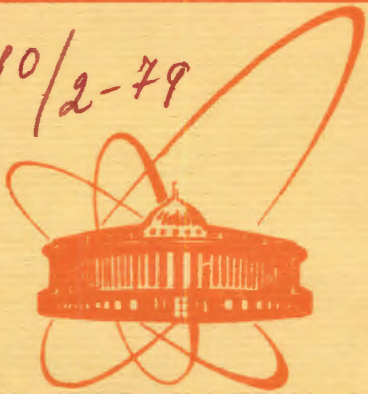

объединенный
институт
ядерных
исследований
дубна

Р11 -12336

Н.И.Янев

МИНИМИЗАЦИЯ ПСЕВДОБУЛЕВСКОЙ ФУНКЦИИ МАКСИМУМА

1979

P11 -12336

Н.И.Янев

МИНИМИЗАЦИЯ ПСЕВДОБУЛЕВСКОЙ ФУНКЦИИ МАКСИМУМА

Направлено в "Известия АН СССР, сер. Техническая кибернетика"

Янев Н.И.

Р11 - 12336

Минимизация псевдобулевской функции максимума

Описан алгоритм решения задачи минимизации функции максимума на множестве двоичных переменных, связанных ограничениями типа "Общие верхние границы" ($\Sigma x_{ij} = 1$). Поскольку максимум берется на множестве линейных функций, алгоритм может быть использован для решения задачи линейного целочисленного программирования при упомянутых ограничениях. Приводятся результаты решения на ЭВМ CDC-6500 ряда задач указанного класса, по которым можно судить об эффективности предложенного алгоритма. Программная реализация алгоритма выполнена на языке ФОРТРАН IV.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1979

Janev N.I.

P11 - 12336

Minimizing the Pseudoboolean Maximum Function

An algorithm is presented for solving a problem of minimizing a maximum of linear functions over a set of binary variables and "general upper bounds" constraints ($\Sigma x_{ij} = 1$). The algorithm can be used for solving linear integer programming problems containing above-mentioned constraints. Results are given from solving numerous problems on a CDC-6500 computer with the FORTRAN realization of suggested algorithm. The algorithm effectiveness is demonstrated.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1979

Ряд полезных практических задач может быть сформулирован следующим образом:

В n -мерном линейном пространстве с равномерной нормой $\|x\| = \max_i |x_i|$, где $x = (x_1, x_2, \ldots x_n)$, заданы m конечных подмножеств. Требуется найти по одному вектору из каждого подмножества, так чтобы норма их суммы была минимальна.

Одна из возможных интерпретаций этой задачи выглядит так: пусть для выполнения m работ (порядок выполнения несуществен) необходимо потребление некоторого однородного ресурса. Потребность в ресурсе для i -ой работы в n дискретных моментов времени задана упорядоченным набором из n чисел. Из каждого набора генерируются при помощи циклического сдвига вправо t_i наборов, задающих потребление ресурса в зависимости от начала выполнения i -ой работы. Необходимо найти такое расписание выполнения работ, которое минимизирует максимальное моментное потребление ресурса.

Меняя определения ресурса и работ, можем получить разные практические задачи. Например, в /1/ потребляемым ресурсом является оперативная память ЭВМ, работающая в мультипрограммном режиме, а работами - набор программ математического обеспечения некоторой системы. В /2/ указан ряд других прикладных задач из рассмотренного класса. Там же описан алгоритм решения этой задачи, реализующий схему метода "ветвей и границ" с односторонним ветвлением.

В данной работе задача решается как частный случай решения более общей задачи целочисленного программирования с двоичными переменными и ограничениями типа "общие верхние границы" ($\sum_j x_{ij} = 1$).

Для решения этой задачи приводится алгоритм типа "ветвей и границ" со спецификой в схеме разбиения множества возможных решений. Эффективность алгоритма и особенности его практической реализации обсуждаются в 3 и 4 разделах.

