УДК 004.421:004.652.3

МЕТОДЫ И АЛГОРИТМЫ РАСПРЕДЕЛЕНИЯ РАБОЧЕЙ НАГРУЗКИ В СЕТЕВЫХ БАЗАХ ДАННЫХ

Фильгус Д.И.

ФГБОУ «Московский технологический университет», Москва, e-mail: dmif42@ya.ru

Представлена модель решения задачи оптимизации процесса управления запросами в распределенных информационных системах. Для разрешения очереди запросов необходимо выбрать наиболее приемлемый способ выборки запросов и наилучший метод реализации выбранного способа. Перспективными вариантами способа обслуживания запросов являются способ групповой выборки, а также способ групповой выборки с индивидуальным сегментированием. Сложность рассматриваемых задач, решаемых в этом направлении, обусловлена тем, что большинство из них носит комбинаторный характер, и характеризуется экспоненциальной временной сложностью и относится к классу NP-полных задач, а их решение современными вычислительными средствами сталкивается с трудностями, которые связаны с большими затратами машинного времени и памяти. Еще одним препятствием при решении этой проблемы является отсутствие достаточно адекватных математических моделей процессов обслуживания запросов в этих информационных системах. Показана возможность применения методов решения задач булевого линейного и нелинейного программирования на основе рангового подхода, обладающих малой временной сложностью и обеспечивающих требуемую точность решения этих задач. Предложенная модель может использоваться для решения задачи оптимизации процесса управления запросами в распределенных информационных системах. Актуальной является задача усовершенствования математических методов решения задач булевого линейного и нелинейного программирования, обладающих малой временной сложностью и обеспечивающих требуемую точность решения этих задач, разработка на основе предложенных методов математического и программного обеспечения для решения задач обслуживания запросов при управлении базами данных, позволяющего повысить коэффициент использования ресурсов системы.

Ключевые слова: база данных, диссипативные структуры, запрос, оптимизация, ранговый подход

OPTIMISATION OF THE PROCESS OF MANAGING REQUESTS IN DISTRIBUTED INFORMATION SYSTEMS

Filgus D.I.

Moscow Technological University (MIREA), Moscow, e-mail: dmif42@ya.ru

A model for solving the problem of optimizing the process of managing requests in distributed information systems is presented. To resolve the query queue, you must select the most appropriate method of query selection and the best method for implementing the selected method. Prospective variants of the method of servicing requests are the method of group sampling, as well as the method of group sampling with individual segmentation. The complexity of the problems under consideration in this direction is due to the fact that most of them are of a combinatorial nature and are characterized by exponential time complexity and belong to the class of NP-complete problems, and their solution by modern computing means is faced with difficulties that are associated with large expenditures of computer time and memory. Another obstacle in solving this problem is the lack of adequate mathematical models of the processes of servicing requests in these information systems. The possibility of application of methods for solving Boolean linear and nonlinear programming problems based on the rank approach, having a small time complexity and providing the required accuracy of solving these problems is shown. The proposed model can be used to solve the problem of optimizing the process of managing requests in distributed information systems. The actual task is to improve mathematical methods for solving Boolean linear and nonlinear programming problems that have a small time complexity and provide the required accuracy of solving these problems. Developing on the basis of the proposed methods of mathematical and software for solving the problems of servicing queries when managing databases that allows increasing the utilization factor of resources system.

Keywords: database, dissipative structures, query, optimization, ranking approach

На современном этапе развития вычислительной техники получили развитие распределенные системы обработки информации в вычислительных сетях под управлением распределенных операционных систем. От качества программного обеспечения, решающего задачи оптимального планирования и диспетчеризации, в значительной мере зависит эффективность вычислительной системы в целом, так как решение дополнительных системных задач увеличивает дополнительные расходы машинного времени. Следствием неоптимального планирования процесса обслуживания множества запросов пользователей является уменьшение степени использования информационных ресурсов, а также снижение безопасности и живучести информационной системы [1, 2].

Анализ режимов функционирования сетевой базы данных (СБД) показал, что разработка и сопровождение информационных систем такого типа производится из расчета среднестатистической нагрузки, определяющейся соотношением 80/20, т.е. каждым 80 запросам соответствует 20 транзакций,

но такое соотношение не дает реального представления о круглосуточном функционировании информационной системы. В частности, при таком подходе не учитывается почасовая нагрузка в течение суток, когда в периоды реорганизации число транзакций, как правило, превосходит число запросов [3–5].

В качестве примера рассмотрим гистограмму (рис. 1). Ее изучение показывает, что АИУС использующей СБД присущи два критических режима функционирования:

1. Область красного цвета – пиковая рабочая нагрузка, являющаяся суммой пользовательских запросов и запросов на корректировки (транзакций) и определяемая соотношением:

$$T = \sum_{p=1}^{P_0} T_p^{з} + \sum_{s=1}^{S_0} T_s^{к},$$

где $T_p^{з}$ – время реализации p-го запроса пользователя, а $T_s^{к}$ – время реализации p-го задания на корректировку.

