жество параметров для мониторинга и управления.

Литература

1. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. М.: Техносфера, 2003. – 512 с.
2. Гудвин Г.К., Гребе С.Ф., Сальгадо М.Э. Проектирование систем управления. М.: БИНОМ. Лаборатория знаний, 2004. – 911 с.
3. Кофман А., Анри-Лабордер А. Методы и модели исследования операций. М.: Мир, 1977. – 432 с.
4. Леохин Ю.Л. Анализ информационной структуры корпоративной сети // Известия высших учебных заведений. Поволжский регион. Технические науки.№ 4, 2008. – С. 27-40.
5. Ретана А., Слайс Д., Уайт Р. Принципы проектирования корпоративных IP-сетей. М.: ИД «Вильямс», 2002. – 368 с.

УДК 621.391

МЕТОД ОПТИМИЗАЦИИ ТАБЛИЦ МАРШРУТИЗАЦИИ

Смагин А.А., Шиготаров А.В.

В работе рассматривается метод уменьшения размера таблиц маршрутизации, основанный на использовании алгоритма минимизации булевых функций. Экспериментальные результаты, полученные на данных, представляющих собой таблицы маршрутизации из крупных точек обмена Internet-трафиком, показывают высокую эффективность метода для задач большой размерности.

Введение

Использование бесклассовой междоменной маршрутизации (CIDR) отчасти позволяет решить проблему постоянного увеличения размеров таблиц маршрутизации в Internet-роутерах. Тем не менее, количество записей в реальных таблицах маршрутизации может достигать нескольких десятков тысяч, в связи с этим, актуальным вопросом остается разработка эффективных схем хранения и управления таблицами маршрутизации.

Базовым методом Internet-маршрутизации является сопоставление по наиболее длинному префиксу. Каждая запись таблицы маршрутизации содержит префикс (который может соответствовать не только конкретному узлу, а целой сети) узла назначения и соответствующий ему узел, на который необходимо отправлять трафик. Одному адресу назначения может соответствовать несколько записей в таблице – выбирается же из них та, которая имеет наиболее длинную маску подсети. Рассмотрим фрагмент таблицы маршрутизации (см. таблицу 1). Пусть адрес назначения равен 192.168.20.18. В этом случае, в соответствии с алгоритмом поиска наиболее длинного префикса, трафик будет отправлен на узел 1, так как адрес сети, соответствующий первой записи дает большее совпадение разрядов.

Таблица 1. Фрагмент таблицы маршрутизации

Префикс	Узел
192.168.20.16/27	1
192.168.0.0/16	2

Исследователями было предложено несколько программных и аппаратных схем для увеличения скорости поиска в таблицах маршрутизации. Алгоритмические методы включают в себя: локальный поиск, двоичный поиск, использование специальных структур данных (бинарные деревья, трие), хеширование и ряд других. Основным недостатком данных подходов является то, что они требуют нескольких обращений к памяти. Требования к производительности современных сетей являются очень высокими, в то время как время отклика современных архитектур памяти ограничивают возможное количество обращений к памяти.

Одним из наиболее распространенных и перспективных аппаратных решений является контентно-адресуемая память (CAM) [6]. Главной особенностью CAM является то, что адресация осуществляется на основе содержания данных, поиск нужной записи, при этом, может быть осуществлен за одно обращение к памяти. В каждой позиции CAM может быть только 0 или 1. CAM поддерживают только сравнения образцов с фиксированной длиной и, следовательно, в явном виде не подходят для поиска префиксов. Возможным решением является использование нескольких CAM, каждая из которых хранит префиксы определенной длины [3].

Более эффективный подход основан на применении так называемых тернарных CAM. Помимо индекса, TCAM хранят также отдельную

маску для каждой записи. Маска определяет, какие из битов индекса являются активными (см. таблицу 2).

Таблица 2. Таблица префиксов, хранящаяся в TCAM

Запись	Префикс	Маска	Следующий узел
1	10100000	11110000	1
2	10110000	11110000	1
3	11011000	11111000	2
4	10001000	11111000	3
5	10111000	11111000	2

Оптимизация таблиц маршрутизации на основе методов построения минимальной ДНФ булевой функции была впервые предложена в [4]. Однако сам алгоритм минимизации, использовавшийся в ней, требует слишком больших вычислительных ресурсов для решения задач большой размерности. В данной работе исследуется метод, дающий такое же качество решения, расходуя на это значительно меньше времени. Рассматриваемая схема оптимизации может быть применена не только к статическим данным – она может быть легко адаптирована для учета изменений, происходящих в таблице маршрутизации [4].

