Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра Теории функций и приближений

МЕТОД ВЕТВЕЙ И ГРАНИЦ В ЦЕЛОЧИСЛЕННОМ ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 412 группы
направления (специальности) 01.03.02 Прикладная математика и информатика
Механико-математического факультета
Козлитиной Маргариты Игоревны

Научный руководитель
Профессор, д.ф.-м.н. А.Л. Лукашов

Зав. кафедрой
Доцент, д.ф.-м.н. С.П. Сидоров

Саратов 2016

ВВЕДЕНИЕ

Потребности развития математической экономики привели к необходимости исследования таких задач на максимум и минимум, как например, задачи наилучшего приближения функций, оптимального набора параметров итерационного процесса или узлов интерполирования. На математическом языке такие задачи могут быть сформулированы как задачи отыскания экстремума (максимума или минимума) некоторой функции или функционала, выражающего собой качество управления из заданного множества некоторого пространства. Требования принадлежности управления некоторому множеству выражает собой ограничения, обычно вытекающие из законов сохранения ограниченности наличных ресурсов, возможностей технической реализации управления, нежелательности каких - либо запрещенных состояний. Задачи отыскания экстремума функции на множество принято называть экстремальными задачами.

В настоящее время теория экстремальных задач обогатилась фундаментальными результатами, появились ее новые разделы, такие как стохастическое, динамическое, выпуклое программирование, оптимальное управление. Потребности практики способствовали бурному развитию методов приближенного решения экстремальных задач.

Применение традиционных методов оптимизации не всегда позволяет достичь желаемого результата - действительной точки оптимума - за приемлемое время, или, по крайней мере, для этого требуются значительные вычислительные ресурсы. Поэтому имеет смысл разрабатывать новые направления в области решения сложных задач оптимизации, которые бы позволили избежать основных недостатков классических методов. К таким направлениям, например, можно отнести решение задач методом ветвей и границ. Данный метод уже давно зарекомендовал себя в качестве эффективного метода решения широкого класса задач.

Структура дипломной работы состоит из введения, трех разделов: целочисленное линейное программирование, методы решения задач с сепарабельными функциями, пример решения нелинейной целочисленной транспортной задачи, заключения, списка использованных источников, приложения.

Целью работы является изучение метода ветвей и границ в целочисленном линейном программировании. В связи с поставленной целью в дипломной работе решались следующие **задачи**:

– решение нелинейной целочисленной транспортной задачи, используя дельта-метод без ограничений на целочисленность решения

– полученный с помощью дельта-метода результат обработать, используя метод ветвей и границ

– создать программный продукт, реализующий метод ветвей и границ на примере поставленной задачи.

Основное содержание работы.

Общая задача линейного программирования имеет вид:

$$\max(cx)$$
$$\begin{cases} A'x \le b', \\ A''x = b''. \end{cases}$$

Общая постановка задачи целочисленного программирования отличается от общей постановки задачи линейного программирования лишь наличием дополнительного ограничения. Этим ограничением является требование целочисленности, в соответствии с которым значения всех или части переменных модели в оптимальном решении являются целыми неотрицательными числами.

На первый взгляд может показаться, что дополнительное требование целочисленности не усложняет задачу, однако в общем случае задачей целочисленного линейного программирования (ЗЦЛП) оказывается более трудной, чем задача линейного программирования (ЗЛП).

В первом разделе также проведен обзор методов целочисленного линейного программирования, симплекс-метода и выделен метод ветвей и границ для задачи целочисленного линейного программирования.

Одним из самых распространенных алгоритмов линейного программирования является симплекс-метод. Этот метод достаточно эффективен при решении реальных задач, но экспотенциален при некоторых искусственных примерах.

Определение. Множество P векторов в R^n называется (*выпуклым*) полиэдром, если

$$P = \{x | Ax \le b\}$$

для некоторых матрицы aи вектора b, т. е. если P является пересечением конечного числа аффинных полупространств.

