ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—июнь 1998. Серия 2. Том 5, № 1, 19–39

УДК 519.87

МЕТОД ВЕТВЕЙ И ГРАНИЦ ДЛЯ ПРОСТЕЙШЕЙ ДВУХУРОВНЕВОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ ПРЕДПРИЯТИЙ*)

Е. Н. Гончаров

Рассматривается задача двухуровневого размещения предприятий без ограничения на объемы производства. Для ее решения предлагается использовать метод ветвей и границ. Строятся эффективные алгоритмы нахождения нижней оценки для целевой функции задачи. Приводятся результаты численных экспериментов, иллюстрирующие эффективность нижней оценки, получаемой при помощи предложенных алгоритмов. Приводятся также результаты вычислительных экспериментов, позволяющие судить о качестве метода ветвей и границ, использующего рассмотренные алгоритмы вычисления нижней оценки целевой функции.

Введение

Двухуровневая задача размещения предприятий на содержательном уровне формулируется следующим образом.

Имеются множество возможных пунктов размещения предприятий первого уровня и множество возможных пунктов размещения предприятий второго уровня. Предприятия первого уровня поставляют свою продукцию предприятиям второго уровня, которые создают конечную продукцию, предлагаемую потребителям. Каждое предприятие первого уровня может поставлять свою продукцию нескольким предприятиям второго уровня, а каждый потребитель может получать конечную продукцию только от одного предприятия второго уровня. Известны затраты на открытие предприятий первого и второго уровней, а также затраты на удовлетворение спроса потребителей каждым предприятием второго уровня. Требуется выбрать пункты размещения предприятий всех уровней и распределить потребителей по предприятиям второго уровня так, чтобы суммарные затраты были минимальными при условии удовлетворения спроса всех потребителей.

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 96-01-01591).

Двухуровневой задаче размещения посвящено значительное количество работ, например [8–11]. В них приводятся, в частности, описания большого количества конкретных задач по размещению различных объектов (заводов, складов, автопредприятий и т. п.), сводимых к двухуровневой задаче размещения.

Известна также [1] другая интерпретация рассматриваемой задачи. Пусть имеется множество сложных технических средств различных типов (образцов), предназначенных для выполнения заданного множества видов работ. Пусть эти технические средства состоят из составных частей (узлов) и узлы одного типа входят в состав технических средств различных образцов. Другими словами, имеются изделия первого уровня (изделия-узлы), используемые для сборки изделий второго уровня (изделий-комплектов), которые используются для выполнения заданных работ. При этом считаются известными начальные затраты на создание (использование) изделий-комплектов и изделий-узлов всех типов и затраты на выполнение изделиями-комплектами работ каждого вида.

Требуется выбрать множество изделий первого уровня и множество составленных из них изделий второго уровня и для каждого вида работ указать исполнителей этих работ из числа выбранных изделий второго уровня так, чтобы суммарные затраты на разработку изделий и выполнение всех работ были наименьшими.

Эту задачу с учетом области ее практического применения называют *задачей выбора двухуровневой системы изделий* [6, 7], или *двухуровневой задачей стандартизации* [1, 4]. С точки зрения приложений в стандартизации эта задача рассматривалась в [1], где для ее решения предложен алгоритм ветвей и границ. Нижняя оценка для целевой функции задачи в этом алгоритме строится с использованием так называемого тупикового решения двойственной задачи. Среди других методов нахождения нижних оценок для целевой функции двухуровневой задачи и некоторых ее обобщений следует отметить метод лагранжевых релаксаций [6, 7]. Двухуровневые задачи размещения на цепи и на дереве исследованы в [2]. Для этих случаев построены полиномиальные алгоритмы, позволяющие находить точные решения.

В настоящей работе для двухуровневой задачи предлагаются полиномиальные алгоритмы нахождения нижних оценок для целевой функции задачи. Идеи, используемые при построении этих алгоритмов, фактически развивают идеи из [1], а также из [3] для одноуровневой задачи размещения. Кроме того, приводятся результаты большого числа вычислительных экспериментов как с алгоритмами нахождения нижних оценок, так и с алгоритмами ветвей и границ, использующими данные алгоритмы вычисления нижних оценок для целевой функции задачи.

1. Формулировка задачи

Для математической формулировки задачи выбора двухуровневой системы изделий (двухуровневой задачи размещения) введем следующие обозначения.

Пусть $J = \{1, \ldots, n\}$ — множество рассматриваемых видов работ; $I = \{1, \ldots, m\}$ — множество типов (образцов) изделий-узлов, используемых для комплектации изделий-комплектов; $L = \{1, \ldots, p\}$ — множество рассматриваемых типов (образцов) изделий-комплектов, способных выполнять заданные виды работ. Обозначим через L_i, $L_i \subseteq L$, $i \in I$, перечень изделий-комплектов, имеющих в своем составе изделия-узлы i-го типа. Аналогично через I_l, $I_l \subseteq I$, $l \in L$, обозначим множество типов изделий-узлов, используемых для комплектации изделий-комплектов l-го типа.

Пусть

f_i, $i \in I$, — величина начальных затрат, связанных с разработкой и организацией начального цикла производства изделия-узла i-го типа (штраф за ввод изделия-узла i-го типа в систему);

g_l, $l \in L$, — величина начальных затрат, связанных с разработкой и организацией начального цикла производства изделия-комплекта l-го типа (штраф за ввод изделия-комплекта l-го типа в систему);

c_{lj}, $l \in L, j \in J$, — затраты на производство и эксплуатацию комплекта l-го типа при выполнении работы j-го вида.

Введем необходимые переменные:

y_i, $i \in I$, — булева переменная, причем $y_i = 1$, если i-е изделие входит в состав хотя бы одного изделия-комплекта, которое используется для выполнения работ, и $y_i = 0$ в противном случае;

z_l, $l \in L$, — булева переменная, причем $z_l = 1$, если l-е изделие-комплект используется для выполнения работ, и $z_l = 0$ в противном случае;

x_{lj}, $l \in L$, $j \in J$, — булева переменная, причем $x_{lj} = 1$, если l-е изделие-комплект выбрано для выполнения работы j-го вида, и $x_{lj} = 0$ в противном случае.

С использованием введенных обозначений задача выбора двухуровневой системы изделий (двухуровневая задача размещения предприятий) записывается следующим образом.

