

УДК 519.6

СИНТЕЗ СИМПЛЕКС-МЕТОДА И МЕТОДА ЛИНЕЙНОЙ КОРРЕКЦИИ В ЗАДАЧАХ ЛИНЕЙНОЙ ОПТИМИЗАЦИИ С НЕФОРМАЛИЗОВАННЫМИ ОГРАНИЧЕНИЯМИ**И. М. Соколинская¹**

Статья посвящена проблеме решения задач линейной оптимизации при наличии неформализованного ограничения. Для решения таких задач предлагается метод, базирующийся на синтезе симплекс-метода и метода линейной коррекции. Данный метод требует наличия *эксперта*, определяющего допустимость точки относительно неформализованного ограничения. Описывается реализация предложенного метода в виде комплекса программ на языке Си. Приводятся результаты численных экспериментов на реальных и модельных задачах, подтверждающие эффективность описанного подхода. Работа выполнена при финансовой поддержке РФФИ (проект № 03-01-00565).

1. Введение. В практике экономико-математического моделирования часто возникают задачи линейного программирования (ЛП) с ограничениями, не поддающимися полной формализации [4, 6, 7]. Рассмотрим случай, когда только одно ограничение не может быть формализовано. Данную ситуацию мы можем описать в виде следующей задачи *линейного программирования с неформализованным ограничением* (ЛПНО) в пространстве \mathbb{R}^n :

$$\max\{(c, x) \mid Ax \leq b, \varphi(x) \leq 0\}. \quad (1)$$

Здесь неравенство $\varphi(x) \leq 0$ играет роль неформализованного ограничения в том смысле, что нам не известен вид функции $\varphi(x)$. Будем предполагать, что $\varphi(x)$ принадлежит к классу аффинных функций. Мы рассматриваем случай, когда задача (1) имеет единственное решение:

$$\bar{x} = \arg \max\{(c, x) \mid Ax \leq b, \varphi(x) \leq 0\}.$$

Для работы с неформализованным ограничением мы можем использовать некоторый информационный ресурс, называемый *экспертом*. В качестве абстрактной модели эксперта рассмотрим функцию $\epsilon(x) = \text{sgn}(\varphi(x))$. С помощью функции эксперта $\epsilon(x)$ мы можем для любого $\tilde{x} \in \mathbb{R}^n$ определить, удовлетворяет ли \tilde{x} неформализованному ограничению или нет, используя следующую систему правил:

$$\begin{cases} \epsilon(\tilde{x}) = 1 & \Leftrightarrow \varphi(\tilde{x}) > 0; \\ \epsilon(\tilde{x}) = 0 & \Leftrightarrow \varphi(\tilde{x}) = 0; \\ \epsilon(\tilde{x}) = -1 & \Leftrightarrow \varphi(\tilde{x}) < 0. \end{cases}$$

На практике роль эксперта может играть нейронная сеть, экспертная система или человек. В общем случае мы должны допускать существование определенных ограничений на использование эксперта. Например, общее количество обращений к эксперту может быть ограничено или ответы эксперта являются верными с некоторой вероятностью.

Для решения задачи ЛПНО может быть использован следующий общий алгоритм [4], основанный на методах дискриминантного анализа (см. рис. 1). На первом шаге формируется начальный *набор образцов* $\mathfrak{B} = \{\mathfrak{A}, \mathfrak{B}\}$ путем задания пары конечных непересекающихся множеств \mathfrak{A} и \mathfrak{B} из пространства \mathbb{R}^n , удовлетворяющих условиям

$$\begin{cases} \epsilon(x) = 1, & \forall x \in \mathfrak{A}; \\ \epsilon(x) = -1, & \forall x \in \mathfrak{B}. \end{cases}$$

Существуют различные способы выбора точек для начального набора образцов. Это могут быть точки, полученные в результате предшествующего опыта, либо набор точек, заданных экспертом, либо множество точек, генерируемых некоторым специальным алгоритмом. Пример такого алгоритма будет приведен ниже.

¹ Челябинский государственный университет, ул. Бр. Кашириных, 129, 454021, г. Челябинск; e-mail: sokolinsky@acm.org

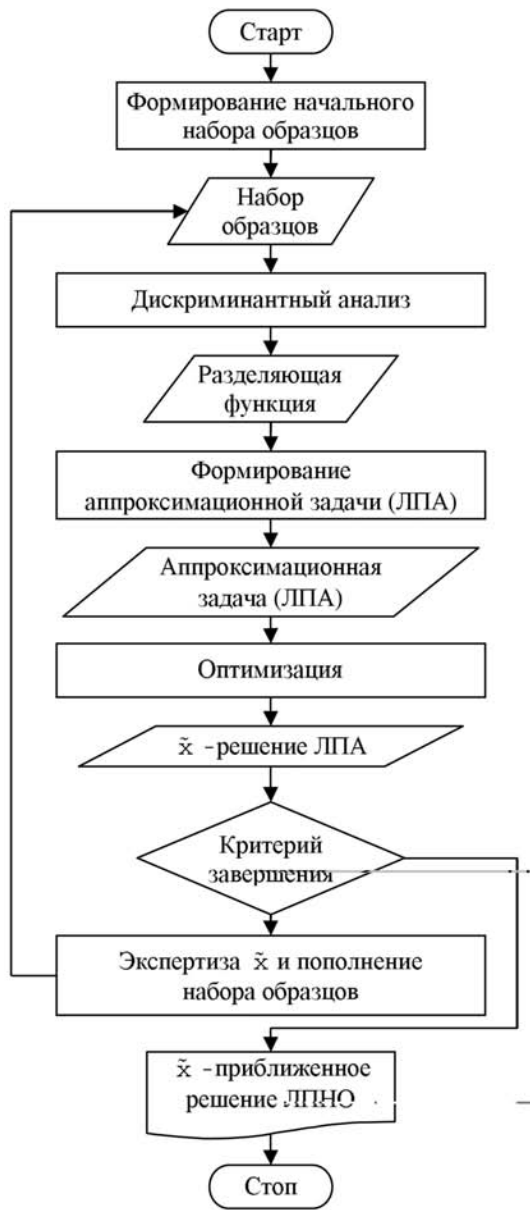


Рис. 1. Общий алгоритм решения задачи ЛПНО

На следующем шаге с помощью какого-либо метода дискриминантного анализа [5] строится аффинная функция $\tilde{\varphi}(x)$, разделяющая множества \mathcal{A} и \mathcal{B} :

$$\begin{cases} \tilde{\varphi}(x) > 0, & \forall x \in \mathcal{A}; \\ \tilde{\varphi}(x) < 0, & \forall x \in \mathcal{B}. \end{cases}$$

Такая функция существует, так как мы предположили, что неформализованное ограничение $\varphi(x) \leq 0$ определяется аффинной функцией $\varphi(x)$. В качестве метода дискриминантного анализа могут фигурировать, например, метод линейной коррекции [5, 9], метод комитетов [7] или фейеровские методы [2, 4].

Далее с использованием полученной разделяющей функции строится *аппроксимационная задача* линейного программирования (ЛПА):

$$\tilde{x} \in \arg \max \{c, x \mid Ax \leq b, \tilde{\varphi}(x) \leq 0\}. \quad (2)$$

Аппроксимационная задача решается в блоке *оптимизации* одним из методов линейного программирования. Здесь может использоваться, например, симплекс-метод [1, 13, 14] или методы фейеровского типа.

