Оглавление

[Введение 3](#_Toc126352266)

[Практическая и математическая постановка задачи 4](#_Toc126352267)

[Анализ существующих алгоритмов решения 5](#_Toc126352268)

[Прямые методы. 5](#_Toc126352269)

[Методы порогов несравнимости. 6](#_Toc126352270)

[Описание разрабатываемого алгоритма 10](#_Toc126352271)

[Укрупненная схема 10](#_Toc126352272)

[Развернутая схема 11](#_Toc126352273)

[Решение контрольного примера 13](#_Toc126352274)

[Листинг 15](#_Toc126352275)

[Результат машинного решения 20](#_Toc126352276)

[Заключение 21](#_Toc126352277)

[Список литературы 22](#_Toc126352278)

# Введение

При широком применении методов исследования операций аналитики стали сталкиваться с задачами, где имеется не один, а несколько критериев оценки качества решения. Опыт использования методов математического моделирования и компьютеров в различных областях человеческой деятельности привел к пониманию многих принципиальных трудностей, возникающих при их внедрении в реальную практику. Оказалось, что ЛПР, при принятии решения учитывает огромное число показателей, которые только в редких случаях удаётся представить в виде одного критерия. Стало очевидно, что методы исследования операций, которые успешно применялись при моделировании различных ситуаций, совершенно недостаточны для решения более сложных проблем, которые по сути своей являются многокритериальными. Многие факторы (социальные, организационные, политические, психологические и т. д.), имеющие существенное влияние на альтернативы не поддаются формализации. Такого рода задачи имеют следующую характерную особенность – модель, описывающая множество допустимых решений объективна, но качество решения оценивается по многим критериям.

# Практическая и математическая постановка задачи

Для выбора наилучшего варианта решения необходим компромисс между оценками по разным критериям. В условиях задачи отсутствует информация, позволяющая найти такой компромисс. Следовательно, он не может быть определен на основе объективных расчетов. Анализ многих реальных практических проблем, с которыми сталкивались специалисты, естественным образом привел к появлению класса многокритериальных задач. Задачи со многими критериями имеют следующие особенности:

- Задача имеет уникальный, новый характер – нет статистических данных, позволяющих обосновать соотношения между различными критериями.

- На момент принятия решения принципиально отсутствует информация, позволяющая объективно оценить возможные последствия выбора того или иного варианта решения. Это может быть сделано лишь людьми на основе их опыта и интуиции.

# Анализ существующих алгоритмов решения

В многокритериальных задачах появляются вопросы: как формализовать задачу? Как согласовать противоречивые стремления? Как принять решение?

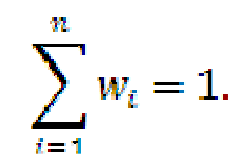
Основные проблемы методов оценки и сравнения многокритериальных альтернатив состоят в следующем: как получить оценки по отдельным критериям и как агрегировать эти оценки в общую оценку полезности альтернативы.

Многочисленные методы принятия решений при многих критериях различаются способом перехода к единой оценке полезности альтернатив. Среди этих методов можно выделить: прямые методы, методы порогов несравнимости, методы компенсации и др. Ниже представлены наиболее часто используемые методы многокритериальной оценки альтернатив.

## Прямые методы.

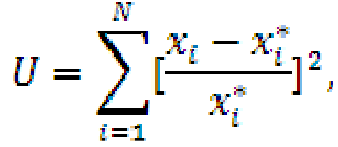
Примерный алгоритм многокритериальной оценки альтернатив, следующий:

* определить критерий оценки альтернатив;
* ранжировать критерий по важности;
* отбросить маловажные критерии;
* назначить числа, соответствующие относительной важности критериев;
* нормировать коэффициенты (wi) по важности из условия:

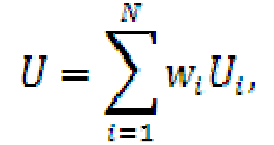


где wi – вес i-го критерия, назначаемый ЛПР;

* произвести предварительное отсечение альтернатив по качеству (на шкалах критериев определяется индекс качества);
* определить функции U полезности для каждого из критериев



* определить полезность каждой из альтернатив по формуле:



## Методы порогов несравнимости.

Существует подход к решению задачи многокритериального выбора на основе попарного сравнения альтернатив. Данный подход реализован в виде методов ЭЛЕКТРА (ELECTRE — Elimination Et Choix Traduisant la Realite — исключение и выбор, отражающие реальность) [68].

