УДК 681.324

С.А.ПАРТЫКА

МЕТОД УСКОРЕННОЙ КОРРЕКЦИИ SPT С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКИХ АЛГОРИТМОВ

Предлагается модификация алгоритма D'Esopo-Pape с применением метода трансформации статических SPT алгоритмов в динамические. Отличительной особенностью данного метода является поддержка специальных структур данных для коррекции текущих значений длин маршрутов. Описываются зависимости числа операций сравнения от количества вершин графа для статических и динамических алгоритмов. Приводится краткий анализ полученных результатов.

1. Введение

В современной телекоммуникационной индустрии существует множество транспортных проблем, в частности проблем маршрутизации [1-3]. Загрузка и пропускная способность линий связи современных компьютерных сетей динамически меняются, что, в свою очередь, может приводить к относительно частой рассылке служебной информации об изменении маршрутов. Традиционно применяемые методы маршрутизации оказываются неэффективными. Изменения характеристик каналов связи, модификация структуры сети, включение в нее новых узлов и линий связи приводят к полному пересчету таблиц маршрутизации. Когда топология сети в области маршрутизации изменяется (например, происходит отказ канала или изменяется его стоимость), каждый маршрутизатор области уведомляется об изменении. После модификации базы данных состояния каналов каждый маршрутизатор повторно вычисляет свой SPT. В современных маршрутизаторах пересчет вызывает удаление текущего SPT с повторным его построением с помощью алгоритма поиска кратчайших путей, например алгоритма Дейкстры [4].

Разработка новых, более эффективных алгоритмов поиска кратчайших путей позволяет повысить быстродействие современных компьютерных сетей. Под ускоренной маршрутизацией понимается метод поиска оптимальных маршрутов для передачи пакетов данных от узла-отправителя к узлу-получателю в условиях динамически изменяющейся структуры сети и характеристик линий связи, позволяющий сократить трудоемкость построения таблиц маршрутизации путем частичного изменения дерева кратчайших путей с помощью дополнительной информации о конфигурации сети [5].

Цель исследования. Преобразование статических SPT алгоритмов с организацией очередей в виде связанных списков в динамические в целях снижения алгоритмической сложности путем использования доступной информации в SPT [6].

2. Метод преобразования алгоритмов

Пусть $G = (V, \varepsilon)$ есть ориентированный граф, где V – набор узлов, а ε – набор ребер в графе. Пусть N – общее количество узлов в V и E – общее количество ребер в ε. Граф G состоит из корневого узла n_0 и всех других узлов $n_1,...,n_{N-1}$, доступных из корневого узла 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, ассоциированный с e, а $S(e)$ и $E(e)$ обозначают начальный и конечный узел ребра e. Пусть 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.

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 атрибут расстояния. 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 Q имеет форму $\{n,(x,D)\}$, где x определяет потенциального родителя для узла n, а D определяет потенциальное расстояние к узлу n.

В случае смены родительского атрибута узла n процедура $B_{par}(n,T)$ переносит n из списка потомков родительского узла в соответствующий список потомков, а также производит коррекцию атрибута $P(n,T)$.

Инструкция EXTRACT(Q) выбирает первый (верхний) элемент и удаляет его из очереди. Вспомогательная процедура $B_{child}(n,T)$ находит множество узлов (включая n), которые являются потомками n. Это множество можно легко вычислить, спускаясь непосредственно вниз по T.