Найти

$$\min_{x,y,z} \left\{ \sum_{i \in I} f_i y_i + \sum_{l \in L} g_l z_l + \sum_{j \in J} \sum_{l \in L} c_{lj} x_{lj} \right\} \qquad (1)$$

при условиях

$$\sum_{l \in L} x_{lj} = 1, \quad j \in J; \tag{2}$$

$$z_l \geqslant x_{lj}, \quad j \in J, \quad l \in L; \tag{3}$$

$$y_i \geqslant z_l, \quad i \in I, \quad l \in L_i; \tag{4}$$

$$x_{lj},\ z_l,\ y_i \in \{0,1\}, \quad i \in I, \quad l \in L, \quad j \in J. \tag{5}$$

Целевая функция (1) данной задачи выражает суммарные затраты на создание и функционирование всей системы; ограничение (2) означает, что все работы должны быть выполнены; ограничение (3) показывает, что невозможно назначить какой-либо комплект на выполнение любой заданной работы, если этот комплект не выбран для обслуживания потребителей; ограничение (4) означает, что какой-либо комплект не может быть использован для выполнения работ, если хотя бы одно изделие, входящее в его состав, не применяется (не выбрано) для формирования комплектов.

Задача (1)–(5) принадлежит к классу NP-трудных задач [5]. Для ее решения предлагается использовать метод ветвей и границ (МВГ). Общая схема работы алгоритма МВГ для такого типа задач известна и достаточно хорошо проявила себя на практике. В настоящей работе мы не будем на ней останавливаться. Подробно с алгоритмом МВГ для данной задачи можно ознакомиться в [1] и [4].

При дальнейшем изложении исходную задачу (1)–(5) будем представлять в эквивалентном виде [4].

Найти

$$\min_{x,y,z} \left\{ \sum_{i \in I} f_i y_i + \sum_{l \in L} g_l z_l + \sum_{j \in J} \sum_{l \in L} c_{lj} x_{lj} \right\} \tag{6}$$

при условии

$$\sum_{l \in L} x_{lj} = 1, \quad j \in J; \tag{7}$$

$$\sum_{l \in L_i} x_{lj} \leqslant y_i, \quad i \in I, \quad j \in J, \tag{8}$$

$$z_l \geqslant x_{lj}, \quad j \in J, \quad l \in L; \tag{9}$$

$$x_{lj},\ z_l,\ y_i \in \{0,1\}, \quad i \in I, \quad l \in L, \quad j \in J. \tag{10}$$

Большое значение для эффективности работы МВГ имеет то обстоятельство, насколько успешно производится вычисление нижней оценки для целевой функции. Алгоритмам вычисления таких оценок и выяснению их качества, т. е. тому, насколько правильно найден баланс между точностью этих оценок и временем, за которое они получены, будет посвящена основная часть этой работы.

2. Алгоритм нахождения нижней оценки для целевой функции

Рассмотрим частный случай задачи, когда начальные затраты на формирование комплекта равны нулю. Заметим, что общий случай задачи сводится к указанному частному случаю. Начальные затраты на изделия-комплекты можно считать равными нулю, если расширить множество разновидностей изделий-узлов, введя в это множество для каждого изделия-комплекта некоторый дополнительный фиктивный узел, который входит в состав только данного изделия-комплекта. Начальные затраты для этого фиктивного изделия-узла должны быть равны начальным затратам для соответствующего изделия-комплекта.

Когда $g_l = 0$, $l \in L$, исходная задача (6)–(10) преобразуется к следующему виду.

Найти

$$\min_{x,y} \left\{ \sum_{i \in I} f_i y_i + \sum_{j \in J} \sum_{l \in L} c_{lj} x_{lj} \right\} \tag{11}$$

при условиях

$$\sum_{l \in L} x_{lj} = 1, \quad j \in J; \tag{12}$$

$$\sum_{l \in L_i} x_{lj} \leqslant y_i, \quad i \in I, \quad j \in J, \tag{13}$$

$$x_{lj},\ y_i \in \{0, 1\}, \quad i \in I, \quad l \in L, \quad j \in J. \tag{14}$$

Рассмотрим релаксированную задачу (11)–(14), получаемую заменой условия булевости переменных задачи на условие их неотрицательности. Задача, двойственная к релаксированной задаче (11)–(14), имеет следующий вид.

Найти

$$\max_{v,w} \sum_{j=1}^{n} v_j \tag{15}$$

при условиях

$$v_j - \sum_{i \in I_l} w_{ij} \leqslant c_{lj}, \quad j \in J, \quad l \in L; \tag{16}$$

$$\sum_{j \in J} w_{ij} \leqslant f_i, \quad i \in I; \tag{17}$$

$$w_{ij} \geqslant 0, \quad i \in I, \quad j \in J. \tag{18}$$

Значение целевой функции задачи (15)–(18) является нижней оценкой для целевой функции задачи (11)–(14). Оптимальное решение задачи (15)–(18) дает наилучшую такую оценку. Однако мы не будем решать эту задачу линейного программирования точно, а будем находить

ее приближенное решение, основанное на построении так называемого тупикового решения.

Нетрудно заметить, что в рассматриваемой задаче (15)–(18) переменные v_j, $j \in J$, могут быть исключены, а сама задача переписана следующим образом.

Найти

$$\max_{w} \sum_{j=1}^{n} \min_{l \in L} \left\{ c_{lj} + \sum_{i \in I_l} w_{ij} \right\} \tag{19}$$

при условиях

$$\sum_{j \in J} w_{ij} \leqslant f_i, \quad i \in I; \tag{20}$$

$$w_{ij} \geqslant 0, \quad i \in I, \quad j \in J. \tag{21}$$

Дадим определение тупикового решения задачи (19)–(21). Для допустимого решения (w_{ij}) задачи (19)–(21) положим

$$v_j = \min_{l \in L} \left\{ c_{lj} + \sum_{i \in I_l} w_{ij} \right\}, \quad j \in J,$$

$$r_i = f_i - \sum_{j \in J} w_{ij}, \quad i \in I,$$

и рассмотрим следующие множества:

$$I^0 = \{ i \in I \mid r_i = 0 \};$$

$$L^0 = \{ l \in L \mid I_l \subset I^0 \};$$

$$L^j = \left\{ l \in L \mid c_{lj} + \sum_{i \in I_l} w_{ij} = v_j \right\}, \quad j \in J;$$

$$J^0 = \{ j \in J \mid L^j \cap L^0 = \varnothing \}.$$

Допустимое решение (w_{ij}) задачи (19)–(21) назовем *тупиковым*, если для всякого $j \in J$ выполняются следующие условия:

1) $L^j \cap L^0 \neq \varnothing$;

2) $w_{ij} = 0$, если $i \in I \setminus I_l$ для каждого $l \in L^j$.

