщение по дуге, указывает её номер. Будем говорить, что дуга занята, если по ней передаётся сообщение. Иначе дуга свободна. Посылать сообщение можно только по свободной дуге. В самом начале все

дуги свободны. Когда сообщение передано по дуге, начало дуги извещается об этом сигналом *освобождения* дуги с параметром номер дуги.

Дуги графа могут изменяться следующим образом:

- Дуга может появиться, о чём начало дуги извещается сигналом *появления* дуги с параметром номер дуги.
- Дуга может исчезнуть. Если по дуге передавалось сообщение, то оно пропадает, и в этом случае начало дуги извещается сигналом *исчезновения* дуги с параметром номер дуги. Заметим, что если по дуге никакого сообщения не передавалось, то сигнала исчезновения дуги не будет; однако если вершина попытается послать сообщение по исчезнувшей дуге, сообщение не будет передано, а вершина получит сигнал исчезновения дуги.
- Дуга может поменять свой конец; в этом случае никаких сигналов не предусмотрено.

Из-за того, что дуги могут исчезать и появляться, номера дуг, выходящих из одной вершины и имеющихся в данный момент времени, могут не идти подряд, т.е. не образовывать отрезок натурального ряда, но в любом случае они располагаются на отрезке от 1 до *s*. Хотя конец дуги может меняться, в данный момент времени у дуги может быть только один конец; если дуга исчезла, считается, что её конец «пустой», то есть имеет пустой идентификатор. В каждый момент времени дуга описывается парой (идентификатор дуги, идентификатор конца дуги) или, что то же самое, тройкой (идентификатор начала дуги, номер дуги, идентификатор конца дуги). *Временем существования* дуги будем называть время между изменениями дуги, то есть время от появления дуги или изменения её конца до исчезновения дуги или изменения её конца.

Итак, если не считать идентификатора вершины, который автомат получает с помощью примитива «Дай идентификатор», входными символами автомата, находящегося в вершине, являются сигналы от дуг, выходящих из вершины, и сообщения, получаемые по дугам, входящим в вершину. Срабатывание автомата – это обработка одного такого входного символа. Мы будем предполагать, что входные символы обрабатываются автоматом в порядке их возникновения. Это означает, что существует очередь на обработку автоматом входных символов. Если во время обработки автоматом очередного сигнала или сообщения поступает новый входной символ, он ставится в конец этой очереди. Одновременно поступающие входные символы ставятся в конец очереди в произвольном порядке. При этом сигнал освобождения дуги вырабатывается не тогда, когда сообщение ставится в очередь входных символов, а тогда, когда оно выбирается из этой очереди автоматом конца дуги. Поэтому в очереди входных символов может быть не более одного сообщения для каждой входящей дуги. Если сигнал от некоторой выходящей

дуги вырабатывается в тот момент времени, когда предыдущий сигнал от этой дуги ещё находится в очереди входных символов, то новый сигнал замещает собой старый сигнал. Можно отметить, что таким новым сигналом может быть только сигнал появления дуги. Поэтому в очереди входных символов может быть не более одного сигнала для каждой выходящей дуги. Тем самым, длина очереди входных символов автомата не превосходит степени вершины, то есть не больше $2m$ ($2m$ достигается для $n=1$, когда все дуги – петли, поскольку петля считается два раза: как входящая – для сообщений и как выходящая – для сигналов).

Мы ставим задачу разработки алгоритма работы автоматов, который обеспечивает сбор полной информации о графе в каждой вершине графа. Такая информация может быть задана как набор описаний всех дуг. Описание дуги содержит идентификатор дуги и идентификатор её конца. Понятно, что если граф постоянно меняется, то мы не можем гарантировать, что описания текущего состояния всех его дуг отражены во всех его вершинах: сообщения о последних изменениях дуг просто «не дошли» до некоторых вершин. Поэтому требуется только, чтобы через время T_0 после изменения дуги все вершины графа «узнали» об этом или более позднем изменении дуги. Если после данного изменения дуга больше не меняется, по крайней мере, в течение времени T_0, то во всех вершинах графа будет одинаковое (и верное) описание этой дуги.

Для того чтобы время T_0 было конечным, нужно, чтобы сообщения могли распространяться по графу, доходя до каждой вершины. Для этого нужно, чтобы время x пересылки сообщения по дуге и время y срабатывания автомата были конечными. Кроме того, время z существования дуги должно быть достаточно велико, чтобы по дуге успевало пройти хотя бы одно сообщение. Такими «долгоживущими» дугами могут быть не все дуги: достаточно, чтобы в каждый момент времени «долгоживущие» дуги порождали сильно связный суграф (подграф, содержащий все вершины графа). Это усиление требования сильно связности графа в каждый момент времени.

Для ограниченности времени T_0 необходимо, чтобы x и y были ограничены сверху: $x \le X$ и $y \le Y$, а z – снизу: $z \ge Z$. Выразим эту нижнюю границу Z через X и Y. Сообщение можно посылать по дуге сразу после того, как она появилась или освободилась. Однако в этот момент времени сигнал появления или освобождения дуги ещё только ставится во входную очередь символов автомата начала дуги, длина которой может достигать $2m$. Поэтому сообщение посылается по дуге не сразу, а через время, которое может достигать $2mY$. Следовательно, сообщение дойдёт до конца дуги и будет поставлено во входную очередь символов автомата конца дуги через время, которое может достигать $2mY+X$, если в течение этого времени дуга не менялась. Поэтому будем предполагать, что нижняя граница времени существования «долгоживущих» дуг $Z \ge 2mY+X$.

Мы предложим алгоритм решения поставленной задачи, а время T_0 будем оценивать для простоты в предположении, что временем срабатывания автомата можно пренебречь, то есть $Y=0$, время пересылки по дуге не превосходит 1 такта, то есть $X=1$, а время существования «долгоживущей» дуги может быть минимально, то есть $Z=2mY+X=1$. Кроме того, оценим время T_1, за которое будет выполнен сбор полной информации о графе в каждой вершине графа после того, как вообще прекратились все изменения в графе. Очевидно, что $T_1 \le T_0$, однако прекращение изменений в графе позволяет существенно уменьшить T_1 по сравнению с T_0.

3. Идея алгоритма

Идея алгоритма заключается в том, чтобы каждый раз, когда обнаруживается изменение дуги, корректировать описание дуги и рассылать сообщение с этим описанием «веером» по всем дугам графа для того, чтобы каждая вершина его получила. Мы будем говорить, что рассылается описание дуги. Первое описание дуги создаётся в начале дуги по сигналу появления дуги при первом появлении этой дуги. Дуга, которая имеется в графе с самого начала, считается появляющейся в начальный момент времени.

Как выполнить рассылку описания дуги p «веером»? Для этого вершина, принимая по некоторой входящей дуге q описание дуги p первый раз, рассылает его по всем выходящим дугам, а повторные принимаемые описания дуги p игнорирует. Однако дуга может меняться несколько раз, поэтому если вершина принимает описание более позднего состояния дуги, чем то, о котором она уже «знает», она не должна игнорировать принимаемое описание. Для этого состояния дуги, соответствующие её последовательным изменениям, перенумеровываются. Этот номер называется *рангом* дуги и хранится в описании дуги. Вершина игнорирует принимаемое описание дуги только в том случае, если в нём ранг дуги меньше или равен рангу дуги в её описании, хранящемся в вершине. Если же ранг дуги в принимаемом описании больше, то принимаемое описание дуги замещает собой описание дуги в вершине, и далее вершина должна разослать это обновлённое описание дуги по всем выходящим дугам.

Каким образом обнаруживается изменение дуги? Схема работы алгоритма показана на рис.1.

О появлении дуги извещается начало дуги с помощью сигнала появления дуги. В этот момент времени конец дуги ещё не известен, поэтому в описании дуги идентификатор конца дуги делается пустым. По дуге посылается её описание, причём указывается идентификатор дуги, по которой посылается описание. Когда это описание дойдёт до конца дуги, в описание дуги будет помещён идентификатор конца дуги и далее полученное описание дуги рассылается «веером». Когда это описание дойдёт до начала дуги, там тоже появится непустой идентификатор конца дуги.

Для того чтобы отслеживать изменение конца дуги, по дуге периодически посылается её описание. Если конец дуги не меняется, идентификатор конца дуги в посылаемом описании совпадает с идентификатором вершины, получившей описание, то есть с идентификатором конца дуги. В этом случае ничего делать не нужно. Если же конец дуги сменился, то идентификатор конца дуги в посылаемом описании будет отличаться от идентификатора вершины, получившей опрос, т.е. от идентификатора нового конца дуги. В этом случае новый конец дуги рассылает «веером» описание дуги со своим собственным идентификатором как идентификатором конца дуги. Когда такое описание получит начало дуги, оно поменяет идентификатор конца дуги в своём описании дуги.