С помощью некоторого *критерия завершения* итерационного процесса проверяется, является ли полученное решение \tilde{x} допустимым приближением к точному решению \bar{x} . Ниже мы приведем конкретный пример критерия завершения.

Если точка \tilde{x} не удовлетворяет критерию завершения, то она направляется в блок *экспертизы*, где к ней применяется функция эксперта $\epsilon(x)$. Если $\epsilon(\tilde{x})=1$, то точка \tilde{x} добавляется в множество \mathcal{A} , если $\epsilon(\tilde{x}) = -1$, то точка \tilde{x} добавляется в множество \mathcal{B} . После этого описанный процесс повторяется применительно к расширенному набору образцов.

При проведении экспертизы и пополнении набора образцов могут возникать коллизии следующих трех типов:

- 1) $\epsilon(\tilde{x}) = 0$;
- 2) $\epsilon(\tilde{x}) = 1$ и $\tilde{x} \in \mathcal{A}$ (добавляемый образец уже присутствует в множестве \mathcal{A});
- 3) $\epsilon(\tilde{x}) = -1$ и $\tilde{x} \in \mathcal{B}$ (добавляемый образец уже присутствует в множестве \mathcal{B}).

В этом случае к \tilde{x} применяется *процедура рандомизации* [17]. Точки, полученные в результате процедуры рандомизации, вновь подвергаются экспертизе и используются для пополнения набора образцов.

Сходимость общего алгоритма, изображенного на рис. 1, была нами доказана в [17].

В данной работе рассматривается вариант описанного алгоритма, в котором при реализации блока дискриминантного анализа использован метод линейной коррекции, а в качестве метода линейной оптимизации выбран симплекс-метод. Такую комбинацию мы будем называть *алгоритмом СМ-ЛК*. В настоящей статье мы рассматриваем реализационные аспекты алгоритма СМ-ЛК и возможность его применения при решении практических задач. Статья организована следующим образом. В разделе 2 описываются подходы, позволяющие использовать алгоритм СМ-ЛК как основу эффективного вычислительного процесса. В разделе 3 обсуждается структура программного комплекса для решения задач линейного программирования с неформализованным ограничением на основе алгоритма СМ-ЛК. В разделе 4 приводится описание реальных и искусственных задач, использованных для проверки эффективности предложенных подходов, и обсуждаются результаты проведенных экспериментов. В заключении суммируются основные результаты и намечаются направления дальнейших исследований.

2. Реализация алгоритма СМ-ЛК. Реализация алгоритма СМ-ЛК в виде программы требует решения следующих задач:

- формирование начального набора образцов;
- выбор критерия завершения итерационного процесса;
- определение процедуры рандомизации;
- обеспечение быстрой сходимости к точному решению;
- решение проблемы погрешности вычислений.

В этом разделе мы обсудим возможные подходы к решению указанных задач.

2.1. Формирование начального набора образцов. Для эффективной работы алгоритма СМ-ЛК необходимо задать начальный набор образцов $\mathfrak{P} = \{\mathfrak{A}, \mathfrak{B}\}$. Очевидно, что множества \mathfrak{A} и \mathfrak{B} должны быть не пусты. Однако это не является достаточным условием для обеспечения эффективной работы алгоритма СМ-ЛК. Рассмотрим пример, изображенный на рис. 2. Здесь $D = \{x \mid Ax \leq b\}$ — допустимое множество точек формализованной части задачи ЛПНО; $P = \{x \mid \varphi(x) \leq 0\}$ — полупространство, определяемое неформализованным ограничением $\varphi(x) \leq 0$; $\tilde{P} = \{x \mid \tilde{\varphi}(x) \leq 0\}$ — полупространство, задаваемое разделяющей функцией $\tilde{\varphi}(x)$. Мы видим, что в данном случае $\tilde{\varphi}(x)$ является допустимой разделяющей функцией для множеств \mathfrak{A} и \mathfrak{B} . Тем не менее, аппроксимационная задача (2) в этом случае является некорректной (не имеет решения).

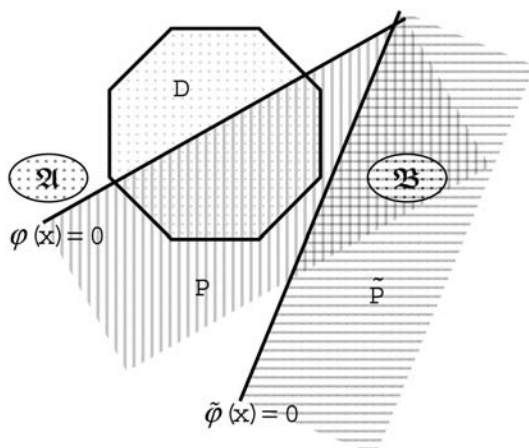


Рис. 2. Некорректный начальный набор образцов

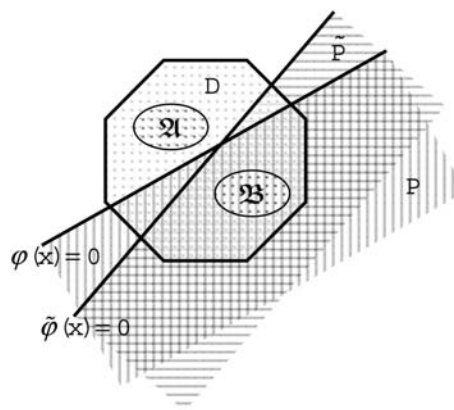


Рис. 3. Корректный начальный набор образцов

Очевидно, что эффективную работу алгоритма СМ-ЛК обеспечивает набор образцов $\mathfrak{P} = \{\mathfrak{A}, \mathfrak{B}\}$, удовлетворяющий следующим условиям (рис. 3):

$$\mathfrak{A} \subset D \setminus P; \quad \mathfrak{B} \subset D \cap (P \setminus H); \quad \mathfrak{A} \neq \emptyset, \quad \mathfrak{B} \neq \emptyset. \quad (3)$$

Здесь $H = \{x \mid \varphi(x) = 0\}$ — гиперплоскость, определяемая неформализованным ограничением. При этом мы предполагаем, что множества $D \setminus P$ и $D \cap (P \setminus H)$ имеют внутренние точки. В этом случае аппроксимационная задача (2) всегда будет иметь решение.

В соответствии с этим можно использовать следующий алгоритм для формирования начального набора образцов. Мы находим все вершины многогранника D , применяем к ним функцию эксперта $\epsilon(x)$ и формируем множества \mathfrak{A} и \mathfrak{B} по следующим правилам:

$$\mathfrak{A} = \{x \mid \epsilon(x) = 1\}; \quad \mathfrak{B} = \{x \mid \epsilon(x) = -1\}.$$

Точки, для которых $\epsilon(x) = 0$, отбрасываются. Если множества $D \setminus P$ и $D \cap (P \setminus H)$ имеют внутренние точки, то в результате мы получим набор образцов, удовлетворяющий условиям (3).

2.2. Критерий завершения итерационного процесса. Для реализации алгоритма СМ-ЛК в виде программы нам необходим некоторый критерий близости решения аппроксимационной задачи к точному решению задачи ЛПНО. Этот критерий необходим для завершения итерационного процесса.