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)

$\forall e_i \in \mathcal{E}^+$ or $\mathcal{E}^-$

$N \leftarrow 0$

$W(e_i) \leftarrow W(e_i) + \Delta_i$ (Ребро e_i изменяет свой вес на Δ_i):

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

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

$\forall n \in N$

$D(n,T) = D(n,T) + \Delta_i$

Случай 1 (Ребро $e_i \in \mathcal{E}^-$ уменьшает свой вес на Δ_i):

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

Случай 2 (Ребро $e_i \in \mathcal{E}^+$ увеличивает свой вес на Δ_i):

$\forall e \in I(N)$

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

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

$B_{par}(E(e),T)$

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

else

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

$B_{par}(E(e_i),T)$

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

$\forall n \in N$

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

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

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 $D(E(e),T) > D(y,T) + W(e)$
$D(E(e),T) = D(y,T) + W(e)$
$B_{par}(E(e),T)$
$D(n,T) = D(S(e),T) + W(e)$
$ENQUEUE(Q\{E(e),y,D)\})$
Go To Step 2

В начале работы алгоритма происходит инициализация начальных установок для статической версии алгоритма, а также корректируются значения атрибутов затронутых изменениями весов ребер графа узлов для динамической версии. Предполагается, что в существующем SPT все узлы имеют родительский атрибут $P(n,T)$, имеют список потомков $C(n,T)$, а также атрибут расстояния $D(n,T)$.

В случае изменения веса ребра первоначально производится проверка на принадлежность ребра существующему SPT. Если условие выполняется, то выбирается узел $E(e_i)$ и все его потомки в существующем SPT. Все такие узлы включаются процедурой $B_{child}(n,T)$ в множество N для дальнейшего обновления их расстояний.

Случай 1 (вес ребра уменьшается). Узел – предок и все его потомки в старом SPT помещаются с помощью инструкции ENQUEUE в очередь для последующей обработки.

Случай 2 (вес ребра увеличивается). В этом случае 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 $S(e)$:

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

В дальнейшем выполняется инструкция ENQUEUE.

Если ребро, изменившее свой вес, не принадлежит существующему SPT, то производится сравнение текущего атрибута расстояния $D(n,T)$ с потенциально возможным расстоянием через $S(e_i)$. При необходимости производится коррекция родительского атрибута с помощью процедуры $B_{par}(n,T)$, выполняется процедура $B_{child}(n,T)$ и множество найденных потомков N помещается в очередь.

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.

Приведенный выше алгоритм использует трансформацию статических алгоритмов в динамические. Для дальнейшего снижения вычислительных затрат вводится обновление текущих кратчайших расстояний узлов – потомков на шаге 3с помощью процедуры $B_{child}(n,T)$:

$\forall e \in O(y)$
if $D(E(e),T) > D(y,T) + W(e)$
$D(E(e),T) = D(y,T) + W(e)$
$B_{par}(E(e),T)$
$N \leftarrow B_{child}(E(e),T)$
$\forall n \in N$
$D(n,T) = D(S(e),T) + W(e)$
$ENQUEUE(Q\{E(e),y,D)\})$
Go To Step 2

В зависимости от реализуемого алгоритма очередь Q имеет различную организацию. В случае алгоритма Дейкстры организуется отсортированный связанный список. В случае алгоритма Беллмана-Форда каждый новый элемент помещается в конец очереди. Согласно D'Esopo-Pape новый элемент также помещается в конец очереди. Если n уже присутствует в очереди, то новые атрибуты заменяют старые. Если же узел уже ранее бывал в очереди, то помещается в начало связанного списка. Для модифицированного D'Esopo-Pape поддерживается два списка, с новыми элементами и уже ранее рассматриваемыми узлами.

4. Методика моделирования и анализ полученных результатов

Было проведено экспериментальное моделирование для подтверждения теоретических выкладок, изложенных выше, проверки эффективности предложенного метода преобразования статических алгоритмов поиска кратчайших путей на графах в динамические, а также для сравнительного анализа работы алгоритмов с точки зрения количества операций 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 [5,6]. Средняя степень вершин составляла 10. Веса ребер графа устанавливались равными евклидовому расстоянию между вершинами.

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.