Рассмотрим алгоритм построения тупикового решения (w_{ij}) задачи (19)–(21), дающий «хорошую» нижнюю оценку для целевой функции.

Алгоритм состоит из конечного числа однотипных шагов, на каждом из которых производится увеличение некоторых компонент текущего решения (w_{ij}), приводящее к увеличению величины v_{j_0} на некоторую величину Δ, где j_0 — номер выбранного на этом шаге столбца.

На первом шаге алгоритма имеется решение (w_{ij}), $w_{ij} = 0$, $i \in I$, $j \in J$.

Пусть к очередному шагу получено решение (w_{ij}). Если $J^0 = \varnothing$, то найденное решение является тупиковым и алгоритм заканчивает работу. В противном случае отыскивается номер $j_0 \in J^0$ такой, что $|L^{j_0}| = \min_{j \in J^0} |L^j|$. Далее определяется величина $\Delta = \min\{\delta_1, \delta_2\}$. Для этого вычисляется величина

$$\delta_1 = \min_{l \in L \setminus L^{j_0}} c_{lj_0} - v_{j_0},$$

которая является минимальной разностью между элементами c_{lj_0}, превышающими достигнутое к данному шагу значение переменной v_{j_0}, и самим значением переменной v_{j_0}, и величина

$$\delta_2 = \min_{l \in L^{j_0}} \left(\sum_{i \in I_l} r_i \right),$$

указывающая на имеющийся «запас ресурсов», который может быть использован для увеличения значения v_{j_0}, исходя из нераспределенных к данному шагу ресурсов.

Далее определяются значения приращений z_i, $i \in I$, переменных w_{ij_0}, $i \in I$, реализующие необходимое увеличение величины v_{j_0}, а также новые значения $w_{ij_0} = w_{ij_0} + z_i$, $i \in I$, этих переменных. После этого начинается следующий шаг.

В приведенном выше описании алгоритма действия, связанные с определением приращений z_i, $i \in I$, нуждаются в уточнении. Действительно, выбор значений переменных z_i, $i \in I$, таких, что

$$\sum_{i \in I_l} z_i \geqslant \Delta, \quad l \in L^{j_0}; \tag{22}$$

$$0 \leqslant z_i \leqslant r_i, \quad i \in I, \tag{23}$$

может быть осуществлен неоднозначно. Вместе с тем мы заинтересованы в нахождении нижней оценки с наибольшим значением. Поэтому выбор приращения переменных (w_{ij_0}), $i \in I$, на каждом шаге должен быть не просто таким, чтобы обеспечить увеличение значения переменных v_{j_0} на величину Δ, но и таким, чтобы на выходе алгоритма обеспечить по возможности наибольшее значение $\sum\limits_{j \in J} v_j$.

Сформулируем некоторые локальные оптимизационные задачи, которые, как представляется, позволяют достичь поставленной цели.

Первая такая задача состоит в минимизации величины

$$\sum_{i \in I} z_i, \tag{24}$$

что соответствует наименьшему расходованию «ресурсов» при увеличении значения переменной v_{j_0} в j_0-м столбце.

Вторая задача учитывает не только используемое количество ресурсов, но и его «дефицитность», и состоит в минимизации величины

$$\sum_{i \in I \setminus I^0} \frac{z_i}{r_i}. \tag{25}$$

Наконец, третья локальная оптимизационная задача состоит в минимизации величины

$$\sum_{i \in I \setminus I^0} \frac{z_i}{r_i \omega_i}, \tag{26}$$

где ω_i — число комплектов из множества L^{j_0}, в состав которых входит изделие i-го типа:

$$\omega_i = |\{l \in L^{j_0} \mid i \in I_l\}|, \quad i \in I.$$

Используя третий критерий, мы стремимся минимизировать суммарные приращения двойственных переменных w_{ij_0}, расходуя в первую очередь наименее «дефицитные» ресурсы.

Приведем приближенный алгоритм решения задачи определения значений переменных z_i, $i \in I$, удовлетворяющих ограничениям (22), (23) и доставляющих наименьшее значение рассмотренным целевым функциям. Алгоритм основывается на использовании перестановки $\sigma = (\sigma_1, \ldots, \sigma_m)$, задающей приоритет в выборе ненулевых значений переменных z_i, $i \in I$. Предлагаются три правила построения такой перестановки.

Правило 1. Элементы перестановки σ располагаются

а) по невозрастанию ω_i;

б) для изделий с одинаковым ω_i — по невозрастанию r_i.

Правило 2. Элементы перестановки σ располагаются по невозрастанию r_i.

Правило 3. Элементы перестановки σ располагаются по невозрастанию $r_i \omega_i$.

Отметим, что первый способ формирования перестановки σ в изложенном ниже алгоритме нахождения приращения двойственных переменных w_{ij_0} реализует приближенное решение задачи (24), (22)–(23), второй способ формирования перестановки σ реализует приближенное решение задачи (25), (22)–(23), а третий способ формирования перестановки σ реализует приближенное решение задачи (26), (22)–(23). Результаты исследования качества нижней оценки, получаемой с использованием каждого из трех способов, приводятся в разд. 4.1.

Алгоритм состоит из конечного числа основных шагов, равных числу ограничений (22) или числу элементов в множестве L^{j_0}. На предварительном шаге фиксируется перестановка $\sigma = (\sigma_1, \ldots, \sigma_m)$ и полагается $z_i = 0$, $i \in I$.

На очередном основном шаге рассматривается элемент $l \in L^{j_0}$. Если $\sum\limits_{i \in I_l} z_i < \Delta$, то начинается процедура пересчета величин z_i, $i \in I$, в порядке, задаваемом перестановкой $\sigma = (\sigma_1, \ldots, \sigma_m)$. Возьмем очередной номер i_k, $1 \leqslant k \leqslant m$. Если $i_k \in I_l$, то полагаем

$$z_{i_k} = z_{i_k} + \min\left(r_{i_k};\ \Delta - \sum_{i \in I_l} z_i\right);$$

$$r_{i_k} = r_{i_k} - z_{i_k}.$$

Если $\sum\limits_{i \in I_l} z_i < \Delta$, то начинается пересчет величины $z_{i_{k+1}}$. В противном случае процедура пересчета величин z_i, $i \in I$, на данном шаге заканчивается.

