Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра мережних технологій

ПРЯМА ТА ОБЕРНЕНА ЗАДАЧА РАНЖУВАННЯ АЛЬТЕРНАТИВ ЗА СУКУПНІСТЮ ПОКАЗНИКІВ

Текстова частина до магістерської роботи (тези) за спеціальністю „Інженерія програмного забезпечення ”

Виконав: студент 2-го року навчання
Безштанько Володимир Віталійович

Керівник Франчук О. В., доцент, канд. фіз.-мат. наук

Магістерська робота захищена з оцінкою
«_____»

Секретар ДЕК _____
«____» _____2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

доц., канд. фіз.-мат. наук

_____ О. В. Франчук

(підпис)

9 листопада 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на дипломну роботу

студенту Безштаньку В.В. факультету інформатики 2 курсу МП

ТЕМА Пряма та обернена задача ранжування альтернатив за сукупністю показників

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Пряма задача ранжування альтернатив

2. Обернена задача ранжування альтернатив

3. Реалізація алгоритмів прямої та оберненої задач

Висновки

Список використаних джерел

Дата видачі 9 листопада 2020 р.

Керівник _____ (підпис)

Завдання отримав _____ (підпис)

Тема: Пряма та обернена задачі ранжування альтернатив за сукупністю показників

Календарний план виконання роботи:

№ п/п	Назва етапу роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на магістерську роботу.	9.11.2020	
2.	Огляд технічних статей за темою роботи.	08.02.2021	
3.	Збір інформації	10.03.2021	
4.	Розробка алгоритмів	30.03.2021	
5.	Тестування алгоритмів	05.04.2021	
6.	Попередній захист	14.05.2021	
7.	Аналіз отриманих даних	18.05.2021	
8.	Створення слайдів для доповіді та написання доповіді.	31.05.2021	
9.	Аналіз отриманих результатів з керівником, написання доповіді та попередній захист курсової роботи.	2.06.2021	
10.	Коригування роботи	05.06.2021	
11.	Остаточне оформлення пояснювальної роботи та слайдів.	13.06.2021	
12.	Захист роботи	15.06.2021	

Зміст

Анотація.....5

Вступ.....6

Розділ 1. Пряма задача ранжування альтернатив.....8

Відомі методи.....8

Постановка задачі.....10

Метод ELECTRE III.....11

Модифікований метод.....13

Висновки.....15

Розділ 2. Обернена задача ранжування альтернатив.....16

Причини виникнення оберненої задачі.....16

Постановка задачі.....16

Алгоритм розв’язку оберненої задачі ранжування альтернатив.....17

Висновки.....20

Розділ 3. Реалізація алгоритмів прямої та оберненої задач.....21

Опис класів.....21

Приклад роботи алгоритмів.....24

Дослідження складності розв'язку прямої задачі.....29

Порівняльний аналіз розв’язку оберненої задачі ранжування методом прямого перебору та застосуванням фільтрації.....32

Висновки.....40

Висновки.....41

Список використаних джерел.....42

Анотація

Робота присвячена прямій та оберненій задачі ранжування альтернатив. У межах роботи сформульовано та розв’язано пряму та обернену задачі.

Перший розділ присвячено прямій задачі ранжування альтернатив за сукупністю показників. Сформульовано пряму задачу, розглянуто відомі методи її розв’язання, обґрунтовано вибір модифікованого методу ELECTRE III для виконання задачі ранжування.

Другий розділ присвячено оберненій задачі ранжування альтернатив. У межах розділу сформульовано обернену задачу та описано алгоритм оптимізації розв’язку оберненої задачі.

Третій розділ присвячений дослідженню роботи алгоритмів. Наведено приклад роботи прямої та оберненої задач. Досліджено роботу алгоритмів за різних умов.

Вступ

Багатокритеріальне прийняття рішень (багатокритеріальний аналіз) є розділом теорії досліджень операцій, спрямованим на прийняття рішень за наявності конкуруючих критеріїв. Засоби дослідження операцій взагалі покликані забезпечити осіб, що приймають рішення, актуальною та математично обґрунтованою базою для прийняття рішень, і багатокритеріальний аналіз – один з невід'ємних розділів дисципліни дослідження операцій. Задача ранжування альтернатив належить до ключових задач, що розглядають в рамках багатокритеріального аналізу.

