Введение

Представленное решение подойдет для любых данных, представленных в нужном формате.

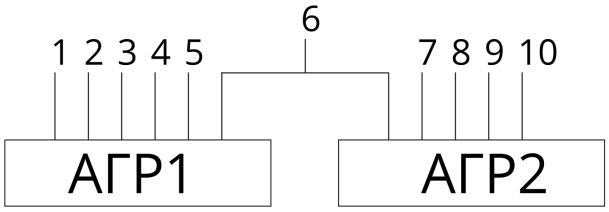
В конце рассчетно-графической работы в приложении будет представлен пример работы решения в условиях использования трех автоматических раскройных комплексов (в будущем будем называть их АГР).

1. Проблема

Задача построения оптимального плана.

При этом даны:

- Функциональная схема расположения АГР:



- Фонд свободного времени для каждого АГР:

  $T_1 = 4$ ч

  $T_2 = 6$ ч

- Указанные в таблице
  - План производства продукции, перевыполнять который нежелательно;
  - Производительность работ для каждого варианта технологии;
  - Число отходов для каждого варианта технологии;
  - Варианты технологий.

Продукт / Вариант	L_1,Т	L_2,Т	L_3,Т	L_4,Т	L_5,Т	L_6,Т	отходы	Р Т/час
1	0	0,962	0	0	0	0	0,038	7
2	0,981	0	0	0	0	0	0,019	6
3	0	0	0	0,963	0	0	0,037	8
4	0	0	0,667	0,316	0	0	0,017	8
5	0	0,12	0	0	0	0,845	0,035	6
6	0,497	0,481	0	0	0	0	0,022	10
7	0,736	0,237	0	0	0	0	0,027	11
8	0	0	0,54	0	0	0,423	0,037	8
9	0	0	0,135	0	0,845	0	0,02	7
10	0	0,356	0	0	0,625	0	0,019	9
b,т план	100	100	100	100	50	150		

2. Содержательная постановка

Необходимо определить, как нужно распределить время работы АГР по различным вариантам работы, чтобы выполнить план, уложиться в ограничения по времени работы каждого АГР.

В качестве дополнительной задачи, необходимо разработать план работы АГР, чтобы:

1. минимизировать расход исходного материала
2. минимизировать отходы производства

3. Формальная постановка

3.0 Обозначения

Введем формальные обозначения для решения задачи в общем и в частном виде:

- $m = 10$ - число различных вариантов работы АГР;
- $n = 2$ - число различных АГР;
- $k = 6$ - число различных продуктов выхода АГР;
- $L = ||l_{ij}||,\ i \in \{1, 2, \dots, k\},\ j \in \{1, 2, \dots, m\}$ - Двумерное пространство долей выхода продуктов технологий производства;
- $P = (p_1, p_2, \dots, p_m)$ - Одномерное пространство мощностей каждого варианта;
- $B = (b_1, b_2, \dots, b_k)$ - Одномерное пространство плана производства продуктов;
- $W = (w_1, w_2, \dots, w_m)$ - Одномерное пространство отходов производства при каждом варианте;
- $T = (t_1, t_2, \dots, t_n)$ - Одномерное пространство фонда свободного времени АГР;
- $\mathcal{N} = (N_1, N_2, \dots, N_n)$ - Двумерное пространство вариантов работы каждого АГР. Здесь N_i - Одномерное подпространства пространства $\mathcal{N}$ номеров вариантов работы для i АГР соответственно;
- $X = ||x_{ij}||,\ i \in \{1, 2, \dots, n\},\ j \in \{1, 2, \dots, m\}$ - Двумерное пространство числа часов работы АГР на варианте;
- $E_{\text{len}} = (1, 1, \dots, 1)$ - Одномерное пространство единиц размера len.

3.1 Постановка задачи минимизации отходов

Запишем математическую модель задачи минимизации отходов, поскольку ресурс может быть ограничен:

Целевая функция:

- $\sum\limits_{i=1}^{m} w_i p_i (E_n, X)_i \to \min$

Условия:

- $\forall i, j:\ x_{ij} \geq 0$ - неотрицательность числа единиц времени;
- $\forall i:\ ((E_n, X), L)_i \leq b_i$ - неперевыполнение плана;
- $\forall i:\ \sum\limits_{j \in N_i} x_{ij} \leq t_i$ - ограниченность фонда свободного времени всех АГР.
- $\forall i, \forall j \notin N_i:\ x_{ij} = 0$ - неиспользование неподдерживаемых технологий производства.

4. Алгоритм и ПО

Поставленная задача интерпретируется как задача линейного программирования с системой ограничений в виде неравенств.