Далее, если не все элементы $l \in L^{j_0}$ уже просмотрены, начинается следующий шаг. В противном случае работа алгоритма заканчивается.

Отметим, что процедура нахождения приращения двойственных переменных w_{ij_0} состоит не более чем из p основных шагов, временная сложность каждого из которых равна $O(m)$. Временная сложность построения на предварительном шаге перестановки $\sigma = (\sigma_1, \ldots, \sigma_m)$ равна $O(m \log m)$. Поэтому временная сложность процедуры нахождения приращения двойственных переменных w_{ij_0} не превосходит $O(mp + m \log m)$. Нетрудно заметить, что верхняя оценка суммарного числа обращений к этому алгоритму не превышает pn.

Таким образом, временная сложность алгоритма вычисления нижней оценки для целевой функции рассматриваемой задачи без учета начальных затрат на изделия-комплекты равна $O(mpn(p + \log m))$.

3. Об алгоритме нахождения нижних оценок задачи в общем случае

Поскольку задача общего вида сводится к задаче с нулевыми начальными затратами на изделия-комплекты, то рассмотренный выше алгоритм применим и для вычисления нижней оценки задачи общего вида. Соответствующая задача с нулевыми начальными затратами на изделия-комплекты имеет расширенное множество изделий-узлов $\mathscr{I} = I \cup I'$, где $I = \{1, \ldots, m\}$, $I' = \{m + 1, \ldots, m + p\}$. Поэтому в этом случае временная сложность алгоритма нахождения нижней оценки для целевой функции не превосходит $O((m + p)pn(p + \log(m)))$.

Следует подчеркнуть, что для $i \in I'$ множество L_i является одноточечным. Эта специфика множеств L_i является существенной и может быть использована при нахождении приращений переменных z_i, $i \in \mathscr{I}$, двойственным переменным w_{ij_0}. Модификация алгоритма вычисления приращений связана с изменением задания приоритетов в выборе ненулевых значений переменных z_i, $i \in \mathscr{I}$. Если в случае задачи с нулевыми начальными затратами на изделия-комплекты на каждом шаге алгоритма приоритеты задавались перестановкой $\sigma = (\sigma_1, \ldots, \sigma_m)$, то на каждом шаге модифицированного алгоритма рассматривается расширенная перестановка $\sigma' = (\sigma_1, \ldots, \sigma_k, m+l, \sigma_{k+1}, \ldots \sigma_m)$, где $\sigma = (\sigma_1, \ldots, \sigma_m)$ — фиксированная перестановка множества I; l — номер рассматриваемого на данном шаге комплекта; k — выбираемый по некоторому правилу номер, $0 \leqslant k \leqslant m$.

Предлагаются следующие правила выбора номера k для каждого комплекта $l \in L^{j_0}$.

Правило A. $k = m$.

Правило B. $k = 0$.

Правило C. Если $\sum\limits_{i \in I_l} r_i > r_{m+l}$, то $k = m$, иначе $k = 0$.

В случае, когда перестановка $\sigma = (\sigma_1, \ldots, \sigma_m)$ построена по правилу 1, то предлагаются также

Правило D. Номер k выбирается так, что $\omega_{\sigma_s} > 1$, $s = 1, \ldots, k$ и $\omega_{\sigma_s} \leqslant 1$, $s = k+1, \ldots, m$.

Правило E. Номер k выбирается так, что

а) $\omega_{\sigma_s} > 1$, $s = 1, \ldots, k'$, $k' \leqslant k$;

б) $\omega_{\sigma_s} = 1$, и $r_s > r_{m+l}$, $s = k'+1, \ldots, k$;

в) $r_s \leqslant r_{m+l}$, $s = k, \ldots, m$.

Теперь для построения расширенной перестановки $\sigma' = (\sigma_1, \ldots, \sigma_k, m+l, \sigma_{k+1}, \ldots \sigma_m)$ мы имеем три правила формирования перестановки σ, представленных в предыдущем разделе, и пять правил для помещения в расширенную перестановку номера $(m+l)$. Различные допустимые комбинации этих двух типов правил дают различные варианты построения расширенной перестановки σ'. Назовем их A1, A2, A3, $B1, \ldots$. Например, запись «расширенная перестановка $B1$» означает, что она сформирована на основе перестановки σ, построенной по правилу 1, и правила формирования расширенной перестановки B. Алгоритм нахождения нижней оценки с использованием этой расширенной перестановки будем называть A_{B1}. В разд. 4 приведены результаты тестовых расчетов по некоторым (лучшим) из этих вариантов. Отметим, что среди них нет варианта, реализующего нахождение наименьшего суммарного приращения двойственных переменных w_{ij_0} (см. задачу (24)–(23)).

4. Результаты тестовых расчетов

Для исследования качества представленного в настоящей работе алгоритма вычисления нижней оценки целевой функции задачи (1)–(5) была проведена серия тестовых расчетов. Тестировался как сам этот алгоритм, так и алгоритм ветвей и границ, использующий данный алгоритм нахождения нижней оценки для целевой функции.

Тестовые расчеты проводились на персональном компьютере в конфигурации Mb ASUS P55T2P4 Pentium 430HX, L2 cache 256 Kb PB, CPU Intel 100 MHz, RAM 16 Mb.

Компьютерные программы, по которым проводились расчеты, были созданы в системе Borland C++ 3.1 под операционной системой DOS 6.22.

Тестирование алгоритмов проводилось на двух классах задач $R(m,p,n,Q)$ и $E(m,p,n,Q)$, где

- m — число изделий первого уровня (изделий-узлов);
- p — число изделий второго уровня (изделий-комплектов);
- n — число потребителей;
- Q — фиксированное число изделий первого уровня в каждом изделии второго уровня.

В обоих классах задач предполагается, что
$f_i = 1000,\ i \in I,$
$g_l = 1000,\ l \in L,$
а номера Q изделий-узлов, входящих в состав изделий-комплектов, выбираются равновероятно из множества I.

