

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"МИРЭА - Российский технологический университет"РТУ МИРЭА |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА № 6**

**Двойственные Задачи**

**по дисциплине**

**«Теория принятия решений»**

Студент группы: ИКБО-05-19 Выонг Чыонг Шон *(Фамилия студента)*

Руководитель работы Железняк Л.М.\_

*(Фамилия преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2021

**Двойственные Задачи**

С каждой ЗЛП тесно связана другая линейная задача, называемая двойственной. Тогда первоначальная задача называется исходной или прямой

**Задача**: Фирма выпускает изделия четырех типов. При этом используется сырье двух видов, запасы которого соответственно 1200 и 1000 единиц. Нормы расхода сырья на изготовление каждого типа продукции, а также доход, полученный от выпуска единицы каждого типа продукции, заданы таблицей:

*Таблица П.3.1* Нормы расхода сырья:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сырье | Нормы расхода | | | | Обьем ресурсов |
| I | II | III | IV |
| 1 | 4 | 2 | 1 | 4 | 1200 |
| 2 | 1 | 5 | 3 | 1 | 1000 |
| Доход | 15 | 5 | 3 | 20 |

Составить план производства, обеспечивающий фирме наибольший суммарный доход.

**Общие правила составления двойственных задач**

Целевая функция:

F(x) = 15x1 + 5x2 + 3x3 + 20x4 → max

Ограничения:

Рассматриваемая задача относится к симметричным двойственным задачам на отыскание максимального значения целевой функции.

Найдем соответствующую двойственную задачу. Введем вектор двойственных переменных размерности 3 (по числу уравнений

F(x) = 15x1 + 5x2 + 3x3 + 20x4 + 0x5 + 0x6

Теперь построим начальную симплекс-таблицу. Запишем систему (3.4) в векторной форме:

Векторы 𝐴5, 𝐴6 являются линейно независимыми единичными векторами 2х-мерного пространства и образуют базис этого пространства. Поэтому за базисные переменные выбираем переменные x5, x6. Небазисными переменными являются x1, x2, x3, x4.

Потом мы найдем первое базисное допустимое решение. Для этого свободные переменные x1, x2, x3, x4 приравниваем нулю. В результате получим разложение

которому соответствует первоначальный опорный план:

Для проверки плана на оптимальность построим первую симплекстаблицу. Введем в рассмотрение вектор коэффициентов целевой функции при базисных переменных

*Таблица 3.2.* Начальная симплекс-таблица задачи о максимальном доходе

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 15 | 5 | 3 | 20 |  |  |
|  |  | x1 | x2 | x3 | x4 |  |  |
| 0 | x5 | 4 | 2 | 1 | 4 | 1200 | 1200/4 = 300 min |
| 0 | x6 | 1 | 5 | 3 | 1 | 1000 | 1000/1 = 1000 |
|  | ***f*** | -15 | -5 | -3 | -20 | 0 |  |
|  |  |  |  |  |  | Q |  |

Найдем относительные оценки ∆1, ∆­2 и значение целевой функции 𝑄:

Для оптимальности опорного решения в задаче на максимум требуется выполнение неотрицательности всех относительных оценок ∆j ≥ 0. Так как оценки ∆1= −15, ∆2= −5, ∆3= −3 и ∆4= −20 в f-строке отрицательны, то это свидетельствуют о возможности улучшения полученного решения.

Наибольшая по модулю отрицательная оценка ∆4= −20. В базис будет включена соответствующая ей небазисная переменная 𝑥­4. Составим отношения свободных членов к положительным элементам разрешающего столбца (3.2) Данные отношения приведены справа от таблицы. Наименьшему частному min (300, 1000) =3000 соответствует строка с переменной 𝑥5. Эта переменная исключается из базиса. Разрешающим элементом является число 𝑎21 = 4.

Далее построим новую симплекс-таблицу.

*Таблица 3.3.* Симплекс преобразования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 15 | 5 | 3 | 0 |  |
|  |  | x1 | x2 | x3 | x5 |  |
| 20 | x4 | 1 | 1/2 | 1/4 | 1/4 | 300 |
| 0 | x6 | 0 | 9/2 | 11/4 | -1/4 | 700 |
|  | ***f*** | 5 | 5 | 2 | 5 | 6000 |
|  |  |  |  |  |  | Q |

В таблице 3.3 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

Остальные элементы рассчитываются по «правилу прямоугольника».

В последней таблице f-строка не содержит отрицательных оценок, что свидетельствует об оптимальности полученного решения:

Таким образом, фирма должна выпускать в сутки 𝑥­4 = 300. Тогда фирма получит максимальный доход от продажи 6000.

