УДК 681.324

Ю.Ю. ЗАВИЗИСТУП, С.А.ПАРТЫКА

МЕТОД ПРЕОБРАЗОВАНИЯ СТАТИЧЕСКИХ SPT АЛГОРИТМОВ В ДИНАМИЧЕСКИЕ

Предлагается метод трансформации статических алгоритмов поиска кратчайших путей на графах в динамические. Отличительной особенностью данного метода является поддержка специальных структур данных для коррекции текущих значений длин маршрутов. Приводятся зависимости числа операций сравнения от количества вершин графа для статических и динамических алгоритмов. Проводится краткий анализ полученных результатов.

1. Введение

В современной телекоммуникационной индустрии множество транспортных проблем, в частности проблем маршрутизации, нуждаются в очень гибких и эффективных решениях задач, связанных с поиском кратчайших путей на графах. При этом ключевым показателем эффективности решения задачи является, как правило, значение вычислительной сложности, с точки зрения продолжительности работы алгоритмов, количества операций сравнения, а также системных требований к оперативной памяти.

Непрерывный исследовательский интерес к проблемам маршрутизации в сетях передачи данных и работе алгоритмов SPT, в частности, сохраняется в течение последних десятилетий [1-3], и что частично мотивируется проблемой неустойчивости маршрутизации в сети Интернет [5].

Основной недостаток используемых в процессе маршрутизации статических алгоритмов заключается в необходимости полного пересчета дерева кратчайших путей (SPT) в случае изменения топологии сети. Фактически после некоторых изменений в состояниях связей топология нового SPT отличается от старого не значительно. Поэтому статические алгоритмы, которые повторно вычисляют SPT, явно неэффективны.

Цель исследования – преобразование статических SPT алгоритмов с организацией очередей в виде связанных списков в динамические для снижения алгоритмической сложности путем использования доступной информации SPT.

2. Метод преобразования алгоритмов

Пусть $G = (V, \varepsilon)$ обозначает ориентированный граф, где V – набор узлов, а E – набор ребер в графе. Пусть N – общее количество узлов в V и E – общее количество ребер в ε. Граф G состоит из корневого узла, обозначенного n_0, и всех других узлов $n_1,...,n_{N-1}$, доступных из n_0 по ориентированным дугам в G. Каждое ребро e_i $(i = 0,...,E-1)$ имеет положительный вес ω_i.

Для каждой дуги $e \in E$, пусть $W(e)$ обозначает вес (расстояние), ассоциированный с e, $S(e)$ и $E(e)$ обозначают соответственно начальный и конечный узел ребра e. Длина или

расстояние направленного пути – сумма весов ребер в пути. Дан ряд узлов $N \subseteq V$, который ассоциируется с двумя наборами ребер графа: $I(N) = \{e \in \varepsilon | E(e) \in N\}$ (набор ребер, направленных в узлы N) и $O(N) = \{e \in \varepsilon | S(e) \in N\}$ (набор ребер, направленных из узлов N). Пусть n_0 определяет корневой узел SPT.

Пусть древовидная структура данных T поддерживается алгоритмом для того, чтобы следить за существующим деревом кратчайших путей. Пусть каждый узел графа G, наряду с его родительским атрибутом $P(n,T)$, имеет список потомков $C(n,T)$, а также его атрибут расстояния $D(n,T)$. Структура данных T изменяется прогрессивно в течение вычисления, и когда работа алгоритма завершена, представляет собой SPT.

В дополнение к структуре данных T алгоритм также поддерживает связанный список Q, который временно содержит подмножество узлов вместе с двумя признаками. Каждый элемент в Q имеет форму $\{n,(x,D)\}$, где x определяет потенциального родителя для узла n, а D определяет потенциальное расстояние к узлу n.

Инструкция $ENQUEUE(Q\{n,(x,D)\})$ добавляет еще один элемент к Q. Если узел уже присутствует в очереди, то новые атрибуты заменяют старые. В любой момент только один набор атрибутов доступен для каждого узла в Q. Когда выполняется инструкция $EXTRACT(Q)$, выбирается первый (верхний) элемент и удаляется из очереди.