Она характеризуется возрастанием числа запросов до 96 в минуту и числа транзакций до 18 в минуту. Длительность функционирования информационной системы в режиме пиковой нагрузки около 5 часов с 11.00 часов до 16.00 часов. Соотношение соответственно составляет 96/18.

2. Область зеленого цвета – режим реорганизации базы данных. Она характеризуется возрастанием числа транзакций до 45 в секунду и падением числа запросов до 5 в секунду. Продолжительность функционирования информационной системы в режиме реорганизации составляет около 3 часов соответственно с 00.00 до 03.00 часов. Соотношение составляет 5/45.

Таким образом, соответствие правила 80/20, используемого разработчиками и администраторами для СБД АИУС, справедливо для достаточно идеализированной системы и соответствует истинному только на некотором участке времени. Если предположить, что информационная система функционирует в режиме близком к идеальному все оставшееся время, то даже такое предположение ставит под сомнение правомочность использования данного правила поскольку 8 часов работы информационной системы при критической нагрузке составляет 1/3 суток, при условии, что 5 часов из 8 система функционирует в режиме обслуживания множества запросов пользователей (рис. 2).

Как противопоставление высказанному предположению об идеализированном состоянии информационной системы рассмотрим графики анализа рабочей нагрузки отдельно для транзакций и отдельно для запросов. Как видно из графика (рис. 3), рекомендуемое правило не учитывает возрастание пользовательской нагрузки в системе на указанном участке времени. Такое же состояние наблюдается и для реализации транзакций в системе, что и отображено на графике (рис. 4) соответственно. Исследование поведения рабочей нагрузки, возникающей в СБД для более длительных промежутков времени, показало, что в целом характер ее поведения не изменяется график (рис. 5), и наблюдается только некоторое снижение в конце недели с 56 до 47 запросов в мин., что не является достаточно существенным.

Также в классических моделях не учитываются такие важные ситуации, как возможность захвата некоторыми запросами одновременно пересекаемых областей данных. И наконец, это уровень требований к качеству обслуживания, предъявляемый различными классами запросов пользователей. Последнее обстоятельство снижает точность получаемых результатов и соответственно увеличивает число сбоев и конфликтов, возникающих в исследуемых системах.

Оценка и анализ процессов управления реализацией рабочей нагрузкой в сетевых информационных системах позволили выделить в отдельный класс запросы на корректировку на основе их семантики. Такое разделение рабочей нагрузки позволяет рассмотреть запросы и транзакции как две составляющих, исключив их влияние друг на друга. В классических же моделях указанное разграничение производится посредством применения различного рода блокировок, не семантических свойств языковых операторов.

Все это свидетельствует о необходимости повышения адекватности математических моделей процессов обслуживания запросов в структурно-сложных системах, разработке методов расчета и оптимизации этих систем и алгоритмов оптимального управления ими, обеспечивающих повышение их качества и оперативности [6].

Основной материал

База данных состоит из M таблиц T_i, $i = \overline{1..M}$ содержащие информацию и системы управления базой данных. К базе данных имеют доступ K операторов O_j, $j = \overline{1..K}$, которые формируют запросы к базе данных. Каждый запрос $З_р$ от любого оператора O_j имеет свой приоритет $С_р$, зависящий от уровня привилегий оператора. Момент времени, в который от оператора поступает запрос – величина случайная. Поэтому на входе системы управления при высокой интенсивности запросов образуются очереди запросов.