Задачі багатокритеріального аналізу почали розглядати ще на початку 50их років XX сторіччя, і продовжують досліджувати сьогодні. Сам термін «багатокритеріальне прийняття рішень» був популяризований у 70их роках XX сторіччя. За цей час сформувались основні підходи до вирішення проблем даної області, серед них: аналіз ієрархій, цільове програмування, теорії нечітких множин, французька школа підтримки прийняття рішень (методи ELECTRE), тощо.

Пряма задача ранжування полягає у визначенні такого впорядкування альтернатив, коли кожна наступна альтернатива не краща (не гірша) за сукупністю критеріїв. Пряма задача достатньо ґрунтовно представлена у наукових роботах та знаходить практичне застосування у різних сферах(стратегічне планування, екологія, дизайн продуктів, тощо [3,4,5]).

Обернена задача ж має за мету визначити, які найменші можливі зміни окремої альтернативи потрібно застосувати, щоб ця альтернатива зайняла ранг не нижче бажаного.

Ця робота присвячена розгляду прямої та оберненої задач ранжування альтернатив. Розгляд виконано, спираючись на модифікований алгоритм

популярної сім’ї методів ELECTRE. В межах роботи описано та реалізовано алгоритми вирішення прямої та оберненої задач ранжування альтернатив за сукупністю показників.

Об’єктом дослідження цієї роботи є багатокритеріальне прийняття рішень.

Предметом дослідження є розв’язання прямої та оберненої задачі ранжування альтернатив за сукупністю показників.

Метою цієї роботи буде розробка програмної системи для розв’язку прямої та оберненої задач ранжування альтернатив.

Задачі, що розв’язано у цій роботі: дослідження існуючих методів розв’язання прямої та оберненої задач ранжування альтернатив; розробка програмної системи для розв’язання прямої та оберненої задач ранжування альтернатив.

Розділ 1. Пряма задача ранжування альтернатив

Відомі методи

Пряма задача ранжування альтернатив полягає у визначенні впорядкування, за яким кожен наступний елемент є не кращим (або не гіршим) за попередній, спираючись на сукупне значення всіх критеріїв. Коротко розглянемо основні методи, що дозволяють здійснити ранжування альтернатив.

- Метод аналізу ієрархій (Analytic Hierarchy Process) уведений Сааті. Основною ідеєю методу є попарне порівняння альтернатив. Метод здобув широку популярність, тому був детально досліджений. Серед недоліків методу: потенційна можливість виникнення транзитивної неузгодженості між альтернативами та можливість допущення помилки при введенні даних особою, що приймає рішення, що суттєво вплине на результати порівняння.

Групи багатоатрибутних методів MAUT та MAVT будують функції відображення у простір чисел. Для відображення уводять функцію цінності або функцію корисності критеріїв. Значення критеріїв мають належати до спільної шкали, або приводитись до неї.

- Метод зважених сум (Weighted Sum Method) є найпростішим і найпоширенішим з багатоатрибутних методів. Кожному критерію ставиться у відповідність його ваговий коефіцієнт, а числова оцінка альтернативи є зваженою сумою його критеріїв.
- Методи ідеальної точки відштовхуються від дистанції до деякої уявної альтернативи, введеної особою, що приймає рішення, у якості взірця. Серед найвідоміших методів цієї групи – TOPSIS та VICOR. Відмінність між TOPSIS і VIKOR полягає у підходах до нормалізації значень критеріїв та обрахунках відстаней до ідеальних точок. [1]

Група методів Outranking сформувалась завдяки роботам французького вченого Бернарда Роя, автора сімейства методів ELECTRE. Серед найвизначніших методів цієї групи – ELECTRE та PROMETEE.