Для класса задач $R(m,p,n,Q)$ элементы матрицы (c_{lj}) являются случайными целыми числами со значениями от 0 до 1000 и равномерным распределением. В случае класса $E(m,p,n,Q)$ элементы матрицы (c_{lj}) строятся следующим образом. В квадрате 1000×1000 случайным образом размещаются $\max(p,n)$ точек и в качестве величины c_{lj} принимается округленное до целого евклидово расстояние между l-й и j-й точками.

Кроме указанных классов особо рассматривались также классы $R_0(m,p,n,Q)$ и $E_0(m,p,n,Q)$, для которых $g_l = 0,\ l \in L$.

4.1. Тесты качества алгоритма нахождения нижней оценки

Для классов задач $R_0(30,50,50,Q)$ и $E_0(30,50,50,Q)$ приведем графики среднего значения нижней оценки (в процентах от оптимального значения целевой функции задачи) в зависимости от значений параметра Q для различных правил формирования перестановки σ (рис. 1, 2).

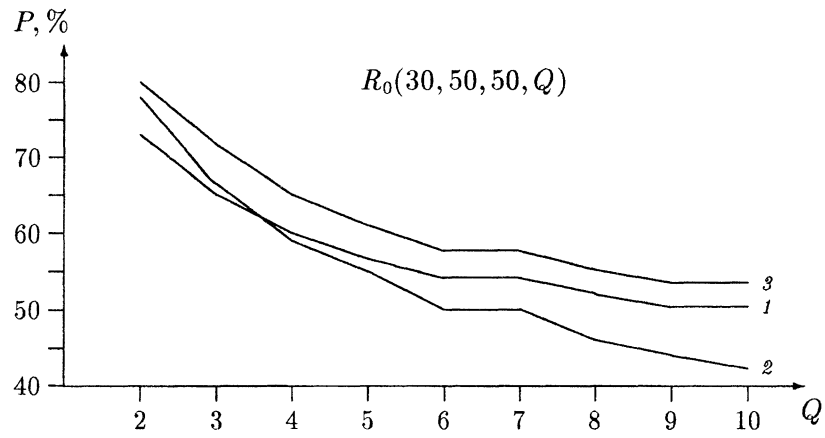

Рис. 1

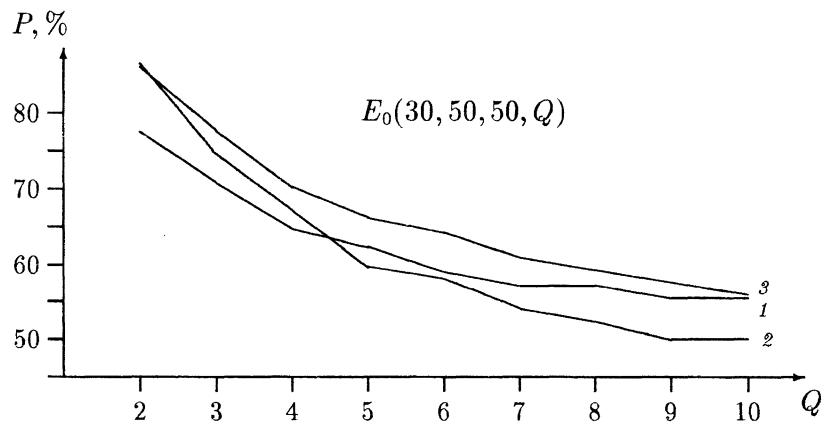

Рис. 2

Рисунки созданы на основе серии расчетов, проведенных для задачи (11)–(14), с целью получения сравнительных характеристик качества рассмотренных выше правил формирования перестановки $\sigma = (\sigma_1 \ldots \sigma_m)$, используемых при построении нижней оценки для целевой функции. Расчеты проводились для задач из класса $R_0(30, 50, 50, Q)$ и $E_0(30, 50, 50, Q)$ с различными значениями параметра Q в пределах от 2 до 10. В каждом случае рассматривалась выборка из 30 задач.

Из рис. 1 и 2 следует, что во всех случаях использование алгоритма A_3 приводит к лучшей нижней оценке. С применением этого способа нахождения нижней оценки для целевой функции проводились все включенные в настоящую работу численные эксперименты для задач без начальных затрат на изделия-комплекты.

Для четырех различных алгоритмов вычисления нижних оценок для задачи (11)–(14) приведем средние значения нижних оценок (в процентах от оптимального решения задачи) и среднее время (в секундах), за которое эти оценки были получены (табл. 1).

Рассматривались следующие алгоритмы:

1) алгоритм из [1];

2) алгоритм из [4];

3) алгоритм A_3, представленный в настоящей работе;

4) алгоритм A_L, основанный на методе лагранжевых релаксаций [7].

Тестовые расчеты проводились для задач из класса $R_0(30, 100, 100, Q)$ со значениями параметра Q от 2 до 7. В каждом случае рассматривалась выборка из 30 задач.

Таблица 1

Q	Алгоритм [1]		Алгоритм [4]		Алгоритм A_3		Алгоритм A_L	
	НГ	Время	НГ	Время	НГ	Время	НГ	Время
2	67,96	0,44	71,3	0,18	75,21	0,18	80,49	771
3	55,78	0,54	58,6	0,30	63,09	0,28	66,68	849
4	53,40	0,6	54,78	0,4	57,2	0,33	64,3	939
5	51,35	0,66	52,14	0,44	54,5	0,37	64,1	1036
6	49,73	0,76	49,83	0,47	51,26	0,39	61,16	1136
7	49,52	0,81	49,18	0,49	49,91	0,41	60,33	1212

Отметим, что алгоритм A_L, основанный на методе лагранжевых релаксаций, был настроен на получение как можно лучшей нижней оценки для целевой функции, не считаясь со временем, которое тратится на его получение. Эта наилучшая нижняя оценка иллюстрирует разрыв двойственности, наблюдаемый на задачах рассматриваемого класса. Как видно, с ростом параметра Q разрыв двойственности возрастает, что сказывается на качестве работы всех представленных алгоритмов.

Для классов задач $R(30, 50, 50, Q)$ и $E(30, 50, 50, Q)$ приведем графики среднего значения нижней оценки (в процентах от оптимального значения целевой функции задачи) в зависимости от значений параметра Q для различных правил формирования расширенной перестановки σ' (рис. 3, 4).