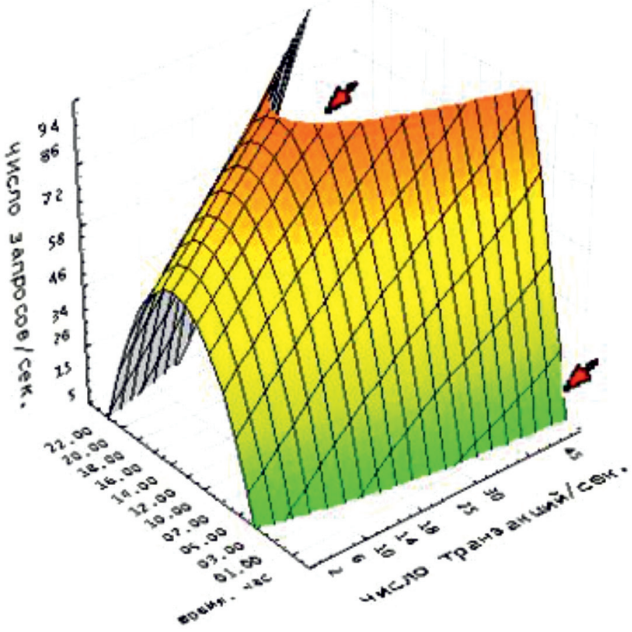

Рис. 1. Диаграмма соотношения запросы/транзакции в течение суток

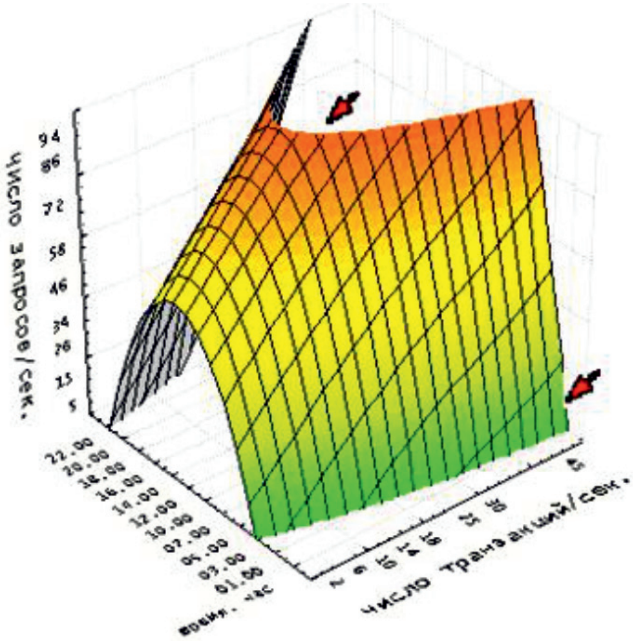

Рис. 2. Диаграмма соотношения запросы/транзакции в течение суток с учетом параметра 80/20

Схема управления распределенной базой данных показана на рис. 6.

При наличии ограничений на размер хранимой очереди в системе управления может возникнуть ситуация, при которой очередной запрос, поступивший на вход системы управления, не будет принят к обслуживанию, что приведет к его потере или к задержке в обслуживании. Таким образом, диссипативные структуры могут перейти из одного стационарного состояния в другое в результате неустойчивости предыдущего неупорядоченного состояния при критическом значении некоторого параметра, отвечающего точке бифуркации. В точке бифуркации невозможно предсказать, в каком направлении будет развиваться система: станет ли состояние хаотическим или система перейдет на новый, более высокий уровень упорядоченности [7, 8].

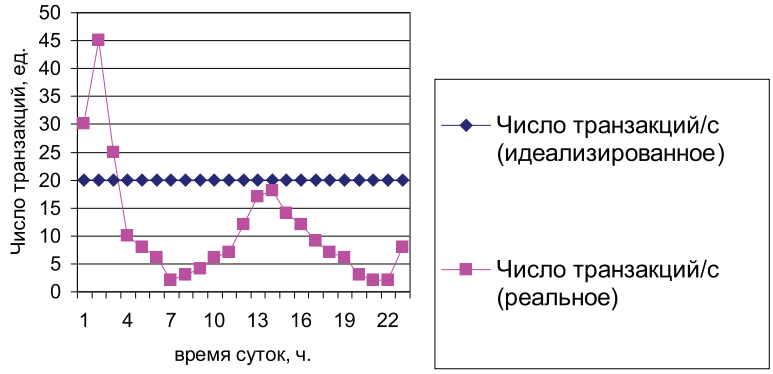

Рис. 3. График соответствия правила 80/20 расчета динамической нагрузки транзакций реальной нагрузке

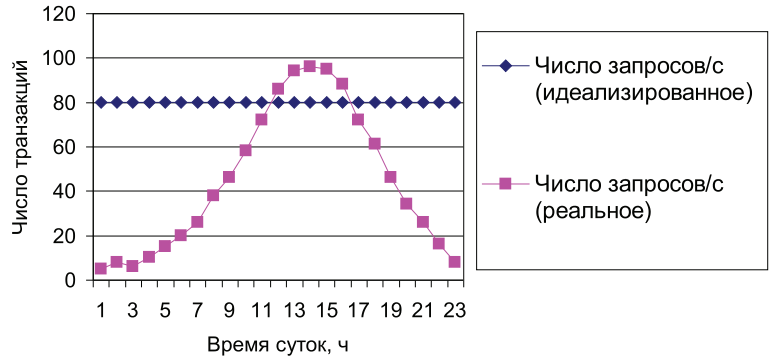

Рис. 4. График соответствия правила 80/20 расчета динамической нагрузки запросов реальной нагрузке