Об исчезновении дуги извещается начало дуги с помощью сигнала исчезновения дуги. Это происходит тогда, когда по дуге передаётся сообщение. Начало дуги делает пустым идентификатор конца дуги в своём описании дуги и рассылает это описание «веером».

Для того, чтобы описанная схема алгоритма работала, нужна правильная процедура изменения ранга дуги. Смысл этой процедуры в том, чтобы в описании более нового состояния дуги ранг был больше, чем во всех других имеющихся описаниях дуги.

Пусть в начале дуги в описании этой дуги указан максимальный ранг *j*. Если идентификатор конца дуги в этом описании неправильный (пустой после появления дуги или старый после изменения конца дуги), то конец дуги, получив по дуге такое её описание и обновив его (добавив свой идентификатор как идентификатор конца дуги), должен увеличить на *1* ранг дуги. Далее обновлённое описание дуги с максимальным рангом *j+1* будет распространяться по графу «веером», в том числе и до начала дуги.

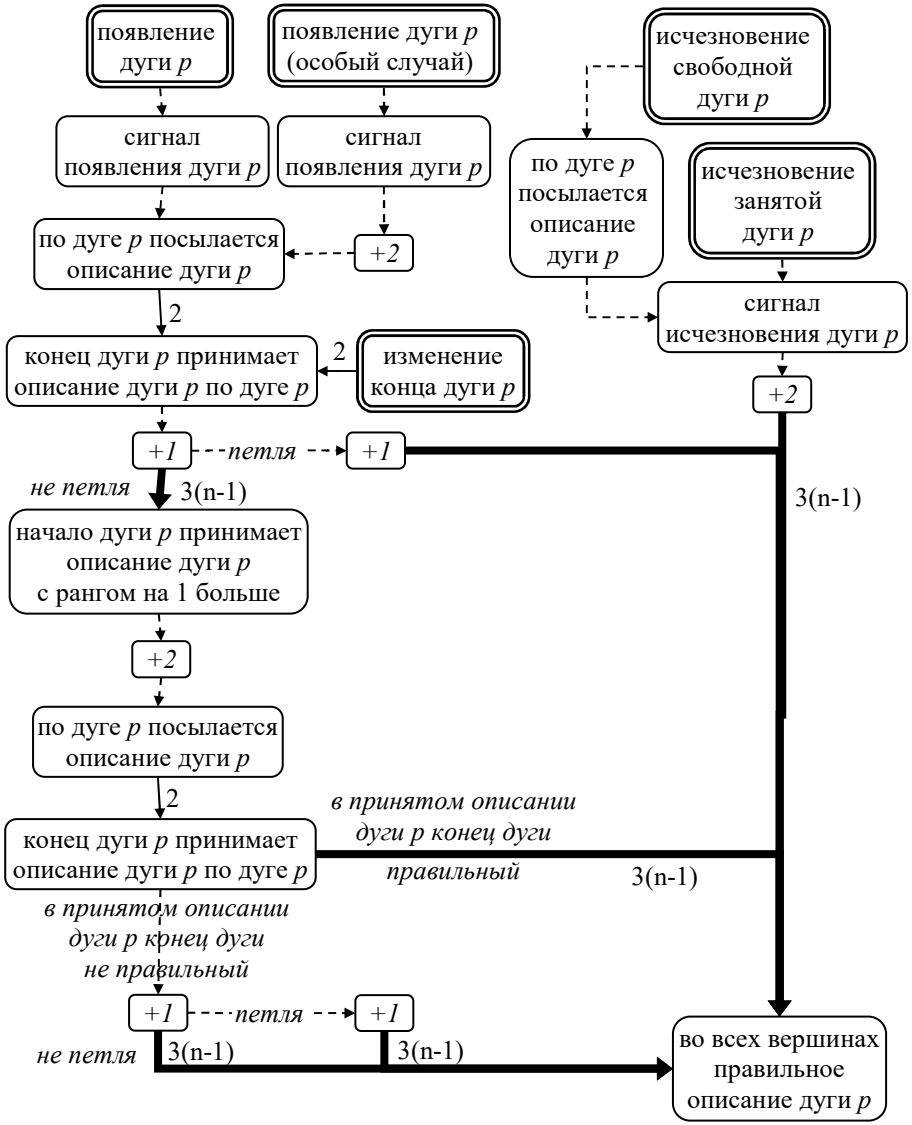
Однако здесь возникает проблема многократного изменения конца дуги. В этом случае уже не в одной, а в нескольких вершинах может оказаться максимальный ранг *j+1*. Поскольку только одна из этих вершин в данный момент времени является текущим концом дуги, нужно, чтобы по графу распространялось описание дуги именно с текущим концом дуги, а не с каким-то другим, более ранним. Для решения этой проблемы начало дуги, получив описание с рангом *j+1* от какого-нибудь, возможно, более раннего конца дуги, ещё раз увеличивает ранг дуги на *1* так, что он становится равным *j+2*. Но теперь такой ранг не просто максимальный: он имеется только в начале дуги. Если идентификатор конца дуги в этом описании правильный (т.е. начало дуги получило описание с рангом *j+1* от текущего конца дуги), то это описание с максимальным рангом *j+2* будет распространяться по всему графу, пока не произойдёт следующего изменения дуги. Если же идентификатор конца дуги в описании не правильный (т.е. начало дуги получило описание с рангом *j+1* от более раннего конца дуги), то это описание с максимальным рангом *j+2* снова по дуге попадёт в текущий конец

дуги, где ранг ещё раз будет увеличен на 1. Теперь в описании в конце дуги идентификатор конца правильный, а ранг, равный *j+3*, не просто максимальный: он имеется только в конце дуги. Это описание будет распространяться по всему графу, пока не произойдёт следующего изменения дуги. В частности, такое описание с рангом *j+3* попадёт в начало дуги, где ранг снова будет увеличен до *j+4*, а идентификатор конца дуги останется правильным.

Разумеется, во время выполнения этой процедуры возможно исчезновение дуги. В начале дуги идентификатор конца дуги в её описании станет пустым, что правильно после исчезновения дуги. Но нужно ещё увеличить ранг дуги, чтобы он стал больше, чем в любой другой вершине. Поскольку по приведённому выше описанию процедуры изменения ранга видно, что ранг дуги в начале дуги либо максимальный, либо на *1* меньше максимального, при исчезновении дуги в начале дуги достаточно увеличить ранг дуги на *2*.

Что делать с рангом дуги, если сигнал исчезновения дуги пропущен, поскольку был замещён последующим сигналом появления дуги? Пропущенный сигнал исчезновения дуги означает и пропущенное состояние дуги – «дуга исчезла». Этому состоянию дуги соответствует описание с пустым идентификатором конца дуги. Поэтому, если при появлении дуги идентификатор конца дуги в её описании в начале дуги не пустой, нужно в начале дуги не только сделать идентификатор конца дуги пустым, но и увеличить ранг так, чтобы он стал больше ранга дуги в любой другой вершине, то есть увеличить на 2. Это как раз и будет соответствовать пропущенному состоянию дуги. Если же при появлении дуги идентификатор конца дуги в её описании в начале дуги пустой, в начале дуги достаточно сделать идентификатор конца дуги пустым, а ранг увеличивать не нужно, поскольку описание дуги в её начале уже соответствует пропущенному состоянию дуги.

На рис.1 выделен также случай, когда дуга является петлёй, то есть конец дуги совпадает с её началом. В этом случае не нужно сначала увеличить ранг на *1* в конце дуги, затем распространить это описание по графу до начала дуги, после чего ещё раз увеличить ранг на *1*. Вместо этого нужно сразу увеличить ранг на *2*.



Пунктирная линия – переход за 0 тактов, сплошная – за ≤ 2 такта, жирная – за ≤ 3(n-1) тактов. В двойной рамке – изменения дуги. Особый случай появления дуги: конец дуги известен, а сигнал исчезновения пропущен.

Рис. 1. Схема работы алгоритма

Также на рис.1 показаны переходы, которые происходят либо за нулевое время (при условии, что временем срабатывания автомата мы пренебрегаем), либо за время не более *2* тактов, либо за время не более *3(n-1)* тактов. На рисунке видно, что максимальная длительность цепочки переходов от изменения дуги до распространения описания этой дуги по состоянию после этого или последующих изменений дуги равна *2+3(n-1)+3(n-1)=6n-4*. Если добавить ещё один такт, то информация о дуге окажется гарантированно правильной также и во всех описаниях, передаваемых по дугам. Строгое доказательство этих оценок приведено ниже в разделе 5.1.

До сих пор мы молчаливо предполагали, что распространению по графу описания дуги ничто не мешает, кроме описаний этой же дуги с большими рангами, что соответствует более поздним состояниям дуги. Однако такому распространению могут препятствовать также изменения дуг. По «короткоживущим» дугам сообщения могут просто не успевать проходить, поэтому вся надежда на «долгоживущие» дуги, которые не меняются, по крайней мере, в течение 1 такта. В постановке задачи предполагается, что в каждый момент времени суграф, порождённый «долгоживущими» дугами, сильно связен.