На рисунках изображены три наиболее интересных варианта алгоритма: A_{B1}, A_{D1} и A_{D3}. Эти рисунки были созданы на основе серии расчетов, проведенных для задачи (6)–(10), с целью получения сравнительных характеристик качества рассмотренных выше правил формирования расширенной перестановки σ', используемых при построении

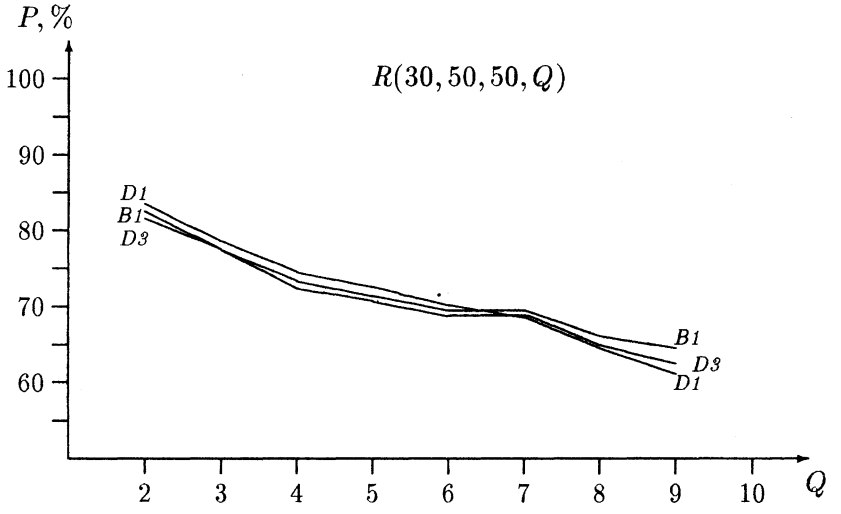

Рис. 3

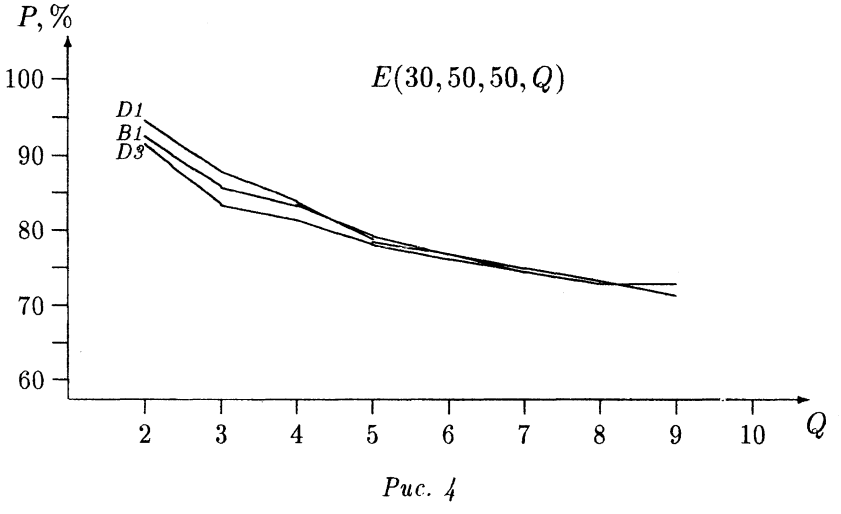

Рис. 4

нижней оценки для целевой функции. Расчеты проводились для задач из класса $R(30, 50, 50, Q)$ и $E(30, 50, 50, Q)$ с различными значениями параметра Q в пределах от 2 до 9. В каждом случае рассматривалась выборка из 30 задач.

Из рис. 3 и 4 следует, что в задаче (6)–(10) в отличие от задачи (11)–(14) трудно выделить явного лидера среди алгоритмов построения нижней оценки для целевой функции.

Кроме того, сравнивая рис. 3 и 4 с рис. 1 и 2, можно увидеть, что представленный в этой работе алгоритм для построения тупикового решения для задачи (6)–(10) дает значительно лучшую нижнюю оценку, чем для задачи (11)–(14).

Результаты серии расчетов, дающие сравнительную характеристику качества построения нижней оценки различными алгоритмами для задачи (6)–(10), приведены в табл. 2. Сравниваются алгоритм из [1], алгоритм из [7] и алгоритмы A_{B1} и A_{D1}. Расчеты проводились на классе задач $R(30, 100, 100, Q)$ с объемом выборки 30 задач.

Таблица 2

Q	Алгоритм [1]		Алгоритм A_{B1}		Алгоритм A_{D1}		Алгоритм A_L	
	НГ	Время	НГ	Время	НГ	Время	НГ	Время
2	77,83	1,04	79,07	0,6	79,52	0,55	86,89	819
3	73,11	1,1	76,63	0,77	77,50	0,66	86,12	911
4	67,67	1,1	69,69	0,77	77,50	0,66	80,34	1006
5	63,52	1,21	66,18	0,82	66,96	0,77	77,51	1098
6	60,79	1,26	64,45	0,82	63,45	0,77	76,05	1185
7	58,18	1,32	62,31	0,93	59,68	0,82	73,60	1268

Как и ранее, алгоритм A_L привлечен для демонстрации наилучшей нижней оценки для целевой функции, иллюстрирующей разрыв двойственности прямой и двойственной задач, время работы этого алгоритма преднамеренно велико. Из табл. 2 следует, что с ростом параметра Q точность нижней оценки у всех представленных алгоритмов падает, однако она заметно превышает аналогичные показатели для задачи (11)–(14) (см. табл. 1).

Результаты, приведенные на рис. 3, 4 и в табл. 2, свидетельствуют о том, что вопрос выбора наилучшего алгоритма для нахождения нижней оценки для задачи (6)–(10) не имеет однозначного ответа. Целесообразно применение различных вариантов алгоритма для разных классов задач. Для проведения дальнейших тестовых расчетов использовался алгоритм A_{D1}. Этот алгоритм на рассматриваемом классе задач продемонстрировал несколько лучшее «поведение» в ходе работы метода ветвей и границ, чем другие алгоритмы.

4.2. Тесты качества метода ветвей и границ

В настоящем разделе приводятся результаты расчетов, иллюстрирующие качество алгоритма ветвей и границ, использующего предложенный выше способ вычисления нижней оценки. Показатели качества алгоритма характеризуются следующими величинами:

Δ — величина относительного отклонения (в процентах) от оптимального решения приближенного решения, построенного согласно [4] по тупиковому решению на первой итерации алгоритма;

K — число итераций (шагов) алгоритма;

H — максимальная мощность множества неотсеченных частичных решений, наблюдаемых в ходе работы алгоритма;

t — время (в секундах) работы алгоритма.

