Литература

1. *Grant K. S*. Developing with Ext GWT Enterprise RIA Development. Apress. 2009.
2. *Dewsbury R*. Google Web Toolkit. Prentice Hall. 2007.
3. *Фридл Дж*. Регулярные выражения. Символ-Плюс. 2008.
4. *Брайан А*. Тестирование и оптимизация веб-сайтов: руководство по Google Website Optimizer. Диалектика. 2009.
5. *Стотлемайер Д.* Тестирование Web-приложений. КУДИЦ-Образ. 2003.

ЗАДАЧА ОПТИМАЛЬНОГО ПОКРЫТИЯ ИЗМЕНЕННЫХ ДАННЫХ

Столяров В. О., Шавлак М. Ю.

ВВЕДЕНИЕ

Задача оптимального покрытия измененных данных рассматривается при передаче модифицированного изображения с экрана одного устройства на экран другого устройства и является частным случаем задачи покрытия множества. Для ее решения были рассмотрены следующие алгоритмы:

- Сеточный алгоритм,
- Жадный алгоритм.

Введем определение эффективности решения. Под эффективностью здесь понимается совокупность атрибутов решения: интерактивность, рациональное использование канала передачи данных (оптимальность сжатия), экономия трафика (минимизация данных на отправку).

Для построения эффективного решения требуется решить две алгоритмические проблемы. Первая – как эффективно покрыть изменения на экране, произошедшие за определенное время прямоугольниками пикселей. Критерий эффективности в данном случае – количество прямоугольников. Задача алгоритма – минимизировать это количество. Вторая проблема – как эффективно сжать эти прямоугольники перед отправкой на устройство клиента. В этом случае критерий эффективности – баланс между степенью сжатия и временем сжатия/распаковки.

Постановка алгоритмической задачи

В результате очередного обновления экрана перед серверным приложением ставится задача оптимального покрытия изменённых точек прямоугольниками, в общем случае известная как “задача о минимальном покрытии множества”. Формулируется она следующим образом. Пусть даны множество $M = \{1,...,m\}$ и набор его подмножеств $M_1,...,M_n$ таких,

что $\bigcup_{j=1}^{n} M_j = M$. Совокупность подмножеств $M_j, j \in J \subseteq \{1,...,n\}$, называется покрытием множества M, если $\bigcup_{j \in J} M_j = M$. Каждому M_j приписан вес $c_j \geq 0$. Требуется найти покрытие минимального суммарного веса. Задача называется невзвешенной, если все подмножества M_j имеют единичные веса.

В общем виде задача о покрытии NP-трудна, поэтому логично предположить, что следует искать приближенные алгоритмы полиномиальной сложности, доставляющие решения со значениями целевой функции, близкими к оптимуму. Рассмотрим некоторые примеры приближённых алгоритмов для решения поставленной задачи.

СЕТОЧНЫЙ АЛГОРИТМ

Самый простой на первый взгляд и, тем не менее, достаточно эффективный алгоритм, разработанный в настоящей работе, заключается в разбиении экрана на множество непересекающихся прямоугольников по сетке. Очевидно, что при таком выборе семейства множеств M_j, существует единственное покрытие M. Алгоритм необычайно прост в реализации: достаточно посетить каждую точку единожды, чтобы построить покрывающее множество. Таким образом, он имеет минимальную возможную сложность O(NxM).

Быстрота работы делает его хорошим выбором на высокоскоростных соединениях, где время, затраченное на вычисления может быть больше задержки передачи данных по сети. Алгоритм в худшем случае даёт четырёхкратный проигрыш по сравнению с оптимальным решением. Однако эта граница редко достигается на практике. На рисунке 1 показано сеточное покрытие текстовых изменений (ввод A,B,C).

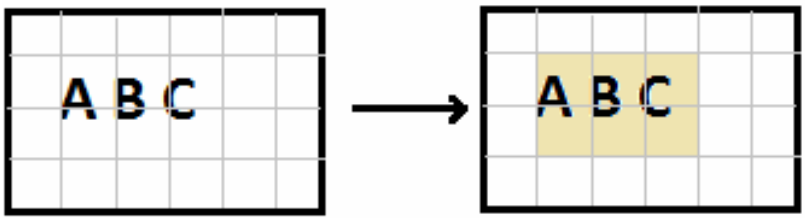

Рис. 1. Схема работы сеточного алгоритма

ЖАДНЫЙ АЛГОРИТМ

Рассмотрим жадный алгоритм для покрытия изменившихся точек экрана размером MxN квадратами шириной W. Идея алгоритма проста: на каждой итерации выбирать квадраты с максимальным количеством покрываемых точек. На рисунке 2 представлена демонстрация его работы на измененной области в форме треугольника.