Пусть $\tilde{x}^0, \tilde{x}^1, \dots, \tilde{x}^k$ — последовательность решений аппроксимационных задач, полученных при работе алгоритма СМ-ЛК. Тогда условие вида $\sum_{i=1}^k \|\tilde{x}^k - \tilde{x}^{k-i}\| < \epsilon$ может выступать в качестве искомого

критерия, где $\varepsilon > 0$ — заданное вещественное число, определяющее точность вычислений. Здесь l — фиксированное целое положительное число, являющееся параметром алгоритма СМ-ЛК. Корректность предложенного критерия обеспечивается следующей теоремой.

Теорема 1. Пусть \bar{x} — решение задачи ЛПНО, $\{\tilde{x}^k\}_{k=0}^\infty$ — последовательность решений аппроксимационных задач, порождаемых алгоритмом СМ-ЛК, i — фиксированное целое положительное число.

$$\text{Тогда } \lim_{k \rightarrow \infty} \|\bar{x} - \tilde{x}^k\| = 0 \quad \Rightarrow \quad \lim_{k \rightarrow \infty} \sum_{i=1}^l \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0.$$

Доказательство. Пусть $\lim_{k \rightarrow \infty} \|\bar{x} - \tilde{x}^k\| = 0$. Это означает, что $\forall \varepsilon > 0 : \exists k' : \forall k > k' : \|\bar{x} - \tilde{x}^k\| < \varepsilon$. Отсюда следует, что $\forall \varepsilon > 0 : \exists k' : \forall k > k' : \|\bar{x} - \tilde{x}^k\| < \varepsilon \cap \|\bar{x} - \tilde{x}^{k+i}\| < \varepsilon$, т.е. $\forall \varepsilon > 0 : \exists k' : \forall k > k' : \|\tilde{x}^k - \tilde{x}^{k+i}\| < 2\varepsilon$, откуда получаем $\lim_{k \rightarrow \infty} \|\tilde{x}^k - \tilde{x}^{k+i}\| = 0$. Это равносильно $\lim_{k \rightarrow \infty} \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0$, откуда

$$\text{следует } \lim_{k \rightarrow \infty} \sum_{i=1}^l \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0. \text{ Теорема доказана.}$$

При реализации алгоритма СМ-ЛК необходимо выбирать значения параметра l большими единицы, так как близость только одной пары точек может оказаться случайностью. С другой стороны, не следует брать значения параметра l слишком большими, так как это может существенно замедлить работу программы и, более того, привести к “зацикливанию” вычислительного процесса в результате накопления ошибок, связанных с погрешностью вычислений. Проведенные нами вычислительные эксперименты на модельных и реальных задачах ЛПНО показали, что в подавляющем большинстве случаев $l = 2$ дает хорошие результаты.

2.3. Рандомизация. На каждой итерации алгоритм СМ-ЛК строит очередную аппроксимационную задачу. Точка \tilde{x} , полученная в качестве решения данной аппроксимационной задачи, используется для пополнения набора образцов, в результате чего формируется аппроксимационная задача. При этом могут возникнуть коллизии трех типов. Коллизия первого типа возникает, когда \tilde{x} принадлежит гиперплоскости H , задаваемой неформализованным ограничением. В этом случае точка \tilde{x} не может быть отнесена ни к множеству \mathcal{A} , ни к множеству \mathcal{B} . Коллизии второго и третьего типов возникают, когда \tilde{x} уже присутствует в множествах \mathcal{A} или \mathcal{B} . Для разрешения этих коллизий мы используем следующую *процедуру рандомизации*.

Определим *случайное* отображение $\mathbf{r}^\rho : \mathbb{R}^n \rightarrow \mathbb{R}^n$, обладающее свойством $\|\mathbf{r}^\rho(x) - x\| < \rho$. Отображение $\mathbf{r}^\rho(x)$ порождает случайные точки внутри сферы радиуса ρ с центром в x . Вещественное число $\rho > 0$ является параметром алгоритма СМ-ЛК, задающим *радиус рандомизации*. На языке Си отображение $\mathbf{r}^\rho(x)$ реализуется путем применения генератора псевдослучайных чисел `random` к каждой координате точки x . Процедура рандомизации заключается в построении с помощью отображения $\mathbf{r}^\rho(\tilde{x})$ множества точек $\mathfrak{R}_m^\rho = \{\tilde{x}^1, \dots, \tilde{x}^m\}$, удовлетворяющих условию

$$\tilde{x}^i \notin \mathcal{A} \cup \mathcal{B} \cup H, \quad \|\tilde{x}^i - \tilde{x}\| < \rho \quad (i = 1, \dots, m).$$

Целое положительное число m является параметром алгоритма СМ-ЛК, задающим *мощность рандомизации*. Все точки множества \mathfrak{R}_m^ρ подвергаются экспертизе и используются для пополнения набора образцов на текущей итерации.

Очевидно, что увеличение мощности рандомизации m должно уменьшать количество итераций метода СМ-ЛК, необходимое для получения приближенного решения с заданной точностью. Однако при этом одновременно будет увеличиваться количество экспертиз, выполняемых в пределах одной итерации.

Увеличение радиуса рандомизации ρ на начальных итерациях также должно приводить к ускорению сходимости. В то же время на более поздних итерациях большой радиус рандомизации может заметно уменьшить скорость сходимости, так как рандомизированные точки будут порождаться далеко от точного решения, что, очевидно, приведет к ухудшению эффективности пополнения набора образцов.

Проблема выбора оптимальных значений параметров ρ и m при решении практических задач с помощью алгоритма СМ-ЛК будет нами исследована в разделах 4.1.2 и 4.1.3 соответственно.

2.4. Метод осцилляций. При применении метода СМ-ЛК к решению практических задач мы можем столкнуться с ограничением на максимально допустимое количество итераций. Данное ограничение может быть обусловлено, например, ограничениями на максимальное количество обращений к эксперту или на максимальное время нахождения решения. Для сокращения количества итераций и, соответственно, для ускорения сходимости метода СМ-ЛК нами было предложено усовершенствование исходного алгоритма, получившее название *метод осцилляций* [10].

Суть метода осцилляций (см. рис. 4) состоит в том, что на каждой итерации алгоритма СМ-ЛК

кроме аппроксимационной задачи (2) решается следующая *дополнительная* задача линейного программирования (ЛПД):

$$\hat{x} \in \arg \min\{(c, x) \mid Ax \leq b, \tilde{\varphi}(x) \geq 0\}. \quad (4)$$

Решение \hat{x} дополнительной задачи подвергается процедуре экспертизы и, в случае отсутствия коллизий, добавляется к набору образцов $\mathfrak{B} = \{\mathfrak{A}, \mathfrak{B}\}$. При возникновении коллизии к точке \hat{x} применяется процедура рандомизации.