- ELECTRE I – найпростіша форма методу. Для визначення переваг уводиться індекс узгодженості (concordance) та неузгодженості (discordance) для кожної пари альтернатив. Індекс конкордації виражає ступінь переваги певної альтернативи над іншою на основі кількості домінуючих критеріїв.[1] Індекс невідповідності враховує кількість критеріїв, за яким альтернатива поступається іншій.[1] Використовуючи порогові значення, встановлені ОПР, для кожної пари альтернатив можна визначити відношення, в якому вона перебуває – перевага, байдужість або незрівнянність.[1] На основі встановлених відношень будується множина альтернатив, які не домінують одна над одною.[1] Серед них ОПР необхідно обрати найкращі.[1]
- ELECTRE IS – модифікація попереднього методу, яка уводить поріг байдужості. Якщо різниця значень пари альтернатив знаходяться у межах цього порогу, альтернативи можна вважати рівнозначними.
- ELECTRE Iv – модифікація, яка уводить поріг вето – коли значення однієї альтернативи строго переважає над іншою.
- ELECTRE II – перша модифікація, яка забезпечує повне ранжування. Ранг вираховують унаслідок виконання процедури дистиляції та двох рівнів ранжувань.
- ELECTRE III – модифікація ELECTRE II, яка використовує псевдокритерії, враховуючи невизначену та обмежену інформацію.
- ELECTRE IV – модифікація ELECTRE II, яка прибирає необхідність налаштування ваг критерії вручну особою, що приймає рішення, як це потребують попередні варіації.

- ELECTRE Tri – модифікація ELECTRE III для кластеризації альтернатив.

Таким чином, методи сімейства ELECTRE можна розділити на три групи: вибір (I , IS , Iv), ранжування (II , III , IV) і кластеризація (Tri).[1] З точки зору практичного застосування найбільш вживаними є група методів ранжування, серед яких можна виділити ELECTRE III як найбільш досконалий метод, що дозволяє враховувати невизначеність і надає ОПР контроль по встановленню ваг критеріїв. [1]

- Методи PROMETHEE порівнюють альтернативи попарно, використовуючи функції переваг.

У межах цієї роботи зосередимось на методі ELECTRE III як найбільш досконалому та популярному з Outranking методів.

Постановка задачі

Нехай кожна альтернатива x характеризується набором показників f(x) її критеріїв:

$$x \in A = \{A_1, A_2, A_3, .., A_n\}$$

$$f_1(x), f_2(x), f_3(x), .., f_m(x) - \text{критерії } x.$$

Пряма задача ранжування альтернатив за сукупністю показників полягає у визначенні впорядкування:

$$A_{i_1} \succcurlyeq A_{i_2} \succcurlyeq \cdots \succcurlyeq A_{i_n},$$
$$\text{де } i_1, i_2, \dots, i_n \in \{1, 2, \dots, n\}$$

На основі обчислення де-якої величини G(x), як узагальненого показника для кожного елемента множини A:

$$G(x) = G(f(x), W) = G\left(\left(f_1(x), f_2(x), \dots, f_m(x)\right), (w_1, w_2, \dots, w_m)\right),$$
$$x \in A = \{A_1, A_2, A_3, .., A_n\},$$

$$W = (w_1, w_2, \dots, w_m), \sum_{j=1}^{m} w_j = 1, \qquad w_j > 0.$$

Де G(x) обчислюють за певним правилом, окремим у кожному випадку, але таким, для якого справджується:

$$G(A_{i_1}) \geq G(A_{i_2}) \geq \cdots \geq G(A_{i_n}).$$

W є нормованим вектором вагових коефіцієнтів. [1]

Метод ELECTRE III

Метод ELECTRE III узагальнив здобутки всіх попередніх методів, тому має широкі можливості контролю виконання. Метод оперує наступними параметрами, доступними до налаштування особою, що приймає рішення: поріг переваги (preference threshold), поріг нейтральності(indifference threshold), поріг вето (veto threshold) та вектор вагових коефіцієнтів.

Найбільша універсальність методу з поміж групи методів ELECTRE, а також новаторський підхід автора, сприяла найбільшому поширенню цього методу серед усіх outranking методів, тому вивчення цього методу та розв’язання оберненої задачі саме для цього методу представляє найбільший інтерес.

Уведімо наступні позначення. Позначимо:

- preference threshold як p,
- indifference threshold як q,
- veto threshold як v,
- вектор вагових коефіцієнтів w.

