**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.БАУМАНА**

**Факультет**: Информатика и системы управления  
**Кафедра**: ИУ8 - Информационная безопасность

**МЕТОДЫ ОПТИМИЗАЦИИ**

**Лабораторная работа №4**  
«Решение задачи многокритериальной оптимизации»

**Вариант** 3

**Преподаватель:**   
Коннова Н.С.

**Студент:**   
Волков Г.А.

**Группа:**   
ИУ8-35

Москва, 2019

**СОДЕРЖАНИЕ**

[**ОСНОВНАЯ ЧАСТЬ** 3](#_Toc21702608)

[Цель работы 3](#_Toc21702609)

[Постановка задачи 3](#_Toc21702610)

[Ход работы 4](#_Toc21702611)

[**ЗАКЛЮЧЕНИЕ** 9](#_Toc21702612)

[**ПРИЛОЖЕНИЕ** 10](#_Toc21702613)

# **ОСНОВНАЯ ЧАСТЬ**

## Цель работы

Изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения задач МКО с помощью различных методов, осуществить сравнительный анализ результатов, полученных при помощи разных методов.

## Постановка задачи

Выбрать лучшую из альтернатив решения предложенной задачи по варианту с точки зрения указанных критериев следующими методами:

1) заменой критериев ограничениями;

2) формированием и сужением множества Парето;

3) методом взвешивания и объединения критериев;

4) методом анализа иерархий.

По условию:

«Выбор спутницы жизни»

Альтернативы: А. Татьяна; В. Лариса; С. Наталья; D. Ольга.

Критерии: 1. Внешность; 2. Финансовые запросы; 3. Домовитость; 4. Характер.

Описание предпочтений: Внешность: Лариса – красавица, Татьяна – довольно миловидна, Ольга – симпатична и стройна, Наталья – менее привлекательна. Финансовые запросы: самые большие – у Ольги, чуть менее – у Татьяны, выше среднего – у Натальи, ниже среднего – у Ларисы. Домовитость: самая хозяйственная – Наталья, чуть менее – Ольга, существенно менее – Татьяна, еще менее – Лариса. Характер: Лариса – мягкая и покладистая, Наталья – строгая, но справедливая, Татьяна – «душа компании», но «себе на уме», Ольга – с диктаторскими наклонностями.

## Ход работы

Составляем вектор весов критериев (с нашей точки зрения), используя шкалу 1÷10.

|  |  |  |  |
| --- | --- | --- | --- |
| Внешность | Фин.зап | Домовитость | Характер |
| 2 | 6 | 5 | 8 |

Нормализовав, получим вектор (0.095, 0.285, 0.238, 0.38)

1. Метод замены критериев ограничениями

Составим матрицу оценок для альтернатив

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Внешность | Фин.зап | Домовитость | Характер |
| Татьяна | 7 | 8 | 5 | 5 |
| Лариса | 6 | 4 | 2 | 8 |
| Наталья | 3 | 6 | 9 | 7 |
| Ольга | 6 | 9 | 8 | 2 |

Выберем в качестве главного критерия характер

Установим минимально допустимые уровни для остальных критериев:

Внешность >= 0.5 \* Amax1

Фин.запросы >= 0.2 \* Amax2

Домовитость >= 0.4 \* Amax3

Проведём нормировку матрицы (кроме столбца главного критерия).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Внешность | Фин.зап | Домовитость | Характер |
| Татьяна | 0.67 | 0.8 | 0.43 | 5 |
| Лариса | 1 | 0 | 0 | 8 |
| Наталья | 0 | 0.4 | 1 | 7 |
| Ольга | 0.5 | 1 | 0.86 | 2 |

При заданных условиях, лучшим выбором будет Татьяна.

1. Формирование и сужение множества Парето

Выберем в качестве главных критериев для данного метода внешность и домовитость. Внешность – по оси х, домовитость – по оси у. Сформируем множество Парето графическим методом. Об критерия максимизируются, поэтому точка утопии находится в правом верхнем углу графика.

Точка утопии имеет координаты (9; 9).

Найдём расстояние для каждой из альтернатив:

Татьяна 4.47

Лариса 7.00

Наталья 6.00

Ольга 3.16

Наименьшее расстояние у Ольги (6; 8), значит, это оптимальный выбор.