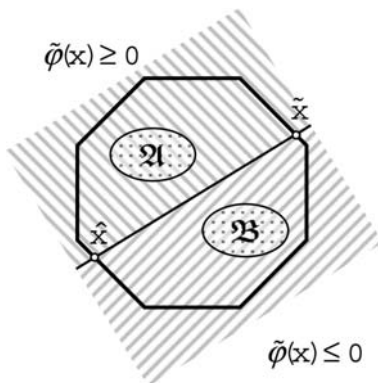


Рис. 4. Метод осцилляций

Метод осцилляций может значительно ускорить сходимость алгоритма СМ-ЛК на начальных итерациях, когда набор образцов не является еще достаточно представительным. Это было подтверждено проведенными нами вычислительными экспериментами (см. раздел 4). Однако на завершающем этапе вычислений метод осцилляций теряет свою эффективность и может привести к замедлению итерационного процесса. Следовательно, при реализации алгоритма СМ-ЛК необходимо на определенном этапе итерационного процесса отключать процедуру осцилляций. С этой целью мы ввели параметр λ , задающий максимально допустимое количество последовательных итераций, на которых возникли коллизии с решением \hat{x} дополнительной задачи. Если количество последовательных коллизий \hat{x} превысило λ , то процедура осцилляций отключается до завершения итерационного процесса. Проведенные нами вычислительные эксперименты на реальных и искусственных задачах показали, что в большинстве случаев значение $\lambda=2$ является хорошим выбором.

2.5. Проблема погрешности вычислений. Известно, что задача линейного программирования является устойчивой по всей совокупности данных, если оптимальные множества исходной и двойственной задач — ограничены [3]. Тем не менее, симплекс-метод является нестабильным по отношению к нулевым значениям. Это означает, что если некоторое нулевое значение, полученное в ходе выполнения итераций симплекс-метода, заменить на сколь угодно малое ненулевое значение, то это может привести к тому, что либо полученное решение не будет являться решением исходной задачи, либо исходная разрешимая задача превратится в неразрешимую. Замена нулевого значения на ненулевое происходит в компьютерных вычислениях в результате неточного представления вещественных чисел в ячейках памяти. В соответствии с этим при реализации симплекс-метода мы использовали следующие допуски на ошибки округления, применяемые вместо точной проверки на нуль:

$\delta_{a_{ij}}$ — допуск для элементов a_{ij} симплекс-таблицы: если после выполнения очередной итерации абсолютное значение элемента a_{ij} симплекс-таблицы становится меньше $\delta_{a_{ij}}$, то a_{ij} полагается равным нулю;

δ_{col} — допуск для выбора разрешающего столбца: при выборе разрешающего столбца не рассматриваются те элементы, значения которых по абсолютной величине меньше δ_{col} ;

δ_{row} — допуск для выбора разрешающей строки: при выборе разрешающей строки не рассматриваются те элементы, значения которых меньше δ_{row} .

В табл. 1 приведены величины указанных допусков в зависимости от разрядности машинного слова. Это — типичные значения для представления чисел в формате с плавающей точкой [8]. При выборе указанных значений мы исходили из того, что 32-разрядное слово позволяет хранить числа примерно с девятью десятичными значащими цифрами, а 64-разрядное — с семнадцатью. В языке Си 32-разрядному представлению обычно соответствует тип `float`, а 64-разрядному представлению — тип `double`. Различные величины допусков для одной разрядности отражают относительную важность допустимой ошибки.

В отличие от симплекс-метода, метод СМ-ЛК требует введения еще трех дополнительных допусков:

$\delta_{\text{лпа}}$ — допуск для решения \tilde{x} аппроксимационной задачи (2): если после выполнения очередной итерации метода СМ-ЛК выполнено $|\lceil \tilde{x}_j \rceil - \tilde{x}_j| < \delta_{\text{лпа}}$, то необходимо \tilde{x}_j присвоить значение $\lceil \tilde{x}_j \rceil$, $j = 1, \dots, n$ (здесь $\lceil x \rceil$ обозначает число, получающееся из x путем его округления до целого);

$\delta_{\text{лпд}}$ — допуск для решения \hat{x} дополнительной задачи (4): если после выполнения очередной итерации метода СМ-ЛК выполнено $|\lceil \hat{x}_j \rceil - \hat{x}_j| < \delta_{\text{лпд}}$, то необходимо \hat{x}_j присвоить значение $\lceil \hat{x}_j \rceil$ ($j = 1, \dots, n$);

δ_H — допуск для попадания \tilde{x} на гиперплоскость H , задаваемую неформализованным ограничением: если $\text{dist}(\tilde{x}, H) = \inf\{\|\tilde{x} - h\| : h \in H\} < \delta_H$, то к \tilde{x} необходимо применить процедуру рандомизации.

Последний допуск δ_H должен использоваться экспертом при квалификации новых образцов. В табл. 2 приведены значения дополнительных допусков в зависимости от разрядности машинного слова. Данные

Таблица 1

Величины допусков для симплекс-метода

Допуск	Число разрядов в машинном слове		
	32	60	64
δ_{aij}	10^{-6}	10^{-10}	10^{-11}
δ_{col}	10^{-5}	10^{-8}	10^{-9}
δ_{row}	10^{-3}	10^{-5}	10^{-6}

Таблица 2

Величины дополнительных допусков для метода СМ-ЛК

Допуск	Число разрядов в машинном слове		
	32	60	64
$\delta_{ЛПА}$	10^{-2}	10^{-3}	10^{-4}
$\delta_{ЛПД}$	10^{-1}	10^{-2}	10^{-2}
δ_H	10^{-3}	10^{-5}	10^{-6}

значения были получены нами экспериментальным путем.

Механизм допусков хорошо работает для задач ЛП, представленных *большими числами* (значительно превосходящими по абсолютной величине значения допусков). Однако если задача ЛП представлена *малыми числами* (сравнимыми с величинами допусков), то указанный прием не срабатывает, так как в результате погрешностей вычислений мы перестаем различать ненулевые и нулевые значения. В этом случае можно использовать модифицированный симплекс-метод с *LU-декомпозицией* [12], являющийся более стабильным по отношению к нулевым значениям (но и более медленным по сравнению со стандартным симплекс-методом). Другим решением проблемы малых чисел является *масштабирование* исходной задачи [19].

Эксперименты, проведенные на реальных и искусственных задачах ЛП, показали, что подход, основанный на использовании допусков, приведенных в табл. 1 и табл. 2, обеспечивает высокую стабильность алгоритма СМ-ЛК по отношению к нулевым значениям.

3. Программный комплекс. На основе описанной методики нами был спроектирован программный комплекс для решения задач линейного программирования с неформализованными ограничениями.

3.1. Модульная структура комплекса. Иерархическая модульная структура программного комплекса изображена на рис. 5. Программный комплекс включает в себя следующие модули: головной модуль; модуль универсального алгоритма; модуль экспертизы; модуль оптимизации; модуль дискриминации.



Рис. 5. Модульная структура программного процесса

Модуль ввода данных выполняет ввод исходных данных задачи ЛПНО из текстового файла. Входной файл данных задачи ЛПНО представляется в *формате LPNC*, разработанном нами специально для представления задач линейного программирования с неформализованными ограничениями. Формат LPNC является расширением промышленного *стандарта MPS* [8, 15], который используется для подготовки данных в большинстве коммерческих систем линейного программирования. Детально формат LPNC будет описан нами в разделе 3.2.

Модуль универсального алгоритма реализует общий алгоритм решения задачи ЛПНО, изображен-

ный на рис. 1. На фазе оптимизации модуль универсального алгоритма вызывает *метод оптимизации* из *модуля оптимизации*. Реализация метода оптимизации скрыта в модуле оптимизации. На фазе дискриминантного анализа модуль универсального алгоритма вызывает *метод дискриминантного анализа* из *модуля дискриминации*. Реализация этого метода скрыта в модуле дискриминации.

