Теорія оптимальних рішень

УДК 519.85

В.Н. КУЗЬМЕНКО, Э.И. НЕНАХОВ

АЛГОРИТМ РЕШЕНИЯ КВАДРАТИЧНОЙ ЗАДАЧИ В PNK-МЕТОДЕ

Строится вычислительно эффективный алгоритм решения квадратичной подзадачи, решаемой на итерациях PNK-метода. При этом учитывается диагональность квадратичной матрицы, границы переменных, незначительность изменения подзадачи на последовательных итерациях. Приводятся результаты вычислительных экспериментов.

Введение. Ключевым моментом различных алгоритмов, использующих решение квадратичной подзадачи, есть эффективность решателя этой подзадачи [1–3]. Под решателем понимается алгоритм, реализующий метод решения и использующий особенности постановки подзадачи и ее данных. Поскольку практический интерес представляют задачи большой размерности [2, 3], то решатель должен обладать такими свойствами: он должен реализовывать вычислительно эффективный метод, быть приспособленным к конкретной постановке задачи, компактно представлять разреженные данные и быстро с ними оперировать, находить решение подзадачи с заданной точностью, в том числе быстро находить приближенные решения.

Методы решения квадратичной задачи в различных ее постановках разработаны достаточно глубоко [4–10]. Обширная библиография приведена в обзоре [11]. Поэтому в данной работе речь идет об эффективном использовании одного из методов активного набора в PNK-методе [12, 13]. Возможность такого использования обусловлена, в частности, сходством логики работы обоих методов. Оба метода используют наборы линейных ограничений, разделяя их на активное и неактивное подмножества, и в этом дополняют друг друга. PNK-метод на каждой итерации модифицирует квадратичную задачу и добавляет новые неактивные, но нарушенные ограничения, а квадратичный метод строит последовательность точек, сходящуюся к решению задачи.

Подзадача, решаемая на итерации PNK-метода, может быть записана следующим образом:

$$\min_{x,\,\xi_1,\xi_2} \{0.5 \left\| x - x_r \right\|^2 + h_k \xi_1 + h_k N_k\, \xi_2\}, \tag{1}$$

$$f(x_i) + (f'(x_i),\ x - x_i) \le \xi_1,\ \forall x_i \in X_k, \tag{2}$$

$$\varphi(x_i) + (\varphi'(x_i),\ x - x_i) \le \xi_2,\ \forall x_i \in X_k, \tag{3}$$

$$0 \le \xi_2,\ \ \underline{x} \le x \le \bar{x}, \tag{4}$$

где переменная ξ_1 с учетом ограничений (2) аппроксимирует целевую функцию $f(x)$; переменная ξ_2 с учетом ограничений (3) – обобщенную функцию ограничений $\varphi(x)$; X_k – текущее (на итерации k) множество точек аппроксимации; $x_r = \arg\min\{F_k(x) : x \in X_k\}$ – вычисленная точка минимума штрафной функции $F_k(x) = f(x) + N_k \max\{0; \varphi(x)\}$ на X_k; $f'(x)$, $\varphi'(x)$ – элементы субдифференциалов $\partial f(x)$, $\partial \varphi(x)$; N_k – штрафной множитель; h_k –множитель, выполняющий роль величины шага.

Запишем квадратичную задачу (1)–(4) в более общем виде

$$\min_{x,\xi} \{0.5 \cdot x^T x + d\,x + p\,\xi \mid Ax + G\xi \ge b\}. \tag{5}$$

В постановке (5) ограничения (4) учтены как общие, переменные ξ_1, ξ_2 собраны в вектор ξ, d и p – вектор-строки, собранные по линейной части (1).

Функция Лагранжа для задачи (5) имеет вид

$$L(x,\xi,\lambda) = 0.5 \cdot x^T x + d\,x + p\,\xi + \lambda(b - Ax - G\xi),\ \lambda \ge 0.$$

Необходимые условия оптимальности Каруша - Куна - Таккера имеют вид

$$\frac{\partial L}{\lambda_i} = \begin{cases} b_i - a^i x - g^i \xi = 0,\ \lambda_i \ge 0 \\ b_i - a^i x - g^i \xi < 0,\ \lambda_i = 0 \end{cases},\ i \in I = \{1,\ldots,m\}, \qquad \begin{matrix} \nabla_x L = x + (d - \lambda A)^T = 0 \\ \nabla_\xi L = (p - \lambda G)^T = 0 \end{matrix}, \tag{6}$$

где – a^i, g^i строки i матриц A и G; m – количество строк ограничений в (5).