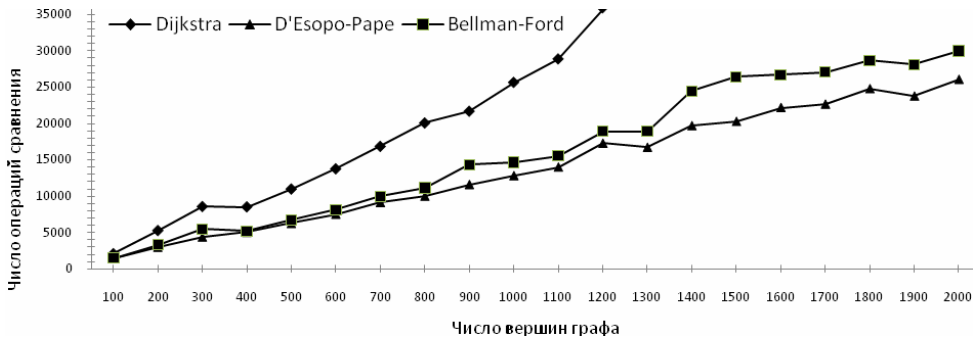

Рис. 1. Трудоемкость статических алгоритмов

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.

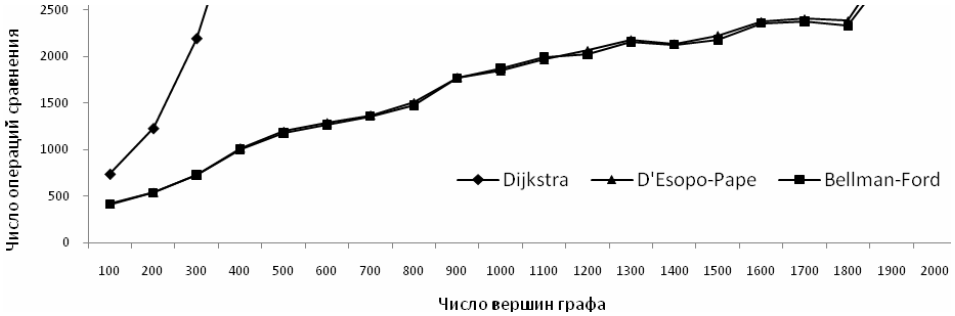

Рис. 2. Трудоемкость динамических алгоритмов при изменяющейся топологии графа

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.

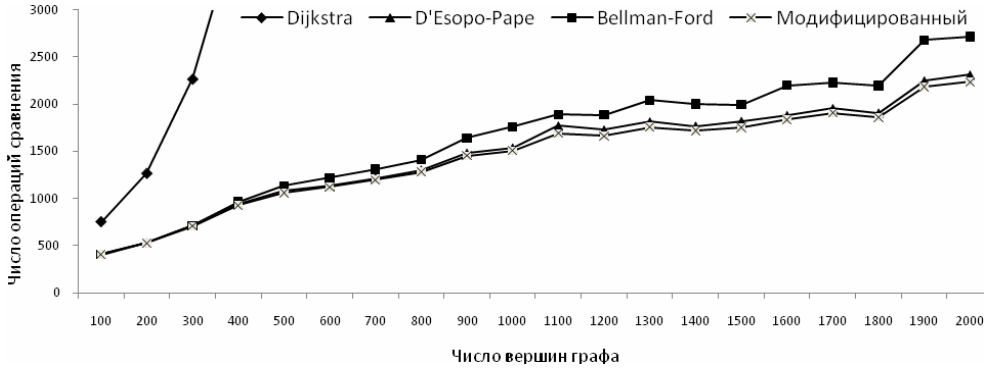

Рис. 3. Трудоемкость динамических алгоритмов при дополнительном обновлении процедурой $B_{child}(n, T)$

Как видно на рис. 3, использование дополнительного обновления текущих длин кратчайших путей на шаге 3 для узлов – потомков позволяет дополнительно снизить вычислительные затраты алгоритмов D'Esopo-Pape и Беллмана-Форда.

