Игорь Бурдонов, Александр Косачев

Слайд 1. Название доклада: Исследование ориентированного графа коллективом неподвижных автоматов.

Слайд 2. Что было неделю назад: исследование графа двигающимися автоматами.

Неделю назад я рассказывал об исследовании графа коллективом двигающихся автоматов.

Было несколько моделей в зависимости от того

1) неориентированный граф или ориентированный,
2) известный или неизвестный до начала исследования,
3) конечный автомат или нет,
4) если не конечный, то большая у него память или маленькая по сравнению с размером графа,
5) может ли автомат читать и писать в вершины графа или не может,
6) имеется один автомат или много автоматов,
7) детерминированный граф или недетерминированный.

Слайд 3. Одна модель с прошлого раза для сравнения с сегодняшней.

Я повторю параметры только одной модели для сравнения с тем, о чём пойдёт речь сегодня. Эти параметры такие.

Граф ориентированный с выделенной начальной вершиной – *корнем*.

Вначале в корне находится автомат-генератор, который может генерировать другие автоматы.

Каждый сгенерированный автомат может двигаться по дугам графа в направлении ориентации этих дуг, перемещаясь из вершины в вершину.

Для этого дуги, выходящие из вершины, перенумерованы: 1, 2, 3 и так далее. Такой граф называется *упорядоченным*.

В вершине автомат может узнать её полустепень исхода, а для движения по дуге указывает номер дуги.

Кроме того, автоматы могут обмениваться между собой сообщениями по сети связи, независимой от графа и обеспечивающей доставку сообщения от точки к точке.

Для оценки сложности алгоритма временем срабатывания автомата мы пренебрегали, а считали только проходы по дугам и передачу сообщений.

Также мы оценивали память автомата и число автоматов.

Наиболее интересным был случай, когда граф не помещается в память одного автомата, но помещается в суммарную память автоматов.

Слайд 4. Какая модель будет сегодня.

Сегодня автоматы не будут двигаться по графу, они будут неподвижно сидеть в вершинах: в каждой вершине по одному автомату.

Зато обмен сообщениями будет вестись только по дугам графа, то есть каждая дуга – это симплексный канал связи, по которому автомат из начала дуги может передавать сообщения автомату в конце дуги.

Это, в каком-то смысле, более суровая постановка задачи, потому что нет никакой независимой от графа сети связи автоматов, сам исследуемый граф и есть такая сеть связи.

Слайд 5. Два типа графа и две задачи.

Сначала я буду рассказывать про исследование статического детерминированного графа.

Статического, потому что он не меняется в процессе его исследования.
Прошлый раз все графы были статическими.

Потом я расскажу про исследование динамического графа, когда его дуги могут появляться, исчезать и менять свой конец.
Вершины с автоматами остаются на месте.

В обоих случаях рассматриваются две задачи.

Первая задача – это исследование графа с целью узнать, как он устроен.

Вторая задача – это параллельные вычисления на графе.

Слайд 6. Неподвижные автоматы на статическом графе.

Пусть имеется ориентированный упорядоченный граф, в каждой вершине которого находится автомат.

Автомат «знает» только полустепень исхода вершины.

За одно срабатывание автомат принимает сразу все поступившие к нему сообщения по всем входящим дугам и может послать сразу несколько сообщений по выходящим дугам с указанием номеров этих дуг.

Дуга – это буфер сообщений ёмкостью k, то есть на ней может одновременно находиться не более k сообщений.
Если на ней уже k сообщений, ещё не дошедших до автомата конца дуги, то автомат начала дуги не может посылать по этой дуге сообщения.

см. след. страницу

Два замечания о ёмкости дуги и освобождении дуги.

Первое замечание.

Говоря о ёмкости дуги, мы предполагаем, что размер сообщения ограничен сверху.

Очевидно, что посылка k сообщений максимального размера эквивалентна посылке одного большого сообщения, максимальный размер которого в k раз больше.
Отличие лишь в том, что если на дуге меньше k маленьких сообщений, то на неё можно добавлять сообщения.

Но я буду рассказывать об алгоритме, для которого оценка времени работы не меняется по порядку, если такого добавления не делать и не посылать по дуге сообщения, пока она не освободится, а потом сразу послать несколько сообщений, но, конечно, не больше k.

Второе замечание.
Одни сообщения посылаются из автомата вершины сразу по всем выходящим дугам, другие – только по части выходящих дуг, а третьи – только по одной выходящей дуге.

Тем не менее, для простоты мы будем считать, что сообщение ожидает в вершине освобождения сразу всех выходящих дуг.

Понятно, что время работы алгоритма от этого не уменьшится.

Таким образом, для автомата вершины не нужны сигналы об освобождении каждой выходящей дуги, а достаточно общего сигнала об освобождении всех выходящих дуг.

Для оценки мы будем пренебрегать временем срабатывания автомата и считать, что сообщение находится на дуге не более 1 такта, т.е. не более чем через 1 такт после посылки сообщения из автомата начала дуги оно будет принято автоматов конца дуги.

Слайд 7. Разметка статического графа. Постановка задачи.

Задача ставится так: когда извне в автомат корня придёт сообщение, дающее старт работе автоматов, нужно выполнить разметку графа и сообщить об этом в ответном сообщении.

Разметка графа включает в себя:

1) *Прямой остов* графа, ориентированный от корня.
В автомате каждой вершины отмечаются все выходящие *прямые* дуги, т.е. дуги прямого остова. Остальные выходящие дуги – *хорды*.