Однако для того, чтобы воспользоваться этими предположениями, нужно успевать передавать по «долгоживущей» дуге все требуемые описания. А это означает, что в одном сообщении, посылаемом по некоторой дуге, нам нужно объединить все описания дуг, которые нужно передать по этой дуге. Что это за описания? Если дуга *q* появляется или меняет свой конец, то по ней нужно передавать все имеющиеся в начале дуги описания, поскольку возможен случай, когда после такого изменения дуги *q* в её конец ещё не передавались эти описания. Если дуга *q* исчезает, то по ней, конечно, ничего передавать нельзя. И только в том случае, когда не происходит изменение дуги *q*, по ней можно было бы передавать только те описания, которые по ней ещё не передавались.

Однако когда мы посылаем сообщение по дуге, содержащее лишь часть хранящихся в начале дуги описаний, мы не знаем, сменит ли дуга свой конец до завершения передачи сообщения. Если через время $x \le 1$ тактов после посылки по дуге сообщения дуга меняет свой конец, то дуга ещё не освобождается; освобождение может произойти через *1* такт, после чего нам следует послать по дуге второе сообщение со всеми (или со всеми остальными) описаниями. Это второе сообщение может передаваться тоже 1 такт, поэтому в конец дуги все описания могут попасть через 2 такта. Однако время существования «долгоживущей» дуги ограничено снизу одним тактом, а не двумя.

Но даже если бы мы увеличили эту нижнюю границу времени существования «долгоживущей» дуги до 2 тактов, это не помогло бы. Дело в том, что о смене конца дуги узнаёт лишь автомат в конце дуги, а в начало дуги информация об

этом попадает через длительное время в результате распространения описания по графу. В течение этого длительного времени дуга может много раз менять свой конец, и каждый раз по дуге в её очередной конец попадёт только часть описаний.

Эти эвристические соображения приводят к выводу, что по дуге нужно посылать сообщение, содержащее сразу все описания, хранящиеся в начале дуги.

4. Описание алгоритма

4.1 Сообщения

Сообщение содержит:

- $I^\sim$ – идентификатор начала дуги, по которой сообщение передаётся,
- $i^\sim$ – номер дуги, по которой сообщение передаётся,
- $l^\sim$ – длина массива описаний дуг,
- $L^\sim[0..l^\sim-1]$ – массив описаний дуг.

Описание e дуги содержит: 1) идентификатор начала дуги, обозначаемый $N(e)$, 2) номер дуги, обозначаемый $i(e)$, 3) идентификатор конца дуги (быть может, пустой), обозначаемый $K(e)$ и 4) ранг дуги, обозначаемый $r(e)$. Идентификатор дуги – $(N(e),i(e))$.

4.2 Память автомата

Память автомата содержит:

- управляющее состояние (начальное или рабочее),
- I – идентификатор вершины,
- l – длина массива описаний дуг, хранящихся в вершине,
- $L[0..l-1]$ – массив описаний дуг.

4.3 Работа автомата

4.3.1 Начальное состояние

Автомат опрашивает идентификатор вершины и запоминает его в I. Инициализируется: $l:=0$. Автомат переходит в рабочее состояние.

4.3.2 Рабочее состояние

В этом состоянии автомат ожидает любого сигнала и любого сообщения.

4.3.2.1 Сигнал появления дуги, параметр: номер дуги *i*.

В массиве L ищется описание e с идентификатором дуги (I,i).

Если такого описания нет, то в массив L вставляется описание $e = (I,i,\varepsilon,0)$, $l:=l+1$.

Если $K(e)\neq\varepsilon$, то $r(e):=r(e)+2$ и $K(e):=\varepsilon$.

В любом случае по дуге i посылается сообщение (I,i,l,L).

4.3.2.2 Сигнал исчезновения дуги, параметр: номер дуги *i*.

В массиве L ищется описание дуги e с идентификатором дуги (I,i).

$r(e):=r(e)+2$ и $K(e):=\varepsilon$.

4.3.2.3 Сигнал освобождения дуги, параметр: номер дуги *i*.

По дуге i посылается сообщение (I,i,l,L).

4.3.2.4 Сообщение

Просматривается массив $L^\sim$ описаний дуг в принятом сообщении. Для каждого $e^\sim=L^\sim(j^\sim)$, где $j^\sim=1..l^\sim-1$ выполняются следующие действия.

1. В массиве L ищется описание дуги e с тем же идентификатором дуги, что в $e^\sim$. Если такого описания нет, то в массив L вставляется описание $e^\sim$, $l:=l+1$.

   Далее п.2, п.3, п.4 или п.5 в зависимости от условий этих пунктов.
2. Дуга e – это петля, по которой пришло сообщение, т.е. идентификатор дуги в e равен $(I^\sim,i^\sim)$ и $I^\sim=I$.

   Если $K(e)\neq I$, то $r(e):=r(e)+2$ и $K(e):=I$.
3. Дуга e – не петля (т.е. $I^\sim\neq I$) и это дуга, по которой пришло сообщение, т.е. идентификатор дуги в e равен $(I^\sim,i^\sim)$.

   Если $r(e^\sim)\geq r(e)$ и $K(e^\sim)=I$, то $r(e):=r(e^\sim)$ и $K(e):=I$.

   Если $r(e^\sim)\geq r(e)$ и $K(e^\sim)\neq I$, то $r(e):=r(e^\sim)+1$ и $K(e):=I$.
4. Дуга e – не петля (т.е. $I^\sim\neq I$), это не дуга, по которой пришло сообщение, т.е. идентификатор дуги в e не равен $(I^\sim,i^\sim)$, и это выходящая дуга ($N(e)=I$).

   Если $r(e^\sim)>r(e)$, то $r(e):=r(e^\sim)+1$ и $K(e):=K(e^\sim)$.
5. Дуга e – не петля (т.е. $I^\sim\neq I$), это не дуга, по которой пришло сообщение, т.е. идентификатор дуги в e не равен $(I^\sim,i^\sim)$, и это не выходящая дуга ($N(e)\neq I$).

   Если $r(e^\sim) > r(e)$, то $r(e):=r(e^\sim)$, $K(e):=K(e^\sim)$.

5. Оценка времени и памяти

5.1 Время работы алгоритма

Пусть выбрана дуга p и некоторый момент времени t_0, который будем называть *началом отсчёта*. Мы будем рассматривать распространение по графу информации о дуге p после момента времени t_0. До первого появления дуги p во всех вершинах отсутствует описание дуги p. Такое «отсутствие описания» распространять не требуется, поэтому можно считать, что t_0 – момент времени после первого появления дуги p, включая обработку автоматом сигнала появления дуги. Текущий момент времени обозначим через t, $t \geq t_0$.

Начало дуги p обозначим через a, а её текущий конец (конец в момент времени t) – через b. Если дуги нет, будем считать, что $b=\varepsilon$. Через B обозначим множество концов, которые дуга p имела в интервале времени $[t_0,t]$, включая ε, если дуга p исчезала в этом интервале времени. Заметим, что множество B может только расти, за исключением случая выбора другого начала отсчёта.

Обозначим: V – множество идентификаторов вершин, E – множество идентификаторов дуг графа.

Для краткости ранг дуги p в её описании в вершине x будем называть *рангом в вершине x* и обозначать r_x. Если такое описание отсутствует, то будем считать, что $r_x = -1$. *Рангом на дуге q* будем называть ранг дуги p в её описании в сообщении, которое передаётся или (если сейчас по дуге сообщение не передаётся) последний раз передавалось по дуге q, и обозначать r_q. Если по дуге q ещё не передавалось ни одного сообщения или ни в одном сообщении не было описания дуги p, то будем считать, что $r_q = -1$. Просто *рангом* будем называть ранг в вершине или ранг на дуге. *Максимальным рангом* будем называть такой ранг, что во всех вершинах и на всех дугах ранг не больше.

Идентификатор конца дуги p в её описании в вершине x будем называть *концом в вершине x*, и обозначать K_x. Если такое описание отсутствует, то будем считать, что K_x не определено. *Концом на дуге q* будем называть идентификатор конца дуги p в её описании в сообщении, которое передаётся или (если сейчас по дуге сообщение не передаётся) последний раз передавалось по дуге q, и обозначать K_q. Если по дуге q ещё не передавалось ни одного сообщения или ни в одном сообщении не было описания дуги p, то будем считать, что K_q не определено. Просто *концом* будем называть конец в вершине или на дуге. Будем говорить, что конец K_z *правильный*, если $K_z \in B$, иначе конец K_z *– неправильный*.

