УДК 001.57:519.876.5:004.942

УНИВЕРСАЛЬНЫЙ СПОСОБ ВОССТАНОВЛЕНИЯ РЕЗЕРВИРУЕМЫХ ДАННЫХ С ОПТИМИЗАЦИЕЙ ПО СКОРОСТИ ВЫПОЛНЕНИЯ

Казаков В.Г., Федосин С.А.

В работе представлена математическая модель процесса восстановления резервируемых данных. Произведен анализ факторов, влияющих на его скорость. Осуществлен анализ и обоснован выбор алгоритмов теории графов для применения. На основе предложенной модели разработан универсальный способ восстановления, не зависящий от применяемого алгоритма резервирования.

Введение

Важность проблемы сохранности данных обусловлена принципиальной неустранимостью возможности их утраты. Наиболее распространенным способом повышения безопасности данных является периодическое создание резервных копий (далее РК).

Процесс восстановления данных сводится к объединению последовательности созданных резервных копий, называемой путем восстановления. Одной из основных характеристик процесса восстановления является скорость его выполнения. При автоматизированном восстановлении производится специализированная процедура для каждого алгоритма резервного копирования. Для традиционно используемых алгоритмов, вопрос поиска пути восстановления данных не стоит. Так, при работе инкрементной схемы создается единственная цепочка копий, и при восстановлении к состоянию на любой момент резервирования существует лишь единственный путь. Путь восстановления для полного резервирования редуцируется в единственную полную копию данных на момент времени восстановления. Однако создание дополнительных избыточных копий по требованию пользователя, а также возможность утраты некоторой части резервных данных образуют отклонения от работы алгоритмов резервного копирования, поэтому применение специализированных процедур восстановления значительно усложняется [1]. В таких случаях возможно, что выбранный системой путь восстановления не будет являться наилучшим по скорости. Возможно также, что система не сможет найти путь и управление будет передано пользователю. При работе более сложных алгоритмов создания резервных копий также появляется возможность восстановления данных по нескольким путям [2]. Таким образом, в общем случае может существовать несколько различных путей, с различной скоростью восстановления, что обуславливает проблему выбора оптимального среди них.

Появляется необходимость в универсальном адаптивном способе, учитывающим возможности утраты резервных данных и дополнительного создания копий, подходящим не только для восстановления данных при работе алгоритмов резервного копирования, но для любого набора копий вообще. Требуется также возможность оптимизации поиска пути восстановления по времени выполнения. Существование унифицированной процедуры восстановления позволило бы отказаться от разработки специализированных процедур для каждого реализованного алгоритма резервирования.

Постановка задачи

Для совершенствования процесса восстановления данных в указанном направлении, необходим аппарат математического моделирования работы систем резервного копирования (далее СРК). В зарубежных источниках, посвященных резервному копированию (A. L. Chervenak, V. Vellanki, Z. Kurmas, V. Gupta, A. Costello, C. Umans, F. Wu, S. Romig, L. P. Cox), повсеместно используется словесно-графический способ описания работы алгоритмов, процессов резервирования и восстановления данных. Для исследования применяется эмпирический подход, то есть алгоритмы резервирования реализуются в составе некоторой системы, где их качества тестируются на выбранных данных. Средства компьютерного моделирования необходимо дополнять построением математических моделей. Существующие узкоспециализированные способы формализации (X. Wei, W. Min, А. Ю. Телешев) не подходят для решения поставленных задач. Требуется разработка модели процесса восстановления данных в системах резервного копирования. Необходимо выделить факторы, влияющие на скорость восстановления и учитывать их при выборе пути восстановления данных.