2) *Обратный остов* графа, ориентированный к корню.
В автомате каждой вершины, кроме корня, хранится номер выходящей *обратной* дуги, т.е. дуги обратного остова. Из вершины может выходить не более одной обратной дуги.

3) В автомате каждой вершины запоминается число входящих обратных дуг.

Такую разметку графа можно понимать как результат исследования графа. Когда разметка сделана, гарантируется, что по каждой дуге передано хотя бы одно сообщение, т.е. совершён обход графа.
Более важно, что такая разметка используется в дальнейшем для параллельных вычислений на графе.

Слайд 8. Разметка статического графа. 3 части алгоритма.

Условно разобьём алгоритм на 3 части:

1) Построение прямого и обратного остова.
2) Определение конца построения.
3) Установка счётчиков числа входящих обратных дуг.

Слайд 9. Разметка статического графа. 1. Построение остовов. Сообщения.

Построение остовов использует 4 типа сообщений:

1. ***Старт*** – идентифицирует вершины графа.
2. ***Поиск*** – ищет пути от вершин к корню.
3. ***Прямое*** – размечает прямой остов.
4. ***Обратное*** – размечает обратный остов.

Слайд 10. Разметка статического графа. 1. Построение остовов. *Старт*.

Сообщение ***Старт*** идентифицирует вершины графа.

Автомат корня получает ***Старт*** извне.

Сообщение ***Старт*** рассылается «веером»: получив первый раз это сообщение, автомат вершины рассылает его по всем выходящим дугам.
Повторный ***Старт*** игнорируется.

Сообщение ***Старт*** накапливает в себе *вектор маршрута*, который оно проходит, как список номеров дуг маршрута.

Идентификатором вершины становится *вектор вершины*, который определяется как вектор маршрута из первого полученного автоматом вершины сообщения ***Старт***.

Слайд 11. Разметка статического графа. 1. Построение остовов. *Поиск*.

Сообщение ***Поиск*** ищет путь от вершины к корню.

Автомат вершины, получив ***Старт***, инициирует ***Поиск***.

Сообщение ***Поиск*** содержит вектор инициатора и накапливает в себе *обратный вектор* = вектор маршрута, который проходит.

Поиск рассылается «веером».

Повторный ***Поиск*** от того же инициатора игнорируется.
Но инициаторов много – это все вершины.

Поэтому, чтобы игнорировать повторные сообщения, автомат вершины хранит множество векторов инициаторов из проходивших через него сообщений ***Поиск***.

Автомат корня, приняв сообщение ***Поиск***, получает вектор инициатора, то есть вектор прямого пути от корня до инициатора, и обратный вектор пути от инициатора до корня. Автомат корня создаёт сообщение ***Прямое***, переписывая в него вектор инициатора и обратный вектор.

Слайд 12. Разметка статического графа. 1. Построение остовов. *Прямое* и *Обратное*.

Сообщение ***Прямое*** размечает прямой путь от корня до инициатора.

Это сообщение двигается по прямому пути от корня до инициатора. Автомат каждой вершины отмечает как прямую ту дугу, по которой посылает сообщение ***Прямое***.

Инициатор, получив сообщение ***Прямое***, создаёт сообщение ***Обратное***, переписывая в него обратный вектор.

Сообщение ***Обратное*** двигается по обратному пути и размечает обратные дуги.

Это сообщение содержит остаток обратного вектора.
Автомат каждой вершины запоминает номер той дуги, по которой посылает сообщение ***Обратное***, как номер обратной дуги, и удаляет этот номер из вектора.
Если раньше был запомнен другой номер, то он забывается.

Слайд 13. Разметка статического графа. 1. Построение остовов. *Обратное*.

Здесь может возникнуть вопрос: почему сообщение ***Обратное*** идёт до корня, а не до вершины, где уже есть обратная дуга?
Ответ: потому что иначе может возникнуть цикл обратных дуг.

Это видно на слайде. Здесь одновременно размечаются два обратных пути от разных инициаторов: красный и чёрный путь. Пунктир – ещё не пройденная часть пути.
Если идём не до корня, то может получиться цикл обратных дуг.

- Если же продолжаем, то старые обратные дуги забываются, а новые возникают.
- И дальше до самого корня получается фрагмент обратного остова.

Слайд 14. Разметка статического графа. 2. Определение конца построения. Сообщения.

Как определить конец построения прямого и обратного остовов?

Идея основана на том, что у дуги есть одно начало и один конец. Считаем дуги по их началу, то есть выходящие дуги, и по их концу, то есть входящие дуги. Оба числа должны, в конце концов, совпасть.

Для этого в автомате корня ведётся подсчёт числа дуг графа.

Сначала счётчик равен числу дуг, выходящих из корня.

Модифицируем сообщение ***Поиск***, добавив в него параметр: число дуг, выходящих из вершины-инициатора.
Это число автомат корня прибавит к счётчику дуг, когда получит сообщение ***Поиск***. Это подсчёт дуг по их началу.

Ещё добавим два новых сообщения: ***Финиш*** и ***Минус***.

Сообщение ***Финиш*** посылается из начала дуги в её конец при получении сообщения ***Прямое*** и заставляет конец дуги учесть эту входящую дугу.