Состояние графа относительно выбранной дуги p в текущий момент времени t полностью описывается величинами V, E, a, t_0, b, B, K_z, r_z, где $z \in V \cup E$. При этом V, E, a не зависят от t. Для события, которое может произойти в момент

времени t и изменить состояние графа, обозначим значения величин после события через, соответственно, $t`_0$, $b`$, $B`$, $K`_z$, $r`_z$, где $z \in V \cup E$. Также обозначим: $r = r_a$, $r` = r`_a$, $K = K_a$, $K` = K`_a$.

Из правил изменения дуг и описания алгоритма следует следующий перечень событий, меняющих состояние графа. Некоторые из этих событий являются последовательностью «микрособытий», приводящих к обработке автоматом входного символа: сигнала или принимаемого сообщения. Например, повторное появление дуги p – это последовательность следующих «микрособытий»: исчезновение дуги, выработка сигнала исчезновения дуги и постановка его в очередь входных символов автомата, появление дуги, выработка сигнала появления дуги и замещение им в очереди входных символов автомата находящегося там сигнала исчезновения, обработка автоматом сигнала появления. Важно отметить, что в любом случае событие происходит в течение времени, сравнимого с временем срабатывания автомата, которым мы пренебрегаем; иными словами, мы можем считать, что событие происходит «мгновенно». Мы рассматриваем состояние графа между событиями, но не «внутри события», то есть не между «микрособытиями» внутри одного события. В перечне событий мы указываем для каждого события только те значения величин, определяющих состояние графа, которые меняются.

1. *Смена начала отсчёта* – выбор текущего момента времени как начала отсчёта.
   $t`_0=t$, $B`=\{b\}$.
2. *Появление* дуги p с концом $x \neq \varepsilon$, включая обработку автоматом сигнала появления. Если в начале отсчёта t_0 дуги p не было, то $\varepsilon \in B$, следовательно, $\varepsilon \in B`$. Если в начале отсчёта $t_0$ дуга $p$ была, то это повторное появление дуги, перед которым должно было быть исчезновение дуги после начала отсчёта, даже если сигнал исчезновения замещался в очереди входных символов автомата сигналом появления, и поэтому тоже $\varepsilon \in B`$.
   $b`=x$, $B`=B \cup \{\varepsilon,x\}$. Если $K \neq \varepsilon$, то $r`=r+2$ и $K`=\varepsilon$. В любом случае $K`_p=K`$, $r`_p=r`$.
3. *Исчезновение* дуги p, если сигнал исчезновения не замещается последующим сигналом появления, включая обработку автоматом сигнала исчезновения.
   $b`=\varepsilon$, $B`=B \cup \{\varepsilon\}$, $r`=r+2$, $K`=\varepsilon$.
4. *Смена конца* дуги p на конец $x \neq \varepsilon$.
   $b`=x$, $B`=B \cup \{x\}$.
5. *ab-событие* – поступление сообщения по дуге p в её конец b, когда $b = a$ (дуга p является петлёй) и $K \neq a$, включая срабатывание автомата по приёму этого сообщения. Заметим, что сначала автомат принимает

сообщение, приходящее по дуге-петле, и только после этого вырабатывается сигнал освобождения. Заметим, что если $K = a$, то состояние графа не меняется.

$r`=r+2$, $K`=a$.

6. *b-событие* – поступление сообщения по дуге p в её конец $b \neq \varepsilon$, когда $a \neq b$ (дуга p не является петлёй), включая срабатывание автомата по приёму этого сообщения.

   Если $r_p \geq r_b$ и $K_p=b$, то $r`_b=r_p$ и $K`_b=b$.

   Если $r_p \geq r_b$ и $K_p \neq b$, то $r`_b=r_p+1$ и $K`_b=b$.

7. *a-событие* – поступление сообщения по дуге $q \neq p$ в вершину a, когда $r_q > r$, включая срабатывание автомата по приёму этого сообщения. Заметим, что если $r_q \leq r$, то состояние графа не меняется.

   $r`=r_q+1$ и $K`=K_q$.

8. *Приём* – поступление сообщения по дуге $q \neq p$ в вершину $x \neq a$, когда $r_q > r_x$, включая срабатывание автомата по приёму этого сообщения. Заметим, что если $r_q \leq r_x$, то состояние графа не меняется.

   $r`_x=r_q$, $K`_x=K_q$.

9. *Посылка* – освобождение дуги q или появление дуги $q \neq p$, имеющей начало в x, включая срабатывание автомату по сигналу освобождения или появления дуги, соответственно.

   $r`_q=r_x$, $K`_q=K_x$.

Определим следующие состояния графа:

1. $b \neq \varepsilon \;\&\; \exists z \in V\; r_z > r \;\&\; K_z \notin B$.

2. $b \neq \varepsilon \;\&\; \forall z \in V \cup E\; r_z \leq r \;\&\; K \notin B$.

3(w). $\exists z \in V\; r_z \geq w \;\&\; \forall z \in V \cup E\; (r_z \geq w \Rightarrow K_z \in B)$.

4. $\forall z \in V \cup E\; K_z \in B$.

Состояние 3(w) определяется для некоторого ранга w. Понятно, что если $w`>w$ и $\exists z \in V\; r_z \geq w`$, то состояние 3(w) является также состоянием 3($w`$). Состояние 4 является, на самом деле, подсостоянием состояния 3(w), когда во всех вершинах и на всех дугах ранг не меньше w.

Лемма 1. Ранг дуги в вершине не уменьшается: $\forall x \in V\; r`_x \geq r_x$.

Доказательство. Непосредственно следует из описания событий.

Лемма доказана.

Лемма 2. Ранг на дуге не превосходит ранга в начале дуги.

Доказательство. Как видно из описания событий, ранг на дуге может измениться только при посылке сообщения по этой дуге (события *появление* и *посылка*), а в этот момент времени ранг на дуге равен рангу в начале дуги. Поскольку по лемме 1 ранг в начале дуги не уменьшается, ранг на дуге не превосходит ранга в начале дуги.

Лемма доказана.

Лемма 3.

1) Ранги r и r_p чётны.

2) Любой ранг не превосходит ранга в вершине a плюс *1*: $\forall z \in V \cup E\; r_z \leq r+1$.

3) Если ранг совпадает с рангом вершины a, то концы также совпадают: $\forall z \in V \cup E\; (r_z = r \Rightarrow K_z = K)$.

4) Если событие увеличивает ранг $r`>r$, то $r`=r+2$ и после события ранг $r`$ максимальный и имеется только в вершине a и, быть может, на дуге p.

Доказательство. Будем вести доказательство индукцией по событиям. Как видно из описания алгоритма, после первого появления дуги p имеем: $r_p=r=0$, $K_p=K=\varepsilon$, а для $z \neq a$ и $z \neq p$: $r_z=-1$, следовательно, утверждения леммы верны.

Пусть до события утверждения леммы верны. Утверждения 1 и 4 могли бы быть нарушены только в результате событий *появление*, *исчезновение*, *ab-события* или *a-события*, которые меняют r или r_p. Утверждения 2 и 3 дополнительно могли бы быть нарушены в результате *b-события*, меняющего r_b или K_b.

Появление. Если $K \neq \varepsilon$, то $r`=r+2$, $K`=\varepsilon$. В любом случае $K`_p=K`$, $r`_p=r`$. По предположению шага индукции для утверждения 1 r чётно, следовательно, $r` = r$ или $r` = r+2$ и $r`_p=r`$ тоже чётны, поэтому утверждение 1 остаётся верным. По предположению шага индукции для утверждения 2 $r_z \leq r+1$, следовательно, для $z \neq a$ и $z \neq p$ имеем $r`_z = r_z \leq r+1 \leq r`+1$, а $r`_p=r`$, поэтому утверждение 2 остаётся верным. Если $K = \varepsilon$, то $r`_p=r`$, $K`_p=K`$, а для $z \neq p$ ранг и конец не меняются, поэтому утверждение 3 остаётся верным и, поскольку $r`=r$, утверждение 4 тоже верно. Если $K \neq \varepsilon$, то $r`_p=r`=r+2$, $K`_p=K`$, а для $z \neq a$ и для $z \neq p$ ранг не меняется. Поскольку по предположению шага индукции для утверждения 2 $r_z \leq r+1$, ранг $r`$ после события максимальный и имеется только в вершине a и на дуге p, поэтому утверждение 3 остаётся верным и утверждение 4 тоже верно.