Алгоритм также использует вспомогательную процедуру. Процедура $B_{child}(n,T)$ находит множество узлов (включая n), которые являются потомками n. Это множество можно легко вычислить, спускаясь непосредственно вниз по T. SPT строится в T, изменяя атрибуты узлов в течение каждой итерации алгоритма.

Этот метод может быть использован в нескольких известных статических алгоритмах, например, таких как D'Esopo-Pape, Беллмана-Форда или Дейкстры. Основное различие в работе данных алгоритмов заключается в том, каким способом происходит организация временного связанного списка Q.

3. Спецификация алгоритма

Шаг 1: Инициализация

(A) Статическая версия

$\forall (n \in V) \neq root(G)$

$P(n,T) \leftarrow (O)$

$D(n,T) \leftarrow MaxInt$

$C(n,T) \leftarrow (O)$

$ENQUEUE(Q\{root(G),(O,D)\})$

Go To Step 2

(B) Динамическая версия (при наличии старого SPT)

Случай 1 (Ребро $e_i \in \varepsilon^+$ увеличивает свой вес на Δ_i):

$\hat{N} \leftarrow 0$

$\forall e_i \in \varepsilon^+$

$W(e_i) \leftarrow W(e_i) + \Delta_i$

if $S(e_i) = P(E(e_i),T)$

$N \leftarrow B_{child}(E(e_i),T)$

$\forall n \in N$

$D(n,T) = D(S(e),T) + W(e)$

$\hat{N} \leftarrow \hat{N} \cup N$

$\forall e \in I(\hat{N})$

if $D(E(e),T) > D(S(e),T) + W(e)$

$D(E(e),T) = D(S(e),T) + W(e)$

$C(S(e),T) \leftarrow C(S(e),T) + \{E(e)\}$

$C(P(E(e),T),T) \leftarrow C(P(E(e),T),T) - \{E(e)\}$

$P(E(e),T) \leftarrow S(e)$

$ENQUEUE(Q\{E(e),(S(e),D)\})$

Go To Step 2

Случай 2 (Ребро $e_i \in \varepsilon^-$ уменьшает свой вес на Δ_i):

$\forall e_i \in \varepsilon^-$

$W(e_i) \leftarrow W(e_i) - \Delta_i$

if $D(E(e_i),T) > D(S(e_i),T) + W(e_i)$

$C(S(e_i),T) \leftarrow C(S(e_i),T) + \{E(e_i)\}$

$C(P(E(e_i),T),T) \leftarrow C(P(E(e_i),T),T) - \{E(e_i)\}$

$P(E(e_i),T) \leftarrow S(e_i)$

$N \leftarrow B_{child}(E(e_i),T)$

$\forall n \in N$

$D(n,T) = D(S(e),T) + W(e)$

$ENQUEUE(Q\{E(e),(S(e),D)\})$

Go To Step 2

Шаг 2: Извлечение вершины

if $Q = O$

Terminate

else

$\{y,(x,D)\} \leftarrow EXTRACT(Q)$

Шаг 3: Обновление расстояний

$\forall e \in O(y)$

if $D(E(e),T) > D(y,T) + W(e)$

$D(E(e),T) = D(y,T) + W(e)$

$C(y,T) \leftarrow C(y,T) + \{E(e)\}$

$C(P(E(e),T),T) \leftarrow C(P(E(e),T),T) - \{E(e)\}$

$P(E(e),T) \leftarrow y$

$ENQUEUE(Q\{E(e),y,D)\})$

Go To Step 2

Во время выполнения фазы инициализации производятся начальные установки для статической версии алгоритма. В дальнейшем также корректируются значения атрибутов, затронутых изменениями весов ребер графа узлов для динамической версии. Предполагается, что в существующем (устаревшем) SPT все узлы имеют родительский атрибут $P(n,T)$, имеют список потомков $C(n,T)$, а также атрибут расстояния $D(n,T)$. Алгоритм сначала обновляет SPT со всеми ребрами, веса которых увеличиваются (случай 1), а затем обновляет SPT со всеми ребрами, веса которых уменьшаются (случай 2).