Модуль экспертизы реализует функцию эксперта. В качестве конкретной реализации модуля экспертизы может фигурировать интерфейс, обеспечивающий связь с *внешним* экспертом, нейро-сетевая программа или экспертная система.

Модуль оптимизации реализует метод линейной оптимизации. Программный комплекс предусматривает возможность использования следующих алгоритмов линейного программирования: симплекс-алгоритма с LU-разложением [12]; модифицированного симплекс-алгоритма [1, 13, 18]; алгоритма линейной оптимизации методом Еремина [2, 4]. Данные алгоритмы реализованы в программном комплексе в виде *сменных картриджей*, которые подключаются к модулю оптимизации на этапе компиляции.

Модуль дискриминации реализует метод дискриминантного анализа. Программный комплекс предусматривает возможность использования следующих алгоритмов дискриминантного анализа: линейной коррекции [5, 9]; метода комитетов [7]; алгоритма дискриминантного анализа на базе фейеровских отображений [2, 4]. Указанные алгоритмы также реализованы в виде сменных картриджей, которые подключаются к модулю дискриминации на этапе компиляции.

Технология картриджей позволяет сочетать различные комбинации алгоритмов, что дает возможность строить конфигурации программного комплекса, наилучшим образом учитывающие специфику каждой конкретной задачи.

3.2. Формат входных данных LPNC. В качестве файла исходных данных программный комплекс использует текстовый файл, содержащий описание задачи ЛПНО в *формате LPNC (Linear Programming with Nonformalized Constraints)*. Формат LPNC был нами разработан на базе формата *MPS (Mathematical Programming System)* [8, 15], который является промышленным стандартом для большинства коммерческих программных систем, ориентированных на решение задач математического программирования.

Для каждого образца в формате LPNC вводится уникальное имя (идентификатор), которое вместе с типом образца должно быть указано в секции ROWS. Тип образца записывается в позициях 2, 3, а имя образца — в позициях 5–12. Для спецификации типа образца формат LPNC предусматривает два дополнительных кода, приведенных в табл. 3. Координаты каждого образца задаются в секции COLUMNS таким же образом, как и коэффициенты матрицы формализованных ограничений.

Таблица 3

Коды для спецификации образцов

Код	Тип образца	Семантика
A	$x \in \mathfrak{A}$	Образец не удовлетворяет неформализованным ограничениям
B	$x \in \mathfrak{B}$	Образец удовлетворяет неформализованным ограничениям

При исследовании алгоритмов решения задач ЛПНО иногда полезно моделировать эксперта путем явного задания неформализованных ограничений. Для этих целей в формате LPNC предусмотрены специальные NFC-строки. Данные строки не описываются в секции ROWS LPNC-файла входных данных. Вместо этого в секции COLUMNS выделяются переменные, образующие NFC-множества. Карты-*маркеры*, которые указывают на начало и конец группы столбцов, принадлежащих отдельному NFC-множеству, имеют формат, приведенный в табл. 4. Карты данных имеют тот же формат, который определяется стандартом MPS. В них задаются ненулевые элементы, принадлежащие как обычным строкам, так и NFC-строкам. В качестве имени маркера конца может использоваться любой неиспользованный идентификатор.

Компоненты вектора ограничений каждой NFC-строки специфицируются в секции RHS как обычные строки. При этом в качестве имени строки в поле 3 или 5 указывается имя соответствующей NFC-строки.

Рассмотрим следующий пример задачи ЛПНО. Пусть формализованные ограничения задаются системой

$$\begin{cases} x_1 + x_2 \leq 3, \\ x_1 - x_2 \leq 1, \\ x_1 - 2x_2 \geq -1. \end{cases}$$

Пусть неформализованное ограничение задается неравенством $-1.25x_1 + 3x_2 \leq 3.75$, а целевая функция

Таблица 4

Формат NFC-множества

	Поле 1 (2-3)	Поле 2 (5-12)	Поле 3 (15-22)	Поле 4 (25-36)	Поле 5 (40-47)	Поле 6 (50-61)
Маркер начала	G, LE или E	<Имя NFC-строки>	'MARKER'		'NFCORG'	
Карты данных	G, L или E	<Имя столбца>	<Имя строки>	<Значение>	<Имя строки>	<Значение>
Маркер конца		<Имя маркера>	'MARKER'		'NFCEND'	

имеет вид $Q_{\max} = x_2$. Пусть набор образцов задается множествами $\mathfrak{A} = \{(1; 2)\}$, $\mathfrak{B} = \{(2; 1), (1; 0), (0; 1)\}$. Тогда входной файл задачи будет иметь вид, изображенный на рис. 6.

```

*          1          2          3          4          5          6
*23456789012345678901234567890123456789012345678901234567890
NAME
example
ROWS
L ue_1
L ue_2
G ue_3
N Qmax
A pa_1
B pb_1
B pb_2
B pb_3
COLUMNS
L nc_1          'MARKER'          'NFCORG'
x_1          ue_1          1.0          ue_2          1.0
x_1          ue_3          1.0          nc_1          -1.25
x_1          pa_1          1.0          pb_1          2.0
x_1          pb_2          1.0
x_2          ue_1          1.0          ue_2          -1.0
x_2          ue_3          -2.0          nc_1          3.0
x_2          pa_1          2.0          pb_1          1.0
x_2          pb_3          1.0          Qmax          1.0
nc_1_end    'MARKER'          'NFCEND'
RHS
b          ue_1          3.0          ue_2          1.0
b          ue_1          -1.0          nc_1          3.75
ENDATA
    
```

Рис. 6. LPNC-файл для задачи с неформализованным ограничением

4. Результаты экспериментов. Для исследования влияния различных параметров на время работы и количество итераций алгоритма СМ-ЛК нами была проведена серия вычислительных экспериментов. В качестве тестовых задач мы использовали как реальные, так и искусственные задачи. Искусственные задачи дают хорошую возможность для подбора оптимальных значений параметров алгоритма. Реальные задачи подтверждают практическую полезность предложенного метода.

Для проведения экспериментов мы использовали персональный компьютер под управлением Windows XP со следующими характеристиками: процессор — Intel Pentium IV 3 МГц, оперативная память — 2 Мб, компилятор — Borland C++, Version 5.02.

4.1. Эксперименты на искусственных задачах. Для исследования свойств алгоритма СМ-ЛК мы сконструировали специальную масштабируемую модельную задачу *Mod-n*.