Исчезновение. Имеем $r`=r+2$, $K`=\varepsilon$. По предположению шага индукции для утверждения 1 r чётно, следовательно, $r`=r+2$ тоже чётно, поэтому утверждение 1 остаётся верным. По предположению шага индукции для утверждения 2 $r_z \leq r+1$, следовательно, для $z \neq a$ имеем $r`_z = r_z \leq r+1 < r+2 = r`$, поэтому утверждение 2 остаётся верным. Ранг $r`$ после события максимальный и имеется только в вершине a, поэтому утверждение 3 остаётся верным и утверждение 4 тоже верно.

ab-событие. Имеем $r`=r+2$, $K`=a$. По предположению шага индукции для утверждения 1 r чётно, следовательно, $r`=r+2$ тоже чётно, поэтому утверждение 1 остаётся верным. По предположению шага индукции для утверждения 2 $r_z \leq r+1$, следовательно, для $z \neq a$ имеем $r`_z = r_z \leq r+1 < r+2 = r`$, поэтому утверждение 2 остаётся верным. Ранг $r`$ после события максимальный

и имеется только в вершине a, поэтому утверждение 3 остаётся верным и утверждение 4 тоже верно.

a-событие ($r_q > r$). Имеем $r`=r_q+1$ и $K`=K_q$. Поскольку по предположению шага индукции для утверждения 2 $r_q \le r+1$, имеем $r_q = r+1$ и $r`=r+2$. По предположению шага индукции для утверждения 1 $r$ чётно, следовательно, $r`=r+2$ тоже чётно, поэтому утверждение 1 остаётся верным. Также по предположению шага индукции для утверждения 2 $r_z \le r+1$, следовательно, для $z \ne a$ имеем $r`_z = r_z \le r+1 < r+2 = r`$, поэтому утверждение 2 остаётся верным. Ранг $r`$ после события максимальный и имеется только в вершине a, поэтому утверждение 3 остаётся верным и утверждение 4 тоже верно.

b-событие. 1) Если $r_p \ge r_b$ и $K_p=b$, то $r`_b=r_p$ и $K`_b=b$. 2) Если $r_p \ge r_b$ и $K_p \ne b$, то $r`_b=r_p+1$ и $K`_b=b$. Поскольку *b-событие* не меняет r и r_p, утверждение 1 остаётся верным. По лемме 2 $r_p \le r$. Следовательно, поскольку $r`_b \le r_p+1$, имеем $r`_b \le r+1$, и, поскольку $r`=r$, $r`_b \le r`+1$. По предположению шага индукции для утверждения 2 $r_z \le r+1$, и, поскольку для $z \ne b$ $r`_z=r_z$, имеем $r`_z \le r+1=r`+1$. Поэтому утверждение 2 сохраняется. В случае 1 по предположению шага индукции для утверждения 3, если $r_p=r$, то $K_p=K$, а тогда $r`_b=r$ и $K`_b=b=K_p=K$. В случае 2 $r`_b=r_p+1$ и, поскольку по предположению шага индукции для утверждения 1 $r_p$ и $r$ чётны, $r`_b$ нечётно и, следовательно, $r`_b \ne r=r`$. Поскольку остальные ранги и концы не меняются, утверждение 3 верно. Поскольку *b-событие* не меняется ранг в вершине a, то утверждение 4 верно.

Лемма доказана.

Лемма 4. Если дуги нет ($b=\varepsilon$), то ранг r максимальный $\forall z \in V \cup E\ r_z \le r$ и $K=\varepsilon$.

Доказательство. Поскольку начало отсчёта t_0 – это момент времени после первого появления дуги p, отсутствие дуги p (между событиями, а не «внутри» одного события между «микрособытиями») возможно только в результате события *исчезновение*, при котором $r`=r+2$, $K`=\varepsilon$, а остальные ранги и концы не меняются. Поскольку по лемме 3 (утверждение 2) $r_z \le r+1$, для $z \ne a$ имеем $r`_z = r_z \le r`$. Поэтому, учитывая, что $K`=\varepsilon$, утверждение леммы верно.

Лемма доказана.

Лемма 5. Пусть в некоторый момент времени в некоторой вершине x ранг $r_x = j$. Тогда не более чем через $3(n-1)$ тактов в каждой вершине y ранг $r_y \ge j$.

Доказательство. Пусть в некоторый момент времени $A \ne \varnothing$ – это множество вершин y, в которых дуга p имеет ранг $r_y \ge j$. В каждый момент времени сураф, порождённый «долгоживущими» дугами, сильно связен, время существования «долгоживущей» дуги не меньше 1 такта, время пересылки по дуге не больше 1 такта, а временем срабатывания автомата мы пренебрегаем. Кроме того, в каждой вершине при появлении или освобождении дуги по ней сразу посылается сообщение.

Докажем *утверждение о расширении* множества A: с момента времени t_1, когда образуется множество $A \ne \varnothing$ и $A \ne V$, не более чем через 3 такта множество A будет расширено.

Поскольку по лемме 1 ранг дуги p в вершине не может уменьшаться, множество A может только расширяться. В любой момент времени, в том числе в момент времени t_1+2, должна существовать «долгоживущая» дуга q, ведущая из вершины $x \in A$ «наружу», т.е. в вершину $y \notin A$. В момент времени t_1+2 по дуге q передаётся сообщение S. Это сообщение отправлено из x не ранее момента времени $(t_1+2)-1=t_1+1$, поэтому в S ранг дуги p не меньше j. Возможны два случая.

1) Сообщение S дойдёт до вершины y. Тогда это произойдёт не позднее момента времени $(t_1+2)+1=t_1+3$. Следовательно, в этом случае не более чем через 3 такта после момента времени t_1 множество A будет расширено.
2) Сообщение S не дойдёт до вершины y. Это означает, что дуга q изменится до того, как по ней до вершины y дойдёт сообщение S. А тогда, поскольку q – «долгоживущая» дуга, по ней должно дойти до вершины y предыдущее сообщение $S`$. Поскольку сообщение $S$ отправлено из $x$ не ранее момента времени $t_1+1$, сообщение $S`$ принято в y не ранее этого же момента времени. Следовательно, сообщение $S`$ отправлено из $x$ не ранее момента времени $(t_1+1)-1=t_1$, поэтому в $S`$ ранг дуги p не меньше j. Следовательно, в этом случае не более чем через 1 такт после момента времени t_1 множество A будет расширено.

При каждом расширении множества A в него добавляется хотя бы одна вершина. Поскольку с самого начала множество A содержит хотя бы одну вершину, а общее число вершин равно n, получается, что число расширений множества A не более $n-1$. Таким образом, требуется не более чем $3(n-1)$ тактов, чтобы в каждой вершине y оказался ранг $r_y \ge j$.

Лемма доказана.

Лемма 6 (о состоянии 1). Граф находится в состоянии 1 не более $3(n-1)$ тактов.

Доказательство. Пусть в момент времени t граф находится в состоянии 1, тогда в некоторой вершине $r_x > r$. По лемме 5 в момент времени $t \le t^ \le t+3(n-1)$ во всех вершинах ранг будет не меньше $r+1$, в том числе и в вершине a. Рассмотрим событие, в результате которого в вершине a ранг первый раз увеличивается, т.е. $r` > r$. По лемме 3 (утверждение 4) $r`=r+2$ и после события ранг $r`$ максимальный и имеется только в вершине $a$ и, быть может, на дуге $p$. Если дуга $p$ не исчезла ($b` \ne \varepsilon$) и $K` \notin B`$, то мы имеем состояние 2. Если дуга p не исчезла ($b` \ne \varepsilon$) и $K` \in B`$, то, поскольку по лемме 3 (утверждение 3) $K`_p=K`$, мы имеем состояние 3($r`$). Если дуга исчезла ($b` = \varepsilon$), то $\varepsilon \in B`$, по лемме 4 ранг $r`$ максимальный и $K` = \varepsilon$, а по лемме 3 (утверждение 3) для любого z, где $r`_z = r`$, также $K`_z = \varepsilon$. Следовательно, для любого $z$, где $r`_z$

максимальный ранг, имеет место $K`_z \in B`$, поэтому мы имеем состояние 3($r`$). Таким образом, через время не более *3(n-1)* тактов граф покинет состояние 1. Лемма доказана.

Лемма 7 (о состоянии 2). Граф находится в состоянии 2 не более *2* тактов, после чего переходит в состояние 3(w) для некоторого ранга w.

Доказательство. Рассмотрим различные события.

1. *Смена начала отсчёта*: $t`_0=t$, $B`=\{b\}$.

   Поскольку $b \in B$ и в состоянии 2 $K \notin B$, имеем $K`=K \notin B`$. Поскольку $b`=b$, а в состоянии 2 $b \neq \varepsilon$, имеем $b` \neq \varepsilon$. Поскольку ранги не меняются, граф остаётся в состоянии 2.

2. *Появление* дуги p с концом $x \neq \varepsilon$: $b`=x$, $B`=B \cup \{\varepsilon,x\}$. Если $K \neq \varepsilon$, то $r`=r+2$, $K`=\varepsilon$. В любом случае $K`_p=K`$, $r`_p=r`$.