Моделирование процесса восстановления данных

Рассмотрим систему резервного копирования, которая создает копии данных в последовательные запланированные моменты времени $\{t_k\}$, где $k = 1,2 \ldots T$. Систему данных для резервного копирования на момент времени $t_j \in t_k$ обозначим D_j. Будем отныне всегда предполагать, что начальное (момент t_0) состояние данных D_0 – пустое, а D_1 нет, то есть $D_0 = \emptyset$, $D_1 \neq \emptyset$. Каждая сделанная копия данных сохраняется в некотором хранилище данных (репозитории). Каждую такую копию будем называть элементом репозитория. Элемент репозитория, который содержит изменения между состояниями данных с момента t_s до момента t_{s+n}, то есть от D_s до D_{s+n}, обозначим R_s^n или $R(s, n)$, где s и n – целые, причем $n > 0$, $s \geq 0$. Учитывая, что начальное состояние данных пустое, R_0^n фактически означает полную резервную копию состояния данных в момент t_n, то есть копию D_n. Множество всех элементов репозитория будем обозначать как ***Rep***.

Сущность алгоритма РК заключается в выборе способа создания копий данных, то есть алгоритм определяет, какая часть данных должна быть скопирована на каждый момент РК.

С помощью предложенной математической схемы описания процессов резервирования данных алгоритм может быть описан функцией выхода $R_{scheme}(t_j) \subset 2^{\mathrm{Rep}}$, задающей соответствие каждому моменту резервного копирования t_j множества генерируемых в СРК элементов репозитория. Алгоритм также может быть описан функцией $\breve{R}_{scheme}(t_T) \subset 2^{\boldsymbol{Rep}}$, задающей соответствие каждому моменту резервного копирования t_j множества хранимых в СРК элементов репозитория [3].

Имеет смысл операция объединения элементов репозитория [4]. Например, объединяя полную резервную копию системы данных на момент t_s, то есть R_0^s, с элементом репозитория, содержащим изменения между состояниями данных с момента времени t_s до момента t_{s+n}, то есть с R_s^n, мы получаем полную резервную копию системы данных на момент t_{s+n}, то есть R_0^{s+n}. Следует заметить, что не все элементы могут быть объединены друг с другом. Нет смысла, например, объединять элементы, содержащие изменения данных в непоследовательные периоды времени. Для объединения несочетаемых элементов репозитория введем элемент *XRep*, который будет обозначать некий «испорченный» элемент.

Множество всех элементов репозитория обозначим ***Rep***. ***Rep*** включает все возможные элементы репозитория и «испорченный» элемент *XRep*:

$$\boldsymbol{Rep} = \{X; R_i^j\}_{i=0,1,2\ldots; j=1,2,3\ldots} =$$

$$= \{\mathrm{XRep}; R_0^1; R_0^2; R_0^3 \ldots; R_1^1; R_1^2; R_1^3 \ \ldots \ R_k^1; R_k^2; R_k^3 \ \ldots\}.$$

Операцию объединения элементов репозитория $\oplus$ на множестве всех элементов репозитория ***Rep*** введем следующим образом:

$$XRep \oplus XRep = XRep;$$

$$R_l^n \oplus XRep = XRep \oplus R_l^n = XRep;$$

$$R_i^{n_i} \oplus R_j^{n_j} = \begin{cases} R_i^{n_i+n_j}, (i < j) \wedge (j = i + n_i); \\ XRep. \end{cases}$$

Введенная операция объединения элементов репозитория является алгебраической бинарной ассоциативной операцией, а множество всех элементов репозитория с операцией объединения элементов репозитория, то есть ***Rep***$\oplus$, является полугруппой. Проверим заявленное утверждение.

Ассоциативность операции объединения соответствует возможности непоследовательного ее применения при объединении элементов в пути. Фактически это означает возможность разбить путь восстановления на составные подпути и оперировать с ними отдельно.

Дадим формализованное определение понятия пути восстановления.

Путем восстановления данных к состоянию на момент t_k называется конечная последовательность n элементов репозитория $P=\{R_i \in Rep\}_{i=1\ldots n}$, для которой результат последовательного применения операции объединения равен $R(0, k)$, то есть $R_1 \oplus R_2 \oplus \ldots \oplus R_n = R(0, k)$.