I. Математическая модель задачи

Пусть $S_1, S_2 \dots S_m$ – заданные конечные множества неотрицательных векторов n-мерного линейного пространства с равномерной нормой. В дальнейшем будем решать следующую задачу:

$$\| x_1 + x_2 + \dots x_m \| \longrightarrow \min, \qquad (1)$$

$$x_i \in S_i, \quad i = \overline{1,m}. \qquad (2)$$

Выразим (1),(2) аналитически. Пусть $|S_1| + |S_2| + \dots |S_m| = \ell$ и векторы множеств S_i пронумерованы числами $1,2,\dots \ell$ так, что векторы с номерами $1 + |S_1|$ принадлежат S_2; $c\,|S_1|+1 + |S_1|+|S_2|-$ $- S_2$ и т.д. Тогда задачу (1)-(2) можно представить как задачу целочисленного программирования:

$$V(x) \equiv \max_{j \in \{1,2,\dots n\}} \sum_{i=1}^{\ell} a_{ij} x_i \longrightarrow \min, \qquad (3)$$

$$\sum_{i \in S_j} x_i = 1, \quad j=\overline{1,m}, \qquad (4)$$

$$x_i \in \{0,1\}, \quad i=\overline{1,\ell}, \qquad (5)$$

где $a_{ij} \geqslant 0$, $j=\overline{1,n}$ – компоненты i-го вектора; множества S_i (после нумерации векторов) задают соответствующее разбиение множества $\{1,2,\dots \ell\}$; x_i – двоичные переменные, задающие условия выбора i-го вектора. В дальнейшем, без ограничения общности, числа a_{ij} будем считать целыми.

Известно, что эффективность метода направленного перебора для решения задач этого класса существенно зависит от возможности получения хороших оценок функции $V(x)$ на подмножествах множества возможных решений. В настоящее время для этой цели используются два достаточно универсальных подхода:
первый – при помощи решения задач линейного программирования;
второй – при помощи лагранжевой релаксации.
В данном случае двойственная задача (3)-(5) формулируется как задача о максимизации функции:

$$L(u) = -\sum_{i=1}^{m} u_i + \min_{x \in X} \left\{ \max_j \left[\sum_{i \in S_1} (a_{ij}+u_1) x_i + \sum_{i \in S_2} (a_{ij}+u_2) x_i + \dots + \sum_{i \in S_m} (a_{ij}+u_m) x_i \right] \right\},$$

где X – множество бинарных n-мерных векторов.
Существенным затруднением здесь является решение задачи о вычислении $L(u)$. Очевидно, что эта задача имеет такой же порядок трудности, как и задача (3)-(5). Поэтому, несмотря на то, что $\max_u L(u)$ дает лучшие оценки для $\min V(x)$, чем те, которые можно получить при помощи ослабления условия (5), в изложенном ниже алгоритме для получения оценки целевой функции используется первый подход.

Для этой цели сформулируем задачу (3)-(5) как задачу линейного целочисленного программирования:

$$\xi \longrightarrow \min, \qquad (6)$$

$$\sum_{i=1}^{\ell} a_{ij} x_i - \xi \leq 0, \quad j=\overline{1,n}, \qquad (7)$$

$$\sum_{i \in S_j} x_i = 1, \quad j=\overline{1,m}, \qquad (8)$$

$$x_i \in \{0,1\}, \quad i=\overline{1,\ell}. \qquad (9)$$

Для компактного изложения алгоритма решения задачи (6)-(9) будем использовать обозначения:

a_i —— i – столбец условия (7); $M_j = M_{j-1} \setminus j$ для $j=\overline{1,m}$ и $M_0 = \{1,2,\dots m\}$; $L_j = L_{j-1} \setminus S_j$ для $j=\overline{1,m}$ и $L_0 = \{1,2,\dots \ell\}$. Для фиксированных $j \in M_0, i \in S_j$ и заданного n-мерного вектора b обозначим через $Z_{ij}(b)$ задачу:

$$\xi \longrightarrow \min,$$
$$\sum_{k \in L_j} a_k x_k - \xi \leq -a_i - b,$$
$$\sum_{k \in S_t} x_k = 1, \quad t \in M_j,$$
$$0 \leq x_k \leq 1, \quad k \in L_j.$$

Оптимальное решение $Z_{ij}(b)$ обозначим через $\xi_{ij}(b)$. Ясно, что Z_{ij} является ослаблением подзадачи, которая получается из (6)-(9) при фиксировании некоторых переменных x_i, так

чтобы выполнялась часть условий (8) и (9). Тогда вектор b является суммой столбцов a_i, соответствующих фиксированным $x_i=1$. Оптимальное значение целевой функции задачи (6)-(9) обозначим через ξ^*.