   Покажем, что ранг $r`$ максимальный после события. Действительно, если $K=\varepsilon$, то ранги не меняются, кроме $r`_p=r`$, и поэтому ранг $r`$ максимальный по условию состояния 2. Если $K \neq \varepsilon$, то по лемме 1 ранг $r`>r$ и тогда по лемме 3 (утверждение 4) ранг $r`$ максимальный.

   По лемме 3 (утверждение 3) везде, где ранг равен $r`$, конец равен $K`$. Также в любом случае $K`=\varepsilon$, а, поскольку $\varepsilon \in B`$, имеем $K` \in B`$. Следовательно, граф переходит в состояние 3($r`$).

3. *Исчезновение* дуги p: $b`=\varepsilon$, $B`=B \cup \{\varepsilon\}$, $r`=r+2$, $K`=\varepsilon$.

   Поскольку $r`>r$, по лемме 3 (утверждение 4) ранг $r`$ максимальный. По лемме 3 (утверждение 3) везде, где ранг равен $r`$, конец равен $K`$. Поскольку $K`=\varepsilon$ и $\varepsilon \in B`$, имеем $K` \in B`$. Следовательно, граф переходит в состояние 3($r`$).

4. *Смена конца* дуги p на конец $x \neq \varepsilon$: $b`=x$, $B`=B \cup \{x\}$.

   Если $K=x$, то $K`=K \in B`$. Ранг $r`$ максимальный по условию состояния 2. По лемме 3 (утверждение 3) везде, где ранг равен $r`$, конец равен $K`$. Поэтому граф переходит в состояние 3($r`$).

   Если $K \neq x$, то, поскольку в состоянии 2 $K \notin B$, имеем $K`=K \notin B`$. Ранг $r`$ максимальный по условию состояния 2, $b`=x \neq \varepsilon$. Поэтому граф остаётся в состоянии 2.

5. *ab-событие*: $r`=r+2$, $K`=a$.

   Поскольку, если дуга p петля, то $a \in B=B`$, имеем $K` \in B`$. Поскольку $r`>r$, по лемме 3 (утверждение 4) ранг $r`$ максимальный. По лемме 3 (утверждение 3) везде, где ранг равен $r`$, конец равен $K`$. Следовательно, граф переходит в состояние 3($r`$).

6. *b-событие*: если $r_p \geq r_b$ и $K_p=b$, то $r`_b=r_p$ и $K`_b=b$, а если $r_p \geq r_b$ и $K_p \neq b$, то $r`_b=r_p+1$ и $K`_b=b$.

   6.1. Если $r_p \geq r_b$ и $K_p=b$, то ранг r остаётся максимальным согласно условию состояния 2. Поскольку в состоянии 2 $K \notin B$, и *b-событие* не меняется конец в вершине a, имеем $K`=K \notin B=B`$. Поскольку $b`=b$, а в состоянии 2 $b \neq \varepsilon$, имеем $b` \neq \varepsilon$. Следовательно, граф остаётся в состоянии 2.

   6.2. Если $r_p \geq r_b$ и $K_p \neq b$, то, поскольку в состоянии 2 ранг r максимальный, возможны два случая: $r_p<r$ и $r_p=r$.

      6.2.1. В первом случае ($r_p<r$) ранг r остаётся максимальным согласно условию состояния 2. Поскольку в состоянии 2 $K \notin B$, и *b-событие* не меняет конец в вершине a, имеем $K`=K \notin B=B`$. Поскольку $b`=b$, а в состоянии 2 $b \neq \varepsilon$, имеем $b` \neq \varepsilon$. Следовательно, граф остаётся в состоянии 2.

      6.2.2. Во втором случае ($r_p=r$) имеем $r`_b>r$ и, поскольку в состоянии 2 ранг $r$ до события был максимальным, после события ранг $r`_b$ максимальный и имеется только в вершине b. Поскольку $b`=b \in B=B`$, а $K`_b=b$, имеем $K`_b \in B`$. Следовательно, граф переходит в состояние 3($r`_b$).

7. *a-событие* ($r_q > r$): $r`=r_q+1$ и $K`=K_q$.

   Условие $r_q > r$ противоречит условию максимальности ранга r в состоянии 2, поэтому это событие в состоянии 2 невозможно.

8. *Приём сообщения*: $r`_x=r_q$, $K`_x=K_q$.

   Это событие, очевидно, сохраняет состояние 2.

9. *Посылка сообщения*: $r`_q=r_x$, $K`_q=K_x$.

   Это событие, очевидно, сохраняет состояние 2.

Таким образом, мы показали, что любое событие либо оставляет граф в состоянии 2, либо переводит его в состояние 3(w) для некоторого ранга w, но не в состояние 1. Теперь покажем, что не более чем через 2 такта граф перейдёт в состояние 3(w). Рассмотрим момент времени t_2, когда граф переходит в состояние 2. Начиная с момента времени t_2 ранг r может измениться только после событий *появление*, *исчезновение* или после *ab-события* (*a-событие* невозможно в состоянии 2). Однако каждое из этих событий гарантированно переводит граф в состояние 3(w). Поэтому достаточно рассмотреть случай, когда эти события не происходят. Тогда ранг r не меняется всё время, пока граф остаётся в состоянии 2. Из описания событий следует, что без изменения ранга r не меняется конец K. Также условие $K \notin B$ не меняется, пока граф остаётся в состоянии 2. Поскольку в состоянии 2 дуга p есть ($b \neq \varepsilon$), в момент времени t_2 по дуге p передаётся сообщение, которое, возможно, было послано ещё до момента времени t_2. Поскольку дуга p не появляется и не исчезает, через время не более 1 такта это сообщение гарантированно будет принято в конце дуги p, после чего по этой дуге будет послано второе сообщение. Это второе сообщение будет послано

тогда, когда граф находится в состоянии 2, поэтому в этот момент времени будет $K_p=K \notin B$ и $r_p=r$. Поскольку дуга p не появляется и не исчезает, это второе сообщение также гарантированно будет принято в конце дуги p через время не более 1 такта, причём, поскольку r и K не меняются и не меняется условие $K \notin B$, будет $K_p \notin B$ и $r_p=r$. Поскольку в момент приёма сообщения в вершине b имеет место $b \in B$, имеем $K_p \neq b$. Поскольку в состоянии 2 ранг r максимальный и поскольку $r_p=r$, имеем $r_p \geq r_b$. А тогда, как показано выше для *b-события* (6.2.2), граф переходит в состояние $3(r`_b)$. Итак, мы показали, что через время не более 2 тактов граф переходит из состояния 2 в состояние $3(w)$. Лемма доказана.

Лемма 8 (о состоянии 3). Если не меняется начало отсчёта, то граф находится в состоянии $3(w)$ для любого заданного w не более $3n-2$ тактов, после чего переходит в подсостояние 4, из которого уже не выходит. Смена начала отсчёта переводит состояние $3(w)$ в состояние 1, 2 или $3(w`)$.

Доказательство. Рассмотрим различные события.

1. *Смена начала отсчёта*: $t`_0=t$, $B`=\{b\}$.

   Если $b=\varepsilon$, то по лемме 4 ранг r максимальный и $K=\varepsilon=b \in B`$. По лемме 3 (утверждение 3) везде, где ранг равен $r$, конец равен $K$. Тогда, поскольку $b`=b$, $r`=r$, $K`=K$ и для любого z $K`_z=K_z$, граф переходит в состоянии $3(r)$.

   Если $b \neq \varepsilon$, то возможны четыре случая:

   1) Ранг r максимальный и $K \neq b$.

      Тогда $K \notin B`$. Граф переходит в состояние 2.

   2) Ранг r максимальный и $K=b$.

      Тогда $K \in B`$. По лемме 3 (утверждение 3) для всех z таких, что $r_z=r$, имеет место $K_z=K$. Граф переходит в состоянии $3(r)$.

   3) Максимальный ранг $r_{max} > r$, для некоторого z, где $r_z = r_{max}$, $K_z \neq b$.

      Тогда $K_z \notin B`$. Граф переходит в состояние 1.

   4) Максимальный ранг $r_{max} > r$, для каждого z, где $r_z = r_{max}$, $K_z = b$.

      Тогда $K_z \notin B`$. Граф переходит в состояние $3(r_{max})$.

Поскольку для любого другого события $B \subseteq B`$, нам достаточно показать, что если меняются ранг $r`_z \neq r_z$ или конец $K`_z \neq K_z$, то конец становится правильным $K`_z \notin B`$.

2. *Появление* дуги p с концом $x \neq \varepsilon$: $b`=x$, $B`=B \cup \{\varepsilon,x\}$. Если $K \neq \varepsilon$, то $r`=r+2$, $K`=\varepsilon$. В любом случае $K`_p=K`$, $r`_p=r`$.

   Поскольку $K`_p=K`=\varepsilon$ и $\varepsilon \in B`$, граф остаётся в состоянии $3(w)$.