Алгоритм покладає, що

$$0 \prec q \prec p \prec v$$

Та

$$w = (w_1, w_2, \dots, w_m), w_i \neq 0$$

Розглянемо алгоритм методу, що складається з наступних кроків:

1. Формування матриці узгодженості (concordance index) альтернатив $C(x_i, x_j)$:

$$C(x_i, x_j) = \frac{1}{w} \sum_{k=1}^{m} w_k * c_k(x_i, x_j),$$

$$w = \sum_{i=1}^{m} w_m,$$

$$c_k(x_i, x_j) = \begin{cases} 1, & f_k(x_i) + q_k(f_k(x_i)) \geq f_k(x_j) \\ 0, & f_k(x_i) + p_k(f_k(x_i)) \leq f_k(x_j) \\ \dfrac{p_k(f_k(x_i)) + f_k(x_i) - f_k(x_j)}{p_k(f_k(x_i)) - q_k(f_k(x_i))} & \text{інакше} \end{cases}$$

2. Визначення індексу неузгодженості (discordance index) $d_k(x_i, x_j)$ для кожного критерію $c_k(x_i, x_j)$:

$$d_k(x_i, x_j) = \begin{cases} 1, & f_k(x_i) + v_k(f_k(x_i)) \leq f_k(x_j) \\ 0, & f_k(x_i) + p_k(f_k(x_i)) \geq f_k(x_j) \\ \dfrac{f_k(x_j) - f_k(x_i) - p_k(f_k(x_i))}{v_k(f_k(x_i)) - p_k(f_k(x_i))} & \text{інакше} \end{cases}$$

3. Формування матриці достовірності (credibility matrix) S:

$$S = \begin{bmatrix} S(x_1, x_1) & \cdots & S(x_1, x_n) \\ \vdots & \ddots & \vdots \\ S(x_n, x_1) & \cdots & S(x_n, x_n) \end{bmatrix},$$

$$S(x_i, x_j) = \begin{cases} C(x_i, x_j), & d_k(x_i, x_j) \leq C(x_i, x_j) \\ C(x_i, x_j) * \prod_{k \in K} \dfrac{1 - d_k(x_i, x_j)}{1 - C(x_i, x_j)} \end{cases},$$

Де K – множина усіх критеріїв, для яких справджується $d_k(x_i, x_j) > C(x_i, x_j)$.

4. Формування нової матриці наближених значень:

Уведімо $m = \max\{ S(x_i, x_j) \}$. І уведімо коефіцієнт $s = 0.15$. Сформуємо матрицю наближених значень:

$$T(x_i, x_j) = \begin{cases} 1, & S(x_i, x_j) > m - m * s \\ 0, & \text{інакше} \end{cases}$$

5. Підрахунок параметру кваліфікації(qualification) $Q(x_i)$:

$$Q(x_i) = \sum_{i=1}^{n} T(x_i, x_j) - \sum_{j=1}^{n} T(x_i, x_j)$$

6. Виконання процедури дистиляції: вибір альтернатив (однієї або кількох) з найвищим рівнем кваліфікації.
7. Повторення попередніх кроків без раніше обраних альтернатив для формування повного ранжування.

Модифікований метод

Наведений у попередньому розділі базовий метод потребує поетапного виконання. Для спрощення процедури прямого ранжування, застосуємо модифікацію цього методу, запропоновану Лі та Вангом [2].

1. Повторимо етапи 1-3 оригінального методу.
2. Обрахємо рівень достовірності узгодженості (concordance credibility degree) з наступних міркувань:

$$CCD(x_i) = \sum_{x_j \in X} S(x_i, x_j)$$

3. Та рівень достовірності неузгодженості (discordance credibility degree):

$$DCD(x_i) = \sum_{x_j \in X} S(x_j, x_i)$$

4. Критерієм ранжування буде різниця цих рівнів (так званий net credibility degree):

$$G(x_i) = CCD(x_i) - DCD(x_i)$$

Ця модифікація методу дозволяє виконати ранжування за один прохід, на відміну від базової. Зосередимось на ній у межах цієї роботи.