2. Алгоритм решения задачи (6)-(9)

Шаг I [начальный] . Найти $R \geq \xi^*$ (см.раздел 3). Для $i \in L_0$ положить:

$$x_i = z_i = 0; \quad b = (0,0,\ldots 0); \quad j=0.$$

Шаг 2 [спуск] $j=j+1$; если $j=m-1$, перейти к шагу 5, иначе:

для всех $i \in S_j$ решить задачи $Z_{ij}(b)$

Шаг 3 [выбор подзадачи] . Найти $t(i^*) = \min\limits_{i \in S_j}\{(1-z_i)\xi_{ij}(b)\}$;

если $t(i^*) \geq R$, перейти к шагу 4, иначе: $b=b+a_{i^*}$, $x_{i^*} = 1$, $z_{i^*} = 1$, перейти к шагу 2.

Шаг 4 [возврат] $j=j-1$; если $j=0$, конец, иначе: $x_i=z_i=0$ для $i \in S_{j+1}$; найти i^* так, чтобы $x_{i^*} = 1$ для $i^* \in S_j$; $b=b-a_{i^*}$, $x_{i^*}=0$ и перейти к шагу 3.

Шаг 5 [возможное решение] . Найти $i^* \in S_{m-1}$, для которого целочисленная задача $Z_{i^* m-1}(b)$ имеет минимальное решение z^*; если $z^* < R$, положить $R = z^*$, отпечатать x и z^* и перейти к шагу 4 ; иначе перейти к шагу 4.

Основная особенность алгоритма, в которой и заключается его отличие от известных схем /3/ "ветвей и границ" для решения задач линейного целочисленного программирования состоит в том, что шаг разбиения выбранной подзадачи на подзадачи-наследники предшествует шагу ослабления этой же подзадачи. В данном случае, так как переменные-бинарные и связаны ограничениями типа "общих верхних границ", это дает возможность получения эффективных оценок снизу для целевой функции не только в смысле их близости к оптимальным решениям соответствующих подзадач, но и по числу операций, затраченных на их получение. Последний эффект получается, если в шаге 2 задачи Z_{ij} (для фиксированного j , $i \in S_j$) решаются одновременно. Такую возможность дает двойственный симплексный алгоритм, если его применить к решению задачи с постоянной левой частью в условии (7) и с более чем одним столбцом $b+a_i$ в правой части. На деле это означает, что в симплексных преобразованиях участвуют все столбцы $b+a_i$, для которых не получена

двойственная допустимость (неотрицательность базисных переменных, т.е. $x_B = B^{-1}(b+a_i) \geq 0$).

Переменные z_i служат для индикации запрета фиксации $x_i=1$. При этом предполагается, хотя в алгоритме явно не указано, что в шаге 2, если в процессе решения задач z_{ij} для некоторой итерации значение целевой функции превысит R , полагается $z_i=1$ и столбец $b+a_i$ из дальнейших преобразований исключается. Аналогично поступаем и в случаях, когда решение задачи Z_{ij} целочисленно.

В шаге 3 из списка подзадач, подлежащих решению, выбирается та, которая имеет минимальную оценку. Считается, что такой выбор подзадачи увеличивает вероятность достижения оптимального решения.

Иногда применяется и альтернативная стратегия выбора подзадачи с максимальной оценкой, увеличивающая вероятность усечения множества возможных решений. Для этого необходимо модифицировать шаг 3 следующим образом: заменить min на max и, если $t(i^*) \geq R$, положить $z_{i^*} = 1$ и перейти к шагу 3.

Принцип одновременного решения задач Z_{ij} , предложенный в шаге 2, позволяет существенно уменьшить объем памяти для сохранения промежуточной информации, которая необходима для осуществления ветвления при небольшом увеличении времени для решения задач. Дело в том, что для программной реализации алгоритма необходимо хранить только векторы x, z, b и a_i . Если векторы множества S_j получаются по некоторому рекуррентному правилу (см. задачу во введении), такое сжатие информации позволяет решать задачи большой размерности в объеме оперативной памяти.