Постановка задачи обычно имеет следующий вид.

*Дано:* множество, состоящее из *m* критериев с количественными шкалами оценок; множество номеров критериев ; веса критериев ; множество альтернатив с оценками по критериям .

*Требуется:* выделить группу лучших альтернатив.

Структура метода ЭЛЕКТРА включает следующие этапы.

1. Проводится полное попарное сравнение всех альтернатив. Для каждой пары альтернатив по критериальным оценкам и вычисляются значения двух специальных индексов — согласия и несогласия. Эти индексы определяют согласие и несогласие с гипотезой, что альтернатива превосходит альтернативу .

2. Задаются уровни согласия и несогласия, с которыми сравниваются значения вычисленных индексов для каждой пары альтернатив. Если индекс согласия выше заданного уровня, а индекс несогласия — ниже, то одна из альтернатив превосходит другую. В противном случае альтернативы несравнимы.

3. Из множества альтернатив удаляются доминируемые. Оставшиеся альтернативы образуют ядро. Альтернативы, входящие в ядро, могут быть либо эквивалентными, либо несравнимыми.

4. Вводятся последовательно более «слабые» значения уровней согласия и несогласия (меньший по значению уровень согласия и больший уровень несогласия), при которых выделяются ядра с меньшим количеством альтернатив.

5. Процесс поиска лучших альтернатив прекращают, когда число альтернатив в ядре становится приемлемым для ЛПР или их число меньше заранее заданного. В последнее ядро входят наилучшие альтернативы. Последовательность ядер определяет упорядоченность альтернатив по качеству.

Рассмотрим эти этапы подробнее.

В разных методах семейства ЭЛЕКТРА индексы согласия и несогласия строятся по-разному. Рассмотрим подробнее метод ЭЛЕКТРА I [68]**.**

1. Проводится полное попарное сравнение всех альтернатив. Для каждой пары альтернатив по критериальным оценкам и вычисляются значения двух специальных индексов – согласия и несогласия.

Выдвигается гипотеза о превосходстве альтернативы ха над альтернативой . Множество номеров критериев разбивается на три подмножества:

1) – подмножество критериев, по которым предпочтительнее ;

2) – подмножество критериев, по которым эквивалентно ;

3) – подмножество критериев, по которым предпочтительнее .

Далее вводятся – индекс согласия с гипотезой о превосходстве над и – индекс, несогласия с гипотезой о превосходстве х„ над х(г

Индекс согласия подсчитывается на основе весов критериев как отношение суммы весов критериев подмножеств и к общей сумме весов:

Индекс несогласия определяется на основе учета относительных значений проигрышей альтернативы альтернативе Для каждого критерия из подмножества вычисляются разности значений критерия для альтернатив . Полученное значение делится на длину шкалы этого критерия, затем в качестве индекса несогласия принимается наибольшее относительное значение:

где - длина шкалы по *i*-му критерию.

Приведем очевидные свойства индексов согласия и несогласия:

;

, если подмножество пусто;

сохраняет значение при замене одного критерия на несколько с тем же общим весом;

;

сохраняет значение при введении более детальной шкалы по i-му критерию при той же ее длине.

Введенные индексы используются при построении матриц индексов согласия и несогласия для заданных альтернатив.

2. Задаются пороговые значения (отсюда следует название методов) – уровни согласия и несогласия (), с которыми сравниваются значения вычисленных индексов для каждой пары альтернатив. Если и , то альтернатива объявляется лучшей по сравнению с альтернативой т. е. альтернатива — доминируемая. В противном случае альтернативы несравнимы.

3. Из множества альтернатив удаляются доминируемые. Оставшиеся альтернативы образуют ядро, в которое входят доминирующие и несравнимые альтернативы.

4. Вводятся последовательно более «слабые» пороговые значения: уровни согласия и несогласия, удовлетворяющие условиям , при которых выделяются ядра с меньшим количеством альтернатив.

5. Процесс поиска лучших альтернатив прекращают, когда число альтернатив в ядре становится приемлемым дня ЛПР или их число меньше заранее заданного. В последнее ядро входят наилучшие альтернативы. Последовательность ядер определяет упорядоченность альтернатив по качеству.