3. *Исчезновение* дуги p: $b`=\varepsilon$, $B`=B \cup \{\varepsilon\}$, $r`=r+2$, $K`=\varepsilon$.

   Поскольку $K`=\varepsilon$ и $\varepsilon \in B`$, граф остаётся в состоянии $3(w)$.

4. *Смена конца* дуги p на конец x: $b`=x$, $B`=B \cup \{x\}$.

   Граф остаётся в состоянии $3(w)$.

5. *ab-событие*: $r`=r+2$, $K`=a$.

   Поскольку $K`=a$ и для петли $a \in B=B`$, граф остаётся в состоянии $3(w)$.

6. *b-событие*: если $r_p \geq r_b$ и $K_p=b$, то $r`_b=r_p$ и $K`_b=b$, а если $r_p \geq r_b$ и $K_p \neq b$, то $r`_b=r_p+1$ и $K`_b=b$.

   Поскольку $K`_b=b$ и $b \in B=B`$, граф остаётся в состоянии $3(w)$.

7. *a-событие* ($r_q > r$): $r`=r_q+1$ и $K`=K_q$.

   Если $r_q > r$, то, поскольку по лемме 3 (утверждение 2) $r_q \leq r+1$, имеем $r_q = r+1$, следовательно, по лемме 3 (утверждение 2) r_q максимальный ранг. А тогда в состоянии $3(w)$ $K_q \in B$, что влечёт $K`=K_q \in B=B`$. Поэтому граф остаётся в состоянии $3(w)$.

8. *Приём сообщения*: $r`_x=r_q$, $K`_x=K_q$.

   Это событие, очевидно, сохраняет состояние $3(w)$.

9. *Посылка сообщения*: $r`_q=r_x$, $K`_q=K_x$.

   Это событие, очевидно, сохраняет состояние $3(w)$.

Таким образом, мы показали, что любое событие, кроме смены начала отсчёта, оставляет граф в состоянии $3(w)$ без изменения ранга w. Теперь покажем, что не более чем через $3n-2$ тактов граф перейдёт в подсостояние 4. По лемме 2 не более чем через $3(n-1)$ тактов во всех вершинах будет ранг не менее w. А тогда ещё не более чем через 1 такт на каждой дуге тоже будет ранг не менее w. Для состояния $3(w)$ это как раз и означает переход в подсостояние 4 через $3(n-1)+1=3n-2$ тактов.

Лемма доказана.

Теорема 1. Через время не более $6n-3$ тактов в каждой вершине и на каждой дуге z будет $K_z \in B$, т.е. $T_0 = \boldsymbol{O}(n)$.

Доказательство. Состояние 4 – это подсостояние $3(w)$, когда в каждой вершине и на каждой дуге ранг не меньше w. В этом случае как раз будет выполнено условие $K_z \in B$. Поэтому нам надо показать, во-первых, что при любом выборе начала отсчёта граф окажется в одном из состояний 1, 2 или $3(w)$, и, во-вторых, что после этого через время не более $4n-1$ тактов он окажется в состоянии 4.

Если t_0 – момент времени непосредственно после обработки автоматом сигнала первого появления дуги p, то имеем: $r_p=r=0$, $K_p=K=\varepsilon$, а для $z \neq a$ и $z \neq p$: $r_z=-1$, что соответствует состоянию 2. По леммам 6, 7, 8 (о состояниях 1, 2, 3) выбор другого начало отсчёта переводит в одно из состояний 1, 2 или $3(w)$ для некоторого ранга w.

По лемме 6 (о состоянии 1) в состоянии 1 граф находится не более $3(n-1)$ тактов. По лемме 7 (о состоянии 2) в состоянии 2 граф находит не более 2 тактов, после чего переходит в состояние $3(w)$ для некоторого ранга w. По лемме 8 (о состоянии 3) в состоянии $3(w)$ граф находит не более $3n-2$ тактов, после чего переходит в подсостояние 4, из которого уже не выходит, если не

меняется начало отсчёта. Следовательно, через время не более $3(n-1)+2+3n-2=6n-3$ из любого состояния граф переходит в состояние 4. Теорема доказана.

Теорема 2. После прекращения изменений в графе полная информация о графе будет собрана в каждой его вершине через время не более $4D+3$ тактов, т.е. $T_1=\boldsymbol{O}(D)$.

Доказательство. После прекращения изменений графа порядок верхней границы времени распространения информации о дуге снижается с n до D. Действительно, рассмотрим множество вершин A из доказательства леммы 5, содержащее вершины, в которых ранг не меньше j.

Поскольку дуги больше не меняются, модифицируется *утверждение о расширении* в доказательстве леммы 5: с момента времени t_1, когда образуется множество $A\neq\varnothing$ и $A\neq V$, множество A будет расширено не более чем через 2 такта (а не 3 такта).

Действительно, в любой момент времени, в том числе в момент времени t_1+1, должна существовать «долгоживущая» дуга q, ведущая из вершины $x\in A$ «наружу», т.е. в вершину $y\notin A$. В момент времени t_1+1 по дуге q передаётся сообщение S. Это сообщение отправлено из x не ранее момента времени $(t_1+1)-1=t_1$, поэтому в S ранг дуги p не меньше j. Поскольку после момента времени t_1 дуги не меняются, сообщение S дойдёт до вершины y. Тогда это произойдёт не позднее момента времени $(t_1+1)+1=t_1+2$. Следовательно, не более чем через 2 такта после момента времени t_1 множество A будет расширено.

Пусть $A_0\neq\varnothing$ – это множество A с самого начала. Для каждой вершины $x\notin A_0$ выберем входящую дугу, по которой в эту вершину первый раз пришло сообщение с рангом не меньше j. Выбранные дуги, очевидно, образуют лес деревьев, ориентированных от своих корней, которыми являются вершины из множества A_0. Этот лес деревьев содержит все вершины. Поскольку дуги не меняются, с момента времени, когда в некоторой вершине x ранг становится не меньше j, по всем выбранным дугам, выходящим из данной вершины x, пройдут сообщения с рангом не меньше j за время не более 2 тактов, согласно модифицированному утверждению о расширении. А это означает, что сообщения с рангом не меньше j пройдут по всем выбранным дугам за время не более чем $2L$ тактов, где L – максимальная длина маршрута из выбранных дуг. Поскольку выбранные дуги образуют лес деревьев, $L\leq D$.

Следовательно, изменяется оценка времени в лемме 5: вместо $3(n-1)$ будет $2D$ тактов. Соответственно, изменяются оценки в леммах 6 (о состоянии 1) и в лемме 8 (о состоянии 3): вместо $3(n-1)$ и $3n-2$ будет $2D$ и $2D+1$. Поэтому максимальное время работы алгоритма уменьшается с $3(n-1)+2+3n-2=6n-3$ тактов до $2D+2+2D+1=4D+3$ тактов.

Теорема доказана.

Заметим, что в доказательстве теоремы 2 не использовано ограничение на время существования дуги: до того, как прекращаются изменения в графе, время существования дуг может быть произвольным. Также до прекращения изменений граф может быть не сильно связным, и сильная связность графа требуется только после прекращения его изменений.

5.2 Размер сообщения

Сообщение содержит идентификатор вершины $I^{\sim}$, номер дуги $i^{\sim}$ от 1 до s, длину $l^{\sim}$ массива описаний дуг и массив $L^{\sim}[0..l^{\sim}-1]$ описаний дуг.

Обозначим через $sizeofI$ размер (в битах) памяти, требуемой для хранения идентификатора вершины. Поскольку разные вершины имеют разные идентификаторы, а число вершин равно n, имеем $sizeofI \geq \log_2 n$. Нижняя оценка достигается, если вершины просто нумеруются от 0 до $n-1$.

Номер дуги имеет размер $\log_2(s+1)$, поскольку дуги нумеруются от 1 до s.

Как следует из описания алгоритма, каждое изменение дуги приводит к увеличению ранга этой дуги не более чем на 2 (события *появление*, *исчезновение*, *ab-событие* или *b-событие* плюс *a-событие*). Обозначим через v (от *variation*) максимальное число изменений одной дуги. Тогда для данной дуги максимальный ранг не более $2v$, поэтому для ранга достаточно $\log_2 2v$ бит памяти. Длина $l^{\sim}$ массива не превосходит числа дуг m.

Итого: размер сообщения $sizeofI+\log_2(s+1)+\log_2 m + m(2sizeofI+\log_2(s+1)+\log_2 2v) = \boldsymbol{O}(m(sizeofI+\lg s)) + \boldsymbol{O}(m\lg v)$ бит. Поскольку $s\leq m$, это равно $\boldsymbol{O}(m(sizeofI+\lg m)) + \boldsymbol{O}(m\lg v)$. Если идентификатор вершины – это просто её номер от 0 до n, то, учитывая, что $n\leq m$, имеем $\boldsymbol{O}(m(\lg n+\lg m)) + \boldsymbol{O}(m\lg v)= \boldsymbol{O}(m\lg m) + \boldsymbol{O}(m\lg v)$. Второе слагаемое – это память для хранения рангов, размер которой зависит от числа изменений дуги.