Поэтому, для ее решения будем симплекс-метод.

В качестве ПО будем использовать ЯП Python с подключенными модулями:

- numpy - для работы с линейной алгеброй;
- cvxpy - для работы с линейным программированием;
- pandas - для работы с табличными данными.

В качестве среды разработки Jupyter Lab.

5. Решение

5.1 Код программы

In [ ]:
```
def minimize_waste(l,p,b,w,N,t=None, accuracy=3, task='отходы', integer=True):
    '''
    Функция, решающая задачу о раскрое
    в трех вариантах:
    1. минимизация отходов
    2. минимизация затраченного времени АГР
    3. минимизация использования ресурсов.

    Вход программы:

    l - 2D-список
    долей выхода продуктов

    p - список
```

```
мощности производства

b - список
плановых значений

w - список
производимых технологией ресурсов

N - 2D-список
поддерживаемых технологий

t - список (опциональный параметр)
ограничений по времени работы АГР,
если не указать, то ограничений по времени не будет

accuracy - целое число
знаков после запятой
на выводе программы

task - строка
с оптимизируемой функцией.
Возможные значения:
'отходы',
'время',
'ресурсы'.
Если не указать, принимает
значение 'отходы'

integer - булево значение,
целое ли число часов работы АГР

Выход программы:

Словарь формата
{
'Отходы'   : pandas.DataFrame,
'Время'    : pandas.DataFrame,
'Продукты' : pandas.DataFrame
}
'''

import numpy as np
import cvxpy
import pandas as pd

n = len(N)
m = len(l)

l = np.array(l)
p = np.array(p)
b = np.array(b)
w = np.array(w)

task = task.lower()

if not t is None:
    t = np.array(t)

x = cvxpy.Variable(shape=(n,m), integer = integer)

constraints = [(x >= 0),
               (cvxpy.multiply((([1]*n) @ x), p) @ l >= b)]
for i in range(n):
    zero_constraint = (cvxpy.sum([x[i,j-1] for j in range(m) if j not in N[i]]) == 0)
    constraints.append(zero_constraint)
    if not t is None:
        time_constraint = (cvxpy.sum([x[i,j-1] for j in N[i]]) <= t[i])
        constraints.append(time_constraint)

if task == 'отходы':
    total_value = cvxpy.sum(cvxpy.multiply(cvxpy.multiply(w, p),([1]*n) @ x))
elif task == 'время':
    total_value = cvxpy.sum(x)
elif task == 'ресурсы':
    total_value = cvxpy.sum(p @ (([1]*n) @ x))
else:
    print('Введенное задание некорректно. Возможные значения: "Отходы", "Время", "Ресурсы". Работа программы завершена.
    return None

problem = cvxpy.Problem(cvxpy.Minimize(total_value), constraints=constraints)
solution = problem.solve()

if x.value is None:
    print('Поставленная задача не имеет решения. Работа программы прекращена.')
    return None

def get_product_df():
    '''
    Функция, возвращающая
    таблицу произведенных продуктов
    '''

    products = np.around((([1]*n) @ x.value) * p @ l, accuracy)
    product_df = pd.DataFrame()
    product_df['Выход, т.'] = products
    product_df.index.names = ['Ресурс']
    product_df.index += 1
    product_df.loc['Сумма'] = np.sum(products)
```

```
        return product_df

    def get_waste_df():
        '''
        Функция, возвращающая
        таблицу произведенных отходов
        '''

        waste = np.around((([1]*n) @ x.value) * p * w, accuracy)
        waste_df = pd.DataFrame()
        waste_df['Отходы, т.'] = waste
        waste_df.index.names = ['Технология']
        waste_df.index += 1
        waste_df.loc['Сумма'] = np.sum(waste)

        return waste_df


    def get_tec_df():
        '''
        Функция, возвращающая
        таблицу распределения времени работы АГР
        по технологиям
        '''

        tec_matrix = np.abs(np.around(np.array(x.value), accuracy))
        tec_sum = np.sum(tec_matrix, axis = 1)
        tec_df = pd.DataFrame()
        for i in range(n):
            tec_df[f'Время работы АГР № {i+1}, ч.'] = tec_matrix[i]
        tec_df.index.names = ['Технология']
        tec_df.index += 1
        tec_df.loc['Сумма'] = tec_sum

        return tec_df

    return {'Отходы': get_waste_df(),
            'Время': get_tec_df(),
            'Продукты': get_product_df()}
```

5.2 Особенности работы программы

1. Программа возвращает словарь в формате словаря:

   ```
   Отходы   :  таблица отходов
   Время    :  таблица распределения времени работы АГР
   Продукты :  таблица произведенных продуктов
   ```
   Чтобы получить нужную таблицу, необходимо в квадратных скобка указать соответствующий таблице ключ словаря.
   Ввод строго с заглавной буквы.