Сообщение ***Минус*** предназначено для того, чтобы вычитать из счётчика дуг число учтённых дуг, входящих в вершину. Это подсчёт дуг по их концу.

Сообщение ***Минус*** посылается по обратному остову до корня.

До того, как появилась обратная дуга, автомат вершины ведёт счётчик учтённых входящих дуг, значение которого и посылает в первом сообщении ***Минус***, когда появляется обратная дуга..

Но и после этого в автомат вершины могут продолжать приходить сообщения ***Финиш*** по ещё неучтённым входящим дугам, которые будут вызывать уже немедленную отправку сообщений ***Минус***.

Слайд 15. Разметка статического графа. 2. Определение конца построения. Диаграмма времени.

Как это всё работает?

Пусть у нас есть две смежные прямые дуги *ab* и *bc*.

Вот здесь на временной диаграмме показаны последовательности сообщений.

Сообщение ***Поиск*** от автомата вершины *a* прибавляет к счётчику дуг в автомате корня единицу для дуги *ab*.
Сообщение ***Поиск*** от автомата вершины *b* прибавляет к этому счётчику единицу для дуги *bc*.
А сообщение ***Минус*** от автомата вершины *b* вычитает из счётчика единицу для дуги *ab*.

Мы видим, что для каждой дуги *ab* вычитание единицы из счётчика происходит не только после прибавления к нему единицы для этой дуги *ab*, но и после прибавления единицы для каждой следующей дуги *bc*.

Из-за этого счётчик дуг в автомате корня обнулится только после того, как прямой и обратный остовы будут построены.

Слайд 16. Разметка статического графа. 3. Установка счётчиков числа входящих обратных дуг.

После того, как мы построили прямой и обратный остовы и определили конец построения, выполняется установка счётчиков входящих обратных дуг.

Для этого используются два сообщения: ***Начало*** и ***Конец*** подсчёта.

Начало создаётся автоматом корня и рассылается по прямому остову.

Получив такое сообщение, автомат вершины создаёт сообщение ***Конец***, которое посылается по обратному остову до корня.

Сначала в сообщении ***Конец*** есть признак «*первая обратная дуга*».

По этому признаку автомат вершины добавляет единицу к своему счётчику входящих обратных дуг, а дальше ***Конец*** пересылается уже без этого признака.

Сообщения ***Конец*** нужны автомату корня, чтобы определить завершение этого этапа, да и всей разметки. Каким образом?

От каждой вершины до корня дойдёт ровно один ***Конец***. Поэтому автомату корня достаточно знать число вершин. А он это знает – это число элементов во множестве инициаторов, которое хранилось в автомате корня, когда передавались сообщения ***Поиск***.

Слайд 17. Разметка статического графа. Оценки.
На этом слайде представлены оценки времени работы алгоритма, и размеры памяти вершины и сообщения.

n – число вершин,
s_{out} – ограничение сверху на полустепень исхода,
D – диаметр графа (длина максимального пути),
k – ёмкость дуги.
Время работы алгоритма $\boldsymbol{O}(n/k + D)$.
Память автомата вершины $\boldsymbol{O}(nD\log s_{out})$.
Размер сообщения $\boldsymbol{O}(D\log s_{out})$.

Слайд 18. Параллельные вычисления на статическом графе. Агрегатные функции.

Теперь мы переходим ко второй задаче – параллельным вычислениям на статическом графе.

Что будет параллельно вычисляться?
Это будет функция от мультимножества значений в вершинах графа.

Можно считать, что эти значения записаны в вершины графа, а автоматы могут их читать.
Или у каждого автомата есть операция – получить значение для той вершины, в которой он находится.
Способ, каким автомат узнаёт значение, приписанное вершине, не важен.

Сначала посмотрим, как устроены те функции, которые мы будем вычислять.

Пусть X – это некоторое базовое множество.
Вершинам будут потом приписываться значения из этого множества X.

Через $X^{\#}$ (X с решёткой) обозначим все конечные мультимножества элементов из X.

Нам нужно мультимножество, потому что разным вершинам могут быть приписаны одинаковые значения.

Функция g на множестве $X^{\#}$ называется *агрегатной*, если существует такая функция e, что значение функции g от объединения двух мультимножеств a и b можно вычислить так: сначала вычисляем g для каждого из этих мультимножеств, а потом к двум полученным значениям применяем функцию e.

Иными словами, агрегатную функцию можно вычислить «по частям».

На слайде приведены примеры агрегатных функций: сумма, минимум и сумма квадратов.

Слайд 19. Параллельные вычисления на статическом графе. Агрегатное расширение функции.

Понятно, что не все функции являются агрегатными.

Например, сумма агрегатна, а среднее арифметическое не агрегатно.

Однако можно использовать агрегатное расширение функции.

Если функцию f можно представить как композицию двух функций h и g, где g – агрегатная функция, то эта функция g называется агрегатным расширением функции f.

Это значит, что мы можем делать вычисления «по частям», используя агрегатное расширение g, а в конце один раз применяем функцию от одного аргумента – функцию h.

Однако возможны такие расширения, которые на практике ничего не дают.
Например, можно взять в качестве g тождественную функцию на $X^{\#}$, а в качестве h – саму функцию f.

Чтобы избежать этого, используется минимальное агрегатное расширение.
Интуитивно, это агрегатная функция, вычисляющая минимум информации, по которой еще можно восстановить f.