5.3 Память вершины (автомата в вершине)

Для хранения управляющего состояния (начальное или рабочее) достаточно 1 бита. Идентификатор вершины занимает $sizeofI$ бит.

Суммарно имеем размер памяти автомата в вершине:

$1+sizeofI+\log_2 m + m(2sizeofI+\log_2(s+1)+\log_2 2v) =$

$= \boldsymbol{O}(m(sizeofI+\lg s)) + \boldsymbol{O}(m\lg v)$ бит.

Поскольку $s\leq m$, это равно $\boldsymbol{O}(m(sizeofI+\lg m)) + \boldsymbol{O}(m\lg v)$. Если идентификатор вершины – это просто её номер от 0 до n, то, учитывая, что $n\leq m$, имеем $\boldsymbol{O}(m(\lg n+\lg m)) + \boldsymbol{O}(m\lg v)= \boldsymbol{O}(m\lg m) + \boldsymbol{O}(m\lg v)$. Заметим, что память, необходимая для хранения полного описания графа, – это первое из двух слагаемых суммы. Второе слагаемое – это память для хранения рангов, размер которой зависит от числа изменений дуги.

5.4. Модификация алгоритма для уменьшения размера памяти

Как показано выше, размер сообщения и памяти вершины зависит от числа изменений дуги. От этой зависимости можно избавиться, если использовать либо дополнительное предположение об изменениях дуг, либо дополнительную возможность для автомата.

Дополнительное предположение: все дуги долгоживущие. В этом случае каждая дуга меняется не чаще 1 раза в такт. Следовательно, за время работы алгоритма максимум $6n-3$ тактов дуга меняется не более $6n-3$ раз. Поскольку каждое изменение дуги увеличивает ранг не более чем на 2, можно сделать ранг циклическим с максимальным значением $2(6n-3)$. Тогда размер сообщения и размер памяти вершины равен $\boldsymbol{O}(m(sizeofI+lgm)) + \boldsymbol{O}(mlgv) = \boldsymbol{O}(m(sizeofI+lgm)) + \boldsymbol{O}(mlgn)$ или, поскольку $n \le m$, $\boldsymbol{O}(m(sizeofI+lgm))$.

Дополнительная возможность: временные сигналы для автомата. Пусть имеется таймер, посылающий автомату каждый такт специальный временной сигнал. Все те действия, которые автомат выполнял при обработке сигнала освобождения или появления дуги, теперь он будет выполнять при обработке временного сигнала. Это значит, что сообщения будут посылаться по дугам не при их освобождении или появлении, а, вообще говоря, позже – при обработке следующего за освобождением или появлением дуги временного сигнала. Тем самым, сообщения по дугам будут посылаться не чаще, чем раз в такт.

Правда, в этом случае придётся предположить, что время существования «долгоживущих» дуг ограничено снизу не 1, а 2 тактами. Действительно, в противном случае может оказаться, что каждый раз при появлении дуги сообщение посылается по ней не сразу, а спустя какое-то время (до получения временного сигнала), из-за чего сообщение не успевает передаться по дуге до её изменения.

В общем случае рассмотрим τ_1 – интервал между сигналами времени, τ_2 – максимальное время передачи сообщения по дуге, τ_3 – минимальное время жизни долгоживущей дуги. Тогда нужно, чтобы выполнялось следующее соотношение $\tau_1 + \tau_2 \le \tau_3$.

В этом случае максимальное время пребывания графа в состоянии 2 будет не 2 такта, а $2\tau_2+\tau_1$. В доказательстве леммы 5 время расширения множества A меняется с 3 на $3\tau_2+2\tau_1$. Соответственно, оценка времени в лемме 5 меняется с $3(n-1)$ на $(3\tau_2+2\tau_1)(n-1)$, время пребывания графа в состоянии 1 меняется с $3(n-1)$ на $(3\tau_2+2\tau_1)(n-1)$, время пребывания графа в состоянии 3 меняется с $3n-2$ на $(3\tau_2+2\tau_1)(n-1)+(\tau_2+\tau_1)$. Эти утверждения легко доказываются, но эти доказательства мы опускаем.

Итак, максимальное время работы алгоритма, включая время «притормаживания» в ожидании временного сигнала, останется тем же по порядку, что и в теореме 1, $T_0 = (3\tau_2+2\tau_1)(n-1) + 2\tau_2+\tau_1 + (3\tau_2+2\tau_1)(n-1) + (\tau_2+\tau_1) = (6\tau_2+4\tau_1)(n-1) + 3\tau_2+2\tau_1 = (3\tau_2+2\tau_1)(2n-1) = \boldsymbol{O}(n)$. Ранг станет циклическим и будет занимать память $\boldsymbol{O}(lgn)$ бит. Соответственно, размер сообщения и памяти вершины равен $\boldsymbol{O}(m(sizeofI+lgm)) + \boldsymbol{O}(mlgn)$ или, поскольку $n \le m$, $\boldsymbol{O}(m(sizeofI+lgm))$.

Соответственно, модифицируется утверждение теоремы 2: $T_1 = (2\tau_2+\tau_1)D + 2\tau_2+\tau_1 + (2\tau_2+\tau_1)D + (\tau_2+\tau_1) = (4\tau_2+2\tau_1)D + 3\tau_2+2\tau_1 = \boldsymbol{O}(D)$.

Список литературы

[1]. Steven S. Skiena. The Algorithm Design Manual. Springer-Verlag, New York, 1997.

[2]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. Программирование, 2003 г., №5, с. 59-69.

[3]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированный случай. Программирование, 2004 г., №1, с. 2-17.

[4]. M.O. Rabin. Maze Threading Automata. An unpublished lecture presented at MIT and UC. Berkeley, 1967.

[5]. И.Б. Бурдонов. Обход неизвестного ориентированного графа конечным роботом. Программирование, 2004 г., № 4, с. 11-34.

[6]. И.Б. Бурдонов. Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. Программирование, 2004 г., № 6, с. 6-29.

[7]. И. Бурдонов, А. Косачев, В. Кулямин. Параллельные вычисления на графе. Программирование, 2015, №1, с. 3-20.

Monitoring of dynamically changed graph

Igor Burdonov <igor@ispras.ru>
Alexander Kossatchev kos@ispras.ru
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Monitoring of oriented graphs is a key task in many applications. Such monitoring is very specific when the graph models a communication network including Internet and GRID. A node of the network has local information about the network: if "knows" only about the arcs outgoing from this node, but does not "know" where (to which nodes) these arcs go. The nodes of the network exchange messages through the network links represented as arcs of the graph and act as message transfer channels. The graph monitoring is based on its traversal when message passes each arc in the graph. While there is an untraversed arc, we cannot be certain that it goes to still unmonitored part of the graph. Usually, the graph traversal is performed with a single message circulating in the network. Traversal can be done

faster if performed in parallel: multiple messages simultaneously circulate in the network. In this paper we consider the parallel monitoring of a graph aimed at not just the graph traversal, but also collection of complete information about the graph in each its node. Another feature of this work is monitoring of a dynamically changing graph: its arcs can disappear, appear or change their destination nodes. An algorithm is proposed, which provides the collection of full information on the graph in each its node.

Keywords: directed graphs; graph exploration, graph traversal, communicating automata, parallel processing, dynamically changed graphs.

DOI: 10.15514/ISPRAS-2015-27(1)-5

For citation: Burdonov Igor, Kosachev Alexander. Monitoring of dynamically changed graph. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 1, 2015, pp. 69-96 (in Russian). DOI: 10.15514/ISPRAS-2015-27(1)-5

References

[1]. Steven S. Skiena. The Algorithm Design Manual. Springer-Verlag, New York, 1997.
[2]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Irredundant Algorithms for Traversing Directed Graphs: The Deterministic Case. Programming and Computer Software, 29(5):245-258, 2003.
[3]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Irredundant Algorithms for Traversing Directed Graphs: The Nondeterministic Case. Programming and Computer Software, 30(1):2-17, 2004.
[4]. M.O. Rabin. Maze Threading Automata. An unpublished lecture presented at MIT and UC. Berkeley, 1967.
[5]. I. B. Burdonov. Traversal of an unknown directed graph by a finite automaton. Programming and Computer Software, 30(4): 11-34, 2004.
[6]. I. B. Burdonov. Backtracking on a tree in traversal of an unknown directed graph by a finite automaton. Programming and Computer Software, 30(6): 6-29, 2004.
[7]. I. B. Burdonov, A. S. Kossatchev, V. V. Kuliamin. Parallel computations on graphs. Programming and computer Software, 41(1): 1-13, 2015.