3. Приближенное решение задачи (6)-(9)

Для получения хорошего начального приближения для ξ^* (см. шаг I алгоритма), а также для возможного улучшения возможных решений, получаемых в ходе решения задачи (6)-(9) по любому алгоритму, можно использовать следующую процедуру:

Пусть $\bar{x}$ – любое базисное решение системы (8),(9). Обозначим через $B = \{i_1, i_2, \ldots i_m\}$ множество индексов компонентов $\bar{x}$, для которых $x_{i_k} = 1$. Очевидно, что $i_k \in S_k, k=\overline{1,m}$. Тогда функцию (3) можно выразить через внебазисные переменные:

$$V(\bar{x}) = \Big\| \sum_{k=1}^{m} a_{i_k} + \sum_{i \in S_1 \setminus i_1} (a_i - a_{i_1}) x_i + \ldots + \sum_{i \in S_m \setminus i_m} (a_i - a_{i_m}) x_i \Big\| . \qquad (10)$$

Пусть $D=\{j \mid \sum_{k=1}^{m} a_{i_k j} = V(\bar{x})\}$. Из (I0) видно, что $V(\bar{x})$ можно уменьшить, если существует $i^* \in S_{j^*}$, для которого:

$$a_{i^* j} - a_{i_k j} < 0 \quad , \; j \in D \; , \tag{II}$$

$$\max_{j \in M_0 \setminus D} \left\{ \sum a_{i_k j} + (a_{i^* j} - a_{i_k j}) \right\} < V(\bar{x}) . \tag{I2}$$

Тогда новое решение $\bar{\bar{x}}$ получается из $\bar{x}$, если положим $x_{i^*} = 1$, т.е. из базиса B исключается $i_k \in S_{j^*}$, а на его место входит i^* . Выражение для $V(\bar{\bar{x}})$ получается из (I0), если: столбец $a_{i^*} - a_{i_k}$ добавить к первому столбцу, а ко всем столбцам, соответствующим S_{j^*} , добавить $a_{i_k} - a_{i^*}$.

Повторяя процедуру, пока условия (II),(I2)не удовлетворятся, получим локально-минимальное решение задачи (6)-(9). Здесь термин "локально-минимальное решение" употребляется в смысле возможного решения задачи (6)-(9), которое нельзя улучшить методом координатного спуска.

4. Структура программы и результаты машинного эксперимента

По описанному алгоритму создана программа на языке ФОРТРАН IУ. Кроме основного модуля, программа содержит 3 подпрограммы, выполняющие следующие функции:

первая подпрограмма – решает задачу линейного программирования по двойственному симплексному методу /4/,/5/;

вторая подпрограмма – генерирует задачи $z_{ij}(b)$ (см. п.I);

третья подпрограмма – по заданному начальному базисному решению системы (8),(9) отыскивает локально-минимальное решение по описанной в п.3 процедуре. Эта подпрограмма работает в двух режимах в зависимости от обращения.

В первом режиме начальное базисное решение генерируется случайным образом. По заданному в обращении параметру N в этом режиме получается N локально-минимальных решений, минимальное из которых задает начальное приближение к ξ^*. В экспериментах N выбиралось равным от 2 до 20 пропорционально наращиванию числа возможных решений. Во всех решенных задачах найденное этой подпрограммой начальное решение превышало оптимальное не более чем на 5% (см. таблицу).

Во втором режиме для начального базисного решения используется найденное в шаге 5 или в первой подпрограмме возможное решение. Интересными особенностями алгоритма оказались:

I) возможные решения, получаемые в шаге 5 или в шаге 2, почти во всех случаях были локально -минимальными;

2) число найденных возможных решений не превышало 5, что объясняется быстрым нахождением оптимума и хорошим начальным приближением.

Машинные эксперименты проводились на ЭВМ CDC-6500 . Результаты сведены в таблицу, где:

- m – число множества S_j;
- ℓ – число двоичных переменных ($\sum |S_j|$);
- n – размерность векторов a_i (этот параметр не является показательным для эффективности алгоритма, но влияет на время вычислений);
- $\xi_n(i)$ – начальное приближение, найденное третьей подпрограммой для i обращений;
- ξ – оптимальное решение задачи;
- L – число найденных возможных решений;
- L_1 – число улучшенных возможных решений;
- P – эффект одной симплексной итерации, т.е. $\mu N/R$, где N – число возможных решений, усеченных при помощи решения задач линейного программирования, R – общее число симплексных итераций до решения задачи, μ – коэффициент отношения числа арифметических операций, затрачиваемых на одну итерацию, к числу операций для прямого получения возможного решения;
- t – среднее время решения одной задачи в сек.;
- t_1 – среднее время для получения оптимального решения в сек.