Длиной пути будем называть количество элементов репозитория в пути.

Система резервного копирования, в которой были произведены операции РК в моменты времени $\{t_k\}$, где $k = 0; 1; 2 \ldots T$, может произвести восстановление данных к состоянию на момент t_j, где $0 < j \leq T$, если в репозитории существуют элементы, образующие путь к состоянию данных на момент РК t_j.

Процесс восстановления заключается в выборе последовательности элементов репозитория и последующем их объединении. Для построения структурной модели процесса восстановления обратимся к разделу теоретической кибернетики – теории автоматов.

Представим процесс восстановления системы данных к некоторому моменту t_k в виде дискрет-

нодетерминированной модели конечного автомата Мили (finite automata):

$$F = <Z, X, Y, \varphi(z,x), \psi(z,x), z_0>,$$

где $x \in X$ и $z, z_0 \in Z$ [5].

Тактами функционирования дискретного автомата будут являться моменты подачи на вход каждого нового элемента из последовательности элементов репозитория, образующих путь.

Множество входных сигналов (входной алфавит) X, множество выходных сигналов (выходной алфавит) Y, множество внутренних состояний (внутренний алфавит, или алфавит состояний) Z, начальное состояние z_0, $z_0 \in Z$, функцию переходов $\varphi(z,x)$ и функцию выходов $\psi(z,x)$ определим следующим образом:

$X = \breve{R}_{scheme}(t_k) \subset \boldsymbol{Rep}$ – все имеющиеся элементы репозитория для некоторого алгоритма *«scheme»* на момент восстановления;

$Y = \{-1, 0, 1\}$ – выходной сигнал, представляет собой статус произведенной операции. При этом обозначим: «–1» – результат неудачного применения элемента репозитория, подаваемого на вход, то есть результат равен *XRep*; «0» – результат удачного применения, то есть результат не равен *XRep*; «1» – достижение целевого состояния автомата, что означает завершение работы процедуры восстановления, то есть в результате получен $R(0, k)$;

$Z = \{D_t \equiv R(0, t) \in \boldsymbol{Rep}\}_{t = 1; 2 \ldots k}$ – копии данных на моменты D_t, при этом целевым состоянием является D_k; $z_0 = R(0,1)$ – начальное состояние, равное имеющейся на момент восстановления полной резервной копии, как правило, это $R(0,1)$;

$\varphi(R_0^n \in Z, R_m^h \in X)$ – фактически означает переход в новое состояние, если объединение элементов репозитория не является «испорченным состоянием» и не «дальше» требуемого состояния: то есть $(n+h) < k$ для получаемого R_0^{n+h};

$\psi(R_0^n \in Z, R_m^h \in X)$ – функция результата перехода, определяемая выражением (2).

Полученный конечный автомат Мили первого рода может быть представлен в виде матрицы соединений или в виде графа, отражающего схему связей элементов репозитория для процедуры восстановления.

При представлении автомата в виде матрицы соединений строки соответствуют исходным состояниям, а столбцы – состояниям перехода. Элемент $A_{ij} = x_k/y_s$, стоящий на пересечении i-ой стоки и j-го столбца, в случае автомата Мили соответствует входному сигналу x_k, вызывающему переход из состояния z_i в состояние z_j, и выходному сигналу y_s, выдаваемому при переходе.

Граф автомата представляет собой набор вершин, соответствующих различным состояниям автомата и соединяющих вершины его дуг, соответствующих тем или иным переходам автомата. Если входной сигнал x_k вызывает переход из состояния z_i в состояние z_j, то на графе автомата дуга, соединяющая вершину z_i с вершиной z_j, обозначается x_k. Для того чтобы отобразить (задать) функцию выходов, дуги графа необходимо отметить соответствующими выходными сигналами. Маркировка дуг в данном случае может быть представлена как x_k/y_s.