2. Чтобы решить нужную задачу, нужно указать опциональный параметр `task` . Возможные значения:

   ```
   task='Время'
   task='Отходы'
   task='Ресурсы'
   ```
   Допускается ввод параметра в любом регистре.

3. В случае, если нет ограничений по времени, их можно просто не указывать при запуске.

4. В случае, если есть возможность работать АГР не целое число часов, стоит указать значение параметра `integer=False` . По умолчание его значение `True` .

6. Анализ

Запишем данные в представлении ЯП:

In [ ]:
```
l = [[0,0.962,0,0,0,0],
     [0.981,0,0,0,0,0],
     [0,0,0,0.963,0,0],
     [0,0,0.667,0.316,0,0],
     [0,0.12,0,0,0,0.845],
     [0.497,0.481,0,0,0,0],
     [0.736,0.237,0,0,0,0],
     [0,0,0.54,0,0,0.423],
     [0,0,0.135,0,0.845,0],
     [0,0.356,0,0,0.625,0]]

p = [7,6,8,8,6,10,11,8,7,9]

b = [100,100,100,100,50,150]

w = [0.038,0.019,0.037,0.017,0.035,0.022,0.027,0.037,0.02,0.019]

t = [4,6]

N = [[1,2,3,4,5,6],[6,7,8,9,10]]
```

Применим к ним нашу функцию:

In [3]:
```
answer = minimize_waste(l,p,b,w,N,t)
```

```
Поставленная задача не имеет решения. Работа программы прекращена.
```

Получаем вывод результата программой "Поставленная задача не имеет решения.". Это значит, что в поставленных условиях, задача линейного программирования не имеет вещественного решения.

Отсюда напрашивается корректировка задачи некоторыми уступками.

7. Корректировка задачи

7.1. Предположение

Предположим, что дополнительные закупки для производства продукции обходятся дороже, чем увеличение рабочего времени АГР. В таком случае уберем ограничение на время и будем минимизировать затраченные ресурсы на производство.

Кроме этого также решим задачу минимизации времени работы АГР, а также минимизации отходов производства для случая, если этот критерий производства очень важен.

7.2. Содержательная постановка откорректированной задачи

Необходимо определить, как нужно распределить время работы АГР по различным вариантам работы, чтобы выполнить план. При этом используя минимальное значение:

1. ресурсов
2. времени
3. произведенных отходов

7.3. Формальная постановка откорректированной задачи

Не будем вводить в уже введенные в П.3.0 обозначения, запишем лишь постановку задачи:

Целевая функция:

- $\sum\limits_{i=1}^{m} p_i(E_n, X)_i \to \min$ - в случае, если нам важно минимизировать затраченные ресурсы;
- $\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} x_{ij} \to \min$ - в случае, если нам важно минимизировать суммарное время работы АГР;
- $\sum\limits_{i=1}^{m} w_i p_i(E_n, X)_i \to \min$ - в случае, если нам важно минимизировать отходы.

Условия:

- $\forall i, j: \ x_{ij} \geq 0$ - неотрицательность числа единиц времени;
- $\forall i: \ ((E_2, X), L)_i \leq b_i$ - неперевыполнение плана;
- $\forall i, \forall j \notin N_i: \ x_{ij} = 0$ - неиспользование неподдерживаемых технологий производства.

7.4 Алгоритм и ПО откорректированной задачи

Здесь ничего не изменится.

7.5 Решение откорректированной задачи

Поскольку вариант отсутствия ограничений на время предусмотрен в решении, а также там предусмотрен выбор целевой функции, в анализе просто запустим программу, не указывая параметр `t`.

7.6 Анализ решения откорректированной задачи

Запустим программу, не указывая параметр `t`.

7.6.1 Минимизация затраченных ресурсов

In [4]:
```
answer = minimize_waste(l,p,b,w,N,task='Ресурсы',integer=True)
```

Посмотрим на таблицу выхода продукта:

In [5]:
```
answer['Продукты']
```

Out[5]:

Выход, т.

Ресурс	Выход, т.
1	100.108
2	100.498
3	100.489
4	104.728
5	53.235
6	150.426
Сумма	609.484

В этом случае наблюдаем незначительное превышение плана. Всего примерно на 9.5 тонн, что составляет примерно 1.6% от плана.

