НУЛП, ІКНІ, САП		Тема	оцінка	підпис
КН-406	1(номер лаб)	Методи розв'язання задач багатокритеріальної оптимізації на основі зведення до розв'язання однокритеріальних		
Ваврик Р.Р.				
№ залікової:				
Методи багатокритеріальної оптимізації			Викладач:	
			к.т.н., доц. каф. Мельник М.Р.	

Мета: Ознайомитися з основними методами розв'язання ЗБО шляхом зведення до розв'язання послідовності однокритеріальних оптимізаційних задач.

Завдання:

1. Ознайомитися з основними методами розв'язання ЗБО шляхом зведення до розв'язання послідовності однокритеріальних оптимізаційних задач
2. Отримати завдання від викладача
3. Сформулювати ЗБО.
4. Розв'язати ЗБО з використанням зазначеного методу.
5. Провести аналіз отриманих результатів та оформити звіт.

Варіант 1

№	Цільова функція	Обмеження
1	$Q_1 = 2x_1 + 3x_2 + x_3 + 2x_4 \to \max$ $Q_2 = 3x_1 - x_2 \to \max$ $Q_3 = x_1 - 4x_2 \to \max$	$x_1 + 2x_2 + x_3 = 6$ $2x_1 + x_2 + x_4 = 8$ $-x_1 + x_2 + x_5 = 4$

Виконання роботи:

Метод головної компоненти

Суть методу полягає в тому, що узагальнений критерій береться рандомно із одного із критеріїв, що вибраний в ролі головного. Інші критерії стають

Цільові функції:

$2x_1 + 3x_2+x3+2x4 \rightarrow max$

$3x_1-x2 \rightarrow max$

$x1 -4x_2 \rightarrow max$

Нижче наведено приклад створення частини з постійних обмежень:

```
int j = 0;
column[j] = 1; row[j++] = 1;
column[j] = 2; row[j++] = 2;
column[j] = 3; row[j++] = 1;
column[j] = 4; row[j++] = 0;
column[j] = 5; row[j++] = 0;

if (lp.add_constraintex(j, row, column, lpsolve_constr_types.EQ, 6) == false)
{
    return 3;
}

j = 0;
column[j] = 1; row[j++] = 2;
column[j] = 2; row[j++] = 1;
column[j] = 3; row[j++] = 0;
column[j] = 4; row[j++] = 1;
column[j] = 5; row[j++] = 0;

if (lp.add_constraintex(j, row, column, lpsolve_constr_types.EQ, 8) == false)
{
    return 3;
}

j = 0;
column[j] = 1; row[j++] = -1;
column[j] = 2; row[j++] = 1;
column[j] = 3; row[j++] = 0;
column[j] = 4; row[j++] = 0;
column[j] = 5; row[j++] = 1;

if (lp.add_constraintex(j, row, column, lpsolve_constr_types.EQ, 4) == false)
{
    return 3;
}

column[j] = 1; row[j++] = 1;
column[j] = 2; row[j++] = -4;
column[j] = 3; row[j++] = 0;
column[j] = 4; row[j++] = 0;
column[j] = 5; row[j++] = 0;
```

Рис. 1 Формування цільової функції

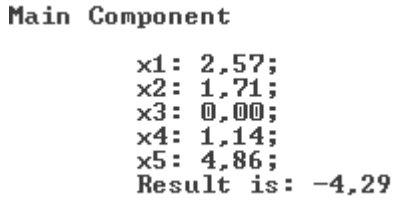

Рис. 2 Результат розв’язання задачі лінійного програмування для головної компоненти.

Лексикографічний метод

Ідея методу ґрунтується на тому, що спершу часткові критерії оптимізації ранжують по їх відносній важливості і поступово розв’язують задачі однокритеріальної оптимізації, починаючи з найважливішого критерія оптимізації.

На першому кроці необхідно розв’язати таку задачу однокритеріальної оптимізації:

$2x_1 + 3x_2+x3+2x4 \rightarrow max$

І основні обмеження.

Дана задача уже була задана і зображена на Рис.4

Наступний крок алгоритму - ведення першого критерію оптимізації в список обмежень. Цільова функція змінюється, до основного списку обмежень додається перша цільова функція з результатом її попереднього розв'язання

Останній етап розв’язання задачі з використанням лексикографічного методу - введення другої цільової функції в список обмежень і розв’язання оптимізаційної задачі лінійного програмування

```
Lexicography:

        Priority function #1:
        x1: 0,00;
        x2: 0,00;
        x3: 6,00;
        x4: 8,00;
        x5: 4,00;
        Result is: 22

        Priority function #2:
        x1: 0,0000;
        x2: 0,0000;
        x3: 6,0000;
        x4: 8,0000;
        x5: 4,0000;
        Result is: 0

        Priority function #3:
        x1: 0,00;
        x2: 0,00;
        x3: 6,00;
        x4: 8,00;
        x5: 4,00;
        Result is: 0
```

Рис. 3 Результат розв’язання задачі Лексикографічним методом.