На слайде приведено формальное определение минимального агрегатного расширения.
Такое расширение для любой функции f существует и единственно с точностью до изоморфизма.

Слайд 20. Параллельные вычисления на статическом графе. Примеры агрегатных расширений.

Вот здесь на слайде приведены примеры минимальных агрегатных расширений для среднего арифметического, среднего геометрического и среднего квадратичного.

Такая теория агрегатных функций – это модификация теории индуктивных функций.
Здесь на слайде приведена ссылка на литературу.

Слайд 21. Параллельные вычисления на статическом графе. Алгоритм пульсации.

Теперь перейдём к алгоритму вычисления функции.

Мы будем использовать разметку графа, о которой я рассказал.
Заметим, что разметка делается один раз, а потом может использоваться для многократного вычисления различных функций.

Пусть вершинам приписаны значения.

Вычисление начинается, когда автомат корня получает извне сообщение, содержащее три функции: h, g и e.

На практике могут присылаться программы вычисления значений этих функций, или какие-то ссылки на эти программы, по которым автоматы могут получить к ним доступ.

см. след. страницу

Вычисление основано на алгоритме пульсации.

От корня вверх по прямому остову распространяется сообщение ***Вопрос***, содержащее две функции: *g* и *e*.

А вниз по обратному остову распространяется сообщение ***Ответ***, содержащее значение функции *g* от мультимножества значений в вышележащих вершинах обратного остова.

Автомат листовой вершины обратного остова, получив ***Вопрос***, вычисляет функцию *g* от значения, записанного в этой вершине, и посылает его как параметр сообщения ***Ответ*** по обратной дуге.

Автомат нелистовой вершины точно также вычисляет функцию *g* от значения в этой вершине, запоминает его, но не посылает, пока не получит ***Ответы*** по всем входящим обратным дугам.

Число таких дуг мы уже установили в автоматах вершин при разметке графа.

Каждый раз, получив ***Ответ*** по входящей обратной дуге, автомат вершины применяет функцию *e*, параметрами которой будет значение, полученное в сообщении ***Ответ***, и запомненное значение.

Когда получены ***Ответы*** по всем входящим обратным дугам, автомат вершины сам посылает по выходящей обратной дуге ***Ответ*** с накопленным значением функции *g*.

Автомат корня посылает ***Ответ*** вовне, предварительно применив к накопленному значению функции *g* завершающую вычисления функцию *h*.

Время вычисления не более *3D*.

Слайд 22. Параллельные вычисления на статическом графе. Пример: вычисление среднего арифметического.

Вот здесь на слайде пример параллельного вычисления среднего арифметического от значений в вершинах графа.

- Сначала выполняется разметка графа, а потом вычисления с использованием этой разметки.

По окончании вычисления граф готов к новым вычислениям, повторная разметка не требуется.

Слайд 23. Динамический граф. Изменение дуг.

Теперь переходим к динамическим графам.
Динамическим графом будем называть такой граф, дуги которого могут меняться со временем.
При этом мы будем считать, что вершины графа, в которых сидят автоматы, не изменяются.

Есть три вида изменения дуг:

1) Дуга может *появиться*.
Об этом автомат начала дуги извещается специальным сигналом *появление дуги* с параметром номер дуги.

2) Дуга может исчезнуть.
Если по дуге передавалось сообщение, то оно пропадает, о чём автомат начала дуги извещается сигналом *исчезновение дуги* с параметром номер дуги.
Если на дуге не было сообщений, то при её исчезновении никаких сигналов не возникает. Однако если автомат вершины попытается послать сообщение по исчезнувшей выходящей дуге, сообщение не будет передано, а автомат получит сигнал исчезновение дуги.

3) Дуга может поменять свою конечную вершину.
В этом случае никаких сигналов не предусмотрено.

Мы выбрали модель с минимальным набором сигналов, всё ещё позволяющих рано или поздно распознавать изменение дуги.

Слайд 24. Динамический граф. Замещение сигналов.

Кроме указанных, есть ещё сигнал освобождения дуги, означающий, что сообщение, которое по дуге передавалось, уже передано и можно послать новое сообщение.

Мы будем предполагать, что ёмкость дуги равна 1: одновременно по дуге передаётся не более одного сообщения.

Если дуга меняется слишком часто и автомат не успевает обрабатывать сигналы от этой дуги, то могут возникнуть два необработанных сигнала: старый и новый.

В этом случае работает правило замещения, которое говорит, какой из двух сигналов остаётся, а какой теряется.

Замечу, что новым сигналом может быть только сигнал появления дуги.

Для целей мониторинга важно, чтобы после сигнала исчезновения дуги не потерялся сигнал её появления, иначе автомат останется в убеждении, что дуги нет.

Если же после сигнала освобождения дуги возникает сигнал её появления, то это означает, что сообщение передано по дуге, затем дуга исчезла (без сигнала), а потом опять появилась.
Это эквивалентно смене конца дуги.
Поэтому всё равно, какой из двух сигналов остаётся, а какой теряется.

Слайд 25. Динамический граф. Информация в вершине.

Граф будет *нумерованным*: все вершины пронумерованы и в каждой вершине записан её номер, который автомат может прочитать.

Слайд 26.Динамический граф. Постановка задачи мониторинга графа.

Ставится задача мониторинга графа: сбор полной информации о графе и отслеживание изменений дуг.
Мы будем предполагать, что такая информация собирается не в каком-то выделенном автомате, а в каждом автомате.
На самом деле для динамического графа это всё равно.