Суть метода активного набора можно сформулировать как поиск множества $I_a^* \subset I$ ограничений задачи (5), которые в оптимальном решении выполнены как равенства. Для произвольного активного набора I_a переменные x, ξ, λ_a определяются решением следующей линейной системы, полученной из (6):

$$\begin{pmatrix} E & 0 & -A_a^T \\ 0 & 0 & -G_a^T \\ A_a & G_a & 0 \end{pmatrix} \begin{pmatrix} x \\ \xi \\ \lambda_a^T \end{pmatrix} = \begin{pmatrix} -d^T \\ -p^T \\ b_a \end{pmatrix}, \tag{7}$$

где A_a, G_a – подматрицы матриц A, G, b_a, λ_a – части векторов b, λ, соответствующие подмножеству I_a. Остальные переменные λ_i полагаются равными 0.

Оптимальность набора I_a проверяется по оставшимся условиям из (6).

Необходимым условием разрешимости системы (7) есть наличие в ней хотя бы одного ограничения, включающего переменные ξ_1, ξ_2, и не более n+2 ограничений из системы (5), где n – длина вектора x.

При поиске оптимального активного набора система (7) должна решаться многократно прямо и косвенно. Поэтому важно рационально изменять систему и пересчитывать ее решение, основываясь на предыдущем наборе и решении. Выполним преобразование пространства переменных x таким образом, чтобы система активных ограничений приняла вид

$$A_a x + G_a \xi = A_a Q y + G_a \xi = (L \;\; 0)\, y + G_a \xi = b_a ,$$

где Q – ортогональная матрица преобразования пространства $x = Qy$; $(L \;\; 0)$ – матрица, состоящая из нижней треугольной L и нулевой подматриц. Такое преобразование возможно, если ранг A_a равен числу ее строк, т.е. ограничение $\xi_2 = 0$ не активно.

В преобразованном пространстве изменяется вид градиентов функции Лагранжа по переменным y, λ, а именно:

$$\nabla_\lambda L = b - \begin{pmatrix} L & 0 \\ S & \end{pmatrix} y - G\xi ,$$

$$\nabla_y L = y + (dQ)^T - \begin{pmatrix} L^T & S^T \\ 0 & \end{pmatrix} \lambda^T , \qquad (8)$$

где $\begin{pmatrix} L & 0 \\ S & \end{pmatrix} = AQ$.

В таком представлении вектор y разделяется на компоненты, соответствующие I_a и нет, $y = (y_a \;\; y_N)$. Более того можно установить взаимнооднозначное соответствие между переменными y_a и ограничениями I_a.

Система (7) приобретает вид

$$\begin{pmatrix} E_a & 0 & 0 & -L^T \\ 0 & E_N & 0 & 0 \\ 0 & 0 & 0 & -G_a^T \\ L & 0 & G_a & 0 \end{pmatrix} \begin{pmatrix} y_a \\ y_N \\ \xi \\ \lambda_a^T \end{pmatrix} = \begin{pmatrix} -(dQ)_a^T \\ -(dQ)_N^T \\ -p^T \\ b_a \end{pmatrix} , \qquad (9)$$

и решается в следующем порядке: $y_a = L^{-1}(b_a - G_a\xi)$, $\lambda_a^T = L^{-1T}(y_a + (dQ)_a^T)$, $p^T = G_a^T\lambda_a^T = G_a^T L^{-1T}(y_a + (dQ)_a^T) = G_a^T L^{-1T}\left(L^{-1}(b_a - G_a\xi) + (dQ)_a^T\right)$. Из последнего находим $\xi = \left(G_a^T L^{-1T} L^{-1} G_a\right)^{-1}\left(G_a^T L^{-1T}\left(L^{-1}b_a + (dQ)_a^T\right) - p^T\right)$. Далее, подставляя ξ, находим y_a и λ_a^T. y_N считается независимо $y_N = -(dQ)_N^T$. Несмотря на некоторую громоздкость формул система решается эффективно, так как не используется явное обращение треугольной матрицы L, матрица $L^{-1}G_a$ имеет размер $|I_a| \times 2$, а матрица $\left(G_a^T L^{-1T} L^{-1} G_a\right)$ – 2×2.