Рассмотрим для примера задание одного простого автомата A в матричном виде:

$$A = \begin{array}{c|ccccc} & D_1 & D_2 & D_3 & D_4 & D_5 \\ \hline D_1 & - & R_1^1/0 & R_2^1/0 & R_3^1/0 & R_4^1/1 \\ D_2 & - & - & - & - & - \\ D_3 & - & - & - & R_3^2/1 & - \\ D_4 & - & - & - & - & - \\ D_5 & - & - & - & - & - \end{array}.$$

Представление в виде ориентированного графа показано на рис. 1, при этом отображены только некоторые переходы с неудачными исходами ($y = -1$), которые не меняют состояния. Для примера приведены таковые только для D_1 и D_5. В дальнейшем опустим маркировку исходов на рисунках графов, так как она прослеживается умозрительно. Не будем также изображать петли переходов с неудачными исходами.

Представление работы процедуры восстановления в виде ориентированного графа конечного автомата Мили удобно для дальнейшего ее исследования на предмет поиска путей восстановления, оптимальных по скорости. Нетрудно заметить, что при данной модельной интерпретации в виде вершин отображены состояния

$$\varphi(R_0^n \in Z, R_m^h \in X) = \begin{cases} R_0^n \oplus R_m^h, \ (R_0^n \oplus R_m^h \neq \mathrm{XRep}) \wedge (n + h \leq k); \\ R_0^n. \end{cases} \quad (1)$$

$$\psi(R_0^n \in Z, R_m^h \in X) = \begin{cases} 1, (R_0^n \oplus R_m^h \neq \mathrm{XRep}) \wedge (n + h = k); \\ 0, (R_0^n \oplus R_m^h \neq \mathrm{XRep}) \wedge (n + h < k); \\ -1, (R_0^n \oplus R_m^h = \mathrm{XRep}) \vee (n + h > k). \end{cases} \quad (2)$$

данных, а в виде дуг – элементы репозитория, хранящие данные об изменении между двумя их состояниями. При этом номера вершинам присвоены последовательно в соответствии с моментами резервного копирования t_k. Так, началом дуги, изображающей элемент репозитория $R(l, m)$, является вершина, соответствующая D_l, а концом – соответствующая D_{l+m}.

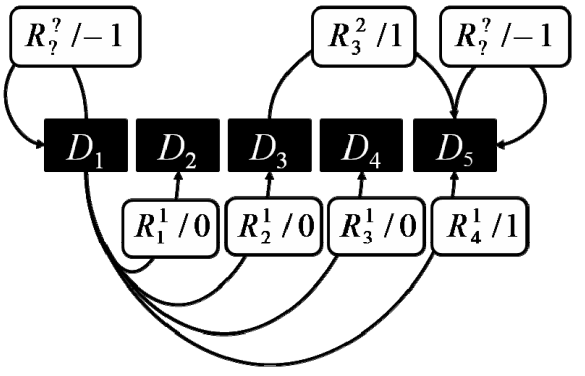

Рис. 1. Граф переходов состояний

Понятие пути восстановления в такой интерпретации становится наглядным и совпадает с понятием пути в орграфах. При этом возможно применение аппарата теории графов для отыскания кратчайших путей и путей, соответствующих некоторым наперед заданным условиям.

Факторы, влияющие на скорость операции восстановления

При отыскании оптимального пути восстановления может учитываться ряд факторов, влияющих на скорость восстановления. Объем данных для восстановления – основной фактор для решений на основе переносных носителей с последовательным доступом (магнитные ленты), так как за счет разных уровней избыточности в путях скорость восстановления может существенно отличаться.

Количество элементов репозитория в пути и объемы файловых индексов влияют на количество логических операций сравнения и объединения, что особенно важно в условиях хранилищ с произвольным доступом.