Понятно, что если граф постоянно меняется, то мы не можем гарантировать, что описания текущего состояния всех его дуг отражены во всех автоматах: сообщения о последних изменениях дуг могут просто не дойти до каких-то автоматов.

Поэтому требуется только, чтобы через некоторое конечное время T_0 после изменения дуги все автоматы «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 автоматах будет одинаковое (и верное) описание этой дуги.

Если дуга меняется так часто, что никакое сообщение по ней не успевает дойти или уходит неизвестно куда, то нельзя гарантировать возможность доставки сообщения в любую вершину. Это нужно как-то ограничить.

Будем говорить, что дуга *долгоживущая*, если за то время, пока она не меняется, по ней может успеть пройти хотя бы одно сообщение.

Мы будем пренебрегать временем срабатывания автомата, а время передачи сообщения по дуге ограничим сверху 1 тактом.
Так что долгоживущая дуга должна не меняться хотя бы в течение 1 такта.

Ограничение, которое нам нужно, звучит так: в каждый момент времени подграф, порождённый долгоживущими дугами, является сильно связным суграфом, то есть содержит все вершины графа.

Слайд 27. Мониторинг динамического графа. Описания дуг и сообщения.

Полная информация о графе, которую нужно собрать, это множество описаний всех его дуг.

Описание дуги – это тройка: номер начала дуги, то есть начальной вершины дуги, номер дуги (в её начале) и номер конца дуги, то есть конечной вершины дуги.

Если номер конца дуги ещё не известен, то указывается номер 0. Мы считаем, что вершины пронумерованы, начиная с 1.

Сразу можно сказать, что в одном сообщении автомат должен посылать сразу описания всех известных ему дуг.
Это *правило одного сообщения*.

Оно объясняется тем, что по долгоживущей дуге гарантированно проходит только одно сообщение.
Поэтому в него надо поместить всю имеющуюся информацию.

Во-вторых, сообщение нужно посылать по дуге не тогда, когда информация в автомате обновилась, а каждый раз, когда появляется такая возможность, то есть при появлении дуги и при освобождении дуги.
В противном случае информация от автомата данной вершины может не достигнуть других автоматов, поскольку выходящая из вершины дуга может несколько раз менять свой конец без какого-либо сигнала для автомата этой вершины.

см. след. страницу

Кто и когда обнаруживает изменение дуги, в результате чего создаёт или корректирует описание дуги?

Если дуга появляется, то это обнаруживает автомат в начале дуги по сигналу появления дуги.
Но он ещё не знает конца дуги.

Если дуга исчезает, то это обнаружит тоже автомат начала дуги по сигналу исчезновения дуги.
Этот сигнал он получит либо сразу, если по дуге передавалось сообщение, либо позже, когда при обработке предыдущего сигнала освобождения дуги попытается послать по дуге сообщение.

Если дуга меняет свой конец, то это обнаруживает автомат нового конца дуги, когда получает по дуге сообщение, в котором описание дуги содержит другой номер конца.
В частности, номер конца будет нулевым, если автомат начала дуги ещё не знает конца дуги, то есть сразу после появления дуги.

Для того, чтобы автомат конца дуги мог это делать, он должен знать, по какой дуге передавалось сообщение.

Для этого сообщение, кроме множества описаний дуг, содержит указание на описание дуги, по которой оно передаётся.

Слайд 28.Мониторинг динамического графа. Ранг (описания) дуги.

Поскольку дуга может меняться несколько раз, автомат, получая сообщение, должен понять, получил ли он описание дуги более новое, чем то, что у него есть, или нет.

Новое описание дуги должно заменять собой старое описание.

Для этого вводится ранг описания дуги, который я буду называть просто рангом дуги.

Каждый раз, когда кто-то обнаруживает изменение дуги и поэтому создаёт или корректирует её описание, ранг дуги в этом описании увеличивается.

Однако недостаточно просто увеличить ранг дуги, например, на 1.

Нужно быть уверенным, что после такого увеличения в любом другом автомате дуга имеет строго меньший ранг.

Здесь возникает проблема из-за серии изменений конца дуги.

Автоматы этих сменяющих друг друга концов дуги увеличивают один и тот же ранг, естественно, одинаково, например, на 1.

Поэтому сразу в нескольких автоматах эта дуга будет иметь одинаковый, увеличенный на 1, ранг.

Эту проблему решает автомат начала дуги, который оставляет последнее слово всегда за собой.

Каждый раз, когда он получает описание дуги с большим рангом, чем у него хранится, он ещё раз увеличивает ранг дуги на 1.

Соответственно, при появлении или исчезновении дуги автомат начала дуги увеличивает её ранг сразу на 2.

Слайд 29. Мониторинг динамического графа. Оценки.

Все оценки приведены на слайде.

Время T_0 имеет порядок числа вершин.

Но если все изменения в графе прекращаются, то время, которое нужно, чтобы во всех автоматах оказалась правильная информация, учитывающая последние изменения, ещё меньше – порядка диаметра графа.

Размер памяти автомата довольно большой, но это и естественно, поскольку в памяти автомата создаётся описание всего графа.
Это же описание, по сути, передаётся в сообщении.

Можно обратить внимание, что размер памяти зависит от максимального числа изменений одной дуги.
Это затраты на ранг дуги.