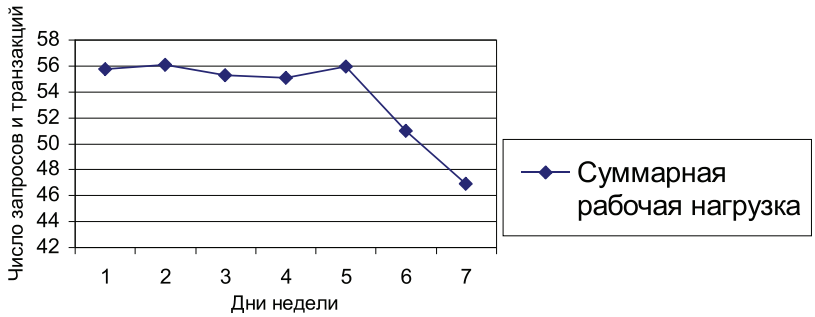

Рис. 5. График суммарной рабочей нагрузки по дням недели

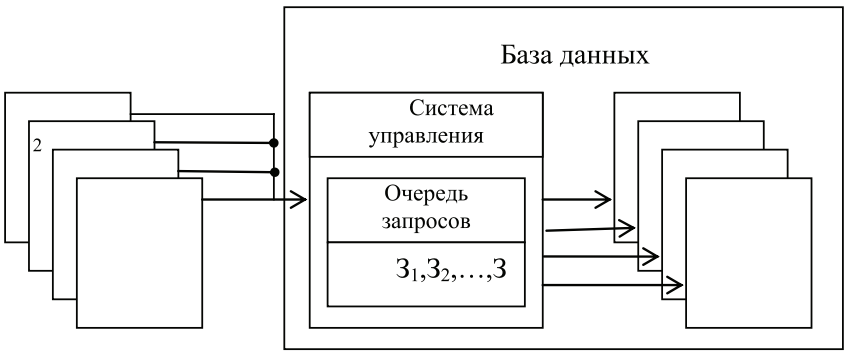

Рис. 6. Схема управления базой данных

Следует отметить дополнительное требование к системе управления базой данных – система управления должна иметь одновременный доступ к любому количеству таблиц.

Определимся, что под разрешением очереди будем понимать процесс поэтапной выборки запросов из очереди запросов, которые необходимо и возможно выполнить на данном этапе. Запросы в очередь поступают в любой момент времени, но состояние очереди определяется только между этапами.

Для разрешения очереди запросов необходимо выбрать наиболее приемлемый способ выборки запросов и наилучший метод реализации выбранного способа.

Для оценки способа выборки запросов будем использовать коэффициент использования ресурсов $K_{\text{ИР}}$. Данный коэффициент показывает, какая часть таблиц из общего количества таблиц, к которым обращаются стоящие в очереди запросы, будет использована. Если $K_{\text{ИР}} = 0$, то ни одна таблица не будет использована, а если $K_{\text{ИР}} = 1$, то используются все таблицы, к которым существуют запросы на данный момент. Но не всегда возможно выбрать такие запросы, чтобы все таблицы были задействованы. Поэтому перед нами стоит задача выбора такого способа разрешения очереди запросов, при котором $K_{\text{ИР}}$ стремилось к 1. В общем виде $K_{\text{ИР}}$ определяется следующим образом:

$$K_{\text{ИР}} = \frac{N_{\text{И}}}{N_{\text{О}}}, \qquad (1)$$

где $N_{\text{И}}$ – количество таблиц, которые будут задействованы при реализации определенной выборки; $N_{\text{О}}$ – количество таблиц, к которым существуют запросы из очереди.

Но в данном виде $K_{\text{ИР}}$ не учитывает приоритеты запросов, поэтому необходимо изменить коэффициент (1), для учета приоритета запросов. Но если учитывать приоритеты запросов, то представление коэффициента $K_{\text{ИР}}$ зависит от выбранного способа выборки. От выборки будет зависеть только числитель, а знаменатель будет одинаковым. Определим его: пусть Y_i – максимальная величина приоритета запроса из запросов очереди, обращающихся к таблице T_i, тогда знаменатель примет вид

$$\sum_{i=1}^{M} Y_i \,. \qquad (2)$$

Наиболее перспективными вариантами способа обслуживания запросов являются:

– способ групповой выборки;

– способ групповой выборки с индивидуальным сегментированием.

Способ групповой выборки – это такой способ, при реализации которого из очереди запросов обслуживается несколько запросов одновременно. Выбираются запросы, которые требуют информацию из разных таблиц, и чтобы сумма их приоритетов была максимальной. В случае наличия равнозначных запросов выбирают более «старые».