В случае 1, после обновления каждого ребра e_i с новым увеличенным весом, проверяется, находится ли ребро в существующем SPT. Если условие выполняется, то выбирается узел $E(e_i)$ и все его потомки в существующем SPT. Все такие узлы включаются процеду-

рой $B_{child}(n,T)$ в множество N для дальнейшего обновления их расстояний. Для каждого ребра e, направленного в узел n, принадлежащий множеству N, производится сравнение текущего атрибута расстояния $D(n,T)$ с потенциально возможным расстоянием через $S(e_i)$. В случае, если потенциально возможное расстояние имеет меньшее значение, чем $D(n,T)$, происходит коррекция атрибута $D(n,T)$, узел n удаляется из списка потомков $P(n,T)$ и добавляется к списку потомков $S(e)$. Также изменяется родительский атрибут $P(n,T)$ на $S(e)$ и выполняется инструкция ENQUEUE.

Случай 2 на шаге инициализации более прост. После обновления каждого ребра e_i с новым уменьшенным весом, подобно случаю 1, производится сравнение текущего атрибута расстояния $D(n,T)$ с потенциально возможным, и если выполняется условие неравенства, с помощью процедуры $B_{child}(n,T)$ формируется множество N. Для всех узлов из N выполняется коррекция атрибутов $D(n,T)$, а также инструкция ENQUEUE.

После шага инициализации на шаге 2 из списка Q извлекается первый элемент. Как было замечено ранее, формирование связанного списка Q происходит в соответствии с используемым статическим алгоритмом.

На шаге 3 рассматривается каждый узел $E(e_i)$, который присоединен к ребру e_i, направленному из узла y. Если потенциально новое расстояние $E(e_i)$ (которое является равным $W(e_i)$ плюс расстояние $D(y,T)$) является меньшим, чем его старое значение, то происходит соответствующее корректирование атрибутов и $E(e_i)$ ставится в очередь в список Q. После того, как этот шаг закончен, выполнение шагов 2 и 3 повторяется до тех пор, пока список Q не будет пуст.

4. Методика моделирования и анализ полученных результатов

Было проведено экспериментальное моделирование в целях подтверждения теоретических выкладок, изложенных выше, проверки эффективности предложенного метода преобразования статических алгоритмов поиска кратчайших путей на графах в динамические, а также сравнительный анализ работы алгоритмов с точки зрения количества операция сравнения.

В ходе проведенного моделирования производилась генерация графов со случайной топологией в диапазоне числа вершин от 100 до 2000 с шагом в 100 вершин по методике, указанной в [4]. Средняя степень вершин составляла 8. Веса ребер графа устанавливались равными евклидовому расстоянию между вершинами:

$$L_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

где x_i, x_j, y_i, y_j – соответствующие координаты вершин i и j.

Для каждого значения числа вершин графа генерировалось 50 различных топологий. Для статических алгоритмов на графе 300 раз случайным образом выбиралась начальная вершина. Для динамических версий алгоритмов по каждой топологии графа 300 раз случайным образом менялись наборы весов ребер (число ребер, одновременно меняющих свой вес, находилось в интервале 5 – 30).

На рис. 1 показаны полученные зависимости числа операций сравнения от количества вершин графа для статических алгоритмов.

Из графиков, приведенных на рис.1, следует, что алгоритмы Беллмана-Форда и D'Esopo-Pape, имеющие большую асимптотическую сложность, на практике работают намного эффективнее алгоритма Дейкстры, что связанно с проведением моделирования на разреженных графах. Полученные результаты указывают на то, что D'Esopo-Pape имеет явные преимущества перед алгоритмами Беллмана-Форда и Дейкстры.

В случае изменения состояния связей (весов ребер графа), как и ожидалось, за счет поддержки древовидной структуры T происходит локализация областей графа, для которых требуется повторное построение SPT. Стоит обратить внимание на то, что данный подход позволяет добиться значительного снижения алгоритмической сложности (рис. 2). Кроме того, в отличие от статических алгоритмов предложенный метод приводит к динамическим алгоритмам с сублинейной сложностью относительно размера сети.