Для того, чтобы оценка памяти не зависела от числа изменений, можно предложить две модификации модели.

n – число вершин графа, m – число дуг графа, D – диаметр графа (длина максимального пути), s – ограничение сверху на полустепень исхода вершин, v – максимальное число изменений одной дуги.

Время $T_0 = \boldsymbol{O}(n)$.
Время после прекращения изменений в графе $T_1 = \boldsymbol{O}(D)$.
Размер памяти автомата $\boldsymbol{O}(mlogs)+\boldsymbol{O}(mlogn)+\boldsymbol{O}(mlogv)$.
Размер сообщения такой же.

Слайд 30. Мониторинг динамического графа. Модификации.

Первая модификация самая простая: потребуем, чтобы все дуги были долгоживущими.

Тогда число изменений одной дуги не превосходит времени работы алгоритма в тактах, а оно имеет порядок числа вершин.

Следовательно, ранг можно сделать циклическим с максимальным значением порядка числа вершин.

Вторая модификация – наличие таймера, который посылает каждому автомату временной сигнал каждый такт.

Тогда все те действия, что автомат выполнял при обработке сигнала освобождения или появления дуги, он будет делать при обработке временного сигнала.

Сообщения будут посылаться по дуге не чаще одного за такт/

При этом время T_0 не изменится по порядку.

А тогда опять ранг можно сделать циклическим с максимальным значением порядка числа вершин.

Правда, при этой модификации время существования «долгоживущих» дуг придётся увеличить с одного до двух тактов.

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.

Слайд 31. Параллельные вычисления на динамическом графе.

Теперь мы переходим к параллельным вычислениям на динамическом графе.

Нам тоже потребуется некоторая предварительная разметка графа. Мониторинг не даёт такой разметки – у него вообще другая задача.

На статическом графе вычисления использовали такую разметку графа: прямой и обратный остовы, плюс счётчики обратных дуг.

Но если граф динамически меняется, возникает проблема: когда меняется дуга остова, нужно этот остов скорректировать.

В общем случае это может потребовать полной перестройки остовов, т.е. фактически придётся заново их строить.

Кроме того, не гарантируется доставка сообщений по дугам остова: эти дуги могут исчезать или менять свой конец до того, как по ним будет передано нужное сообщение.

В то же время, как мы видели в случае мониторинга графа, предположение о долгоживущих дугах гарантирует возможность доставки информации из любой вершины в любую вершину.

Но для этого автомат вершины должен следовать *правилу одного сообщения*.
Получив сообщение с нужной информацией, он сохраняет её в своей памяти и далее помещает во все сообщения, которые постоянно посылает по всем выходящим дугам, когда появляется такая возможность, то есть по сигналам освобождения и появления дуги.

Для доставки ***Вопроса*** этого достаточно, и никакого прямого дерева не требуется.

см. след. страницу

Точно также можно было бы доставить в корень значения из всех вершин графа и уже в автомате корня выполнить вычисление функции от получившегося мультимножества.

Однако, поскольку мы всю информацию вынуждены помещать в одно сообщение, размер такого сообщения был бы максимальным и пропорциональным числу вершин графа.

Обратный остов и счётчики обратных дуг позволяют аккумулировать ответы: автомат вершины, получив ответы по всем входящим обратным дугам, вычисляет промежуточное значение агрегатной функции и посылает дальше по выходящей обратной дуге только один ответ.

Тем самым, сообщение содержит ответ только от автомата одной вершины каждой ветви обратного остова от корня до листа.

Размер сообщения становится пропорциональным не числу вершин, а ширине обратного остова.

Итак, обратный остов и счётчики обратных дуг нам нужен только для уменьшения размера сообщения, содержащего ответы, то есть вычисленные промежуточные значения агрегатной функции.

Такой остов может быть виртуальным. Вершины в нём настоящие, а дуги – виртуальные.

Дуга задаётся просто парой её вершин: начальной и конечной.

Поскольку остов виртуальный, у нас появляется свобода: какой остов лучше построить.

Его ширина определяет размер сообщения, а высота – время вычисления, поскольку вычисления распараллеливаются только между разными ветвями остова, а по одной ветви выполняются последовательно от листа к корню.

Слайд 32. Параллельные вычисления на динамическом графе. Сбалансированный веник.

Возникает задача: для данного числа вершин определить минимальную высоту дерева при заданной ширине, и описать вид такого дерева.

Оно будет выглядеть как «сбалансированный веник» на слайде.

Слайд 33. Параллельные вычисления на динамическом графе. Разметка: индексы вершин и счётчики обратных дуг.

Веник можно задать с помощью двухуровневых индексов вершин, отличных от корня.

Индекс вершины состоит из номера ветви веника от 1 до w и номера на этой ветви от 1 до h, считая от корня.

В таком венике счётчик входящих обратных дуг нужен только в автомате корня, его значение равно w.

Автомату другой вершины достаточно «знать», является вершина листовой (счётчик равен 0) или внутренней (счётчик равен 1) вершиной веника.

Слайд 34. Параллельные вычисления на динамическом графе. Алгоритм разметки: 2 этапа.

Разметка выполняется в 2 этапа:

1) Сначала в автомате корня собирается информация о всех вершинах. Используется сообщение ***Прямое***.

2) Затем автомат корня строит в своей памяти веник и рассылает его во все вершины. Используется сообщение ***Обратное***.

Слайд 35.Параллельные вычисления на динамическом графе. 1. Сбор информации о вершинах. Статическая замкнутость.