Пусть $\{\vec{X}\}$ – множество всех вариантов выборки запросов из очереди, $\vec{X}$ – один из вариантов выборки запросов. Причем

$$\vec{X} = \{x_1, x_2, ..., x_p, ..., x_N\}, \qquad (3)$$

где $p = \overline{1..N}$, N – количество запросов в очереди, x_p – булева переменная равная 1, если запрос $З_р$ выбран в этом варианте, и 0, если нет. $С_р$ – приоритет запроса $З_р$. Тогда числитель из (1) примет вид

$$\sum_{p=1}^{N} C_p x_p \,. \qquad (4)$$

Получаем $K_{\text{ИР}}$ следующего вида с учетом (4) и (2):

$$K_{\text{ИР}} = \frac{\sum_{p=1}^{N} C_p x_p}{\sum_{i=1}^{M} Y_i}. \qquad (5)$$

Для того чтобы $K_{\text{ИР}}$ приняло единичное значение, необходимо, чтобы числитель (4) из (5) был бы равен знаменателю (2) из (5). Но это в лучшем случае, а в общем случае числитель должен стремиться к знаменателю. Так как знаменатель для конкретного момента времени это константа, то необходимо сделать такую выборку запросов $\vec{X}$ из очереди, чтобы числитель принял максимальное значение, причем это значение никогда не превысит знаменатель.

Это означает, что необходимо выбрать из очереди как можно большее количество запросов, которые обращаются к разным таблицам, и сумма приоритетов выбранных запросов была бы максимальна. Причем стремление к максимуму суммы приоритетов выбранных запросов является главным критерием при выборе запросов из очереди.

Обозначим числитель (5) переменной F, которая стремится к максимальному значению. Получаем функционал:

$$F = \sum_{p=1}^{N} C_p x_p \to \max \,. \qquad (6)$$

Пусть A_{kg} – булевая переменная равная 1, если $З_k$ использует таблицу T_g, и 0, если нет. B_g – количество копий таблицы T_g. Тогда исходя из условия, что в любой момент времени любая таблица может быть

использована для одного запроса получаем M ограничений вида

$$\sum_{k=1}^{p} A_{kg} x_k \le B_g, g = \overline{1..M}. \qquad (7)$$

Следовательно, нам необходимо найти такую выборку $\vec{X}$ из множества $\{\vec{X}\}$, для которой функционал (6) примет максимальное значение, при выполнении всех ограничений (7). Мы получили задачу линейного программирования с булевыми переменными.

Способ групповой выборки с индивидуальной сегментацией – это такой способ, при реализации которого запросы, находящиеся в очереди, разбиваются на подзапросы. Из очереди полученных подзапросов выбираются подзапросы, требующие информацию из различных таблиц, и чтобы сумма их приоритетов была максимальной. В случае наличия равнозначных подзапросов выбирают более «старые».

Подзапрос – это часть запроса, которая запрашивает информацию из одной конкретной таблицы. Если запрос требует информацию из R таблиц, то он разбивается на R подзапросов.

В этом варианте нам надо выбрать из очереди как можно большее количество подзапросов, которые обращаются к разным таблицам, и сумма приоритетов выбранных подзапросов была бы максимальна. Причем стремление к максимуму суммы приоритетов выбранных подзапросов является главным критерием при выборе подзапросов из очереди.

Пусть $З_{kg}$ – подзапрос запроса $З_k$, обращающийся к таблице T_g. C_{kg} – приоритет запроса $З_{kg}$. $\{\vec{X}\}$ – множество всех вариантов выбора подзапросов из очереди, $\vec{X}$ – один из вариантов выбора подзапросов. Причем $\vec{X} = \{x_{11}, x_{12}, ..., x_{kg}, ..., x_s\}$, где $k = \overline{1..p}$, $g = \overline{1..M}$, p – количество запросов в очереди, M – количество таблиц, x_{kg} – булевая переменная, равная 1, если соответствующий подзапрос $З_{kg}$ выбран в этом варианте, и 0, если нет, тогда коэффициент (5) примет вид

$$K_{ИР} = \frac{\sum_{j=1}^{p_1} C_{1j} S_1(C_n^1) + \sum_{j=1}^{p_2} C_{2j} S_2(C_n^2) + ... + \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k) + ... + \sum_{j=1}^{p_n} C_{nj} S_n(C_n^n)}{\sum_{i=1}^{M} Y_i}, \qquad (8)$$

где $Sr(C_n^r) = S_1 + S_2 + ... + S_{p_r}$ – сумма всех возможных сочетаний произведений переменных содержащих в каждом произведении $Sr = X_p X_к ... X_m$ r различных переменных; $p_r = \frac{n!}{r!(n-r)!}$;

C_{rj} – коэффициенты, стоящие в произведениях Sr, содержащих r переменных.