Значимость факторов может значительно отличаться в зависимости от реализации системы, при этом могут существовать дополнительные факторы. Переход к единому показателю практически невозможен, поэтому при поиске пути восстановления следует выделять наиболее значимые факторы и устанавливать для них приоритет. Далее производить поиск ряда оптимальных путей по первичному фактору, и при необходимости вести дополнительную оптимизацию по дополнительному фактору, имеющему наибольший вес в конкретном случае реализации системы.

В случае, когда данные всех элементов репозитория равнодоступны (например, если хранятся на жестком диске), процесс копирования будет занимать одинаковое количество времени для любого корректного пути, так как в любом случае будет скопировано одинаковое количество файлов. Разница во времени работы процесса восстановления будет обусловлена вычислительной скоростью алгоритмов поиска пути восстановления и скоростью работы операции определения места нахождения нужных версий файлов, зависящей от факторов длины пути и объема файловых индексов резервных данных.

Для иллюстрации важности учета обоих факторов, рассмотрим следующий пример. Начальное состояние данных D_0 пустое. D_1 включает только некоторый файл A, то есть $D_1 = \{A\}$ и $R(0, 1) = \{A\}$. Далее этот файл изменяется и $D_2 = \{B\}$, создается элемент репозитория $R(0, 2)$, содержащий изменение данных от D_0 до D_2 состоящее из B. Далее процесс продолжается так, как отображено на рис. 2. Причем A^- в элементе $R(1, 2)$ означает данные об удалении A. В результате для восстановления к моменту времени t_3 имеются два пути: $P_1 = \{R(0,1); R(1,2)\}$ и $P_2 = \{R(0,2); R(2,1)\}$. Оба пути состоят из двух элементов репозитория каждый. Однако для объединения элементов репозитория для P_1 требуется проанализировать больше на $\{A\}$ данных, чем для P_2. Если предположить что $\{A\}$ является не файлом, а совокупностью из нескольких тысяч файлов, то отличие времени работы процедуры восстановления для рассматриваемых путей может оказаться значительным.

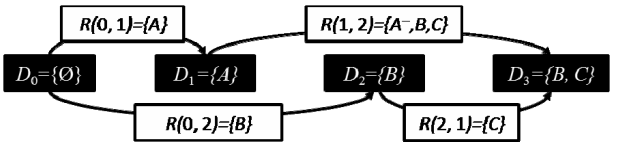

Рис. 2. Пример влияния фактора объема элементов репозитория в пути

Поэтому, при поиске оптимального пути нужно учитывать не только фактор количества элементов репозитория входящих в путь, но и их суммарный объем.

Выбор и применение алгоритмов поиска кратчайших путей

Требуется найти кратчайший путь, обладающий дополнительным свойством. Данную задачу можно рассматривать как задачу с некоторыми дополнительными ограничениями или как многоцелевую задачу, в которой учитывается не только длина, но и свойство объема файлов в элементах репозитория. Эти усложнения будут, вообще говоря, сильно увеличивать вычислительную работу, и с практической точки зрения более просто найти кратчайшие пути с минимальной длиной и выбрать среди них тот, который обладает нужным свойством.

Вообще говоря, необходимо получать простые цепи, так как наличие циклов в пути означает непригодность для восстановления.

Существует много алгоритмов, в которых заложено требование получения простых цепей, однако, как показывают исследования, это требование значительно осложняет работу алгоритма, что влечет за собой значительное увеличение времени работы алгоритма.

Следует заметить, что, как правило, для алгоритмов резервного копирования нет возможности получить цикл в пути восстановления, однако в принципе это не исключено. Ведь существует принципиальная возможность создавать элементы репозитория хранящие изменения по отношению к предыдущему состоянию данных как, к примеру, в обратной инкрементной схеме (англ. reverse incremental). Нужно иметь в виду такую возможность и в действительно универсальном алгоритме восстановления это нужно учитывать.

Можно было бы создать процедуру восстановления с использованием алгоритмов, исключающих циклы для тех схем резервного копирования, где наличие таких путей не исключено и другой алгоритм в противном случае. Но чтобы добиться действительной универсальности процедуры нахождения пути необходимо избавиться от требования излишних пользовательских параметров.