Висновки

У розділі описано основні підходи до вирішення задачі ранжування альтернатив за сукупністю критеріїв. Сформульовано пряму задачу ранжування альтернатив. Описано причини вибору модифікованого методу ELECTRE III для розв’язання прямої задачі. Вказано алгоритм роботи базового та модифікованого методів.

Розділ 2. Обернена задача ранжування альтернатив

Причини виникнення оберненої задачі

Унаслідок розв'язання прямої задачі ранжування альтернатив отримують деяке впорядкування. Нехай альтернатива A_i зайняла місце r у цьому впорядкуванні. Будемо казати, що альтернатива A_i посіла ранг r.

Отримане впорядкування може не задовільняти потреб особи, що приймає рішення – бажана альтернатива може зайняти занадто низький ранг. Тому є сенс розглянути, які найменші зміни потрібно застосувати до критеріїв бажаної альтернативи, щоб отриманий у результаті повторної роботи алгоритму ранжування результат гарантував цій альтернативі ранг не нижче бажаного.

Постановка задачі

Отже, аналізується певна альтернатива a', яка фігурує у реальній задачі прийняття рішень, а бажаним результатом розв'язку нової задачі вважається отримання даною альтернативою рейтингу r, не нижче від наперед заданого значення p ($1 \leq p < n$).[1]

Покладемо, що значення показника $f_i(x)$ тим краще, чим воно більше.

Уведімо множину можливих покращень Q_k відповідного критерію k, задану особою, що приймає рішення:

$$Q_k = \{Q_{k,1}, Q_{k,2}, \dots, Q_{k,t_k}\},$$
$$Q_{k,1} < Q_{k,2} < \dots < Q_{k,t_k}$$

Уведімо вектор V – вектор «ціни» зусиль, які потрібно застосувати до зміни критеріїв на $Q_{k,i}$:

$$V = (v_1, v_2, \dots, v_m),$$

$$\sum_{j=1}^{m} v_j = 1$$

Сформулюємо обернену задачу ранжування альтернатив наступним чином:

$$h\left(\overrightarrow{Q_{i,*}}, V\right) = \sum_{j=1}^{m} v_j Q_{i,*} \to min$$

$$G\left(f\left(a', \overrightarrow{Q_{i,*}}\right), W\right) > G\left(f\left(A_{i_p}\right), W\right)$$

$$\overrightarrow{Q_{i,*}} = \left(Q_{1,*}, Q_{2,*}, \dots, Q_{m,*}\right) \in Q_1 \times Q_2 \times \dots \times Q_m$$

$$f\left(a', \overrightarrow{Q_{i,*}}\right) = \left(f_1(a') + Q_{1,*}, f_2(a') + Q_{2,*}, \dots, f_m(a') + Q_{m,*}\right)$$

Для розв'язання задачі застосовуватимемо метод запропонований [1] на базі ідеології послідовного аналізу та відсіву варіантів.

Алгоритм розв'язку оберненої задачі ранжування альтернатив

Найпростішим рішенням для розв'язання оберненої задачі ранжування альтернатив буде повний перебір варіантів. Складність виконання такої операції буде зростати пропорційно збільшенню варіантів комбінацій модифікацій, введених особою, що приймає рішення. Наведемо алгоритм, що дозволить зменшити складність поточного завдання.

Алгоритм складається з наступних кроків:

1. Перевіримо розв'язність завдання.

Нехай $Q_{max} = (\max(Q_1), \max(Q_2), \dots, \max(Q_m))$.

Якщо

$$G(f(a', Q_{max}), W) \le G\left(f\left(A_{i_p}\right), W\right)$$

То за заданих значень не існує розв'язку оберненої задачі ранжування альтернатив.

2. Спробуємо виконати відсів варіантів знизу.

Основна проблема, що постає перед нами – узагальнена величина G, яка визначає порядок ранжування, тісно взаємопов'язана з кожною альтернативою, яка присутня у вхідних даних. Таким чином, ми маємо проводити перевірку кожного конкретного варіанту.

Однак, можна з упевненістю сказати, що зростання кожного окремого критерію не призведе до спадання величини G загалом.