И, соответственно, исходя из (6), получаем функционал

$$F(x) = \sum_{j=1}^{p_1} C_{1j} S_1(C_n^1) + \sum_{j=1}^{p_2} C_{2j} S_2(C_n^2) + ... + \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k) + ... + \sum_{j=1}^{p_n} C_{nj} S_n(C_n^n) \to \max. \qquad (9)$$

Таким образом, мы получаем задачу нелинейного программирования с булевыми переменными, с ограничениями вида (7).

В настоящее время наиболее эффективными методами решения рассматриваемых классов задач являются методы «рангового подхода» и «ветвей и границ». Данные методы обладают наименьшей вычислительной сложностью, при наименьшей погрешности приближенных решений. Основным недостатком в алгоритмах, основанных на методе «ветвей и границ», является их экспоненциальная сложность, а в алгоритмах, основанных на методе «рангового подхода» на данный момент не определено, каким образом реализовать принцип выделения «коридора» в задачах квадратичного программирования. При реализации обоих методов время получения решения фиксированно зависит от размерности решаемой задачи. Поэтому актуальным был вопрос разработки нового метода, лишенного указанных недостатков, причем была бы предусмотрена такая возможность, как задавать требуемое время решения задачи. Необходимо было также сохранить ряд достоинств указанных выше методов.

Предложенный метод основан на комбинированном использовании способа отсечения неперспективных вариантов решения, используя идеи метода «рангового подхода», и способа построения возможных вариантов решений, используя идеи метода «ветвей и границ» [9]. Суть разработанного

метода заключается в том, что при формировании множества возможных решений на основе «рангового подхода» отсев неперспективных вариантов, внутри множеств, происходит на основе метода «ветвей и границ». При этом для определения верхней и нижней оценки, при реализации метода «ветвей и границ», были использованы разработанные для алгоритмов метода «рангового подхода» оптимистический (P_O) и гарантированный прогнозы (P_G) соответственно.

Оптимистический прогноз – это такая величина решения задачи, основанного на векторе (3), для которого определяется прогноз, которая обязательно не будет превышена.

Гарантированный прогноз – это такая величина решения задачи, основанного на векторе (3), для которого определяется прогноз, которая обязательно будет получена.

Для корректировки времени решения предложено использовать величину, названную степенью огрубления ($\varepsilon_{огр}$). Данная величина измеряется в процентах и показывает степень приближения верхней оценки к нижней оценке, путем принудительного уменьшения верхней.

При изменении $\varepsilon_{огр}$ время решения задачи обратно пропорционально погрешности решения.

Для формализации метода были разработаны две процедуры $П_1$ и $П_2$. $П_1$ применяется для построения цепочки векторов и, соответственно, для выбора из нее оптимального решения, а $П_2$ применяется для построения прогнозов и отсечения неперспективных векторов. Необходимо отметить, что для каждого вектора из цепочки полученной с помощью процедуры $П_1$ выполняется процедура $П_2$.

Рассмотрим каждую процедуру подробней:

$П_1$: При построении цепочки векторов есть необходимость хранить в памяти вычислительного средства все вектора из цепочки, которые не были отсечены с помощью процедуры $П_2$. Поэтому необходимо выбрать такой вариант построения векторов, который бы требовал как можно меньше физической памяти для их хранения. Возможны два варианта построения цепочки: «в ширину» и «в глубину». Данные варианты различаются только способом формирования цепочки векторов и не влияют на вычислительную сложность алгоритмов, построенных на основе разработанного метода. Вариант «в ширину» имеет хорошую наглядность при построении, но требует значительные затраты физической памяти, а вариант «в глубину» менее нагляден, но менее требователен к объему физической памяти. К тому же данный вариант позволяет создавать алгоритмы на основе разработанного метода с использованием рекурсивных функций, что также упрощает программный код и увеличивает быстродействие разрабатываемых алгоритмов.

Работа процедуры $П_1$ начинается с так называемого нулевого вектора. Нулевой вектор это такой вектор, который независимо от исходных данных задачи всегда удовлетворяет ограничениям.

$П_2$: Данная процедура используется для построения прогнозов и отсечения неперспективных векторов. Причем корректировка времени решения, за счет которой время решения задачи не будет превышать требуемое время, происходит за счет ужесточения требований, при отсечении неперспективных векторов, используя величину $\varepsilon_{огр.}$

Для формализации процедуры введем понятие текущего максимума. Текущий максимум – это максимальная величина решения задачи, из всех решений, получаемых при подстановке в функционал (6) или (9) векторов уже побывавших в цепочке векторов.