1. Взвешивание и объединение критериев.

Составим матрицу рейтингов альтернатив по критериям, используя шкалу 1-10 и нормализуем её

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Внешность | Фин.зап | Домовитость | Характер |
| Татьяна | 0.28 | 0.30 | 0.21 | 0.23 |
| Лариса | 0.36 | 0.15 | 0.08 | 0.36 |
| Наталья | 0.12 | 0.22 | 0.38 | 0.32 |
| Ольга | 0.24 | 0.33 | 0.33 | 0.09 |

Составим экспертную оценку критериев (по методу попарного сравнения) и получим вектор весов критериев: (1.50, 1.00, 2.00, 0.50)

Нормализуем его (0.30, 0.20, 0.40, 0.10)

Умножим нормализованную матрицу на нормализованный вектор весов критериев и получим значения объединенного критерия для всех альтернатив: (0.25, 0.21, 0.26, 0.28)

Как видно из полученной интегральной оценки, наиболее приемлемой альтернативой является Ольга.

1. Метод анализа иерархий

Для каждого из критериев составим и нормализуем матрицу попарного сравнения альтернатив:

Внешность

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Татьяна | Лариса | Наталья | Ольга | Сумма  строки | Нормир.  Стр. |
| Татьяна | 1 | 8 | 6 | 3 | 18 | 0.42 |
| Лариса | 0.12 | 1 | 9 | 8 | 18.12 | 0.42 |
| Наталья | 0.17 | 0.11 | 1 | 4 | 5.28 | 0.12 |
| Ольга | 0.33 | 0.12 | 0.25 | 1 | 1.71 | 0.04 |

Отношение согласованности = 0.04%

Фин. Зап.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Татьяна | Лариса | Наталья | Ольга | Сумма  строки | Нормир.  Стр. |
| Татьяна | 1 | 8 | 7 | 4 | 20 | 0.60 |
| Лариса | 0.12 | 1 | 3 | 2 | 6.12 | 0.18 |
| Наталья | 0.14 | 0.33 | 1 | 4 | 5.48 | 0.16 |
| Ольга | 0.25 | 0.5 | 0.25 | 1 | 2.00 | 0.06 |

Отношение согласованности = 2.07%

Домовитость

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Татьяна | Лариса | Наталья | Ольга | Сумма  строки | Нормир.  Стр. |
| Татьяна | 1 | 8 | 3 | 2 | 14 | 0.43 |
| Лариса | 0.12 | 1 | 2 | 3 | 6.12 | 0.19 |
| Наталья | 0.33 | 0.5 | 1 | 9 | 10.83 | 0.33 |
| Ольга | 0.5 | 0.33 | 0.11 | 1 | 1.94 | 0.06 |

Отношение согласованности = 1.42%

Характер

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Татьяна | Лариса | Наталья | Ольга | Сумма  строки | Нормир.  Стр. |
| Татьяна | 1 | 5 | 6 | 8 | 20 | 0.43 |
| Лариса | 0.20 | 1 | 7 | 9 | 17.20 | 0.37 |
| Наталья | 0.17 | 0.14 | 1 | 7 | 8.31 | 0.18 |
| Ольга | 0.12 | 0.11 | 0.14 | 1 | 1.38 | 0.03 |

Отношение согласованности = 5.13%

Оценка приоритетов критериев

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Внешность | Фин.зап. | Домовитость | Характер | Сумма  строки | Нормир.  Стр. |
| Внешность | 1 | 8 | 0.5 | 0.12 | 9.62 | 0.26 |
| Фин.зап. | 0.12 | 1 | 3 | 0.11 | 4.24 | 0.11 |
| Домовитость | 2 | 0.33 | 1 | 1 | 4.33 | 0.12 |
| Характер | 8 | 9 | 1 | 1 | 19.01 | 0.51 |

Составим матрицу (i – альтернатива, j - критерий) и умножим на столбец оценки приоритетов, получим (0.44, 0.34, 0.18, 0.04)

Оценив полученный вектор, можем сделать вывод, что оптимальным вариантом является Татьяна.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения лабораторной работы мы изучили основные приёмы различных методов многокритериальной оптимизации. Результаты могут отличаться в зависимости от применяемых методов решения, главным образом, в зависимости от оценки критериев. В таких методах, как анализ иерархий и линейная свертка критериев, также сравниваются между собой и сами критерии. Для приведенных данных можно заметить, что отношение согласованности и вектор весов критериев α показывают, что сравнение альтернатив, которые сильно отличаются между собой по нескольким критериям, может оказываться неточным.