Рис. 2. Работа жадного алгоритма

В [1] показано, что относительная погрешность $\delta \le 1 + \ln \max_{j=1,..,n} |M_j|$. Также было доказано, что в невзвешенном случае для этого алгоритма справедливо неравенство $\delta \le \ln m - \ln \ln m + 0{,}78$, что близко к известной нижней оценке $\delta \ge (1-\varepsilon)\ln m$.

Решение можно разбить на два этапа: построение множества квадратов с ассоциированным весом и сортировка их по убыванию. Весом c(A) квадрата A назовём количество точек, которые он покрывает. Как и в случае с кэшированием, функция c(A) обладает свойством, позволяющим за один проход по экрану вычислить её для всех квадратов.

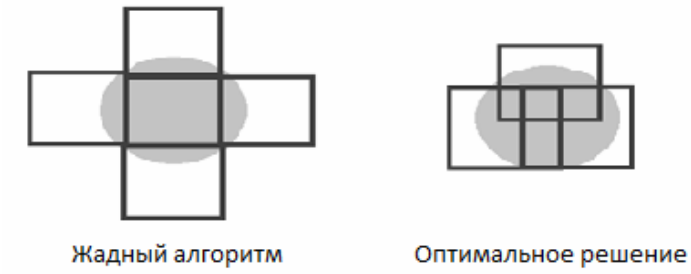

Рис. 3. Сравнение результатов работы жадного алгоритма и оптимального решения

Таким образом, сложность построения множества квадратов порядка O(N*M). В результате также получим количество квадратов порядка O(N*M). Сортировка их по убыванию в среднем O(N*M*log(N*M)). Как альтернатива, можно использовать бинарное дерево поиска для сортировки, однако этот подход не даст нам уменьшения алгоритмической сложности задачи. Выборку из отсортированной последовательности можно произвести за время порядка O(N*M). Таким образом, алгоритм

даст нам результат за время порядка O(N*M*log(N*M)), что достаточно эффективно для целей, поставленных в настоящей работе, однако алгоритм не лишён недостатков. На рисунке 3 представлен случай, когда такой алгоритм даёт неоптимальный результат.

ДОПОЛНИТЕЛЬНЫЕ ОПТИМИЗАЦИИ

В качестве дополнительной оптимизации был использован кэш. В кэше находятся наиболее часто используемые измененные данные. В него попадают все изменения, но остаются только наиболее часто используемые. Были расмотрены две реализации кэша контекстно-зависимая и контекстно-инвариантная. В контекстно-зависимой реализации измененные данные покрываются прямоугольными множествами, и далее выполняется поиск вхождения этих покрытий в кэше. Эта реализация отвечает требованиям высокой эффективности по скорости работы кэша. В контекстно-независимой реализации строится максимальное покрытие измененных данных существующими элементами в кэше. Несмотря на то, что построение этого покрытия требует дополнительных вычислительных ресурсов, эта реализация дает максимально возможное использование данных из кэша, что отвечает требованиям минимизации отправляемых данных.

ЗАКЛЮЧЕНИЕ

В работе реализованы оба алгоритма и протестированы в контексте задачи удаленного управления компьютером посредством портативных устройств, произведена оценка их эффективности. К преимуществам сеточного алгоритма можно отнести простоту реализации; он имеет наименьшую сложность из всех алгоритмов покрытия. Но недостатком является то, что в худшем случае он демонстрирует результат в четыре раза хуже оптимального решения.

Жадный алгоритм более сложен в реализации, но в условиях ограниченной пропускной способности сети он более эффективен, хотя в некоторых случаях явно уступает сеточному в построении наиболее компактного покрытия.

Литература

1. *Еремеев А. В., Заозерская Л. А. ,Колоколов А. А.* Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования //Дискретный анализ и исследование операций. Сер. 2. 2000. № 2. С. 22–46.