На основе формализованного метода был заработан точный алгоритм решения задач линейного и квадратичного программирования. Разработка алгоритма включает в себя реализацию процедур $П_1$ и $П_2$. Для рационального использования памяти построение цепочки векторов производится в «глубину», в этом случае необходимо будет хранить не более N векторов в любой момент времени. Для дальнейшего уменьшения требуемого объема памяти, так как алгоритм является итерационным, реализован рекурсивный вызов функции $П_1$. В теле которой выполняется функция $П_2$. Входными параметрами для обеих процедур является текущий вектор (текущее решение) вида: $\vec{X} = \{x_1, x_2, ..., x_p, ..., x_N\}$. Причем вторым входным параметром функции $П_1$ является число p. Данное число p сопровождает текущий вектор и говорит о том, что все $x_i, i = \overline{1..p}$ определены и принимают значения 0 либо 1, а $x_i, i = \overline{.(p+1)..N}$ неопределенны. К тому же в очередном вызове процедуры $П_1$ определяется значение именно $x_{(p+1)}$. Процедура $П_1$ не зависит от степени нелинейности функционала. Решение задачи начинается с так называемого нулевого вектора. Нулевой вектор это – вектор (3), у которого все x_i неопределенны. Обозначим $x_i = \varnothing$, если x_i неопределен, но при вычислении значений функционала и ограничений считать $\varnothing \equiv 0$. Поэтому нулевой вектор примет вид

$$\vec{X}_0 = \{\varnothing, \varnothing, ..., \varnothing\}. \qquad (10)$$

Причем p нулевого вектора равен 1.

Обозначим: текущий вектор – $\vec{X}_T$; значение функционала при подстановке $\vec{X}_T$ – F_T; текущий максимум – F^*_{opt}; текущий оптимальный вектор – $\vec{X}^*_{opt}$; некоторый вектор – $\vec{X}$.

Главная программа имеет вид:

Шаг 1. $F^*_{opt} = 0$.

Шаг 2. Вызвать процедуру Π_1 ($\vec{X}_0$, 1).

Шаг 3. Выдать $\vec{X}^*_{opt}$ и F^*_{opt} как решение задачи.

Шаг 1 необходим для инициализации переменной F^*_{opt}.

Шаг 2 процедура Π_1 запускается однократно с входными переменными равными нулевому вектору и его числу p. В дальнейшем данная процедура рекурсивно запускает сама себя.

Шаг 3 после перебора всех векторов на выходе процедуры будем иметь решение задачи.

Процедура Π_1 имеет вид

Заголовок $\Pi_1(\vec{X}_T, p)$

Шаг 1. Определяем выполнимость всех ограничения при $\vec{X}_T$.

Шаг 2. Если хотя бы одно ограничение невыполнено, тогда выход из процедуры.

Шаг 3. Если $F_T > F^*_{opt}$, тогда $F^*_{opt} = F_T$, $\vec{X}^*_{opt} = \vec{X}_T$.

Шаг 4. Если $p = N+1$, тогда выход из процедуры.

Шаг 5. Присвоить $\vec{X} = \vec{X}_T$, у $\vec{X}$ изменить x_p на 0.

Шаг 6. Вызвать процедуру Π_1 ($\vec{X}$, p + 1).

Шаг 7. Присвоить $\vec{X} = \vec{X}_T$, у $\vec{X}$ изменить x_p на 1.

Шаг 8. Вызвать процедуру Π_1 ($\vec{X}$, p + 1).

Шаг 9. Выход из процедуры.

Шаг 1 необходим для проверки является ли $\vec{X}_T$ одним из решений задачи.

Шаг 3 необходим для обновления текущего оптимального решения.

Шаг 4 необходим для проверки глубины выборки.

Шаги 5, 6 и 7, 8 необходимы для формирования следующего текущего вектора и рекурсивного вызова функции.

Если ограничиться реализацией главного модуля и процедуры Π_1, то получим точный алгоритм. Но так как при большой размерности решаемой задачи время, за которое алгоритм выдаст точное решение, превысит допустимое время. Поэтому есть необходимость в разработке приближенного алгоритма. Для этого была разработана процедура Π_2.

Для описания процедуры Π_2 введем следующие переменные:

P_G – гарантированный прогноз вектора $\vec{X}_T$;

P_O – оптимистический прогноз вектора $\vec{X}_T$.

Процедура Π_2 интегрируется в процедуру Π_1 вместо Шага 3.

В этом случае процедура Π_1 с интегрированной процедурой Π_2 примет вид

Заголовок $\Pi_1(\vec{X}_T, p)$

Шаг 1. Определяем выполнимость всех ограничения при $\vec{X}_T$.

Шаг 2. Если хотя бы одно ограничение не выполнено, тогда выход из процедуры.

Шаг 3. Если $P_O \le F^*_{opt}$, тогда выход из процедуры.

Шаг 4. Если $P_G > F^*_{opt}$, тогда $F^*_{opt} = P_G$, $\vec{X}^*_{opt} = \vec{X}_T$.

Шаг 5. Присвоить $P_O + P_O * \varepsilon_{огр}$.