Код, реализующий данные методы, написан на языке Python версии 3.\* и представлен в приложении.

# **ПРИЛОЖЕНИЕ**

**import** copy  
  
**def** normalizeVec(vec):  
 n = len(vec)  
 pam = 0  
 **for** i **in** range(n):  
 pam += vec[i]  
 pam = 1 / pam  
 **for** i **in** range(n):  
 vec[i] = vec[i] \* pam  
 print(**"Normalize Vec:"**)  
 **for** i **in** range(n):  
 print(**'{:>7.2f}'**.format(vec[i]), end=**""**)  
 print()  
 **return** vec  
  
  
**def** normalizeTab(tab, numberMainC):  
 n = len(tab)  
 m = len(tab[0])  
 pamMa = [-100] \* m  
 pamMi = [100000] \* m  
 **for** j **in** range(m):  
 **if** j == numberMainC - 1:  
 **continue  
 for** i **in** range(n):  
 pamMa[j] = max(pamMa[j], tab[i][j])  
 pamMi[j] = min(pamMi[j], tab[i][j])  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 **if** j == numberMainC - 1:  
 **continue** mi = pamMi[j]  
 ma = pamMa[j]  
 tab[i][j] = (tab[i][j] - mi) / (ma - mi)  
 **return** tab  
  
  
**def** printWithAll(tab, pam):  
 print()  
 pam2 = [**" "**, **"Внешность"**, **"Фин.зап."**, **"Домов."**, **"Характер"**]  
 n = len(tab)  
 m = len(tab[0])  
 **for** j **in** range(m + 1):  
 print(**'{:>9s}'**.format(pam2[j]), end=**""**)  
 print()  
 **for** i **in** range(n):  
 print(**'{:>9s}'**.format(pam[i]), end=**""**)  
 **for** j **in** range(m):  
 print(**'{:>9.2f}'**.format(tab[i][j]), end=**""**)  
 print()  
  
  
**def** findBest(tab, pam):  
 fl = **""** n = len(tab)  
 m = len(tab[0])  
 ma1=-1  
 ma2=-2  
 ma3=-3  
 **for** j **in** range(m):  
 ma1=max(ma1, tab[j][0])  
 **for** j **in** range(m):  
 ma2 = max(ma2, tab[j][1])  
 **for** j **in** range(m):  
 ma3=max(ma3, tab[j][2])  
 print(**"внешность 0.5 \* Amax1"**)  
 print(**"фин.запросы 0.2 \* Amax2"**)  
 print(**"Домовитость 0.4 \* Amax3"**)  
 ma = -1  
 **for** i **in** range(n):  
 **if** tab[i][0]>=0.5\*ma1 **and** tab[i][1]>=0.2\*ma2 **and** tab[i][2]>=0.4\*ma3 **and** tab[i][3]>ma:  
 fl = pam[i]  
 ma = tab[i][3]  
 print(**"The best is: "**, end=**""**)  
 print(**'{:>9s}'**.format(fl))  
  
  
**def** changeForLimit(tab, pam, numbCr):  
 print(**"Start changing Limits"**)  
 print(**"The main crit N:"**, numbCr)  
 printWithAll(tab, pam)  
 normalizeTab(tab, numbCr)  
 printWithAll(tab, pam)  
 findBest(tab, pam)  
  