Посмотрим на производимое по данной схеме число отходов:

In [6]:
```
answer['Отходы']
```

Out[6]:

Технология	Отходы, т.
1	1.064
2	0.228
3	2.664
4	1.904
5	5.670
6	1.760
7	1.782
8	1.184
9	1.260
10	0.000
Сумма	17.516

По данной схеме получаем число отходов, равное примерно 17.5 тонн. Это составляет примерно 3% от использованного ресурса.

Число использованного ресурса равно сумме произведенных продуктов и отходов. Это число равно 627 тоннам.

Посмотрим на распределение времени работы АГР:

In [7]:
```
answer['Время']
```

Out[7]:

Технология	Время работы АГР № 1, ч.	Время работы АГР № 2, ч.
1	4.0	0.0
2	2.0	0.0
3	9.0	0.0
4	14.0	0.0
5	27.0	0.0
6	8.0	0.0
7	0.0	6.0
8	0.0	4.0
9	0.0	9.0
10	0.0	0.0
Сумма	64.0	19.0

В сумме получаем 64 и 19 часов соответственно, что превышает данные в исходной задаче ограничения почти в 11 и 5 раз соответственно.

В соответствии с этим, перейдем к решению задачи минимизации времени.

7.6.2 Минимизация затраченного времени

Запустим программу:

In [8]:
```
answer = minimize_waste(l,p,b,w,N,task='Время',integer=True)
```

Сразу же посмотрим на распределение времени работы АГР:

In [9]:
```
answer['Время']
```

Out[9]:

Технология	Время работы АГР № 1, ч.	Время работы АГР № 2, ч.

Технология	Время работы АГР № 1, ч.	Время работы АГР № 2, ч.
1	1.0	0.0
2	0.0	0.0
3	13.0	0.0
4	0.0	0.0
5	15.0	0.0
6	16.0	0.0
7	0.0	3.0
8	0.0	22.0
9	0.0	9.0
10	0.0	0.0
Сумма	45.0	34.0

Получаем в сумме 45 и 34 часов работы соответственно для каждого АГР. Это распределение уравнивает превышение данных в исходной задаче ограничений. Получаем превышение в 7 и 6 раз соответственно. При этом суммарное время изменилось не сильно. Было 83 часа на оба АГР, стало 79 часов на оба АГР. Т.к. скорее всего АГР могут работать одновременно, данное решение нас устроит больше в этом критерии.

Посмотрим распределение произведенных продуктов и отходов:

In [10]:
```
answer['Продукты']
```

Out[10]:

Ресурс	Выход, т.
1	103.808
2	102.315
3	103.545
4	100.152
5	53.235
6	150.498
Сумма	613.553

Также видим незначительное перевыполнение плана. Всего лишь 2%.

In [11]:
```
answer['Отходы']
```

Out[11]:

Технология	Отходы, т.
1	0.266
2	0.000
3	3.848
4	0.000
5	3.150
6	3.520
7	0.891
8	6.512
9	1.260
10	0.000
Сумма	19.447

Видим, что число производимых отходов тоже увеличивается. По сравнению с предыдущим решением, на 2 тонны.

Итого число затрачиваемых ресурсов по сравнению с предыдущим решением увеличивается на 6 тонн.

7.6.3 Минимизация отходов

Приведем также анализ решения задачи минимизации отходов с учетом отсутствия ограничения по времени.

In [12]:
```
answer = minimize_waste(l,p,b,w,N,task='Отходы',integer=True)
```

Т.к. главным критерием этого решения является число отходов, первым делом посмотрим на него.

In [13]:
```
answer['Отходы']
```

Out[13]:

Технология	Отходы, т.
1	0.000
2	0.342

Технология	Отходы, т.
3	2.072
4	2.584
5	6.300
6	3.740
7	0.000
8	0.000
9	1.260
10	0.000
Сумма	16.298

Из таблицы видим, что снизить отходы по сравнению с прошлыми решениями получится на 1.2 и 3.2 тонны соответственно.

Проверим остальные данные.

In [14]:
```
answer['Продукты']
```

Out[14]:

Ресурс	Выход, т.
1	102.148
2	103.370
3	109.889
4	101.960
5	53.235
6	152.100
Сумма	622.702

Наблюдаем заметный прирост перевыполнения плана. Примерно на 13 тонн. Это составляет перевыполнение плана на 4%.