# Описание разрабатываемого алгоритма

## Укрупненная схема

Укрупненная схема представлена на рисунке 1.



Рисунок 1 - Укрупненная схема алгоритма

## Развернутая схема

Развернутые схемы представлена на рисунках 2-3.



Рисунок 2 – Блок-схема функции расчета матрицы согласия



Рисунок 3 - Блок-схема функции удаления доминируемых альтернатив

# Решение контрольного примера

Рассмотрим пример решения с помощью описанного метода ЭЛЕКТРА задачи выбора комплекса защитных мероприятий. Пусть заданы альтернативы , которые оценены по следующим критериям: — обеспечение конфиденциальности информации; — обеспечение целостности информации; — обеспечение доступности информации.

Результаты оценивания альтернатив приведены в табл. 2.11.

Таблица 1 - Оценка альтернатив по критериям

|  |  |  |  |
| --- | --- | --- | --- |
| Альтернатива | Критерий | | |
|  |  |  |
|  | 180 | 70 | 10 |
|  | 170 | 40 | 15 |
|  | 160 | 55 | 20 |
|  | 150 | 50 | 25 |

Пусть веса критериев , , , а соответствующие длины шкал критериев .

Решим задачу в соответствии со схемой метода.

1. На основании заданных оценок альтернатив вычислим значения индексов согласия и несогласия (табл. 2.12 и 2.13).

Таблица 2 - Значения индексов согласия для примера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Альтернатива |  |  |  |  |
|  | – | 1/6 | 1/6 | 1/6 |
|  | 5/6 | – | 3/6 | 3/6 |
|  | 5/6 | 3/6 | – | 1/6 |
|  | 5/6 | 3/6 | 5/6 | – |

Таблица 3 - Значения индексов несогласия для примера

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Альтернатива |  |  |  |  |
|  | – | 0,6 | 0,3 | 0,4 |
|  | 0,11 | – | 0,1 | 0,2 |
|  | 0,22 | 0,3 | – | 0,1 |
|  | 0,33 | 0,22 | 0,11 | – |

2. Зададим уровни согласия и несогласия, с которыми сравниваются вычисленные индексы для каждой пары альтернатив: и .

3. Из множества альтернатив удалим доминируемые альтернативы и : альтернатива доминирует , так как и альтернатива доминирует , так как и . Оставшиеся альтернативы , образуют первое ядро и являются несравнимыми.

4. Вводим более «слабые» значения уровней согласия и несогласия: . Удалим доминируемую альтернативу : альтернатива доминирует , так как и .

5. В последнее ядро входит наилучшая альтернатива .

# Листинг

#include <iostream>

#include <math.h>

#include <vector>

#include <fstream>

#include <string>

#include <algorithm>

#include <sstream>

#include <iomanip>

#include <numeric>

using namespace std;

//Критериальные методы принятия решений: Методы порогов несравнимости

typedef vector<vector<float>> Matrix;

bool more(float a, float b) {

return a > b;

}

bool equals(float a, float b) {

return a == b;

}

bool Less(float a, float b) {

return a < b;

}

long numberOfDigits(double n) {

std::ostringstream strs;

strs << n;

return strs.str().size();

}

void printMatrix(const Matrix& M) {

long max\_len\_per\_column[M.size()];

long n = M.size(), m = M[0].size();

for (long j = 0; j < m; ++j) {

long max\_len {};

for (long i = 0; i < n; ++i){

long num\_length {numberOfDigits(M[i][j])};

if (num\_length > max\_len)

max\_len = num\_length;

}

max\_len\_per\_column[j] = max\_len;

}

for (long i = 0; i < n; ++i)

for (long j = 0; j < m; ++j)

std::cout << (j == 0 ? "\n| " : "") << std::setw(max\_len\_per\_column[j]) << M[i][j] << (j == m - 1 ? " |" : " ");

std::cout << '\n';

}

vector<float> getIMatrix(vector<float> a, vector<float> b, bool (\*func)(float, float)) {

vector<float> I;

for (int i=0; i<a.size(); i++){

if (func(a[i], b[i])) {

I.push\_back(i);

}

}

return I;

}

Matrix calculateAgreementMatrix(vector<float> A, Matrix X) {

int sizeX = X.size();