  
**def** graphParetto(tab, names, n1, n2):  
 print()  
 print(**"Start Paretto"**)  
 n = len(tab)  
 pamMaX = -1  
 pamMaY = -1  
 **for** i **in** range(n):  
 pamMaX = max(pamMaX, tab[i][n1 - 1])  
 pamMaY = max(pamMaY, tab[i][n2 - 1])  
 print(**"Num for X="**, n1, **"Num for Y="**, n2)  
 print(**"Max X ="**, pamMaX, **"Max Y ="**, pamMaY)  
 pammi = 10000000000  
 pam = **""** print(**"R:"**)  
 **for** i **in** range(n):  
 print(**'{:>9s}'**.format(names[i]), end=**""**)  
 r = (tab[i][n1 - 1] - pamMaX) \*\* 2 + (tab[i][n2 - 1] - pamMaY) \*\* 2  
 print(**'{:>7.2f}'**.format(r \*\* (1 / 2)))  
 **if** r < pammi:  
 pammi = r  
 pam = i  
 print(**"The best is: "**, end=**""**)  
 print(names[i], **" ("**, tab[i][n1 - 1], **" , "**, tab[i][n2 - 1], **")"**, sep=**""**)  
  
  
**def** normalizeTabW(tab):  
 n = len(tab)  
 m = len(tab[0])  
 pa = [0] \* m  
 **for** j **in** range(m):  
 pam = 0  
 **for** i **in** range(n):  
 pam += tab[i][j]  
 pa[j] = 1 / pam  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 tab[i][j] \*= pa[j]  
 **return** tab  
  
  
**def** weightAndAssociation(tab, pam):  
 print()  
 print(**"Start WeightAndAssociation"**)  
 print(**"Default Vec:"**)  
 vec = [0.5 + 1,  
 1,  
 1 + 1,  
 0 + 0.5  
 ]  
 n = len(tab)  
 **for** i **in** range(n):  
 print(**'{:>9.2f}'**.format(vec[i]), end=**""**)  
 print()  
 vec = normalizeVec(vec)  
 vec2 = [[0] **for** i **in** range(n)]  
 **for** i **in** range(n):  
 vec2[i][0] = vec[i]  
 vec = vec2  
 m = len(tab[0])  
 aa = [0] \* n  
 tab = normalizeTabW(tab)  
 print(**"Normalize tab"**)  
 printWithAll(tab, pam)  
 **for** i **in** range(n):  
 **for** j **in** range(m):  
 aa[i] += tab[i][j] \* vec[j][0]  
 tab = aa  
 print(**"Union Crit:"**)  
 print(**"("**, end=**""**)  
 **for** i **in** range(n):  
 print(**'{:>9.2f}'**.format(tab[i]), end=**""**)  
 print(**")"**)  
 ma = -1  
 name = -1  
 **for** i **in** range(n):  
 **if** tab[i] > ma:  
 ma = tab[i]  
 name = i  
 print(**"The best is:"**, pam[name])  
  
  
**def** printCrit(names, crit):  
 n = len(names)  
 print(**'{:>9s}'**.format(**""**), end=**""**)  
 **for** i **in** range(n):  
 print(**'{:>9s}'**.format(names[i]), end=**""**)  
 print(**'{:>9s}'**.format(**"Сум. стр."**), end=**""**)  
 print(**'{:>9s}'**.format(**"Нормир."**), end=**""**)  
 print()  
 **for** i **in** range(n):  
 print(**'{:>9s}'**.format(names[i]), end=**""**)  
 **for** j **in** range(n + 2):  
 print(**'{:>9.2f}'**.format(crit[i][j]), end=**""**)  
 print()  
  
  
**def** fillTabAndGetNormir(names, crit1, wei):  
 n = len(names)  
 **for** i **in** range(n):  
 su = 0  
 **for** j **in** range(n):  
 **if** i == j:  
 crit1[i][j] = 1  
 **if** crit1[i][j] == 0 **and** crit1[j][i] != 0:  
 crit1[i][j] = 1 / crit1[j][i]  
 su += crit1[i][j]  
 crit1[i][-2] = su  
 su = 0  
 **for** i **in** range(n):  
 su += crit1[i][-2]  
 **if** su != 0:  
 su = 1 / su  
 **for** i **in** range(n):  
 crit1[i][-1] = crit1[i][-2] \* su  
 normir = [crit1[i][-1] **for** i **in** range(n)]  
 printCrit(names, crit1)  
 kk=0  
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 kk+= crit1[i][j]\*wei  
 **if** wei!=1:  
 **'''print("ИС =", end='')  
 print('{:>5.2f}'.format((kk-n)/(n-1)), end="")  
 print("%")'''** print(**"Отношение согл.="**, end=**''**)  
 print(**'{:>5.2f}'**.format((kk - n) / (n - 1)/0.9), end=**""**)  
 print(**"%"**)  
 **return** normir  
  