В таком случае имеет смысл использовать гибридные алгоритмы. При работе такого алгоритма, при поиске каждого последующего оптимального пути, изначально работает алгоритм, который не исключает появление циклов, найденный путь проверяется, и в случае обнаружения цикла, оптимальный путь находиться с использованием другого алгоритма, гарантирующего получение простого пути.

Оптимальность выбора гибридного алгоритма поиска K кратчайших путей как основы алгоритма поиска оптимального пути восстановления подтверждается исследованиями [6], проведенных на различных классах графов. По данным исследований гибридный алгоритм в составе с алгоритмом Йена превосходит по скорости поиска другие алгоритмы поиска простых путей, в том числе алгоритм Перко (Perko), алгоритм Йена в оригинале и модифицированный, алгоритм Мартинса и Паскоаля (Martins & Pascoal).

Таким образом, применение гибридного алгоритма отлично подходит для поиска оптимального пути восстановления в системах резервного копирования.

Для поиска путей с требованием минимизации по одному фактору будем использовать алгоритм Дейкстры.

В результате при работе процедуры построения пути восстановления будут работать три алгоритма теории графов: гибридный алгоритм, алгоритм Йена и алгоритм Дейкстры.

Алгоритм Дейкстры строит кратчайший (s, t) – путь для графа $G(N, A)$ за время $O(n^2)$, где N – множество вершин, а A – множество дуг [7]. При этом $n = |N|$ – количество вершин, а $m = |A|$ – количество дуг.

При реализации алгоритма в случае поиска пути восстановления необходимо учесть возможность множественных дуг между двумя вершинами. Поэтому при проверке вершин с временными метками следует рассматривать все возможные варианты дуг. При построении непосредственно пути надо учитывать не только последовательность прохождения вершин, но и использованные при этом дуги.

Удобно связывать с каждой вершиной v еще один список меток $\Theta(v)$, в который следует заносить идентификатор указывающей на v дуги в (s, v)-пути, имеющем минимальный вес среди всех (s, v)-путей, проходящих через вершины, получившие к данному моменту постоянные метки. Таких путей может быть несколько, поэтому $\Theta(v)$ в общем случае является списком, однако если требуется только один путь, то список следует понимать как просто метку. После того как вершина t получила свою постоянную метку, с помощью меток Θ можно легко указать последовательность вершин и дуг, составляющих кратчайший (s, t)-путь.

Заметим, что для работы алгоритма Йена вкупе с алгоритмом Дейкстры для верной работы не требовалось бы указания всех возможных дуг в кратчайшем пути в работе алгоритма Дейкстры,

потому что алгоритм Йена, перед тем как запускать каждый раз алгоритм Дейкстры, удаляет уже просмотренные дуги и рассматривает оставшиеся. Таким образом, нет возможности пропустить нужный путь. Для гибридного алгоритма ситуация сходная. Идеально для задачи подходит представление мультиграфа списками смежности. Так как требуется найти путь, кратчайший в смысле количества использованных элементов репозитория, следует считать все веса равными единице.