4.1.1. Модельная задача Mod-n. Модельная задача Mod-n имеет вид

$$\begin{cases} x_1 - 2x_2 \leq 0, \\ x_1 - 2x_3 \leq 0, \\ \dots \\ x_1 - 2x_n \leq 0, \\ x_1 + 2x_2 \leq 2000, \\ x_1 + 2x_3 \leq 2000, \\ \dots \\ x_1 + 2x_n \leq 2000, \\ x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0, \end{cases}$$

$$Q_{\max} = x_1, \quad \varphi(x) = 2(n-1)x_1 - x_2 - x_3 - \dots - x_n - 800(n-1).$$

Здесь Q_{\max} — целевая функция, для которой необходимо найти максимальное значение. Функция $\varphi(x)$ моделирует эксперта в том смысле, что она явно задает неформализованное ограничение $\varphi(x) \leq 0$. Параметр n задает размерность пространства \mathbb{R}^n , в котором решается задача. Для всех $n \geq 2$ задача Mod-n имеет следующее единственное *точное* решение \bar{x} : $\bar{x}_1 = 720$, $\bar{x}_2 = \dots = \bar{x}_n = 640$. В качестве начального набора образцов мы берем множество всех вершин выпуклого многогранника, задаваемого формализованными ограничениями задачи Mod-n:

$$\mathfrak{A} = \{(1000, 500, \dots, 500)\}; \quad \mathfrak{B} = \{(0, \dots, 0); (0, 1000, 0, \dots, 0); \dots; (0, \dots, 0, 1000); (0, 1000, \dots, 1000)\}.$$

При этом $|\mathfrak{A}| = 1$, $|\mathfrak{B}| = n + 1$.

Задача Mod-n была нами использована для проведения вычислительных экспериментов, в ходе которых исследовались зависимости времени решения и количества итераций от размерности задачи, представительности набора образцов, радиуса и мощности рандомизации.

4.1.2. Влияние радиуса рандомизации на эффективность алгоритма СМ-ЛК. В первой серии экспериментов мы исследовали зависимость количества итераций и времени решения задачи Mod-n от радиуса рандомизации ρ . Этот параметр был нами введен в разделе 2.3. Мы решали задачу Mod-n для значений n , равных 40, 60 и 80. Результаты проведенных экспериментов представлены на рис. 7, 8.

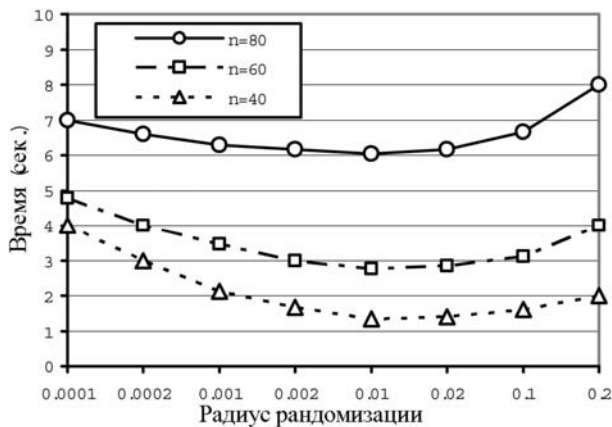


Рис. 7. Зависимость времени решения задачи от радиуса рандомизации ρ

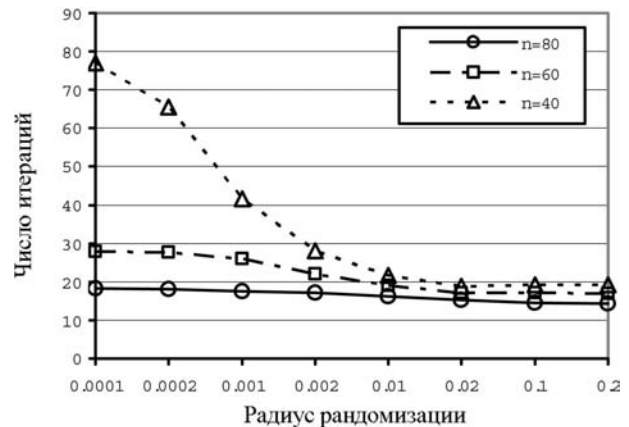


Рис. 8. Зависимость числа итераций от радиуса рандомизации ρ

Графики показывают, что время решения задачи слабо зависит от радиуса рандомизации, достигая минимума приблизительно при $\rho = 0.01$.

Количество итераций для задач небольшой размерности ($n = 40$) может значительно уменьшаться с увеличением радиуса рандомизации, принимая горизонтально-асимптотический характер для значений $\rho \geq 0.01$. Для задач большой размерности ($n = 80$) влияние радиуса рандомизации на количество итераций становится незначительным.

Проведенные эксперименты показывают, что в большинстве случаев хорошим выбором является значение $\rho = 0.01$.

4.1.3. Влияние мощности рандомизации на эффективность алгоритма СМ-ЛК. Во второй серии экспериментов мы исследовали зависимость количества итераций и времени решения задачи Mod- n от мощности рандомизации m . Данный параметр был нами введен в разделе 2.3 и обозначает количество точек, генерируемых процедурой рандомизации. Мы решали задачу Mod- n для значений n , также равных 40, 60 и 80. Результаты этих экспериментов представлены на рис. 9, 10.

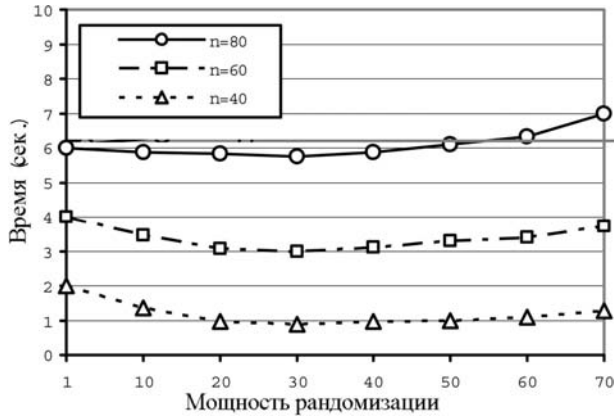


Рис. 9. Зависимость времени решения задачи от мощности рандомизации m

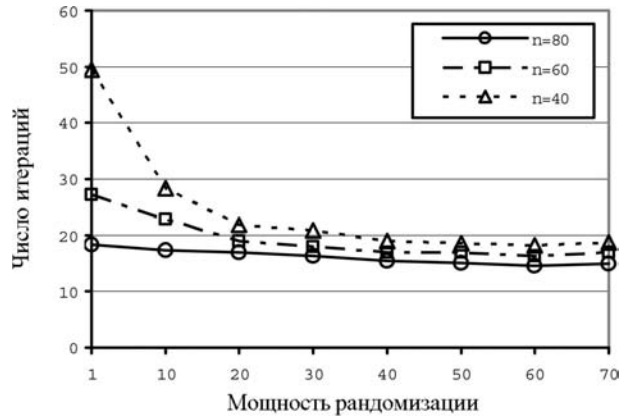


Рис. 10. Зависимость числа итераций от мощности рандомизации m

Графики демонстрируют, что время решения задачи также слабо зависит от мощности рандомизации, достигая минимума приблизительно при значении $m = 30$.

Количество итераций для всех трех размерностей уменьшается с ростом m , принимая горизонтально-асимптотический характер для значений $m > 20$. При этом для задач большой размерности ($n = 80$) влияние мощности рандомизации на количество итераций также становится незначительным.

Проведенные эксперименты показывают, что в большинстве случаев хорошим выбором является значение $m = 30$.

4.1.4. Эффективность осцилляций. В следующей серии экспериментов мы исследовали эффективность применения метода осцилляций при решении задач различной размерности. Мы провели вычислительные эксперименты с задачей Mod- n , варьируя размерность n в диапазоне от 20 до 110. При этом мы использовали два варианта алгоритма СМ-ЛК. В первом варианте метод осцилляций был включен (OSC=ON), а во втором — выключен (OSC=OFF). Результаты этих экспериментов показывают (см. рис. 11, 12), что в обоих вариантах при увеличении размерности время решения задачи растет, а количество итераций уменьшается. Однако, если задача имеет размерность $n < 50$, вариант алгоритма, использующий метод осцилляций, позволяет получить решение при значительно меньшем количестве итераций, что дает возможность существенно сократить число обращений к эксперту.