Шаг 6. Если $P_O \le F^*_{opt}$, тогда выход из процедуры.

Шаг 7. Если $p = N+1$, тогда выход из процедуры.

Шаг 8. Присвоить $\vec{X} = \vec{X}_T$, у $\vec{X}$ изменить x_p на 0.

Шаг 9. Вызвать процедуру Π_1 ($\vec{X}$, p + 1).

Шаг 10. Присвоить $\vec{X} = \vec{X}_T$, у $\vec{X}$ изменить x_p на 1.

Шаг 11. Вызвать процедуру Π_1 ($\vec{X}$, p + 1).

Шаг 12. Выход из процедуры.

Шаг 5 необходим для коррекции оптимистического прогноза с учетом $\varepsilon_{огр}$. Необходимо отметить, что если $\varepsilon_{огр} = 1$, тогда мы получаем точное решение, а при уменьшении $\varepsilon_{огр}$ к 0 будут получены приближенные решения. Данный алгоритм позволяет, задавая значение $\varepsilon_{огр}$, получать решения с заданной погрешностью или за заданное время. Поэтому, корректно варьируя значением $\varepsilon_{огр}$ возможно получать решение задачи любой размерности за заданное время.

Заключение

Необходимо ответить, что при использовании способа групповой выборки с индивидуальной сегментацией коэффициент $К_{ИР}$ больше, чем при использовании способа групповой выборки без индивидуальной сегментации. Но время, необходимое для решения задачи определения оптимальной выборки при выборе различных способов выборки, различно. И помимо выбора способа выборки с максимальным значением коэффициента $K_{ИР}$, необходимо также учитывать то, что время, необходимое для определения оптимальной выборки, не должно превысить допустимое время $T_Д$. А так как для одной и той же очереди запросов время, необходимое для определения оптимальной выборки у способа групповой выборки без индивидуальной сегментации меньше, чем у способа групповой выборки с индивидуальной сегментации, то возможны случаи, когда не-

обходимо для нахождения оптимальной выборки выбирать способ групповой выборки без индивидуальной сегментации.

В различных системах на время обслуживания запросов из очереди накладывается временное ограничение, заключающееся в том, что оно не должно превышать некоторое значение $T_д$. Поэтому решение задачи обслуживания запросов, обеспечивающее максимальное значение коэффициента $K_{ИР}$, должно осуществляться с требуемой оперативностью, которую можно оценить вероятностью решения данной задачи за время, не превышающее $T_д$.

Список литературы

1. Буй Д.Б., Скобелев В.Г. Модели, методы и алгоритмы оптимизации запросов в базах данных // Компьютерные системы и информационные технологии. – 2014. – № 2 (66). – С. 43–58.

2. Listrovoy S.V., Tretiak V.F., Listrovaya E.S. Parallel algorithms of calculation process optimization for the boolean programming problems Engineering Simulation. – 1999. – Vol. 16. – P. 569–579.

3. Третьяк В.Ф., Пашнева А.А. Оптимизация структуры хранилища данных в узлах сети инфокоммуникационной сети облачного хранилища // Системы навигации и связи Полтавского национального университета имени Юрия Кондратюка. – 2017. – № 4(44). – С. 122–128.

4. Горобец В.В. Математические модели и алгоритмы оптимизации размещения данных биллинговых OLTP-систем: дис. ... канд. тех. наук: 05.13.18 / ФГБОУ ВО «Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова. – Новочеркаск, 2014. – 190 с.

5. Федорин А.Н. Многокритериальные задачи ранцевого типа: разработка и сравнительный анализ алгоритмов: дис. ... канд. техн. наук: 05.13.18 / Нижегородский государственный университет им. Н.И. Лобачевского. – Нижний Новгород, 2010. – 132 с.

6. Цвященко Е.В. Модели процессов согласования реплик в базах данных NoSQL: дис. ... канд. техн. наук: 05.13.17 / Московский государственный технический университет им. Н.Э. Баумана. – Москва, 2016. – 157 с.

7. Андрианова Е.Г., Мельников С.В., Раев В.К. Диссипация и энтропия в физических и информационных системах // Фундаментальные исследования. – 2015. – № 8–2. – С. 233–238.

8. Квеглис Л.И. Диссипативные структуры в тонких нанокристаллических пленках: монография / Л.И. Квеглис, В.Б. Кашкин; отв. ред. В.Ф. Шабанов. – Красноярск: Сиб. федер. ун-т, 2011. – 204 с.

9. Патент на полезную модель № 69487, Украина, МПК G06 F15/00. Устройство для решения задач на графах / В.Ф. Третьяк, Д.Ю. Голубничий и др. – № u201113667; заяв. 21.11.2011; опубл. 25.04.2012; Бюл. № 8. – 6 с.