При поиске кратчайших путей по алгоритму Йена [8], работу алгоритма следует останавливать, как только следующий найденный путь будет обладать большей длиной, чем уже найденные. Затем из найденных путей следует выбрать оптимальный по второму фактору. Для этого достаточно найти путь с минимальным суммарным объемом входящих в него элементов репозитория. Так, каждой дуге $(x, y) \in A$ поставлен в соответствие вес $c_{xy} \in IR$. $P^k = \{s, v_2^k, v_3^k, ..., v_{q_k}^k, t\}$ – k-ый кратчайший (s, t)-путь, где v_i^k – его вершины. Пусть P_i^k – отклонение от пути P_i^{k-1} в точке i. Под этим понимается следующее: P_i^k – кратчайший из путей, совпадающий с P_i^{k-1} от s до i-ой вершины, а затем идущий к $(i + 1)$-ой вершине по дуге, отличной от дуг к $(i + 1)$-ой вершине тех (ранее уже построенных) кратчайших путей P^j $(j = 1; 2 \ldots k-1)$, которые имеют те же самые начальные подпути от s к i-ой вершине, что и P^{k-1}. P_i^k приходит в вершину t по кратчайшему подпути, не проходящему ни через одну из вершин $\{s, v_2^{k-1}, v_3^{k-1}, ..., v_i^{k-1}\}$, участвующих в формировании первой части пути P_i^k. Алгоритм начинает работу с нахождения кратчайшего пути P^1 с помощью алгоритма Дейкстры. Далее находят следующий оптимальный путь P^k, $k = 2$. Для этого подбирают лучшие отклонения от заданного пути, используя на каждом шаге алгоритм Дейкстры. Кратчайший среди найденных кандидатов P_i^k и будет являться P^k. Далее переходят к поиску P^k, для $k = 3$. И так далее. Временная сложность алгоритма $O(Kn^3)$ с учетом того, что используется алгоритм Дейкстры с неотрицательными весами.

При работе гибридного алгоритма [6], кратчайший (s, t)-путь, который отклоняется от P^k на вершине v_i^k, то есть P_i^k, ищется в виде $Root_{P^k}(s, v_i^k) \circ (v_i^k, j) \circ T_t(j)$, так, что дуга (v_i^k, j) не принадлежит никакому из кандидатов P^j, найденных до этого. Здесь $Root_{P^k}(s, v_i^k)$ – подпуть P^k от s до v_i^k; операция объединения путей p и q обозначена как $p \circ q$, при этом объединенный путь образован p и следующим за ним q.

Создание каждого нового кандидата таким способом более эффективно, так как процедура отыскания кратчайшего пути на каждом этапе сводится к проблеме выбора нужной дуги (v_i^k, j), что может быть сделано за время $O(1)$, в отличие от применения алгоритма Дейкстры, который в нашем случае с неотрицательными весами выполняется за время $O(n^2)$. Однако такой алгоритм может вернуть путь с циклами, когда $Root_{P^k}(s, v_i^k)$ и $T_t(j)$ содержат одинаковые узлы. В таком случае гибридный алгоритм переключается на использование алгоритма Йена. В худшем случае время работы гибридного алгоритма будет идентично алгоритму Йена, то есть $O(Kn^3)$, в лучшем же случае, когда в отклонениях циклов не будет, следует ожидать $\Omega(Kn + n^2 + m\log n)$.

При практической реализации для тривиальных случаев работы схем РК, когда проблемы множественности путей не стоит и существует лишь единственный путь, процедура гибридного алгоритма может быть отключена и выполнение сводится к алгоритму Дейкстры, что максимизирует производительность.

Способ восстановления данных

Общая процедура восстановления может быть описана следующей последовательностью действий:

1. Определить наиболее значимые факторы, влияющие на скорость восстановления, основываясь на особенностях реализации системы резервного копирования и типе использованных носителей.

2. При создании каждого нового элемента репозитория сохранить метаданные, включающие файловые индексы.

3. Произвести проверку корректности элементов репозитория, идентифицировать их доступность.

4. Построить граф репозитория для автомата процесса восстановления данных к искомому состоянию.

5. Произвести поиск кратчайших путей на графе репозитория с учетом выбранных факторов. В случае, если пути восстановления не существует, следует:

6. Произвести восстановление данных по найденному пути.

7. Проверить результат восстановления данных, используя хэш или другую информацию о целевом состоянии данных. В случае неудачного результата следует перейти к этапу 3. В случае, когда при втором проходе найден идентичный путь и восстановление невозможно следует прервать процедуру восстановления с ошибкой.