Для тестов с номерами I и 2 решено по 2 задачи. Остальные результаты усреднены по 5-ти случайно генерированным задачам.В примерах I и 2 $i=20$, а в остальных – $i=10$.

Были проведены и эксперименты для установления сравнительной эффективности данного алгоритма по отношению к алгоритмам (общего назначения) решения линейно-целочисленных задач. Для 5 решенных случайно генерированных задач время работы данной программы оказалось в среднем в 30 раз меньше, чем время работы программы INTPL /6/, решающей общую задачу частично-целочисленного линейного программирования. До создания данного алгоритма были

испробованы возможности применить к решению задачи (6)-(9) и некоторые известные способы решения целочисленных задач. Внизу перечислены самые существенные недостатки, обнаруженные при этом.

- метод отсечения /3/ - апробированы отсечения типа $S=-f_{i0}+\sum_{j\in R} f_{ij}x_j$, $S \geq 0$ и $\sum_{j\in R} x_j=1$ (R - множество внебазисных переменных). Была установлена практическая несходимость к оптимальному решению даже для задач с небольшими размерностями (число переменных не больше 29);

- алгоритм Балаша /7/ - установлено, что замыкание активных вершин дерева подзадач является маловероятным событием;

- отсечение при помощи решения двойственной задачи /8/ - очень трудно вычисляется лагранжиан;

- спуск по обобщенному градиенту /8,9/ - здесь трудности аналогичны тем, которые встречаются при применении результатов, полученных для непрерывных задач к дискретным задачам;

- уменьшение размерности задачи при помощи заменяющих ограничений /3/ - замена m ограничений (8) одним ограничением настолько расширяет область ограничений ослабленной задачи, что оценки, полученные при помощи решения задач линейного программирования, становятся неиспользуемыми.

Полученные результаты и отмеченные трудности только подтверждают, что создание алгоритмов, разумно учитывающих специфику решаемых дискретных задач, целесообразно и перспективно.

Таблица

Тест №	m	ℓ	n	$\frac{\xi_{n(i)}-\xi}{\xi}100$	L	L_1	$P=\mu N/R$	t	t_1
I	90	270	I0	4	4	2	$2.I0^6$	2047	2I0
2	30	I80	I0	5	5	0	$2.I0^6$	II80	570
3	20	I40	20	4	6	0	$I.5I0^4$	II40	II2
4	40	I30	5	4.5	4	0	$2.I0^{II}$	750	360
5	40	I25	5	5	6	0	$>I0^{I5}$	750	200
6	40	I20	5	3	5	0	$>I0^{I5}$	520	I80
7	I0	70	20	2	I	I	$I0^4$	300	50
8	I0	60	20	I	2	0	$I0^4$	I50	95

Литература

I. Арнаудов Д.Д., Иванчев М.Д. ОИЯИ, РII-I0I59, Дубна, I976.

2. Иванчев М.Д., Янев Н.И., Митев Н.Г. Реализация на метода " разклоняване и ограничеване" за една комбинаторна оптимизационна задача. Системи и управление. София, № 2, I974.

3. Garfinkel R.S., Nemhauser G.L. Integer programming, 1972.

4. Balinski M.L., Gomory R.E. A mutual Primal-Dual Simplex Method . Recent Advances in Math. Program. 1963.

5. Thucker A.W. Combinatorial theory Underling Linear Programs . Recent Advances in Math. Program. 1963.

6. Иванчев М.Д., Митев Н.Г., Янев Н.И. Реализация метода "ветвей и границ" для решения общей задачи частично-целочисленного линейного программирования. ЖВМ, № 3, I976.

7. Geoffrion A.M. Integer programming by Implicit Enumeration and Balas' Method, SIAM Rev., 7, 1967.

8. Fisher M.L., Northup W.D., Shapiro J.F. Using duality to solve discrete optimization problems: theory and computational experiance, Math. progr. study 3, 1975.

9. Демьянов В.Ф. Минимакс: дифференцируемость по направлениям. Изд-во Ленинградского университета, I974.

Рукопись поступила в издательский отдел
26 марта I979 года.