Идея симплекс-метода заключается в движении по полиэдру, описываемому линейной программой, по ребрам от вершины к вершине, пока не будет достигнута оптимальная вершина.

На практике симплекс-метод весьма быстр: опыт подтверждает почти линейную по размерности оценку числа итераций.

Многие задачи линейного программирования, если рассмотреть их содержательную постановку, являются на самом деле целочисленными задачами линейного программирования.

Таким образом, берем обычные задачи линейного программирования и всего лишь добавляем ограничение: все переменные должны принимать целые значения. Но несмотря на такое, казалось бы, небольшое усложнение, само решение задачи становится куда более объемным. Если речь идет о задаче с двумя переменными, проще всего применить графический метод, если же переменных больше, приходится использовать специальные методы для этого класса задач: метод отсечений, метод Гомори; комбинаторные методы, метод ветвей и границ.

Методы отсечений.

Методы отсечений относятся к численным методам решения дискретных задач оптимизации (методам дискретного программирования). Они предназначены для решения целочисленных задач линейного программирования (ЛП).

Метод ветвей и границ.

Метод ветвей и границ относится к группе комбинаторных методов дискретного программирования и является одним из наиболее распространенных методов этой группы. Центральную идею комбинаторных методов составляет замена полного перебора допустимого множества X частичным перебором. В случае метода ветвей и границ это осуществляется путем последовательного разбиения допустимого множества на подмножества (ветвления) и вычисления оценок (границ), позволяющих отбрасывать подмножества, заведомо не содержащие решения задачи. При реализации общей схемы метода ветвей и границ для различных задач дискретного программирования необходимо, исходя из специфики этих задач, конкретизировать правила ветвления и вычисления границ.

Рассмотрим задачу линейного программирования P и соответствующую ей задачу целочисленного линейного программирования P^Z:

$$P:\quad \begin{matrix}\max cx\\ Ax \le b,\end{matrix} \qquad\qquad P^Z:\quad \begin{matrix}\max cx\\ \begin{cases}Ax \le b,\\ x \in Z^n.\end{cases}\end{matrix}$$

Если все компоненты оптимального вектора $\tilde{x}$ задачи Pцелочисленны, то xявляется решением задачи P^Z. В противном случае выберем какую-нибудь нецелую компоненту $\tilde{x}_i$ вектора $\tilde{x}$ и рассмотрим две задачи:

$$P_0:\quad \begin{matrix}\max cx\\ \begin{cases}Ax \le b,\\ x_i \le [\tilde{x}_i],\end{cases}\end{matrix} \qquad\qquad P_1:\quad \begin{matrix}\max cx\\ \begin{cases}Ax \le b,\\ x_i \ge [\tilde{x}_i].\end{cases}\end{matrix}$$

Очевидно, что для того, чтобы решить P^Z, достаточно решить каждую из задач P_0^Z и P_1^Z и выбрать среди их решений то, на котором значение целевой функции больше, Если условия каждой из задач P_0^Z и P_1^Z несовместны, то также несовместными будут и условия задачи P^Z.

К задачам P_0^Z и P_1^Z можно применить те же рассуждения, что и к задаче P^Z. А именно, отбросим требование целочисленности задачи P_0^Z. Решим полученную задачу P_0. Если все компоненты оптимального вектора $\hat{x}$задачи P_0 целочисленны, то x является решением задачи P_0^Z. В противном случае выберем какую-нибудь нецелую компоненту $\hat{x}_j$ вектора $\hat{x}$ и рассмотрим две задачи:

$$P_{00}:\quad \begin{matrix}\max cx\\ \begin{cases}Ax \le b,\\ x_i \le [\tilde{x}_i],\\ x_j \le [\hat{x}_i],\end{cases}\end{matrix} \qquad\qquad P_{01}:\quad \begin{matrix}\max cx\\ \begin{cases}Ax \le b,\\ x_i \le [\tilde{x}_i],\\ x_i \ge [\hat{x}_j],\end{cases}\end{matrix}$$

Для того, чтобы решить P_0^Z, достаточно решить каждую из задач P_{00}^Z и P_{01}^Z и выбрать среди их решений то, на котором значение целевой функции больше. Аналогичные рассуждения можно провести и с задачей P_1^Z и т.д.