Для кожного критерію k розглядатимемо наступні величини:

$$Q_{max} = (\max(Q_1), \max(Q_2), \dots, \max(Q_m))$$

$$Q_{k-min} = (\max(Q_1), \max(Q_2), \dots, \min(Q_k), \dots, \max(Q_m))$$

З попередніх міркувань зрозуміло, що

$$G(f(a', Q_{max}), W) > G\left(f\left(A_{i_p}\right), W\right).$$

Якщо

$$G(f(a', Q_{k-min}), W) \leq G(f\left(A_{i_p}\right), W)$$

То з властивості неперервності величини G можна зробити висновок, що існує точка поділу множини можливих модифікацій Q_k на дві множини Q_{k-} та Q_{k+}, причому всі модифікації Q_{k-} будуть недостатніми для розв'язку оберненої задачі.

Відкинемо модифікації Q_{k-}.

3. Спробуємо виконати відсів варіантів згори.

Розглянемо цільову функцію h. Знайдімо вартість найменшого розв'язку, що точно існує:

$$h' = \min\{ h(Q_{k-min}, V)\}$$

Для кожного критерію k', відмінного від k розглянемо наступні комбінації:

$$Q_{k'} = (\min(Q_1), \min(Q_2), \dots, \max(Q_k), \dots, \min(Q_m))$$

Якщо

$$h(Q_{k'}, V) > h'$$

То можна відкинути найбільшу модифікацію за критерієм k', оскільки точно існує розв'язок «дешевший» даного.

Відкинувши найбільшу модифікацію, потрібно перевірити, чи нова модифікація не більш дорога, ніж існуюча найменша. Виконувати відкидання усіх варіантів, поки вартість модифікацій не буде менша чи рівна h'.

4. Повторювати етапи 2-3 поки можливо.

Етапи 2 та 3 взаємопов'язані, і умови, придатні для повторної фільтрації 2 виникають тільки у випадку виконання 3, а повторне виконання 3 принесе результат тільки після 2. Таким чином можна покроково виконувати етапи 2-3, поки вони змінюють множину можливих значень модифікації.

5. Виконати повний перебір варіантів, що залишились.

Висновки

У розділі сформульовано задачу та описано алгоритм розв’язання оберненої задачі ранжування альтернатив. Алгоритм вдалось удосконалити за рахунок реалізованої процедури відсіву варіантів.

Розділ 3. Реалізація алгоритмів прямої та оберненої задач

Опис класів

Код програми представлений п’ятьма класами:

1. Клас Program.

Містить точку запуску програми – метод Main.

2. Клас Alternative.

Клас Alternative створений для представлення альтернатив у програмі. Альтернативу характеризують її оцінки за критеріями – поле Rates. Оцінки можуть бути представлені дійсними числами, тому поле Rates типу double[].

Метод ToString() створений для зручності виводу даних про альтернативу на екран.

Метод Improve(double[] improvements) слугує для внесення модифікацій у альтернативу.

3. Клас MultiIndex.

Це клас багатовимірних лічильників.

Конструктор MultiIndex(int[] maxIndex) приймає масив верхньої межі можливих значень кожного виміру лічильника.

Метод Index() повертає поточне значення лічильника.

Метод Inc() збільшує значення лічильника. Спочатку збільшується значення лічильника у першому вимірі. Якщо значення досягло межі, заданої при конструюванні, збільшується лічильник у другому вимірі, тощо. В результаті роботи методу повертається значення логічного типу – true, якщо збільшення пройшло успішно, та false, якщо досягнуто максимальне значення за всіма

вимірами лічильника. Значення лічильника у всіх вимірах у такому випадку скидується до нуля.

Клас застосовується для роботи оберненого алгоритму ранжування альтернатив.

4. Клас ELECTRE.

Основний клас, що виконує розв'язання розв’язання прямої задачі ранжування альтернатив.

Конструктор ELECTRE(double[] P, double[] Q, double[] V, double[] W, Alternative[] alternatives) приймає набір альтернатив, значення порогів, та вагу критеріїв у оцінюванні.

Метод Execute() запускає виконання розв’язання прямої задачі ранжування альтернатив та повертає впорядкування альтернатив у вигляді масиву Alternative[], від найбільшого значення величини G до найменшої. Виконання ранжування відбувається за модифікованим алгоритмом ELECTRE III, описаним у цім документі вище.