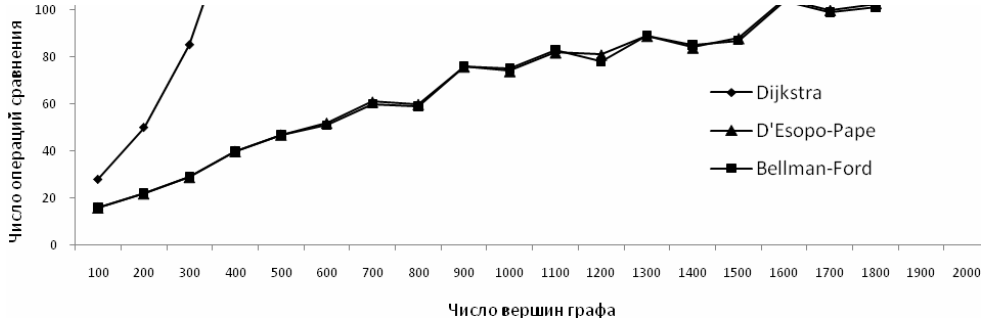

Рис. 4. Число операций сравнения на одно изменение ребра графа

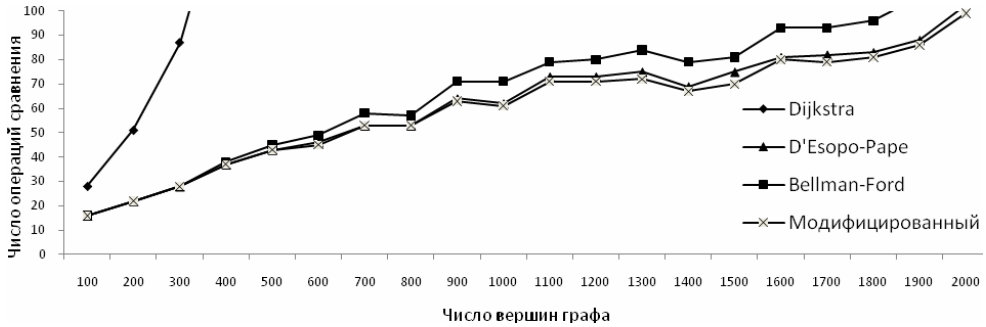

Рис.5. Число операций сравнения на одно изменение ребра графа с процедурой $B_{child}(n, T)$

На рис. 4 и 5 представлены зависимости среднего числа операций сравнения на одно изменение ребра графа для методов трансформации и дополнительного обновления соответственно.

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. Проведенный сравнительный анализ результатов моделирования для различных размерностей графов позволяет сделать вывод о том, что предложенные методы обновления текущих расстояний к узлам для случаев локальных изменений топологии графа позволяют добиться значительного эффекта экономии вычислительных ресурсов.

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. NO. 6. P. 706-718. 2. *T. Cormen, C. Leiserson, and R. Rivest*. Introduction to Algorithms.//Cambridge, MA: The MIT Press, 2001. 3.*R. Perlman, G. Varghese*. "Pitfalls in the design of distributed routing algorithms," in *Proc. SIGCOMM*, vol. Aug., 1988. P. 43–54. 4. *E. Dijkstra*, "A note two problems in connection with graphs." Numerical Mathemat. 1959. Vol. 1. P. 269–271.**5.** *Завизиступ Ю.Ю., Партыка С.А.*Метод преобразования статических SPT алгоритмов в динамические//АСУ и приборы автоматики. Вып. №153. Харків: Харківський національний університет радіоелектроніки, 2010.Вып. 153. С.69-73. **6.** *Завизиступ Ю.Ю., Партыка С.А.* Метод снижения алгоритмической сложности динамического SPT алгоритма // Вестник ХНТУ. 2011. №2(41). С. 318-322.

Поступила в редколлегию 06.03.2012

Партыка Станислав Александрович, асистент 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