Метод Поступок

Волдодіючи результатом першої функції потрібно здійснити операцію виконання поступки. Для розглядуваної задачі беремо поступку в 33%.

```
using (Lp lp = Lp.make_lp(0, Ncol))
            {
                FirstFunctionMP(lp, out Lp lpRes);
                FirstLp = lpRes.get_objective() - (lpRes.get_objective() * 0.33);
                Console.WriteLine($"\t: {(lpRes.get_objective() - (lpRes.get_objective() * 0.35))}");
                Console.WriteLine();
            }
```

Рис. 4 Задача ЛП для першої цільової функції згідно методу поступок

І це модифіковане значення першої функції використовуватиме друга функція для свого обмеження. Аналогічним чином проводиться розв’язання другої цільової функції із новими обмеженням і виконується операція поступки. Для другої цільової функції поступка становитиме 33%.

```
Method Postupok:

        Priority function #1:
        x1: 0,00;
        x2: 0,00;
        x3: 6,00;
        x4: 8,00;
        x5: 4,00;
        Result is: 14,30

        Priority function #2:
        x1: 0,0000;
        x2: 3,0000;
        x3: 0,0000;
        x4: 5,0000;
        x5: 1,0000;
        Result is: -1,95
        Priority function #3:
        x1: 0,00;
        x2: 1,95;
        x3: 2,10;
        x4: 6,05;
        x5: 2,05;
        Result is: -5,07
```

Рис. 5 Результат розв’язання задачі методом поступок

Метод умовного центру

Попередньо створю моделі для збереження усіх змінних та значень функцій для майбутнього обчислення

Для початку потрібно послідовно розв’язати задачі ЛП для кожної із цільових функцій із стандартними обмеженнями. Для методу умовного центру необхідно ввечти поняття умовної маси для кожної із цільових функцій.

```
for (int i = 0; i < FunctionModel.funcObjectives.Length; i++)
                FuncObjectiveSum += FunctionModel.funcObjectives[i];

            for (int i = 0; i < FunctionModel.funcObjectives.Length; i++)
                mFunc[i] = FuncObjectiveSum / FunctionModel.funcObjectives[i];
        System.Console.WriteLine("\n\tWeight:");

            for (int i = 0; i < mFunc.Length; i++)
            {
                System.Console.Write($" \tm{i} = {mFunc[i]}; ");
            }

            double[] variables = new double[5];

            System.Console.WriteLine($"\n\tVariables: ");

            for (int i = 0; i < FunctionModel.rowForFirstFunc.Length; i++)

            {
                variables[i] = FunctionModel.rowForFirstFunc[i] * mFunc[0]
                             + FunctionModel.rowForSecondFunc[i] * mFunc[1]
                             + FunctionModel.rowForThirdFunc[i] * mFunc[2];
                System.Console.Write($" \tX{i} = {variables[i]:0.00}; ");
            }
```

Рис. 6 Визначення умовної маси для кожної ЦФ і параметрів для умовного центру та значення цільових функцій згідно параметрів

```
x1: 0,00; x2: 2,00; x3: 0,00; x4: 2,00; x5: 0,00;
        Result is: 8,00
        Result is: 100000000000000000000000000000000,00
        Result is: -100000000000000000000000000000000,00
```

Рис. 7 Результат розв'язання методом умовного центр.

Метод ідеальної точки

При даному методі необхідно незалежно обчислити значення ЦФ при стандартних обмеженнях. Після цього, згідно із середнім арифметичним, знайти ідеальну точку для знаходження екстремуму даних фунцій.

```
for (int i = 0; i < FunctionModel.rowForFirstFunc.Length; i++)
            {
                variables[i] = FunctionModel.rowForFirstFunc[i]
                    + FunctionModel.rowForSecondFunc[i]
                    + FunctionModel.rowForThirdFunc[i]
                    / 3;
                Console.Write($"\t X{i} = {variables[i]:0.00}; ");
```

```
Function values:
 f1 = 10,00; f2 = -2,00; f3 = -8,00;
```

Рис. 8 Результат розв’язання ЦФ згідно методу ідеальної точки.

Висновки: Перевагами методу головної компоненти простот та наглядність, головний недолік – фактор суб’ктивності при виборі головного критерія . Також метод стає важко реалізовуваним при нагромадженні критреріїв. Чим більше варіантів – тим більше впливає на розв’язок суб’єктивність – тим важче раціонально обрати критерій. Лексикографічний метод реалізується таким чином. Проводиться мінімізація критерію із найбільшим пріоритет. Недоліком цього метод, як і попереднього, є суб’єктивність. В даному випадку при визначенні важливості критеріїв. Метод послідовних поступок доцільно застосовувати для розв’язання тих багатокритеріальних задач, в яких усі часткові критерії природнім чином впорядковані за ступенем важливості, причому кожний критерій настільки істотно більш важливий, ніж наступний, що можна обмежитися врахуванням тільки попарного зв'язку критеріїв і вибирати допустиме зниження чергового критерію з врахуванням поведінки тільки одного наступного критерію. На відміну від двох попередніх методів, в даному не потрібно вручну сортувати або обирати дані для подальших кроків. Фактор суб’єктивності відсутній.

Жоден із методів не є оптимальним та ідеальним у 100 випадках зі 100. Кожен із них годиться для розв'язку задачі багатокритеріальної оптимізації лише за певних умов. Нашим завданням є визначення оптимального розв’язку, базуючись на вибірці вхідних даних.