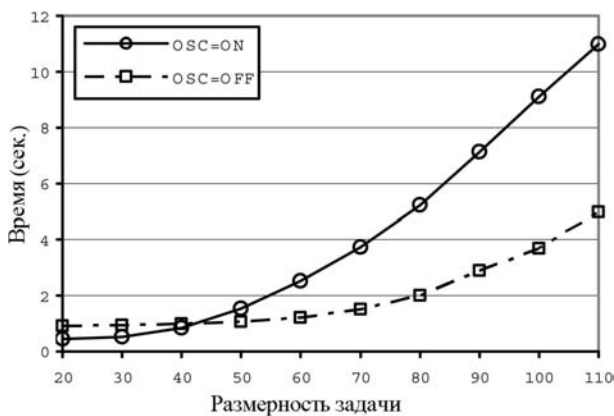


Рис. 11. Зависимость времени решения задачи от размерности n

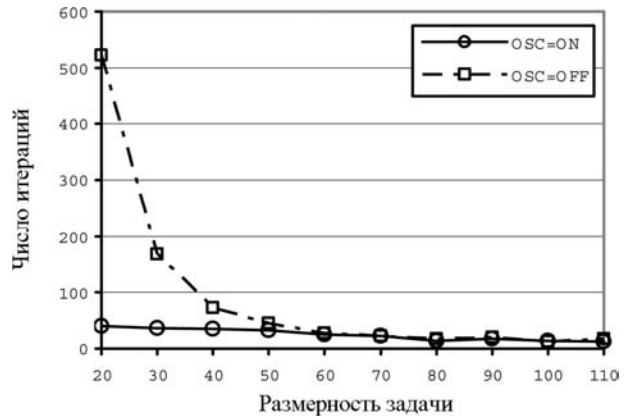


Рис. 12. Зависимость числа итераций от размерности n

4.1.5. Эксперименты с неполными наборами образцов. В данной серии экспериментов мы

исследовали влияние мощности начального набора образцов на эффективность метода СМ-ЛК. В разделе 2.1 нами был предложен метод, в соответствии с которым в качестве начального набора образцов берутся все вершины многогранника D , ограничивающего допустимое множество точек формализованной части исходной задачи. На практике не всегда возможно вычислить все вершины многогранника D . В связи с этим является важным вопрос, насколько метод СМ-ЛК является чувствительным к неполным наборам образцов. Мы провели следующий эксперимент на модельной задаче Mod-50.

Пусть \mathcal{D} обозначает множество всех вершин многогранника D . Определим *репрезентативность* (представительность) начального набора образцов \mathfrak{P} следующим образом: $\text{гер}(\mathfrak{P}) = \frac{|\mathfrak{P}|}{|\mathcal{D}|}$. В частности, при $\mathfrak{P}=\mathcal{D}$ мы будем иметь $\text{гер}(\mathfrak{P})=1$, что соответствует максимальной репрезентативности, а при $\mathfrak{P}=\emptyset$ получаем $\text{гер}(\mathfrak{P})=0$, что соответствует минимальной репрезентативности. Мы варьировали $\text{гер}(\mathfrak{P})$ в диапазоне от 1.0 до 0.6 с шагом 0.5. В каждом случае мы оценивали величину $\Delta = \|\bar{x} - \tilde{x}\|$, задающую расстояние от точного решения \bar{x} до приближенного решения \tilde{x} , полученного в ходе вычислительного эксперимента. Нужные значения $\text{гер}(\mathfrak{P})$ мы получали путем удаления соответствующего количества точек из множества \mathfrak{P} , определенного в разделе 4.1.1.

Мы провели две серии испытаний. В первой использовали режим с включенной осцилляцией (OSC=ON), во второй — с отключенной (OSC=OFF). Результаты испытаний сведены в табл. 5. Полученные экспериментальные данные показывают, что метод осцилляций в значительной мере повышает устойчивость алгоритма СМ-ЛК к неполным начальным наборам образцов.

Таблица 5

Влияние начального набора образцов
на точность решения

гер(\mathfrak{P})	Δ	
	OSC=ON	OSC=OFF
1.0	0.0001	0.021
0.95	0.001	3.5
0.9	0.01	6.2
0.85	0.034	9.76
0.8	0.023	12.4
0.75	0.04	16.67
0.7	0.045	19.55
0.65	0.04	22.8
0.6	0.04	27.3

Таблица 6

Информационное описание прокатного комплекса

Оборудование	Продукция				
	Осевые заготовки	Рельсы	Балки	Конструкционный прокат	Квадратные заготовки
Рельсобалочный стан	6.4	4.0	7.5	7.0	4.4
Участок отделки	0	0	35	20	0
Термоямы	85	0	0	75	18
Прибыль	18	16.8	26.3	36.2	17.5

4.2. Эксперименты на реальной задаче. Мы проверили эффективность алгоритма СМ-ЛК на реальной задаче управления металлургическим предприятием, заимствованной из монографии [11]. Назовем ее задачей Metal. Эта задача моделирует управление прокатным комплексом, включающим рельсобалочный стан, участок отделки фасонных профилей и термоямы. Номенклатура производимой продукции, удельные затраты времени работы оборудования (час/тыс. т) и прибыль (руб/т), получаемая от реализации продукции, приводятся в табл. 6. Фонд фактического времени работы оборудования в рассматриваемый период для всех агрегатов и участков составляет 7 тыс. час. Согласно проекту планового задания, необходимо произвести как минимум (в тыс. т): осевых заготовок — 55, рельсов — 1150, балок — 200, конструкционного проката — 5. Производство квадратных заготовок фиксировано и составляет 125 тыс. т. Необходимо максимизировать эффективность функционирования рельсобалочного комплекса.

Сформулируем задачу ЛПНО для данного примера. Обозначим через x_i объем i -го вида производства. По проекту планового задания для них заданы ограничения $x_1 \geq 55, x_2 \geq 1150, x_3 \geq 200, x_4 \geq 5, x_5 = 125$. С учетом табл. 6 формализованная часть нашей задачи описывается следующей системой неравенств:

$$\begin{cases} 6.4x_1 + 4x_2 + 7.5x_3 + 7x_4 + 4.4x_5 \leq 7000, \\ 35x_3 + 20x_4 \leq 7000, \\ x_1 \leq 50, \\ x_2 \leq 1150, \\ x_3 \leq 150, \\ x_4 \leq 5, \\ x_5 = 125. \end{cases}$$

В качестве неформализованного ограничения фигурирует неравенство $85x_1 + 75x_4 \leq y$. Здесь y — константа, определяемая экспертом по неформализуемым критериям. Поскольку необходимо максимизировать прибыль, то целевую функцию представим в виде

$$Q_{\max} = 18x_1 + 16.8x_2 + 26.3x_3 + 36.2x_4 + 17.5x_5.$$

Множества \mathfrak{A} и \mathfrak{B} задаются как множества недопустимых и допустимых по неформализованному ограничению планов. Для $y=4750$ в качестве набора образцов были взяты следующие множества точек:

$$\mathfrak{A} = \{ (50, 1150, 150, 58); (108, 1150, 150, 5); (53, 1150, 197, 5); (50, 1150, 194, 11) \};$$

$$\mathfrak{B} = \{ (50, 1243, 150, 5); (50, 1154, 197, 5); (0, 0, 0, 0) \}.$$

Данные точки являются вершинами многогранника, ограничивающего область допустимых значений формализованной части задачи Metal. Они были нами получены путем решения задачи ЛПФ симплекс-методом для различных значений коэффициентов целевой функции.