Как автомат корня собирает информацию о вершинах?

Он создаёт сообщение ***Прямое***, которое потом циркулирует по всему графу с помощью постоянной веерной рассылки.

В этом сообщении и в автоматах вершин, через которые оно проходит, накапливаются описания дуг.

Описание дуги содержит номер начальной вершины дуги, номер дуги в её начальной вершине и номер конечной вершины дуги, если он известен, или знак вопрос, если неизвестен.

Долгоживующие дуги гарантирует доставку сообщения до любой вершины, и из любой вершины до корня.

Автомат корня проверяет *замкнутость по дугам*: нет знаков вопроса, то есть для каждой дуги известна конечная вершина этой дуги.

Однако замкнутость гарантирует, что все вершины известны, только для статического графа.

В динамическом графе конец дуги может меняться без всяких сигналов, поэтому замкнутость по дугам ничего не гарантирует.

Слайд 36. Параллельные вычисления на динамическом графе. 1. Сбор информации о вершинах. Статическая замкнутость.

Вот здесь на слайде пример.

Автомат корня, вершины *a*, посылает сообщение № 1, в котором одна дуга с неизвестным концом.

Автомат вершины *b*, получив сообщение, записывает вершину *b* как конец этой дуги и рассылает по двум выходящим дугам сообщение № 2, в котором неизвестны концы этих двух дуг.
Одно направляется в корень *a*, а другое – в вершину *c*.

- Однако две красные дуги меняют свои концы, поэтому оба сообщения № 2 приходят в корень.

И автомат корня, как и положено, записывает корень как конечную вершину этих дуг.

В результате в автомате корня получается замкнутое множество дуг, однако вершина *c* осталась неизвестной.

Слайд 37. Параллельные вычисления на динамическом графе. 1. Сбор информации о вершинах. Два условия.

Для того, чтобы замкнутость гарантировала известность всех вершин, нам требуются два дополнительных условия.

Первое – это правило замещения сигналов.
Нам будет важно знать, дошло сообщение по дуге или нет.

- Поэтому сигнал освобождения не должен теряться.

Во-вторых, нам понадобятся начальные дуги, которые обладают двумя свойствами:

1) Начальная дуга существует с самого начала и не меняется до тех пор, пока по ней не пройдёт первое сообщение.

2) По начальным дугам из корня достижимы все вершины.

Слайд 38.Параллельные вычисления на динамическом графе. 1. Сбор информации о вершинах. Динамическая замкнутость.

Теперь у нас будет модифицированное условие замкнутости:
для каждой известной дуги:

- либо известна конечная вершина этой дуги (по дуге послано первое сообщение, оно дошло до конца дуги и вернулось к корню),
- либо по этой дуге не прошло первое сообщение (дуга гарантированно не начальная).

Важно, что каждая вершина регистрирует только те выходящие из неё дуги, которые появились до получения вершиной первого сообщения. А те, что появляются позже, не регистрирует.

Почему?

Во-первых, этого достаточно, потому что все начальные дуги будут зарегистрированы, так как появляются с самого начала.

Во-вторых, это важно для минимизации времени сбора информации о вершинах.
Оно будет порядка числа вершин n.
Если же регистрировать все дуги, то их число равно m.
Дуги могут всё время появляться, и, если они продолжают регистрироваться, то замкнутость в корне может возникнуть после появления всех m дуг, потому что для каждой появившейся дуги придётся ждать выяснения, какой же у неё конец.

Пример: a корень, дуги ab, ba и много петель bb. Каждый раз появляется одна петля, т.е. когда в корень приходит информация о конце i-ой петли, к нему же приходит информация о появлении $i+1$-ой петли, но без конца. Поэтому корень ждёт, пока появятся все m петель.

Слайд 39.Параллельные вычисления на динамическом графе. Идентификация вершин.

Можно обойтись без нумерованности графа, если использовать идентификацию вершин с помощью динамически создаваемых векторов вершин, как мы это делали для разметки статического графа.

Для этого используется сообщение ***Старт***, которое накапливает в себе вектор маршрута, который оно проходит.
Корень получает ***Старт*** извне с пустым вектором.

Автомат каждой вершины, получив первый раз сообщение ***Старт***, запоминает его вектор маршрута как вектор вершины и посылает ***Старт*** по всем выходящим дугам. Регистрация выходящих дуг прекращается.
Повторные сообщения ***Старт*** игнорируются.

Начальные дуги гарантируют, что каждая вершина получит сообщение ***Старт*** и, тем самым, получит свой вектор.
Поскольку по каждой дуге посылается не более одного сообщения ***Старт***, вектор вершины будет уникальным в графе.

Только после этого автомат вершины начинает работать с сообщением ***Прямое*** так, как я рассказал:

Каждый раз, когда дуга освобождается или появляется, по ней посылается сообщение ***Прямое*** с накопленным множеством описаний дуг.

При получении сообщения ***Прямое*** множество описаний дуг из сообщения объединяется с множеством описаний дуг, хранящимся в памяти автомата.
При этом вектор вершины-получателя вставляется в описание дуги, по которой принято сообщение, как вектор конца дуги.

Слайд 40. Параллельные вычисления на динамическом графе. 2. Рассылка описания виртуального обратного остова.

Когда автомату корня становятся известны все вершины, он создаёт описание виртуального веника, которое рассылается веером по всему графу в сообщении ***Обратное***.