Если ограничение $\xi_2 = 0$ активно, то оно добавляется к системе (9), а также добавляется столбец соответствующей двойственной переменной λ_{ξ_2}, содержащий единственный ненулевой элемент –1 в строке, соответствующей ξ_2. В этом случае первыми находятся переменные ξ_1 и λ_{ξ_2} из системы

$$\begin{pmatrix} \xi_1 \\ 0 \end{pmatrix} = \left(G_a^T L^{-1T} L^{-1} G_a\right)^{-1}\left(G_a^T L^{-1T}\left(L^{-1}b_a + (dQ)_a^T\right) - p^T + \begin{pmatrix} 0 \\ \lambda_{\xi_2} \end{pmatrix}\right).$$

Если активный набор не является оптимальным, то алгоритм выполняет его изменение путем добавления нарушенных ограничений из (5) и вывода ограничений, у которых двойственные переменные отрицательны. Порядок ввода, вывода ограничений и пересчета системы (9) зависит от конкретной реализации алгоритма. Но при любой реализации необходимо выполнить соответствующее изменение матриц Q и L. И если матрица L может иметь небольшую размерность, то матрица Q имеет размерность $n \times n$. Поэтому рассмотрим вопрос: как используя специфику постановки задачи можно уменьшить объем вычислений при изменении матриц Q и L.

Спецификой в постановке задачи (1)-(4) есть наличие границ на переменные и единичная квадратичная матрица. Если среди активных ограничений есть границы переменных и соответствующие ограничения записаны первыми, то равенство $A_a Q = (L \;\; 0)$ более детально представляется в виде

$$A_a Q = \begin{pmatrix} P \\ B \end{pmatrix}\begin{pmatrix} P^T & D \end{pmatrix} = \begin{pmatrix} E & 0 \\ BP^T & L' \;\; 0 \end{pmatrix} = (L \;\; 0),$$

где P – строки ограничений (5), соответствующие границам переменных, B – строки "обычных" ограничений, D – правая часть столбцов матрицы Q. Каждая строка в P содержит только один ненулевой элемент +1 или –1. Отсюда с учетом того, что $PD = 0$, следует, что строки Q, содержащие +1, –1 подматрицы P^T, не содержат других ненулевых элементов. Следовательно, при соответствующем (Π_x) изменении порядка переменных матрица Q может быть пред-

ставлена в виде $A_a Q = A_a \Pi_x \Pi_x^T Q = A_a \Pi_x \begin{pmatrix} E' & 0 \\ 0 & Q' \end{pmatrix}$, т.е. $Q = \Pi_x \begin{pmatrix} E' & 0 \\ 0 & Q' \end{pmatrix}$, где E' – диагональная матрица с элементами +1, –1.

При добавлении нового ограничения к системе активных ограничений матрица Q должна преобразоваться для сохранения треугольного вида результата произведения $A'_a Q_1 = \begin{pmatrix} A_a \\ a_+ \end{pmatrix} Q Q_+ = \begin{pmatrix} L & 0 \\ a_+ Q \end{pmatrix} Q_+ = \begin{pmatrix} L_{+1} & 0 \end{pmatrix}$, где A'_a –матрица увеличенного набора активных ограничений, a_+ – строка добавленного ограничения, Q_1 – соответственно измененная матрица Q, Q_+ – ортогональное преобразование, зануляющее "хвост" строки $a_+ Q$.

Если матрица L_{+1} имеет размер $r \times r$, то необходимо занулить $n - r$ элементов строки $a_+ Q$, начиная с конца. Зануление каждого элемента выполняется с помощью преобразования элементарного поворота, а именно, если s_j – последний ненулевой элемент строки, то преобразование q_j, задаваемое матрицей, полученной из единичной с подматрицей $\frac{1}{\sqrt{s_{j-1}^2 + s_j^2}} \begin{pmatrix} s_{j-1} & -s_j \\ s_j & s_{j-1} \end{pmatrix}$ в строках и столбцах $j-1$, j, даст 0 в позиции j строки $a_+ Q$. Таким образом, матрица Q_+ может быть представлена в виде произведения $Q_+ = q_n q_{n-1} \ldots q_{n-k+1} q_{n-k}$.

Другие, используемые в алгоритме повороты, задаются изменением знаков в подматрице 2×2: $\begin{pmatrix} s_{j-1} & s_j \\ s_j & -s_{j-1} \end{pmatrix}, \begin{pmatrix} -s_{j-1} & -s_j \\ -s_j & s_{j-1} \end{pmatrix}, \begin{pmatrix} -s_{j-1} & s_j \\ -s_j & -s_{j-1} \end{pmatrix}$.

Если $s_{j-1} = 0$, то второй и третий варианты поворота эквивалентны в зависимости от знака s_j перестановке столбцов $j-1$ и j в матрице Q. Но в этом случае более рационально найти ближайший к s_j ненулевой элемент s_γ строки $a_+ Q$, такой, что $r < \gamma < j$. Если такой s_γ существует, то преобразование поворота q выполняется для столбцов γ и j. Если нет, то для столбцов r и j.