Первая серия расчетов показывает влияние параметра Q, т. е. числа изделий-узлов в составе изделия-комплекта, на показатели качества алгоритма. Расчеты проводились для классов задач $R_0(30, 100, 100, Q)$, $E_0(30, 100, 100, Q)$, $R(30, 100, 100, Q)$ и $E(30, 100, 100, Q)$, где Q либо фиксированная величина со значениями от 2 до 9, либо случайная величина, равномерно распределенная в интервале от 3 до 7.

Результаты расчетов приведены в табл. 3 и 4. Видно, что точность Δ первого приближенного решения достаточно высока и не зависит ни от размерности задачи, ни от параметра Q. Вместе с тем можно заметить, что число итераций и другие показатели качества алгоритма зависят от значения параметра Q.

Таблица 3

Q	$R_0(30, 100, 100, Q)$				$E_0(30, 100, 100, Q)$			
	Δ	K	H	t	Δ	K	H	t
2	3,8	164	65	57,7	2,1	170	66	40,5
3	5,1	1695	703	649,7	2,9	1291	502	305,4
4	5,3	4013	1617	1288,0	2,8	2958	1073	697,4
5	8,3	5285	1927	1682,5	3,9	5392	1759	1431,9
6	3,1	10135	3302	2791,0	3,8	7083	2085	1692,4
7	5,4	10481	3030	2612,8	3,0	6032	1504	1083,6
8	3,4	8002	2051	1760,3	4,5	3124	711	512,4
9	3,7	6139	1404	1280,8	4,6	1616	360	320,5
3–7	4,2	2981	1164	961,7	4,5	1846	668	527,5

Следующая серия расчетов исследует зависимость от параметра Q среднего времени работы следующих алгоритмов:

1) алгоритма A_{D1} на задачах из $R(30, 100, 100, 4)$;
2) алгоритма A_3 на задачах из $R_0(30, 100, 100, 4)$;
3) алгоритма A_{D1} на задачах из $E(30, 100, 100, 4)$;
4) алгоритма A_3 на задачах из $E_0(30, 100, 100, 4)$.

Таблица 4

Q	$R(30,100,100,Q)$				$E(30,100,100,Q)$			
	Δ	K	H	t	Δ	K	H	t
2	3,1	401	122	207,7	1,7	49	18	29,4
3	5,0	2132	814	1014,3	1,9	250	102	134,2
4	6,1	4145	1660	1834,7	4,7	733	290	335,5
5	3,1	6880	2580	2817,8	9,1	770	298	349,3
6	8,3	6771	2348	2978,1	6,1	1943	661	757,7
7	5,3	14369	4491	4995,0	6,3	3570	1011	1204,2
8	5,6	8952	3106	3481,6	9,4	2305	697	899,8
3–7	0,8	2279	883	1022,2	7,5	254	101	134,8

Графики среднего времени работы алгоритмов при объеме выборки в 30 задач изображены на рис. 5. Видно, что с ростом Q общее время счета возрастает до некоторого предела, для каждого класса задач своего собственного, а затем начинает уменьшаться. В среднем максимальное время работы алгоритма достигается при $Q = 6 \div 7$.

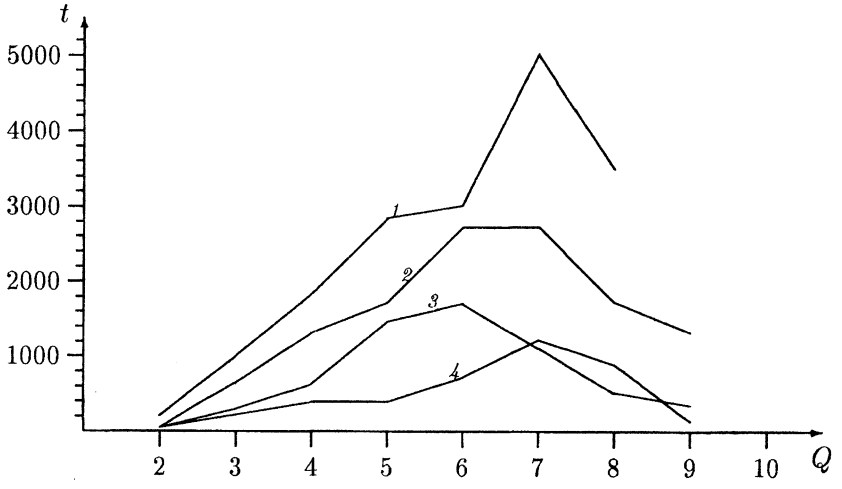

Рис. 5

Результаты серии тестовых расчетов, проводившихся при фиксированном параметре $Q = 4$ и показывающих зависимость основных показателей качества алгоритмов A_3 и A_{D1} от значений показателей размерности исследуемых задач (11)–(14) и (6)–(10), приведены в табл. 5 и 6. Объемы выборок в этих экспериментах также составляли 30 задач за

исключением случаев, помеченных в таблицах звездочкой, для которых объемы выборок составили не более трех задач.

Таблица 5

Размерность задачи	$R_0(m,n,p,4)$				$E_0(m,n,p,4)$			
	Δ	K	H	t	Δ	K	H	t
$20 \times 30 \times 30$	3,3	66	18	1,8	3,9	71	18	1,5
$30 \times 50 \times 50$	8,5	371	145	40,9	5,1	330	122	29,7
$30 \times 100 \times 100$	5,3	4013	1617	1288	2,8	2958	1073	697
$50 \times 100 \times 100$	3,1	16517	7934	9898*	1,7	8272	3180	4828*
$100 \times 100 \times 100$					6,8	6013	2371	8062*

Таблица 6

Размерность задачи	$R(m,n,p,4)$				$E(m,n,p,4)$			
	Δ	K	H	t	Δ	K	H	t
$20 \times 30 \times 30$	2,6	70	22	2,9	5,0	38	12	1,4
$30 \times 50 \times 50$	6,2	322	127	53,4	4,9	88	33	16,4
$30 \times 100 \times 100$	6,1	4145	1660	1834,7	4,7	733	290	335
$50 \times 100 \times 100$	4,2	19730	8077	10122*	8,5	1025	414	1067*
$100 \times 100 \times 100$					1,8	461	183	1058*

Следующая серия расчетов была произведена для сравнения алгоритмов A_3 и A_{D1} с аналогичным алгоритмом из [1]. В табл. 7 приведены результаты расчетов по *единичным* задачам, а не по серии задач, как это делалось раньше.