Описание веника – это множество описаний вершин.

Для каждой вершины по её номеру указывается её индекс в венике и признак: листовая или не листовая.

Когда автомат вершины первый раз получает сообщение с описанием веника, он запоминает отдельно описание своей вершины, удаляет его из описания веника и такое уменьшенное описание веника сохраняет и рассылает дальше.

При получении повторного сообщения автомат вершины строит пересечение множества описаний вершин, которое хранится в нём, и которое содержится в сообщении.

Это пересечение сохраняется и рассылается дальше.

Рано или поздно сообщение становится пустым и через какое-то время описание веника в автомате корня тоже становится пустым.

Это и означает завершение 2-го этапа разметки.

Далее система готова к параллельным вычислениям.

Слайд 41. Параллельные вычисления на динамическом графе. *Вопрос*ы и *Ответ*ы.

По правилу одного сообщения вся нужная информация должна рассылаться по графу в одном сообщении, распространяемом веером с размножением в каждой вершине.
Поэтому на этапе вычисления функции сообщение содержит как вопрос, так и ответы.

При этом для каждой ветви веника в сообщении не больше одного ответа, а именно ответа от автомата вершины с наименьшим номером по ветви.
Ответ задаётся вместе с индексом отправителя.

После завершения вычисления по графу может продолжать циркулировать такое сообщение ***Вопрос-Ответ***.

Оно должно вытесняться сообщением для следующего вычисления.

Для этого автомат корня нумерует вопросы в порядке их поступления извне и номер вопроса помещается в сообщение.

Слайд 42. Параллельные вычисления на динамическом графе. Оценки.

На слайде показаны оценки времени и памяти на этапе разметки и на этапе вычисления.

Слайд 43. Спасибо за внимание !

Слайд 44. Ссылки.

Автоматы неподвижны, сообщения по дугам	
doc, pdf, pdf_1	Параллельные вычисления на динамически меняющемся графе. Труды Института системного программирования РАН, том 27-2, 2015 г., ISSN 2079-8156, с. 189-220. 32 стр.
doc, pdf, pdf_1	Мониторинг динамически меняющегося графа. Труды Института системного программирования РАН, том 27-1, 2015 г., ISSN 2079-8156, с. 69-96. 28 стр.
doc, pdf, pdf_1	Параллельные вычисления на графе. "Программирование", 2015, №1, стр. 3-20. 18 стр.
doc, pdf	Параллельные вычисления автоматами на прямом и обратном остовах графа. Труды Института системного программирования РАН Том 26-6. 2014 г., стр. 63-66. 4 стр.
doc, pdf	Построение прямого и обратного остовов автоматами на графе. Труды Института системного программирования РАН Том 26-6. 2014 г., стр. 57-62. 6 стр.
Автоматы двигаются, сообщения по отдельной системе связи	
doc, pdf, pdf_1	Исследование графа набором автоматов. "Программирование", -2015, №6–стр. 3-8. 6 стр.
doc, pdf, pdf_1	Обход неизвестного графа коллективом автоматов. Недетерминированный случай. Труды Института системного программирования РАН, том 27-1, 2015 г., ISSN 2079-8156, с. 51-68. 18 стр.
doc, pdf, pdf_1	Исследование графа взаимодействующими автоматами. «Вестник Томского государственного университета. Управление, вычислительная техника и информатика», №3, 2014, стр. 67-75. 9 стр.
doc, pdf, pdf_1, ppt	Исследование графа взаимодействующими автоматами. Новые информационные технологии в исследовании сложных структур. Материалы 10-ой российской конференции с международным участием. 2014, изд. Томского госуниверситета, стр.47-48. 2 стр.
doc, pdf, pdf_1	Обход неизвестного графа коллективом автоматов. Труды Института системного программирования РАН, том 26-2, 2014 г., ISSN 2079-8156, с. 43-86. 44 стр.
doc, pdf, pdf_1, ppt	Обход неизвестного графа коллективом автоматов. Труды Международной суперкомпьютерной конференции "Научный сервис в сети Интернет: все грани параллелизма". 2013, изд. МГУ, стр. 228-232. 5 стр.
Тестирование распределенное и параллельное	
doc_1, pdf_1	Параллельное тестирование больших автоматных моделей. Вестник ННГУ, №3(2), 2011 г., стр. 187-193. 7 стр.

doc, pdf, pdf_1	Параллельное тестирование больших автоматных моделей. Научный сервис в сети Интернет: суперкомпьютерные центры и задачи: Труды Всероссийской суперкомпьютерной конференции (21-24 сентября 2010 г., г. Новороссийск). - М.: Изд-во МГУ, 2010, с. 224-228. 5 стр.
Робот	
doc, pdf, ppt	Исследование одно/двунаправленных распределённых сетей конечным роботом. Труды Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ". 2004, стр.53-54. 2 стр.
doc, pdf	Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. "Программирование",-2004, №6, стр.6-29. 24 стр.
doc, pdf	Обход неизвестного ориентированного графа конечным роботом. "Программирование",-2004, №4, стр.11-34. 24 стр.
Обход графа	
doc, pdf	Неизбыточные алгоритмы обхода ориентированных графов. Недетерминированный случай. "Программирование",-2004. №1, стр.2-17. 16 стр.
html,doc, pdf, pdf_1	Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. "Программирование".-2003, №5.- стр.59-69. 11 стр.