  
**def** metHierarchy(names, iVec):  
 normVec = normalizeVec(iVec)  
 print()  
 print(**"Start Hierarchy"**)  
 pam2 = [**"Внешность"**, **"Фин.зап."**, **"Домов."**, **"Характер"**]  
 n = len(names)  
 k = 0  
 print()  
 print(pam2[k])  
 crit1 = [[0] \* (n + 2) **for** i **in** range(n)]  
 crit1[0][1] = 8  
 crit1[0][2] = 6  
 crit1[0][3] = 3  
 crit1[1][2] = 9  
 crit1[1][3] = 8  
 crit1[2][3] = 4  
 norm1 = fillTabAndGetNormir(names, crit1,normVec[k])  
 k += 1  
 print()  
 print(pam2[k])  
 crit1 = [[0] \* (n + 2) **for** i **in** range(n)]  
 crit1[0][1] = 8  
 crit1[0][2] = 7  
 crit1[0][3] = 4  
 crit1[1][2] = 3  
 crit1[1][3] = 2  
 crit1[2][3] = 4  
 norm2 = fillTabAndGetNormir(names, crit1,normVec[k])  
 k += 1  
 print()  
 print(pam2[k])  
 crit1 = [[0] \* (n + 2) **for** i **in** range(n)]  
 crit1[0][1] = 8  
 crit1[0][2] = 3  
 crit1[0][3] = 2  
 crit1[1][2] = 2  
 crit1[1][3] = 3  
 crit1[2][3] = 9  
 norm3 = fillTabAndGetNormir(names, crit1,normVec[k])  
 k += 1  
 print()  
 print(pam2[k])  
 crit1 = [[0] \* (n + 2) **for** i **in** range(n)]  
 crit1[0][1] = 5  
 crit1[0][2] = 6  
 crit1[0][3] = 8  
 crit1[1][2] = 7  
 crit1[1][3] = 9  
 crit1[2][3] = 7  
 norm4 = fillTabAndGetNormir(names, crit1,normVec[k])  
 print()  
 crit1 = [[0] \* (n + 2) **for** i **in** range(n)]  
 crit1[0][1] = 8  
 crit1[0][2] = 0.5  
 crit1[0][3] = 0.125  
 crit1[1][2] = 3  
 crit1[1][3] = 0.111  
 crit1[2][3] = 1  
 norm5 = fillTabAndGetNormir(pam2, crit1, 1)  
  
 tab = [[0]\*n **for** i **in** range(n)]  
 **for** j **in** range(n):  
 tab[j][0] = norm1[j]  
 tab[j][1] = norm2[j]  
 tab[j][2] = norm3[j]  
 tab[j][3] = norm4[j]  
 vec = [0]\*n  
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 vec[i] += tab[i][j] \* norm5[j]  
 print(**"Union Crit:"**)  
 print(**"("**, end=**""**)  
 **for** i **in** range(n):  
 print(**'{:>9.2f}'**.format(vec[i]), end=**""**)  
 print(**")"**)  
 ma = -1  
 name = -1  
 **for** i **in** range(n):  
 **if** vec[i] > ma:  
 ma = vec[i]  
 name = i  
 print(**"The best is:"**, names[name])  
  
  
**def** main():  
 tab = [[7, 8, 5, 5],  
 [9, 4, 2, 8],  
 [3, 6, 9, 7],  
 [6, 9, 8, 2]]  
 names = [**"Татьяна"**, **"Лариса"**, **"Наталья"**, **"Ольга"**]  
 vec = [2, 6, 5, 8]  
 changeForLimit(copy.deepcopy(tab), names, 4)  
 graphParetto(copy.deepcopy(tab), names, 1, 3)  
 weightAndAssociation(copy.deepcopy(tab), names)  
 metHierarchy(names, vec)  
  
  
main()