Способ поиска оптимального пути для восстановления данных был реализован в составе раз-

работанной СРК, на базе которой проведен ряд испытаний на различных алгоритмах РК, среди которых полное, инкрементное, дифференциальное, мультиуровневое копирования, схема Костелло, Z Scheme, а также некоторые другие. Опыт эксплуатации показал применимость предложенного способа.

Заключение

В работе решена задача построения универсального способа поиска оптимального пути восстановления данных, пригодного для реализации автоматизированного восстановления при работе с любым набором элементарных копий в репозитории и не зависящего от используемого алгоритма РК. Для этого была формализована операция объединения элементов репозитория. Процедура восстановления сводится к объединению нужного множества элементарных копий в определенной последовательности. Процесс восстановления представляется конечным автоматом.

Произведен анализ возможных факторов, влияющих на оптимальность скорости восстановления по найденному пути. Ввиду практической невозможности построения агрегированного показателя в общем случае предлагается выделение таковых при реализации системы. Способ отыскания пути остается прежним в каждом случае, в чем и заключается его универсальность.

Для поиска путей, оптимальных по некоторым факторам применяется аппарат теории графов. Было осуществлено исследование современных методов отыскания кратчайших путей, в результате которого выявлена оптимальность применения гибридного алгоритма в составе с алгоритмом Йена для решения задачи отыскания оптимального пути восстановления в СРК.

Литература

1. Казаков В.Г. Избыточность в алгоритмах резервного копирования // Системы управления и информационные технологии. 2.2(36), 2009. – С. 252-256.
2. Kurmas Z., Chervenak A. Evaluating backup algorithms // Proc. of the Eighth Goddard Conference on Mass Storage Systems and Technologies. 2000. URL: www.cis.gvsu.edu/ /~kurmasz/papers/kurmas-MSS00.pdf (дата обращения 05.10.2009).
3. Казаков В.Г., Федосин С.А. Метод моделирования алгоритмов резервного копирования для получения оценок объема репозитория // ИКТ. Т.6, №3, 2008. – С. 126-132.
4. Kazakov V.G., Fedosin S.A. Selecting the Optimal Recovery Path in Backup Systems // Innovations and Advances in Computer Sciences and Engineering. Ed. Sobh, Tarek. Springer, 2009.
5. Глушков, В. М. Синтез цифровых автоматов М.: ГИФМЛ, 1962. – 238 с.
6. Pascoal M. M. B. Implementations and empirical comparison for K shortest loopless path algorithms // The Ninth DIMACS Implementation Challenge: The Shortest Path Problem. 2006. URL: http:// www.dis.uniroma1.it/ ~challenge9/papers/pascoal.pdf (дата обращения 04.10.2009).
7. Емеличев В.А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов. М.: Наука, 1990. – 384 с.
8. Yen J.Y. Finding the K shortest loopless paths in a network // Management Science. №17, 1971. – P. 712-716.

УДК 338.47

ВЛИЯНИЕ МЕЖОПЕРАТОРСКИХ ОТНОШЕНИЙ НА ТАРИФНУЮ ПОЛИТИКУ ТЕЛЕКОММУНИКАЦИОННЫХ КОМПАНИЙ

Трубникова Е.И., Трубников Д.А.

В статье рассматривается и анализируется современное состояние и основные проблемы в процессе ценообразования телекоммуникационных компаний для абонентов и операторов. В работе приводится структура взаимодействия телекоммуникационной компании с потребителями и другими участниками рынка при оказании услуг телефонной связи. Статья посвящена анализу современных тенденций во взаиморасчетах операторов. Центральное место в материале уделяется возможности ценообразования в зависимости от распределения трафика через сети взаимодействующих операторов.

Постановка задачи

Специфика телекоммуникационной отрасли, согласно нормативным законодательным актам [1], имеет существенную особенность, отличающую ее от всех остальных: в предоставлении услуги абоненту, как правило, участвует не один оператор, а сразу несколько, и получаемое вознаграждение за оказанные услуги разделяется между всеми теми компаниями, которые взаимодействуют между собой в процессе предоставления этой услуги.