При удалении ограничения из списка активных матрицу Q также необходимо преобразовать. Если удаляется γ-е ограничение из r активных, то при $\gamma < r$ в матрице L', получаемой из L после удаления строки γ, получится $r - \gamma$ ненулевых элементов справа от диагонали в строках $\gamma, \ldots, r-1$. Для зануления этих элементов необходимо применить один оператор поворота для каждой строки, начиная со строки γ. Соответствующее преобразование матрицы Q задается оператором $Q_- = q_\gamma q_{\gamma+1} \ldots q_{r-1}$, а преобразованная матрица $Q_1 = Q Q_-$.

Так как изначально матрица Q равна единичной и каждое ее преобразование выражается в виде ряда операторов поворота, то в общем виде на итерации k матрица Q может быть представлена как $Q_k = \prod_{t \in T_k} q_t$, где T_k – упорядоченное множество индексов операторов поворотов, задающее порядок операторов q_t в произведении, а операции с ее участием – как последовательное применение операторов q_t. Изменение матрицы Q – как дополнение множества T_k.

Опишем схематически алгоритм решения квадратичной задачи.

0. Инициализация: $I_a = \varnothing$, L – матрица 0×0, $T_0 = \varnothing$ ($Q = E$).

1. ***Start*** – поиск допустимого двойственного решения.

do; Решить (9). Найти x. $I_- := \{i \in I_a \mid \lambda_i < 0\}$,

if $I_- \neq \varnothing$ ***then*** $I_a := I_a \setminus I_-$. Изменить L, Q. Повторить блок. ***end***

end

2. ***Restart*** – поиск оптимального решения.

do while ($\Delta_{i_m} > 0$, $i_m = agr \max\{\Delta_i = b_i - a^i x - g^i \xi \mid i \in I \setminus I_a\}$);

$I_a := I_a \cup \{i_m\}$, $step = 0$.

do while ($step < 1$); $\lambda_{a0} = \lambda_a$, $x_0 = x$.

Изменить Q, L. Решить (9). Найти $\Delta\lambda_a = \lambda_a - \lambda_{a0}$, $\Delta x = x - x_0$.

$i_0 = agr \min\{\lambda_i / \Delta\lambda_i \mid i \in I_a \setminus \{i_m\},\ \Delta\lambda_i < 0\}$.

if i_0 существует ***then*** $step = \min\{-\lambda_{i_0} / \Delta\lambda_{i_0};\ 1\}$ ***else*** $step = 1$ ***end***

$\lambda_a = \lambda_{a0} + step \cdot \Delta\lambda_a$, $x = x_0 + step \cdot \Delta x$.

if $step < 1$ ***then*** $I_a := I_a \setminus \{i_0\}$ ***end***

end

end

Решение квадратичной подзадачи начинается либо с точки 1, либо с точки 2. Точка 1 используется вначале, а также при изменении параметров подзадачи – штрафного множителя N_k, шага – h_k, рекорда – x_r. Если эти параметры не изменились, то решение начинается с точки 2, так как при добавлении новых ограничений линеаризации (2), (3) по точке x_{k+1} активный набор I_a остается двойственно допустимым, а нарушенными только новые ограничения.

Решение системы (9) в блоке ***Restart*** находится существенно проще, чем в блоке ***Start***. Переменные y_i, $i \in I_a \setminus \{i_m\}$ не изменяются, выполнение нарушенного ограничения i_m достигается за счет изменения одной переменной y_{i_m}, которая определяет изменение переменной λ_{i_m}, что в свою очередь определяет изменение переменных λ_i, $i \in I_a \setminus \{i_m\}$.

Еще один важный момент уменьшения вычислительной сложности – обновление множества T_k. Множество T_k увеличивается с добавлением и выводом каждого активного ограничения, с изменением их порядка. Произведение a_+Q требует выполнения $4\times|T_k|$ операций умножения и $2\times|T_k|$ сложения. При значительном росте $|T_k|$ выполняется перестройка множества T_k и пересчет матрицы L, а именно: стартуя с L – 0×0 и $T_k=\varnothing$ имитируется добавление активных ограничений I_a, начиная с границ переменных. При этом отсеиваются все промежуточные операторы поворота, связанные выводом ограничений из списка активных и изменением их порядка в этом списке, матрица L становится более разреженной за счет уменьшения ошибок вычислений.