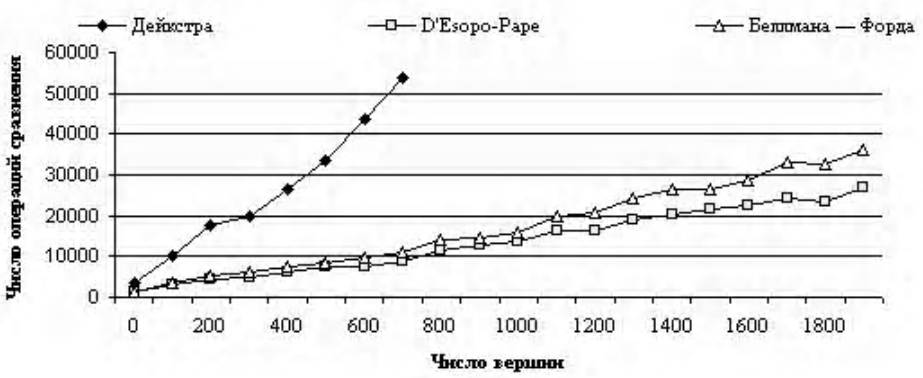

Рис. 1. Трудоемкость статических алгоритмов

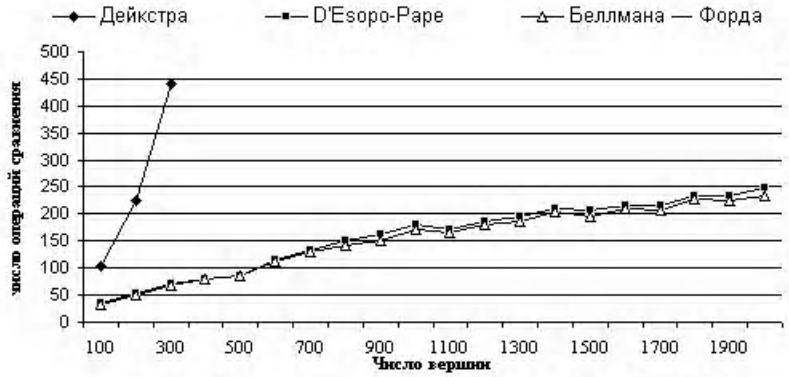

Рис. 2. Трудоемкость динамических алгоритмов при изменяющейся топологии графа

5. Заключение

Научная новизна. Для преобразования статических версий SPT алгоритмов в динамические предложен метод, позволяющий добиться значительного снижения алгоритмической сложности. Приведенные иллюстрации показывают существенное сокращение вычислительных затрат при использовании динамических версий алгоритмов.

Практическое значение. Показано, что использование динамических алгоритмов исключает необходимость полного пересчета дерева кратчайших путей в случае изменения топологии сети за счет использования доступной информации из устаревшего SPT. Таким образом, существует возможность значительного снижения вычислительной нагрузки на процессор маршрутизатора.

Список литературы: 1.*Narvaez P., Kai-Yeung Siu, Hong-Yi Tzeng*. New Dynamic SPT Algorithm Based on a Ball-and-String Model. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 9, DECEMBER 2001. N 6. P. 706-718. **2.** *Breslau L., Estrin D.* Design of inter-administrative domain routing protocolsn // Proc. SIGCOMM, Sept. 1990. P. 231–241. **3.** *Perlman R., Varghese G.* Pitfalls in the design of distributed routing algorithms, in *Proc. SIGCOMM*, vol. Aug., 1988. P. 43–54. **4**. *Агеев Д. В.* Модернизированная методика синтеза начальной структуры транспортной сети передачи данных. Одесса:Труды УНДИРТ, 2001, No 2(26). С.42-47. **5.** *Dilworth S., Kutzarova D., Shuman K., Temlyakov V.N., Wojtaszchyk P.* Weak Convergence of Greedy Algorithms in Banach spaces // Journal of Fourier Analysis and Applications. 2008. Vol. 14. P. 608–629.

Поступила в редколлегию 04.12.2010

Завизиступ Юрий Юрьевич, канд. техн. наук, проф. кафедры ЭВМ ХНУРЭ. Научные интересы: компьютерные сети, надежность и живучесть компьютерных систем. Увлечения: история науки и техники, литература. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 80577021354. E-mail: zaviz@kture.kharkov.ua

Партыка Станислав Александрович, ассистент кафедры ЭВМ ХНУРЭ. Научные интересы: компьютерные сети, математическое моделирование. Увлечения: путешествия, туризм. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 80577021354. E-mail: stas_partyka@mail.ru