Описанный процесс можно описать в виде корневого бинарного дерева. Корню этого дерева соответствует ЗЛП P. Если какой-либо вершине соответствует ЗЛП P' с нецелочисленным оптимальным вектором $\acute{x}$, то ее потомкам (дочерним вершинам) соответствуют задачи P'_0 и P'_1, полученные из P' приписыванием к ограничениям дополнительных неравенств $x_i \leq [\acute{x}_i], x_i \geq [\acute{x}_i]$ соответственно, где $\acute{x}_i$ – какая-либо дробная компонента вектора $\acute{x}$. Вершина является листом, если соответствующая ей задача имеет целочисленное решение, либо условия задачи несовместны.

Алгоритм строит описанное дерево, в каждой вершине решая соответствующую ЗЛП до тех пор, пока дерево не будет построено полностью (далее, однако, мы покажем, что дерево иногда не обязательно строить до конца). Среди всех оптимальных значений целевой функции для задач, соответствующих листьям этого дерева, нужно выбрать максимальное. Это значение является оптимальным значением целевой функции для исходной задачи P^Z, а соответствующий ему вектор – ее решением.

Построение задач P'_0 и P'_1 из ЗЛП P называется ветвлением. Другой принцип, на котором основан метод ветвей и границ, называется оцениванием.

Во втором разделе рассматривается приближенное решение сепарабельных задач нелинейного программирования в дельта-форме. Мы имеем дело с приближенными методами для решения задач нелинейного программирования следующего вида:

$$\sum_{j=1}^{n} g_{ij}(x_j)\{\leq, =, \geq\}b_i, \qquad i = 1, \dots, m,$$
$$x_j \geq 0, \qquad j = 1, \dots, n; \qquad (1)$$
$$\max z = \sum_{j=1}^{n} f_j(x_j).$$

$$\hat{f}_j(x_j) + \sum_{k=0}^{r_j} \delta_{kj} f_{kj}\,, \qquad f_{kj} = f_j(x_k),$$

$$g_{ij}(x_j) = \sum_{k=0}^{r_j} \delta_{kj} g_{kij}\,, \qquad g_{kij} = g_{ij}(x_6), i = 1, \dots, m,$$

$$x_j = \sum_{k=0}^{r_j} \delta_{kj} x_{kj}.$$

Предполагается, что все функции в ограничениях и целевая функция являются сепарабельными. Запишем задачу в другой форме:

$$\sum_{j=1}^{n}\sum_{k=1}^{r_j}(\Delta g_{kij})\delta_{kj}\,\{\le,=,\ge\}b_i-\sum_{j=1}^{n}g_{0ij}\,,\qquad i=1,\dots,m,$$

$$0\le\delta_{kj}\le 1 \text{ при всех } k \text{ и } j, \text{найти} \tag{2}$$

$$\max z=\sum_{j=1}^{n}\sum_{k=1}^{r_j}(\Delta f_{kj})\delta_{kj}\,.$$

Задача (2) называется задачей в δ-форме. Такая задача содержит $\sum_j r_j$ переменных и m ограничений. Кроме того, имеются $\sum_j r_j$ верхних границ, которые нет необходимости рассматривать как ограничения общего вида. Преимущество δ -формы в том, что мы имеем дело с малым базисом; недостаток состоит в необходимости использовать специальный симплексный алгоритм, особым образом учитывающий верхние границы переменных. Также эта форма должна учитывать ограничительные условия на базис.