Метод оптимизации таблиц маршрутизации

Введем ряд определений, которые будут использоваться в работе. Пусть задан алфавит $\{x_1 \dots x_n\}$. Выражение $K = x_{i_1}^{\sigma_1} \& \dots \& x_{i_r}^{\sigma_r}$ ($i_\nu \neq i_\mu$ при $\nu \neq \mu$, $\sigma_i \in \{0,1\}$, $x^0 = \bar{x}$, $x^1 = x$) называется элементарной конъюнкцией. Под булевой функцией будем понимать функцию $f(x_1; x_2 \dots x_n)$ с областью определения $\{0,1\}^n$ и областью значений $\{0, 1, *\}$, где * соответствует безразличному состоянию функции. Дизъюнкция элементарных конъюнкций называется дизъюнктивной нормальной формой (ДНФ). Любая булева функция может быть представлена в виде ДНФ. Например, функция $f(x_1, x_2, x_3)$, принимающая значение 1 на наборах {000, 001, 100} и 0 на остальных, может быть представлена ДНФ $D = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3$. Импликантой булевой функции $f(x_1; x_2 \dots x_n)$ называется элементарная конъюнкция $K = x_{i_1}^{\sigma_1} \& \dots \& x_{i_r}^{\sigma_r}$, такая, что на любом наборе значений переменных $\{x_1 \dots x_n\}$, на котором K принимает значение 1, функция f равна 1 или *. Импликанта называется простой, если при удалении из нее любого символа она перестает быть импликантой. В приведенном выше примере, $\bar{x}_1\bar{x}_2\bar{x}_3$ и $\bar{x}_1\bar{x}_2x_3$ являются импликантами, а $\bar{x}_1\bar{x}_2$ – простой импликантой. В геометрической интерпретации набору значений переменных соответствует вершина n-мерного единичного куба E, а импликанте функции f – грань E такая, что на любой ее точке f принимает значение 1 или *.

Под проблемой минимизации булевых функций понимается задача построения для произвольной булевой функции ее минимальной, относительно заданного критерия, ДНФ.

Метод уменьшения размера таблицы маршрутизации, рассматриваемый в данной работе основан на замене нескольких записей на одну. Так, например, записи 1 и 2 из таблицы 2 отличаются только четвертым битом и могут быть скомбинированы (10100000, 11100000). Такой подход естественным образом сводится к задаче минимизации ДНФ – префиксам из таблицы маршрутизации мы сопоставляем элементарные конъюнкции, а каждому множеству записей, имеющих одинаковые длины префиксов k и следующий узел u, ставится в соответствие отдельная булева функция. Так, например, записям из таблицы 2, имеющим следующий узел 1, будет соответствовать функция $f_{4,1} = x_1\bar{x}_2x_1\bar{x}_2 \vee x_1\bar{x}_2x_1x_2$. Операции же объединения записей осуществляются посредством запуска процедуры минимизации ДНФ (относительно количества конъюнкций) для каждой функции $f_{k,u}$. Такое преобразование, очевидно, не влияет на процесс маршрутизации, то есть выбор узла, на который необходимо отправлять трафик.

Алгоритм минимизации ДНФ

В работе [4] для уменьшения таблиц маршрутизации использовался минимизатор Espresso-Exact [7]. Повышение быстродействия по сравнению с ним достигается за счет применения более эффективных структур данных – двоичных диаграмм решения, а также модифицированного алгоритма построения минимального покрытия, основанного на методе локального поиска.

Предлагаемый алгоритм минимизации использует для представления булевых функций упорядоченные двоичные диаграммы решения (OBDD) [5], а для покрытий – двоичные диаграммы с отбрасыванием незначащих нулей (ZDD) [5].