|  |
| --- |
| import operator  from copy import copy, deepcopy  fa = open("data/function1.txt")  minmax = str(fa.readline().replace("\n", ""))  C = list(map(int, fa.readline().split()))  B = list(map(int, fa.readline().split()))  A = []  for f in fa:      A.append(list(map(int, f.split())))  if minmax == "MIN":      for i in range(0, len(C)):          C[i] = (-1) \* C[i]  Cb = [0] \* len(A)  F = [None] \* (len(C))  Q = 0  Basic = []  Non\_basic = []  for i in range(len(C)):      Non\_basic.append("x" + str(i + 1))  for i in range(len(A)):      Basic.append("x" + str(i + len(C) + 1))  def printTable(Basic, Non\_basic, C, B, A, Cb, F, Q):      print("\t C", end="\t")      for c in C:          print(c, end="\t")      print()      print("Cb \t\t", end ="")      for non\_basic in Non\_basic:          print(non\_basic, end="\t")      print("B")      for i in range(len(A)):          print(Cb[i], end="\t")          print(Basic[i], end="\t")          for j in range(len(C)):              print(A[i][j], end="\t")          print(B[i])      print("\t F", end = "\t")      for f in F:          print(f, end = "\t")      print(Q)  print("Starting Table:")  printTable(Basic, Non\_basic, C, B, A, Cb, F, Q)  for f in range(len(F)):      a = list(map(operator.itemgetter(f), A))      S = 0      for i in range(len(Cb)):          S += (Cb[i] \* a[i])      F[f] = S - C[f]  for i in range(len(Cb)):      Q += (Cb[i] \* B[i])  iter = 1  check2 = False  while(True):      print("----------------------------------")      print("Iteration :", iter)      printTable(Basic, Non\_basic, C, B, A, Cb, F, Q)      Q\_old = Q      for f in range(len(F)):          a = list(map(operator.itemgetter(f), A))          S = 0          for i in range(len(Cb)):              S += (Cb[i] \* a[i])          F[f] = S - C[f]      for i in range(len(Cb)):          Q += (Cb[i] \* B[i])      printIter = False      for f in F:          if f < 0:              printIter = True      if not printIter:          Q = Q\_old          break      min\_f = min(F)      print("Pivot Column: " + Non\_basic[F.index(min\_f)])      r\_min = float("inf")      row = 0      check = False      for i in range(0, len(A)):          if A[i][F.index(min\_f)] == 0:              continue          else:              r = B[i] / A[i][F.index(min\_f)]              #print(r)              if (r > 0 and r < r\_min):                  r\_min = r                  row = i                  check = True      if check == False:          print("Cannot solve this problem!")          check2 = True          break      print("Pivot Row: " + Basic[row])      print("Pivot Element: " + str(A[row][F.index(min\_f)]))      # Swap basic and non-basic of pivot row and col      c\_old = C[F.index(min\_f)]      C[F.index(min\_f)] = Cb[row]      Cb[row] = c\_old      non\_basic\_old = Non\_basic[F.index(min\_f)]      Non\_basic[F.index(min\_f)] = Basic[row]      Basic[row] = non\_basic\_old      A\_new = deepcopy(A)      B\_new = B.copy()      pivot\_ele = A[row][F.index(min\_f)]      # Calculate new matrix      for i in range(0, len(A)):          for j in range(0, len(A[0])):              if i == row:                  if j == F.index(min\_f):                      A[i][j] = 1 / pivot\_ele                  else:                      A[i][j] = A\_new[i][j] / pivot\_ele              elif j == F.index(min\_f):                  A[i][j] = -1 \* (A\_new[i][j] / pivot\_ele)              else:                  A[i][j] = (A\_new[i][j] \* pivot\_ele) - (A\_new[i][F.index(min\_f)] \* A\_new[row][j])                  A[i][j] /= pivot\_ele          if i == row:              B[i] = B\_new[i] / pivot\_ele          else:              B[i] = (B\_new[i] \* pivot\_ele) - (A\_new[i][F.index(min\_f)] \* B\_new[row])              B[i] /= pivot\_ele      F\_new = F.copy()      for i in range(len(F)):          if i == F\_new.index(min\_f):              F[i] = -1 \* (F\_new[i] / pivot\_ele)          else:              F[i] = (F\_new[i] \* pivot\_ele) - (A\_new[row][i] \* F\_new[F\_new.index(min\_f)])              F[i] /= pivot\_ele      Q = 0      for i in range(len(Cb)):          Q += (Cb[i] \* B[i])      iter += 1  if check2 == False:      print("----------------------------------")      print("Final Table reached in", iter, "iterations")      print("Coefficients: ")      for i in range(len(Basic)):          if Cb[i] != 0:              print("\t" + Basic[i] + ": " + str(B[i]))      print("Optimal value: " + str(Q)) |

Код программа

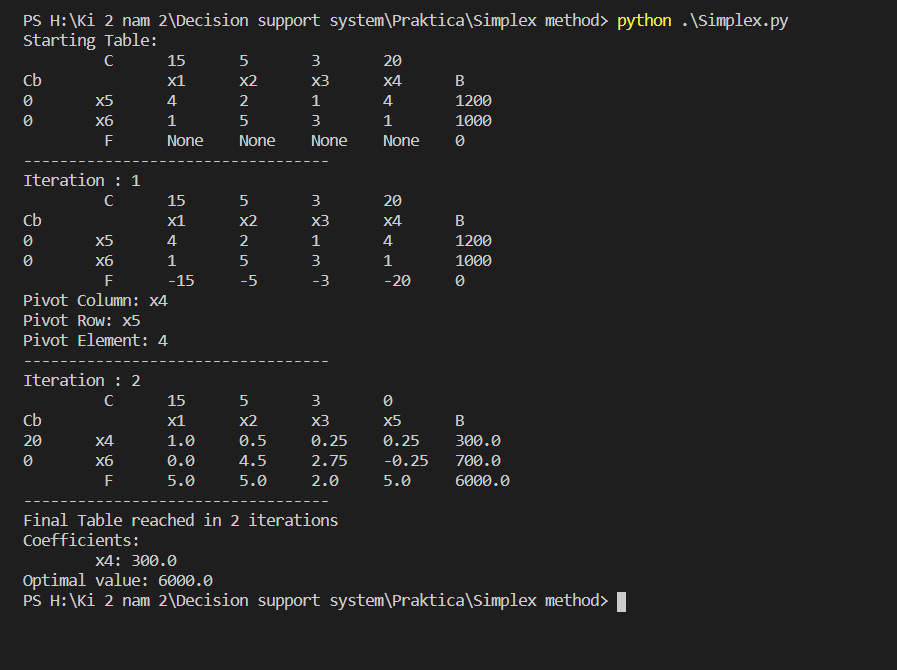


Рис 1: Запустить программу