Метод OddNetCred(Alternative odd) уведений для підтримки процесу розв’язання оберненої задачі ранжування. Специфікою методу ELECTRE III є те, що величина G альтернативи строго залежить від набору альтернатив, з якими її порівнюють. Цей метод вираховує величину G для сторонньої альтернативи, не присутньої у даних, переданих конструкторові прямої задачі, порівнявши цю сторонню альтернативу з альтернативами, присутніми у даних, переданих конструкторові прямої задачі. Метод повертає значення величини G типу double.

Клас містить також приватні методи, коротко розглянемо їх призначення:

- Concordance() повертає матрицю узгодженості виду double[][].

- Credibility(double[][] concordance) приймає матрицю узгодженості, обраховану раніше, повертає матрицю достовірності виду double[][].
- NetCred(double[][] credibility) приймає матрицю достовірності, обраховану раніше, повертає масив double[] значень величини G кожної альтернативи, наданої у конструкторі прямої задачі.
- Conc_k(Alternative A, Alternative B, int k) обраховує узгодженість альтернатив A та B за критерієм k, повертає значення типу double.
- Conc(Alternative A, Alternative B) обраховує узгодженість альтернатив A та B за всіма критеріями, спираючись на попередній метод, повертає значення типу double.
- Disk_k(Alternative A, Alternative B, int k) обраховує неузгодженість альтернатив A та B за критерієм k, повертає значення типу double.
- Cred_mult(Alternative A, Alternative B, double c) обраховує множник, що буде застосований до матриці узгодженості, для перетворення її до матриці достовірності. Параметр c – значення узгодженості альтернатив A та B, обраховане раніше. Застосовує попередній метод для підрахунку множника. Повертає значення типу double.
- Cred(Alternative A, Alternative B, double c) обраховує значення достовірності альтернатив A та B, спираючись на значення узгодженості, обраховане раніше. Повертає значення типу double.
- Cred(Alternative A, Alternative B) обраховує значення достовірності альтернатив A та B, спираючись на попередній метод. Уведена для обрахунку OddNetCred. Повертає значення типу double.

5. Клас Inverse.

Конструктор Inverse(ELECTRE primal, Alternative alt, int thresholdPosition, double[][] possibleImprovements, double[] priceVec, Alternative[] prime_res = null) приймає:

- Початкову задачу primal;
- Альтернативу alt, для якої буде виконано обернену задачу ранжування альтернатив за сукупністю показників;
- Порогову позицію thresholdPosition, ранг не нижче якої повинна зайнята альтернатива alt, після модифікації критеріїв;
- Двовимірний масив модифікацій, заданих особою, що приймає рішення, possibleImprovements, доступних для альтернативи alt;
- Масив коефіцієнтів вартості застосування модифікацій priceVec. Вихідна ціна модифікації i критерію k буде $priceVec[k] * possibleImprovements[k][i]$;
- Та може приймати результат виконаного ранжування primal.

Метод Exec(bool useFilters=true) виконує обрахунок оберненої задачі та повертає масив double[], що містить відповідні модифікації, застосування яких призводить до зайняття альтернативою alt бажаної позиції. Параметр useFilters контролює роботу методу. Якщо параметр має значення хиби, відбувається розв'язання задачі методом повного перебору. Якщо параметр має значення істини, то відбувається робота алгоритму з попереднім відсіканням критеріїв, після чого застосовується повний перебір варіантів, що залишаться. Робота алгоритму описана у розділі, присвяченому оберненій задачі ранжування альтернатив за сукупністю показників.

Методи toCrop(Alternative rawProbe, int k, double[] imps, Func<bool> test) та underThres (Alternative probe, double threshold) – допоміжні приватні методи для виконання процедури відсіву варіантів.

Приклад роботи алгоритмів

Наведемо приклад роботи алгоритмів.