Мы провели с задачей Metal серию вычислительных экспериментов, в которых исследовали зависимость времени решения задачи и количества итераций от значения ϵ , используемого в критерии завершения (см. раздел 2.2). Проведенные эксперименты показали (см. рис. 13, 14), что метод осцилляций во всех случаях обеспечивает примерно двукратное превосходство по скорости сходимости.

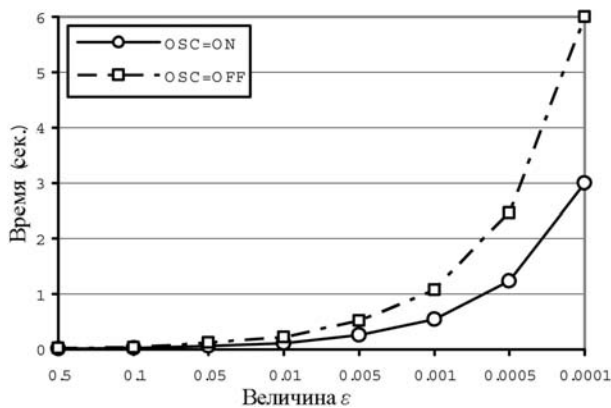


Рис. 13. Зависимость времени решения задачи от величины ϵ

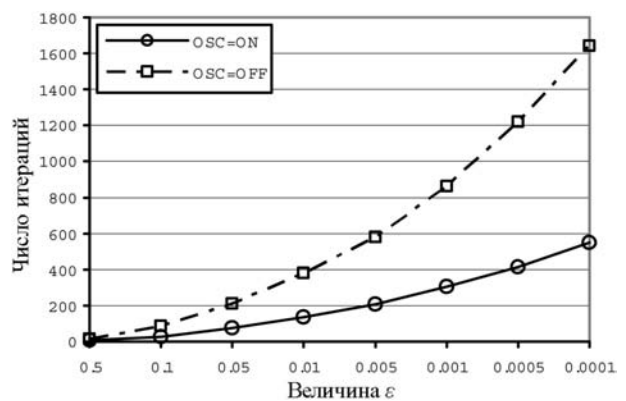


Рис. 14. Зависимость числа итераций от величины ϵ

5. Заключение. В данной работе был рассмотрен итерационный алгоритм СМ-ЛК для решения задач линейного программирования при наличии одного неформализованного ограничения. Этот алгоритм базируется на сочетании симплекс-метода и метода линейной коррекции. Рассмотрены различные аспекты реализации алгоритма СМ-ЛК в виде программы для ЭВМ. Предложен метод формирования начального набора образцов. Введен и математически обоснован критерий завершения итерационного

процесса. Дано формальное описание процедуры рандомизации и введены такие параметры алгоритма, как радиус и мощность рандомизации. Предложен оригинальный метод осцилляций, позволяющий во многих случаях значительно повысить эффективность алгоритма СМ-ЛК. Введена система допусков, обеспечивающих корректную работу алгоритма при возникновении погрешностей вычислений, связанных с неточным представлением вещественных чисел в ячейках памяти ЭВМ. Описана модульная структура программного комплекса для решения задач линейного программирования с неформализованными ограничениями (ЛПНО), использующая технологию картриджей, которая позволяет расширять программный комплекс путем добавления новых методов линейной оптимизации и дискриминантного анализа. На основе стандарта MPS разработан специальный формат входных данных LPNC для представления задач ЛПНО. Выполнена реализация программного комплекса на языке Си и проведено большое количество вычислительных экспериментов на реальных и искусственных задачах, подтверждающих эффективность предложенных подходов. Исходные тексты программ свободно доступны в Интернет по адресу <http://life.susu.ru/lpno/>.

В рамках дальнейших исследований мы предполагаем реализовать метод комитетов и встроить его в наш программный комплекс. Это позволит решать задачи ЛПНО с двумя и более неформализованными ограничениями. Кроме этого, планируется провести тестирование разработанного программного комплекса на задачах, доступных в интернет-репозитории [16].

СПИСОК ЛИТЕРАТУРЫ

1. Данциг Дж. Линейное программирование, его применение и обобщения. М.: Прогресс, 1966.
2. Еремин И.И. Теория линейной оптимизации. Екатеринбург: Изд-во "Екатеринбург", 1999.
3. Еремин И.И. Общая теория устойчивости в линейном программировании // Известия ВУЗов. Математика. 1999. № 12. 43–52.
4. Еремин И.И., Мазуров Вл.Д. Нестационарные процессы математического программирования. М.: Наука, 1979.
5. Еремин И.И., Мазуров Вл.Д., Скарин В.Д., Хачай М.Ю. Математические методы в экономике. Екатеринбург: У-Фактория, 2000.
6. Мазуров Вл.Д. Дискриминантный анализ при математическом моделировании плохо формализуемых ситуаций // Нелинейная оптимизация и приложения в планировании. Свердловск: УНЦ АН СССР, 1973. 26–35.
7. Мазуров Вл.Д. Метод комитетов в задачах оптимизации и классификации. М.: Наука, 1990.
8. Муртаф Б. Современное линейное программирование: теория и практика. М.: Мир, 1984.
9. Нильсон Н. Обучающиеся машины. М.: Мир, 1967.
10. Соколинская И.М. Метод осцилляций в задачах линейного программирования с неформализованным ограничением // Алгоритмический анализ неустойчивых задач. Тез. докл. Всерос. конф. Екатеринбург. 2–6 февр. 2004 г. Екатеринбург: Изд-во Урал. ун-та, 2004. 302–303.
11. Фролов В.Н. Оптимизация плановых программ при слабо согласованных ограничениях. М.: Наука, 1986.
12. Bartels R.H., Golub G.H. The simplex method of linear programming using LU decomposition // Communications of the ACM. 1969. **12**, N 5. 266–268.
13. Gass S.I. Linear programming. New York: McGraw-Hill, 1969.
14. Hadley G. Linear programming. Reading: Addison-Wesley, 1962.
15. Nazareth J.L. Computer solution of linear problems. Oxford: Oxford University Press, 1988.
16. Netlib Repository: lp (Linear Programming) [<http://www.netlib.org/lp/>].
17. Mazurov V.I., Sokolinskaya I.M. Discrimination analysis and randomization in linear optimization problems with not formalized restrictions // Pattern Recognition and Image Analysis. 2005. **15**, N 4. 592–610.
18. Orchard-Hays W. Advanced linear programming computing techniques. New York: McGraw-Hill, 1968.
19. White W.W. A status report on computing algorithms for mathematical programming // ACM Computing Surveys. 1973. **5**, N 3. 135–166.

Поступила в редакцию
09.08.2005