Таблица 7

Задача	Алгоритмы A_3 или $A_{\Gamma 1}$			Алгоритм [1]		
	K	K^0	t	K	K^0	t
$R_0(30,100,100,2)$	68	14	24	4257	2761	157
$R(30,100,100,2)$	323	14	173	10600	445	582
$R_0(30,100,100,5)$	3074	227	1444	636293	498774	34569
$R(30,100,100,7)$	16112	4695	5355	98054	4429	5139
$E(30,100,100,5)$	336	28	183	3960	2272	422
$E_0(30,100,100,5)$	3890	16	1155	11682058	10438020	7 суток

Алгоритм МВГ из работы [1] имеет иную схему ветвления — по объектам второго уровня (изделиям-комплектам), в нем используется

односторонняя схема ветвления. МВГ, используемый в данной работе, имеет двустороннюю схему ветвления по объектам первого уровня (изделиям-узлам), и одна его итерация содержит фактически две итерации в смысле алгоритма [1].

Обозначения K и t в табл. 7 имеют тот же смысл, что и в табл. 5 и 6, а параметр K^0 означает номер итерации, на которой было найдено точное решение задачи.

Из табл. 7 видно, что разница между временем работы алгоритма [1] на различных конкретных задачах может быть непредсказуемо велика.

В заключение приведем еще одну серию тестовых расчетов сравнения алгоритма A_3 с алгоритмом из [8], проведенном на классе задач, также предложенном в [8].

Отметим, что тестовые расчеты в [8] проводились на Sun Sparc System 600 Workstation и все программы были написаны на Sun Pascal. Все тестовые расчеты этой работы проводились на PC AT Pentium с процессором Intel 100 МГц, программы были написаны на Borland C++. Опишем класс задач $\mathscr{BL}$ из [8] в терминах данной статьи.

Таблица 8

Размерность задачи	Время работы алгоритма	
$m \times p \times n$	Алгоритм [8] Sun Sparc System 600	Алгоритм A_3 Pentium/Intel 100
$20 \times 100 \times 30$	103,9	12,9
$25 \times 150 \times 30$	292,6	37,3
$20 \times 100 \times 50$	473,2	31,1

Пусть $J = \{1, \ldots, p_j\}$ и $K = \{1, \ldots, p_k\}$ — два непересекающихся множества изделий-узлов. Изделие-комплект состоит только из двух изделий-узлов, по одному из множеств J и K. Очевидно, что общее число комплектов $p = p_j p_k$. Как и ранее, n — число работ, а общее число изделий-узлов $m = p_j + p_k$.

Начальные затраты на включение в систему изделий-узлов и изделий-комплектов определяются следующим образом.

а. Для $n = 10$ и $n = 20$ f_i, $i \in J$, является случайной величиной, равномерно распределенной в интервале 700–1000 (обозначим $f_i \sim (700 - 1000)$, $i \in J$), а $f_i \sim (300 - 800)$, $i \in K$; $g_l \sim (100 - 400)$, $i \in L$.

б. Для $n = 30$ и $n = 50$ $f_i \sim (2100 - 3000)$, $i \in J$ и $f_i \sim (900 - 2400)$, $i \in K$; $g_l \sim (300 - 1200)$, $i \in L$.

Стоимостная матрица c_{lj} строится следующим образом:

а) для $n = 10$ и $n = 20$ $c_{lj} \sim (100 - 400)$, $l \in L$, $j \in J$;

б) для $n = 30$ и $n = 50$ $c_{lj} \sim (300 - 1200)$, $l \in L$, $j \in J$.

Результаты сравнения двух алгоритмов — алгоритма из [8] и представленного в настоящей работе — показаны в табл. 8.

ЛИТЕРАТУРА

1. **Береснев В. Л., Гимади Э. Х., Дементьев В. Т.** Экстремальные задачи стандартизации. Новосибирск: Наука, 1978.
2. **Гимади Э. Х.** Эффективные алгоритмы для решения многоэтапной задачи размещения на цепи // Дискрет. анализ и исслед. операций. 1995. Т. 2, № 4. С. 13–31.
3. **Гимади Э. Х., Глебов Н. И., Дементьев В. Т.** Об одном методе построения нижней оценки и приближенного решения с апостериорной оценкой точности для задачи стандартизации // Управляемые системы: Сб. науч. тр. Новосибирск: Ин-т математики СО АН СССР, 1974. Вып. 13. С. 26–31.
4. **Гончаров Е. Н.** Математическая модель и метод решения двухуровневой задачи стандартизации // Тр. Ин-та математики / РАН. Сиб. отд-ние. 1994. Т. 28: Модели и методы оптимизации. С. 77–90.
5. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
6. **Кочетов Ю. А., Пащенко М. Г.** Лагранжевы релаксации в задаче выбора оптимального состава системы технических средств // Управляемые системы: Сб. науч. тр. Новосибирск: Ин-т математики СО РАН, 1993. Вып. 31. С. 48–58.
7. **Кочетов Ю. А., Пащенко М. Г.** Нижние границы в задаче выбора состава двухуровневой системы технических средств // Дискрет. анализ и исслед. операций. 1995. Т. 2. № 4. С. 32–41.
8. **Barros A. I., Labbe M.** A general model for the uncapacitated facility and depot location problem // Location Science. 1994. V. 2, N 3. P. 173–191.
9. **Gao L., Robinson E. P. Jr.** A dual-based optimization procedure for the two-echelon uncapacitated facility location problem // Naval Res. Logistics. 1992. V. 397. P. 191–212.

10. **Ro H. B., Tcha D. W.** A branch and bound algorithm for the two-level uncapacitated facility location problem with some side constraints // European J. Oper. Res. 1984. V. 18, N 3. P. 343–358.

11. **Tcha D. W., Lee B.-Y.** A branch-and-bound algorithm for the multilevel uncapacitated facility location problem // European J. Oper. Res. 1984. V. 18, N 1. P. 35–43.

Адрес автора:

Институт математики
им. С. Л. Соболева СО РАН,
пр. Академика Коптюга, 4,
630090 Новосибирск,
Россия

Статья поступила
2 июня 1997 г.,
переработанный вариант —
21 октября 1997 г.