Приближенная задача в δ-форме для задачи (1) может быть записана в виде

$$\sum_{j=1}^{n}\sum_{k=1}^{r_j}\left(\Delta g_{kij}\right)\delta_{kj}\,\{\le,=,\ge\}b_i-\sum_{j=1}^{n}g_{0ij}\,,\qquad i=1,\dots,m,$$

$$0\le\delta_{kj}\le 1 \text{ для всех } k,j, \text{если } \delta_{kj}>0, \text{то } \delta_{uj}=1, u=1,\dots,k-1,(3),$$

$$\text{найти} \max z=\sum_{j=1}^{n}\sum_{k=1}^{r_j}(\Delta f_{kj})\delta_{kj}\,.$$

и сводится к задаче целочисленного линейного программирования:

$$\sum_{j=1}^{n}\sum_{k=1}^{r_j}(\Delta g_{kij}\delta_{kj})\,\{\ge,=,\ge\}b_i-\sum_{j=1}^{n}g_{oij}\,, i=1,\dots m,(4),$$

$$\delta_{kj}-\psi_{kj}\ge 0 \text{ для всех } k,j,$$

$$\delta_{k+1,j}-\psi_{kj}\le 0 \text{ для всех } k,j,$$

$$0\le\delta_{kj}\le 1 \text{ для всех } k,j,$$

$$\psi_{kj}\ge 0, \psi_{kj} \text{ целые для всех } k,j;$$

$$\text{найти } \max z = \sum_{j=1}^{n} \sum_{k=1}^{r_j} (\Delta f_{kj})\, \delta_{kj}$$

Сведение задачи (3) к целочисленной задаче линейного программирования (4) неизбежно влечет за собой значительное увеличение числа ограничений. Даже если m и n крайне малы, например 5 и 20 соответственно, и если для каждой исходной переменной используется пять переменных δ_{kj} , то задача (4) будет иметь 305 ограничений, включая ограничения на верхние границы.

В третьем разделе проводится решение нелинейной целочисленной транспортной задачи. В качестве исходных данных используется, как и в классической транспортной задаче, m обычных пунктов производства с объемом производства a_i, $i = 1, \ldots, m$, и n пунктов потребления с объемом потребления b_j , $j = 1, .., n$. Стоимость перевозки из обычных пунктов производства в обычные пункты потребления линейно зависит от объемов перевозки. Кроме того, у каждого i-го пункта производства имеется свой индивидуальный пункт потребления с объемом y_i . У каждого j-го пункта потребления также кроме обычных поставщиков имеется свой индивидуальный пункт производства с объемом w_j. Стоимость перевозки в дополнительные пункты потребления и из дополнительных пунктов производства линейно зависит от объемов перевозки.

Из условия задачи имеем целевую функцию:

$$\frac{w_2^3}{30} + \frac{w_1^2}{4} + 4x_{11} + 8x_{12} + \frac{y_1^2}{5} + 16x_{21} + 18x_{22} + \frac{y_2^3}{15} \to min$$

и ограничения:

$$x_{11} + x_{12} + y_1 = 150$$

$$x_{21} + x_{22} + y_2 = 170$$

$$w_1 + x_{11} + x_{21} = 140$$

$$w_2 + x_{12} + x_{22} = 180$$

$$x_{ij} \geq 0, y_i \geq 0, w_i \geq 0.$$

Решение задачи будет состоять из двух этапов:

1. Решаем нелинейную задачу приближенными методами решения нелинейных задач с сепарабельными функциями.

2. Если решение задачи получилось нецелочисленное, то применяем метод ветвей и границ.

ЗАКЛЮЧЕНИЕ

В работе рассмотрены метод ветвей и границ целочисленного линейного программирования и приближенный метод решения задач сепарабельного нелинейного программирования в δ-форме.

Показано, что сочетание двух методов достаточно эффективно может быть применено к рассматриваемому классу транспортных задач с дополнительными пунктами производства и потребления. Совпадение результатов с приведенными в работе А.С. Есенкова и др. из журнала «Известия РАН. Серия Проблемы и системы Управления» за 2015 год показывает правильность работы программы.