Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №3
Первая фаза симплекс-метода.

Выполнил:
студент гр. 953505
Басенко К. А.

Руководитель:
доцент
Алёхина А. Э.

Минск 2022

Краткие теоретические сведения

Цель первой фазы симплекс метода - выяснить, является ли задача линейного программирования в канонической форме совместной и в случае совместности найти какой-нибудь базисный допустимый план с множеством базисных индексов. Задача распознавания совместности сводится к решению вспомогательной задачи ЛП. Чтобы выяснить, является ли ЗЛП совместной, необходимо:

- Сделать все свободные члены неотрицательными
- Составить вспомогательную задачу ЛП. Для того, чтобы составить вспомогательную задачу введем новые искусственные переменные. Искусственных переменных ввести понадобится столько, сколько основных ограничений: по одной искусственной переменной на одно основное ограничение.

Решаем задачу симплекс-методом. Для этого понадобится начальный базисный план. Чтобы получить начальный базисный план надо значения неискусственных переменных положить равными нулю. Значения искусственных переменных однозначно определяются из основных ограничений.

В качестве начального базисного плана берем вектор из нулей, количество которых равно количеству неискусственных переменных, и свободных членов. Этому базисному допустимому плану соответствует множество базисных индексов, которые состоят из индексов искусственных переменных. Получили вспомогательную задачу, которую решаем симплекс-методом.

Если вспомогательная задача совместна, и ее целевая функция ограничена сверху, то с помощью симплекс-метода получим оптимальный базисный план с множеством базисных индексов B.

Если в оптимальном плане значения всех искусственных переменных равны 0, то исходная задача ЛП является совместной. Из базисного плана убираем искусственные переменные.

Если в базисном плане значения, соответствующие неискусственным переменным, равны 0, то за множество базисных индексов берем $\boldsymbol{B}$, если там нет индексов искусственных переменных. В ином случае по алгоритму корректируем множество

базисных индексов так, что ***B*** становится текущим множеством базисных индексов.

Алгоритм итерационный: на каждой итерации из множества ***B*** выбирается индекс искусственной переменной, который либо удаляется из ***B***, либо заменяется на индекс неискусственной переменной.

Пусть *B* содержит индекс искусственной переменной , где ***i*** - количество неискусственных переменных. Заменим этот индекс на индекс неискусственной переменной. Для каждого небазисного индекса ***j*** вычислим вектор ***l***: = *.

Два случая:

- Среди небазисных индексов нашли индекс ***j***, для которого . В этом случае в ***B*** заменяем на .
- Для любого небазисного индекса ***j*** . В этом случае система основных ограничений исходной задачи имеет избыточное ограничение, удаление которого не меняет множество допустимых планов. Таким ограничением является ***i***-е ограничение, которое мы убираем. Из вспомогательной задачи удаляем искусственную переменную с индексом . Из ***B*** удаляем .

Программная реализация

Первая фаза симплекс-метода реализована в функциии, на вход в которую идут: матрица условий *_syst*, вектор ограничений *b:*

```
def Simplex_first_phase(_syst, _b):
```

Делаем свободные члены неотрицательными:

```
for i in range(restrictions_count):
    if _b[i] < 0:
        _b[i] *= -1
        _syst[i] = [-j for j in _syst[i]]

    _syst[i] += [0] * restrictions_count
    _syst[i][-restrictions_count + i] = 1
```

Чтобы найти начальный базисный план, значения неискусственных переменных положим равными 0:

```
_x = [0] * n + _b

_Jb = []
for i in range(-restrictions_count, 0):
    _Jb.append(i + len(_x))

_c = [0] * (n + restrictions_count)

for j in _Jb:
    _c[j] = -1
```

Решаем вспомогательную задачу, проверяем совместность:

```
_x, F, _Jb, invAb = Simplex(_syst, _b, _c, _x, _Jb)

for i in range(len(_x) - restrictions_count, len(_x)):
    if (_x[i] != 0):
        print("Задача несовместна")
        return

print("Задача совместна")
```

Находим базисные индексы:

```
while True:
    _syst = np.array(_syst)
```

Вычисляем вектор для каждого небазисного индекса:

```
    have_excess = True
    for j in range(n):
        if (j in _Jb):
            continue

        lj = invAb.dot(_syst[:,j])
```

Если среди небазисных индексов нашли индекс ***j***, для которого :

```
            for i, j in enumerate(_Jb):
                if j > n:
                    if i < len(lj) and lj[i] != 0:
                        _Jb[i] = j
                        have_excess = False
                        break
```

Если есть избыточное ограничение, убираем его; удаляем искусственную переменную из этого ограничения:

```
if (have_excess):
    for j in _Jb:
        if j > n:
            excess = j - n
            _syst = _syst[:excess] + _syst[excess + 1:

            _Jb.remove(j)

            for i in range(len(_syst)):
                _syst[i] = _syst[i][:j] + _syst[i][j +

            break
```

Если в базисном плане значения, соответствующие искусственным переменным, не равны 0, то вычисляем заново базисные индексы:

```
has_artificial_basis = False
for j in _Jb:
    if j > n:
        has_artificial_basis = True

if (has_artificial_basis):
    continue

print(f"Базисный план: {set(_Jb)}")
break
```

Тестовые примеры

Входные данные для тестирования:

```
test_cases = [
    [
        syst := [
            [1, 1, 1, 0],
            [1, -1, 0, -1],
        ],
        b := [2, 3],
        res := "задача несовместна"
    ],
```

```
    [
        syst := [
            [1, 1, 1],
            [2, 2, 2]
        ],
        b := [0, 0],
        res := (
            x := [0, 0, 0],
            Jb := [0]
            )
    ],
```

```
    [
        syst := [
            [-1, 3, 1, 0],
            [4, 3, 0, 1],
            [2, 9, 2, 1],
        ],
        b := [9, 24, 42],
        res:= (
            x := [3, 4, 0, 0],
            Jb := [0, 1]
            ),
    ],
```

```
    [
        syst := [
            [-1, 1, 1, 0],
            [1, 1, 0, 1],
            [0, 2, 1, 1],
        ],
        b := [2, 6, 8],
        res := (
            x := [2, 4, 0, 0],
            Jb := [0, 1],
        ),
    ],
]
```

Выполняется метод с предоставленными данными, возвращается результат, выводится полученный результат, выводится ожидаемый результат:

```python
for test_case in test_cases:
    *data, res = test_case
    Simplex_first_phase(*data)

    if type(res) != str:
        res = f"x' = {tuple(res[0])}, Jb = {set(res[1]
    print("-" * 10)
    print(f'Правильный ответ:\n{res}\n\n')
```

Вывод, соответствующий предоставленным входным данным:

```
План отпимальный
F(2, 0, 0, 0, 0, 1) = -1
Задача несовместна
----------
Правильный ответ:
задача несовместна


План отпимальный
F(0, 0, 0, 0, 0) = 0
Задача совместна
x' = (0, 0, 0)
Базисный план: {0}
----------
Правильный ответ:
x' = (0, 0, 0), Jb = {0}
```

```
План отпимальный
F(3, 4, 0, 0, 0, 0, 0) = 0
Задача совместна
x' = (3, 4, 0, 0)
Базисный план: {0, 1}
----------
Правильный ответ:
x' = (3, 4, 0, 0), Jb = {0, 1}


План отпимальный
F(2, 4, 0, 0, 0, 0, 0) = 0
Задача совместна
x' = (2, 4, 0, 0)
Базисный план: {0, 1}
----------
Правильный ответ:
x' = (2, 4, 0, 0), Jb = {0, 1}
```