In [15]:
```
answer['Время']
```

Out[15]:

Технология	Время работы АГР № 1, ч.	Время работы АГР № 2, ч.
1	0.0	0.0
2	3.0	0.0
3	7.0	0.0
4	19.0	0.0
5	30.0	0.0
6	17.0	0.0
7	0.0	0.0
8	0.0	0.0
9	0.0	9.0
10	0.0	0.0
Сумма	76.0	9.0

Видим еще большее, чем в первом решении, расхождение по использованию времени АГР. 76 и 9 часов соответственно. В сумме не сильно увеличивается: всего на 2 часа по сравнению с первым решением и на 6 часов по сравнению со вторым решением. Превышения ограничений исходной задачи в почти 13 и 2 раз соответственно.

8. Выводы

Снятие ограничений на время работы АГР пусть и позволяет решить поставленные задачи, но затраченное время существенно превышает предоставленный в исходной задаче фонд времени. Если подразумевать выполнение плана хотя бы за неделю без выходных, задача имеет решение.

Исходя из полученных ответов в предыдущем пункте, не имеет смысла использовать схему квазилексикографической оптимизации, которая может быть предложена для рассмотрения.

Дальнейший анализ того, какое из полученных трех решений, следует использовать требует дополнительных данных метрик: стоимость утилизации ресурсов, себестоимость ресурсов, стоимость часа работы АГР.

На основе этих результатов можно предложить:

- частично снизить план производства продуктов
- закупить дополнительное или более производительное оборудование

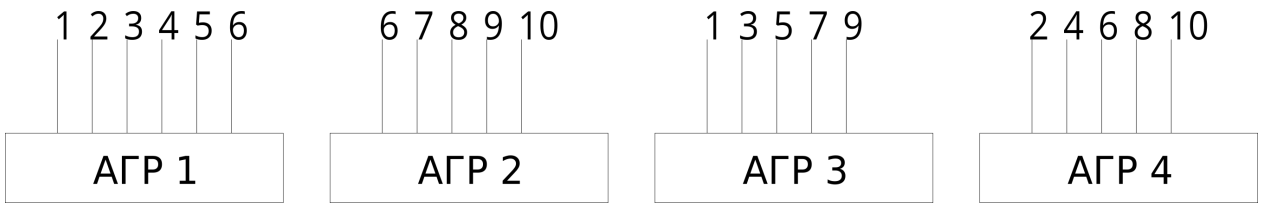
9. Приложение

Продемонстрируем работу программы на более сложных схемах. Увеличим число АГР до 4. Для того, чтобы было решение для задачи с ограничением по времени, просто увеличим фонд времени. Также используем функцию добавления интерактивности программе.

Фонд времени

Номер АГР	Время
1	20
2	22
3	21
4	23

Схема:



Запишем данные в память

In [16]:
```
l = [[0,0.962,0,0,0,0],
     [0.981,0,0,0,0,0],
     [0,0,0,0.963,0,0],
     [0,0,0.667,0.316,0,0],
     [0,0.12,0,0,0,0.845],
     [0.497,0.481,0,0,0,0],
     [0.736,0.237,0,0,0,0],
     [0,0,0.54,0,0,0.423],
     [0,0,0.135,0,0.845,0],
     [0,0.356,0,0,0.625,0]]

p = [7,6,8,8,6,10,11,8,7,9]

b = [100,100,100,100,50,150]

w = [0.038,0.019,0.037,0.017,0.035,0.022,0.027,0.037,0.02,0.019]

t = [20,22,21,23]

N = [[1,2,3,4,5,6],
     [6,7,8,9,10],
     [1,3,5,7,9],
     [2,4,6,8,10]]
```

Применим нашу функцию к этим данным

In [17]:
```
answer = minimize_waste(l,p,b,w,N,t=t,task='Отходы',integer=True)
```

Отобразим таблицы

In [18]:
```
display(answer['Время'])
display(answer['Отходы'])
display(answer['Продукты'])
```

Технология	Время работы АГР № 1, ч.	Время работы АГР № 2, ч.	Время работы АГР № 3, ч.	Время работы АГР № 4, ч.
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	3.0
3	0.0	0.0	7.0	0.0
4	0.0	0.0	0.0	19.0
5	20.0	0.0	10.0	0.0
6	0.0	17.0	0.0	0.0
7	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0
9	0.0	5.0	4.0	0.0
10	0.0	0.0	0.0	0.0
Сумма	20.0	22.0	21.0	22.0

Технология	Отходы, т.
1	0.000
2	0.342
3	2.072
4	2.584

Технология	Отходы, т.
5	6.300
6	3.740
7	0.000
8	0.000
9	1.260
10	0.000
Сумма	16.298

Ресурс	Выход, т.
1	102.148
2	103.370
3	109.889
4	101.960
5	53.235
6	152.100
Сумма	622.702

Видим работоспособность программы и адекватные результаты.