Matrix agrMatrix(sizeX, vector<float>(sizeX));

for (int i=0; i<sizeX; i++){

for (int j=0; j<sizeX; j++){

int jPlus = (j + i) % sizeX;

if (i == j) agrMatrix[i][j] = -1;

if (i == jPlus) continue;

vector<float> IMore = getIMatrix(X[i], X[jPlus], more);

vector<float> IEq = getIMatrix (X[i], X[jPlus], equals);

vector<float> ILess = getIMatrix(X[i], X[jPlus], Less);

float C = 0;

if (!ILess.empty()) {

for (int k=0; k<A.size(); k++) {

if (count(IMore.begin(), IMore.end(), k) > 0 || count(IEq.begin(), IEq.end(), k) > 0) {

C += A[k];

}

}

C /= accumulate(A.begin(), A.end(), 0);

} else {

C = 1;

}

agrMatrix[jPlus][i] = C;

}

}

return agrMatrix;

}

Matrix calculateDisagreementMatrix(vector<float> L, Matrix X) {

int sizeX = X.size();

Matrix disagrMatrix(sizeX, vector<float>(sizeX));

for (int i=0; i<sizeX; i++){

for (int j=0; j<sizeX; j++){

int jPlus = (j + i) % sizeX;

if (i == j) disagrMatrix[i][j] = -1;

if (i == jPlus) continue;

vector<float> ILess = getIMatrix(X[i], X[jPlus], Less);

vector<float> d;

for (int k=0; k<ILess.size(); k++){

d.push\_back(abs(X[jPlus][(int) ILess[k]] - X[i][(int) ILess[k]]) / L[(int) ILess[k]]);

}

disagrMatrix[jPlus][i] = \*max\_element(d.begin(), d.end());

}

}

return disagrMatrix;

}

float getLevelAgreement() {

float C;

cout << "Enter level agreement (0 <= C <= 1)" << endl;

cin >> C;

return C;

}

float getLevelDisagreement() {

float d;

cout << "Enter level disagreement (0 <= d <= 1)" << endl;

cin >> d;

return d;

}

void printCore(int number, vector<int> core) {

cout << number << " Core:" << endl;

for (int i=0; i < core.size(); i++){

cout << core[i] + 1 << " ";

}

cout << endl;

}

vector<int> hardDeleteAlternatives(Matrix& agrMatrix, Matrix& disagrMatrix) {

float C1 = getLevelAgreement(), d1 = getLevelDisagreement();

int sizeX = agrMatrix.size();

vector<int> deletedCriteria;

for (int i=0; i<sizeX; i++){

for (int j=0; j<sizeX; j++){

int jPlus = (j + i) % sizeX;

if (i == jPlus) continue;

if (agrMatrix[i][jPlus] >= C1 && disagrMatrix[i][jPlus] >= d1) {

if (count(deletedCriteria.begin(), deletedCriteria.end(), jPlus) == 0) {

deletedCriteria.push\_back(jPlus);

}

}

}

}

return deletedCriteria;

}

vector<int> softDeleteAlternatives(Matrix& agrMatrix, Matrix& disagrMatrix) {

float C1 = getLevelAgreement(), d1 = getLevelDisagreement();

int sizeX = agrMatrix.size();

vector<int> deletedCriteria;

for (int i=0; i<sizeX; i++){

for (int j=0; j<sizeX; j++){

int jPlus = (j + i) % sizeX;

if (i == jPlus) continue;

if (agrMatrix[i][jPlus] >= C1 && disagrMatrix[i][jPlus] <= d1) {

if (count(deletedCriteria.begin(), deletedCriteria.end(), jPlus) == 0) {

deletedCriteria.push\_back(jPlus);

}

}

}

}

return deletedCriteria;

}

vector<int> compareCores(vector<int> core1, vector<int> core2) {

vector<int> new\_core2;

for (int i=0; i<core2.size(); i++){

if (count(core1.begin(), core1.end(), core2[i]) == 0) {

new\_core2.push\_back(core2[i]);

}

}

return new\_core2;

}

vector<int> createResultCore(vector<int> core1, vector<int> core2, int numberAlternative) {

vector<int> new\_core2;

for (int i=0; i<numberAlternative; i++){

if (count(core1.begin(), core1.end(), i) == 0 &&

count(core2.begin(), core2.end(), i) == 0) {

new\_core2.push\_back(i);