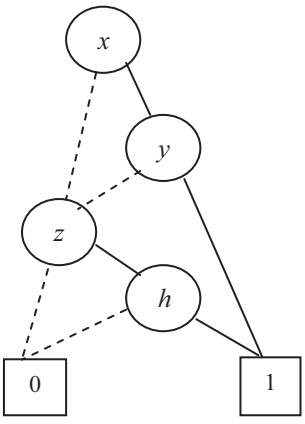

Рис. 1. OBDD–представление для функции $f = xy \vee \bar{x}zh$

Двоичные диаграммы решения (BDD) сочетают в себе два полезных качества – приемлемый расход памяти и, как правило, быстрое выполнение операций над реализуемыми функциями. Упорядоченная двоичная диаграмма решений (OBDD) для заданных функции $f(x_1...x_n)$ и отношения порядка $\pi : V \to \{0;1\ ...\ n\}$, где $V = \{x_1...x_n\}$ – это направленный ациклический граф, состоящий из нетерминальных вершин, отмеченных переменными из V и терминальными вершинами, отмеченными булевыми константами 1 и 0. Каждая нетерминальная вершина имеет два исходящих ребра – 1-ребро и 0-ребро. Начальный узел OBDD называется корнем. Для вычисления значения функции на заданном наборе значений переменных мы формируем путь от корня к терминальной вершине следующим образом: если переменная соответствующая текущей вершине принимает значение 1, то выбирается 1-ребро, в противном случае – 0-ребро. В сформированном пути мы можем встретить каждую переменную не более одного раза. На пути от корня к терминальной вершине, переменные встречаются в соответствии с заданным отношением порядка π.

Двоичные диаграммы решения реализуют разделяемое представление для дискретных объектов, размер которого не зависит линейно от их количества. Пример OBDD-представления приведен на рис. 1, где 0-ребра изображены заштрихованной линией, а 1-ребра – обычной.

Для уменьшения размера OBDD применяют следующие правила:

1. совмещение изоморфных подграфов;

2. удаление узлов, оба исходящих узла которых указывают на один и тот же узел.

В случае двоичных диаграмм с отбрасыванием незначащих нулей (ZDD), первое правило сохраняется, а вместо удаления вершин, исходящие ветви которых указывают на один узел, удаляются узлы с единичным выходом, указывающим на 0. Данная разновидность диаграмм решения более эффективна, по сравнению с OBDD, для представления разреженных множеств, в частности, покрытий булевых функций.

Рассматриваемый алгоритм минимизации ДНФ использует классическую схему работы, состоящую из двух этапов:

- генерация всех простых импликант функции;
- поиск минимального покрытия множества точек, в которых функция принимает значение 1, множеством простых импликант.

Генерация простых импликант, а также удаление доминируемых строк и столбцов осуществляются с помощью методов, предложенных в [1].

Далее задачу построения минимальной ДНФ удобно рассматривать как задачу на покрытие. Пусть M обозначает множество всех точек $\{m_1, m_2, ..., m_l\}$, на которых заданная функция f принимает значение 1, а $P = \{p_1, p_2, ..., p_k\}$ – множество всех простых импликант f. Матрицей покрытия будем называть бинарную матрицу A размерности $l \times k$ такую, что $A_{ij} = 1$ (при этом будем говорить, что j-й столбец покрывает i-ю строку), $1 \le i \le l$, $1 \le j \le k$, если $m_i \subseteq p_j$, в противном случае $A_{ij} = 0$. Требуется найти минимальное количество столбцов, покрывающих все строки матрицы A.

Алгоритм начинается с выбора некоторого допустимого решения s и далее поиск оптимального решения ведется не во всем пространстве поиска, а только в некоторой окрестности s. При этом, если стоимость найденного решения s' меньше, чем у s, то мы устанавливаем в качестве текущего решения s' и запускаем поиск по его окрестности.

Начальное решение строится с помощью вероятностного «жадного» алгоритма, который последовательно выбирает простые импликанты в соответствии с (1) [2] до тех пор, пока не будет сформировано полное покрытие.

$$p = \arg\max \sum_{m \in p} \frac{1}{|\{p \in P \mid m \in p, m \in M\}|}. \quad (1)$$

В случае равенства оценок выбор производится псевдослучайно.