Результаты вычислительных экспериментов. Вычислительные эксперименты проводились с целью демонстрации влияния использования специфики внутренней квадратичной задачи на общее время решения задачи в зависимости от общего количества переменных и количества переменных, лежащих на границе в оптимальном решении. Результаты вычислений приведены в таблице.

Результаты с использованием специфики квадратичной подзадачи			Результаты без использования специфики квадратичной подзадачи		
Размерность задачи, n	Кол-во переменных на границе	Время решения, с	Размерность задачи, n	Кол-во переменных на границе	Время решения, с
10	0	0.5	10	0	0.6
10	5	0.1	10	5	0.5
100	0	3.9	100	0	4.3
100	90	0.9	100	90	4.2
1000	300	22.6	1000	300	62.2
1000	700	10.3	1000	700	68.7
5000	100	198.5	5000	100	328.5
5000	3000	28.5	5000	3000	362.9

Заключение. В результате выполненной работы значительно улучшена эффективность работы PNK-метода. Показано, что учет специфики квадратичной подзадачи, решаемой на каждой итерации, позволяет значительно уменьшить объем вычислений на итерациях метода.

В.М. Кузьменко, Е.І. Ненахов

АЛГОРИТМ РОЗВ'ЯЗУВАННЯ КВАДРАТИЧНОЇ ЗАДАЧІ У PNK-МЕТОДІ

Будується чисельно ефективний алгоритм розв'язування квадратичної підзадачи, яку треба розв'язувати на ітераціях PNK-методу. При цьому враховується діагональність квадратичної

матриці, границі змінних, незначна зміна підзадачі на послідовних ітераціях. Наводяться результати обчислювальних експериментів.

V.M. Kuzmenko, E.I.Nenakhov

AN ALGORYTHM FOR SOLVING QUADRATIC PROBLEM IN PNK-METHOD

Computationally effective algorithm for solving quadratic subproblem on iteration of PNK-method is built. Diagonal property of quadratic matrix, bounds on variables, small change of subproblem are took into account. Results of computational experiments are given.

1. *Murray W., Prieto F.J.* A Sequential Quadratic Programming Algorithm Using an Incomplete Solution of the Subproblem // SIAM J. Optim. – 1995. – **5**, N 3. – P. 590–640.
2. *Boggs P.T., Tolle J.W.* Sequential quadratic programming for large-scale nonlinear optimization // J. Computational and Applied Mathematics. – 2000. – N 124. – P. 123–137.
3. *Kim S.J., Koh K., Lustig M., Boyd S.* An Interior-Point Method for Large-Scale L1-Regularized Least Squares // J. of selected topics in signal processing. – 2007. – **1**, N 4. – P. 606–617
4. *Dax A.* The gradient projection method for quadratic programming // Institute of Mathematics Report. – Jerusalem: The Hebrew University of Jerusalem, 1978. – P. 124–149.
5. *Gill P.E., Murray W.* Numerically Stable Methods for Quadratic Programming // Math. Progr. – 1978. – **14**(3). – P. 349–372.
6. *Goldfarb D., Idnani A.* A numerically stable dual method for solving strictly convex quadratic programs // Math. Progr. – 1983. – N 27. – P. 1–33.
7. *Gill P.E.,Gould N.I.M., Murray W., Saunders M.A., Write M.H.* A weighted Gram-Schmidt method for convex quadratic programming // Math. Progr. – 1984. – **30** (2). – P. 176–195.
8. *Powell M.J.D*. On the quadratic programming algorithm of Goldfarb and Idnani // Mathematical Programming Studies. – 1985. – N 25. – P. 46–61.
9. *Костина Е.А., Костюкова О.И.* Алгоритм решения выпуклой квадратичной задачи с линейными равенствами и неравенствами // ЖВМиМФ. – 2001. – № 42. – С. 1012 – 1026.
10. *Schittkowski K.* QL: A Fortran code for convex quadratic programming - user's guide // Report, Department of Mathematics, University of Bayreuth, 2003. – P. 64–71.
11. *Gould N. I. M., Toint Ph.L.*, A Quadratic Programming Bibliography. – 2001. – http://www.optimization-online.org/DB_HTML/2001/02/2 85.html
12. *Пшеничный Б.Н., Ненахов Э.И., Кузьменко В.Н.* Комбинированный метод решения общей задачи выпуклого программирования // Кибернетика и системный анализ. – 1998. – № 4. – С. 121–134.
13. *Кузьменко В.Н., Бойко В.В.* О применении комбинированного метода выпуклого программирования // Теорія оптимальних рішень. – К.: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2003. – С. 19–24.

Получено 14.04.2011