}

}

return new\_core2;

}

int main() {

// vector<float> A{3, 2, 1};

// vector<float> L{100, 50, 45};

// Matrix X{{180, 70, 10},

// {170, 40, 15},

// {160, 55, 20},

// {150, 50, 25}};

int numberAlternatives, numberCriteria;

cout << "Enter number of criterias:" << endl;

cin >> numberCriteria;

cout << "Enter number of alternatives:" << endl;

cin >> numberAlternatives;

vector<float> A(numberCriteria), L(numberCriteria);

Matrix X(numberAlternatives, vector<float>(numberCriteria));

cout << "Enter criteria weights:" << endl;

for (int i = 0; i < numberCriteria; ++i) {

cin >> A[i];

}

cout << "Enter criteria criteria scale lengths:" << endl;

for (int i = 0; i < numberCriteria; ++i) {

cin >> L[i];

}

cout << "Enter alternatives:" << endl;

for (int i = 0; i < numberAlternatives; ++i) {

for (int j = 0; j < numberCriteria; ++j) {

cin >> X[i][j];

}

}

cout << "Entred matrix (X):" << endl;

printMatrix(X);

Matrix agrMatrix = calculateAgreementMatrix(A, X);

Matrix disagrMatrix = calculateDisagreementMatrix(L, X);

printMatrix(agrMatrix);

printMatrix(disagrMatrix);

vector<int> core1 = hardDeleteAlternatives(agrMatrix, disagrMatrix);

printCore(1, core1);

vector<int> core2 = softDeleteAlternatives(agrMatrix, disagrMatrix);

core2 = compareCores(core1, core2);

printCore(2, core2);

vector<int> resultCore = createResultCore(core1, core2, X.size());

cout << "Result ";

printCore(3, resultCore);

return 0;

}

# Результат машинного решения

Enter number of criterias:

3

Enter number of alternatives:

4

Enter criteria weights:

3 2 1

Enter criteria criteria scale lengths:

100 50 45

Enter alternatives:

180 70 10

170 40 15

160 55 20

150 50 25

Entred matrix (X):

| 180 70 10 |

| 170 40 15 |

| 160 55 20 |

| 150 50 25 |

| -1 0.166667 0.166667 0.166667 |

| 0.833333 -1 0.5 0.5 |

| 0.833333 0.5 -1 0.166667 |

| 0.833333 0.5 0.833333 -1 |

| -1 0.6 0.3 0.4 |

| 0.111111 -1 0.1 0.2 |

| 0.222222 0.3 -1 0.1 |

| 0.333333 0.222222 0.111111 -1 |

Enter level agreement (0 <= C <= 1)

0.8

Enter level disagreement (0 <= d <= 1)

0.1

1 Core:

1 3

Enter level agreement (0 <= C <= 1)

0.5

Enter level disagreement (0 <= d <= 1)

0.2

2 Core:

4

Result 3 Core:

2

# Заключение

Принятие важных решений является неотъемлемой частью не только крупных компаний, но и обычных людей.

В данной работе был выделен объект исследования, проанализирована предметная область, определены цели и задачи работы. Проведен анализ существующих методов многокритериального анализа и реализован наиболее подходящий для реализации цели выпускной квалификационной работы.

При анализе МКА было выявлено, что для решения задач работы потребуется написать свою программную реализацию алгоритма метода анализа иерархий, а не использовать существующие, так как существующие разработки были сделаны для других конкретных целей, поэтому и наша разработка будет носить такой же характер.

Была спроектирована программа и реализована на языке C++.

# Список литературы

1. Борисов А.Н. Обработка нечеткой информации в системах принятия решений / Алексеев А.В., Меркурьева Г.В., Сладзь Н.Н., Глушков В.И. – М.: Радио и связь, 2003.
2. Семенов С.С. Обзор методов принятия решений при разработке сложных технических систем // Функциональная надежность. Теория и практика. / Полтавский А.В., Маклаков В.В., Крянев А.В. – Надежность, 2014.
3. Чопоров О.Н. Стандартизация и методология управления информационными рисками: учеб. пособие [Электронный ресурс]. – Электрон. текстовые, граф. данные / О. Н. Чопоров, Ю.Н. Гузев. – Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2015