Под окрестностью решения s будем рассматривать множество решений, которые можно получить из s путем удаления, либо добавления 1 элемента. Для ускорения процесса поиска, исследуется не вся окрестность: после удаления элемента решения, выбранного псевдослучайно, запускается «жадный» алгоритм построения минимального покрытия. Таким образом, мы получаем новое допустимое решение. Алгоритм прекращает работу, если на текущей итерации не произошло улучшения текущего решения.

Экспериментальные результаты

Программная реализация рассмотренного алгоритма была выполнена на языке C++, проект был скомпилирован с помощью gcc 3.2. При разработке использовались пакеты CUDD [10] и extra [8].

На рис. 2 показаны результаты сравнения времени работы предлагаемого метода MINDNF и метода Espresso-Exact, использовавшегося для оптимизации таблиц маршрутизации в [4]. В качестве тестовых данных были использованы таблицы, полученные из нескольких крупных точек обмена интернет-трафиком – PAIX и Mae-West [9]. При этом суммарное время минимизации при использовании MINDNF меньше в 8,25 раз для задач с количеством префиксов $k < 10000$, и в 22,5 для задач с $k \geq 10000$. Метод Espresso-Exact позволяет уменьшить количество записей в таблицах маршрутизации в два раза. Отличие в качестве сжатия для обоих методов незначительно, и составляет порядка 0,001.

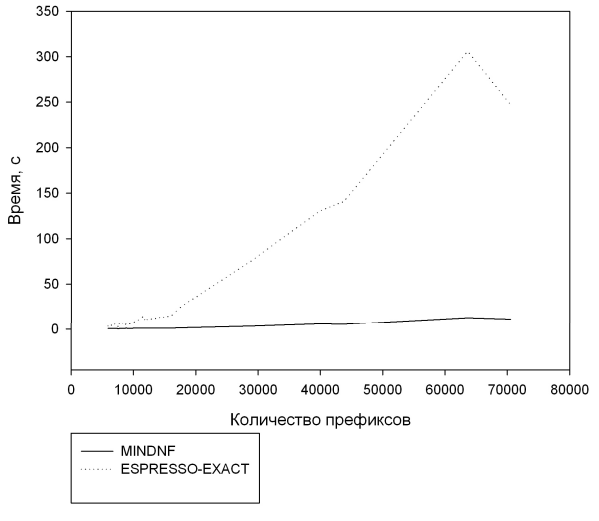

Рис. 2. Зависимость времени минимизации от количества префиксов

Заключение

В данной работе рассмотрен метод уменьшения размера таблиц маршрутизации с помощью применения алгоритма минимизации булевых функций. Экспериментальные результаты показывают, что время работы предлагаемого способа минимизации значительно меньше, чем у алгоритмов, приводимых в литературе [4]. Качество получаемых решений, при этом, практически не отличается от точных решений. Данный метод может быть использован для оптимизации таблиц маршрутизации большой размерности, реализованных на основе тернарной контентно-адресуемой памяти.

Литература

1. Coudert O., Madre J.C. Implicit and incremental computation of primes and essential primes of Boolean functions // Proc. of the Design Automation Conf. Anaheim, CA, 1992. – P. 36-39.

2. Coudert O., Madre J.C. New ideas for solving covering problems // Proc. of the Design Automation Conference, 1995. – P. 641-645.

3. Hayashi T., Miyazaki T. High-Speed Table Lookup Engine for IPv6 Longest Prefix Match // Proc. IEEE Globecom, vol. 2, IEEE Press, Piscataway, N. J., 1999. – P. 1576-1581.

4. Liu H. Routing Table Compaction in Ternary-CAM // IEEE Micro. Jan/Feb, 2002. – P. 58-64.

5. Meinel C., Theobald T. Algorithms and data structures in VLSI design. Springer-Verlag NY, 1998. – 267 p.

6. McAuley A., Francis P. Fast Routing Table Lookup Using CAMs // Proc. IEEE Infocom. Vol. 3. IEEE CS Press, Los Alamitos, Calif., 1993. – P. 1382-1391.

7. Rudell R. Multiple-valued minimization for PLA synthesis. UCB technical report M86/65, 1986.

8. http://www.ee.pdx.edu/~alanmi/research/extra.htm

9. http://www.routeviews.org/vlsi.colorado.edu/~fabio/