Нехай маємо наступні вхідні дані:

	K1	K2	K3	K4	K5	K6
Q	12	12	10	7	0	6
P	16	12	11	16	11	7
V	18	19	14	17	17	14
A1	16	16	8	18	18	11
A2	11	5	1	6	13	13
A3	15	6	3	10	13	7
A4	8	17	14	11	12	10
A5	15	17	13	19	3	9
A6	19	4	11	13	4	16
A7	11	14	11	19	19	8
A8	10	10	3	14	3	10
A9	7	9	3	7	18	8
A10	7	4	3	18	3	0
A11	17	11	2	11	6	9
A12	1	1	2	7	19	18
A13	15	9	18	6	1	2
A14	7	0	14	3	12	13
A15	5	12	5	3	5	19
w	0.012	0.157	0.192	0.315	0.281	0.043

У процесі виконання прямого ранжування відбувається обрахунок величини G для кожної альтернативи:

	G	місце
A1	4,024	2
A2	-0,336	9
A3	0,284	7
A4	2,762	3

A5	0,509	6
A6	-1,699	10
A7	4,291	1
A8	-2,422	14
A9	1,186	5
A10	-2,421	13
A11	-2,151	11
A12	1,192	4
A13	-2,402	12
A14	0,089	8
A15	-2,906	15

Визначимо місце, зайняте кожною альтернативою за значенням її величини G, впорядкованими за спаданням. Упорядкуємо альтернативи за місцем, та отримаємо результат прямої задачі ранжування альтернатив:

місце		К1	К2	К3	К4	К5	К6
1	A7	11	14	11	19	19	8
2	A1	16	16	8	18	18	11
3	A4	8	17	14	11	12	10
4	A12	1	1	2	7	19	18
5	A9	7	9	3	7	18	8
6	A5	15	17	13	19	3	9
7	A3	15	6	3	10	13	7
8	A14	7	0	14	3	12	13
9	A2	11	5	1	6	13	13
10	A6	19	4	11	13	4	16

11	A11	17	11	2	11	6	9
12	A13	15	9	18	6	1	2
13	A10	7	4	3	18	3	0
14	A8	10	10	3	14	3	10
15	A15	5	12	5	3	5	19

Аналізуючи отриманий результат, можемо помітити наступні закономірності:

- Найкращі позиції зайняли альтернативи з гарним результатом у середньому, і видатним – по двом найвпливовішим критеріям (К4 та К5);
- Високий поріг нейтральності зробив критерії К3 та К2 менш впливовими на результат, ніж критерій К6, ваговий коефіцієнт якого на порядок менший. Це добре відображено на прикладі альтернатив А9 та А12, а також А5 та А12;
- У разі приблизної рівності альтернатив за основними факторами у дію вступають другорядні критерії(на прикладі альтернатив А3 та А5).

Розв’яжемо обернену задачу: які модифікації потрібно застосувати до альтернативи А15, щоб вона посіла у рейтингу місце, не нижче першого, і це відбулось з докладанням найменших зусиль.

Нехай особа, що приймає рішення, визначить модифікації, які можливо застосувати до критеріїв, та «ціну» цих модифікацій:

	К1	К2	К3	К4	К5	К6
ціна	0,417	0,021	0,417	0,0417	0,021	0,083
М1	0	0	0	0	0	0
М2	1	1	1	2	1	
М3	2	2	2	3	2	
М4	4	3	3	6	3	
М5	5	4	4	7	4	
М6	6		5	8	6	
М7	7		6	10	7	
М8	8		10	11	8	
М9	9		11	12	9	
М10	10		12	14	10	
М11	13		13	15	13	
М12			14			

Розмірність цієї задачі методом повного перебору

$$11 * 15 * 2 * 11 * 11 * 1 = 39930.$$

Після застосування відсіву варіантів (варіанти, які не пройшли відбір, помічено червоним) розмірність задачі зменшилась до 10x5x9x8x5x1

$$10 * 5 * 9 * 8 * 5 * 1 = 18000.$$

Таким чином вдалось досягти зменшення розмірності задачі у

$$\frac{39930}{18000} = 2{,}218(3) \text{ раз.}$$

Час роботи програми скоротився з 2028 мс до 487 мс.

Розв’язком оберненої задачі за даних умов є:

критерій	модифікація
К1	0
К2	3
К3	3
К4	15
К5	13
К6	0

Дослідження складності розв'язку прямої задачі

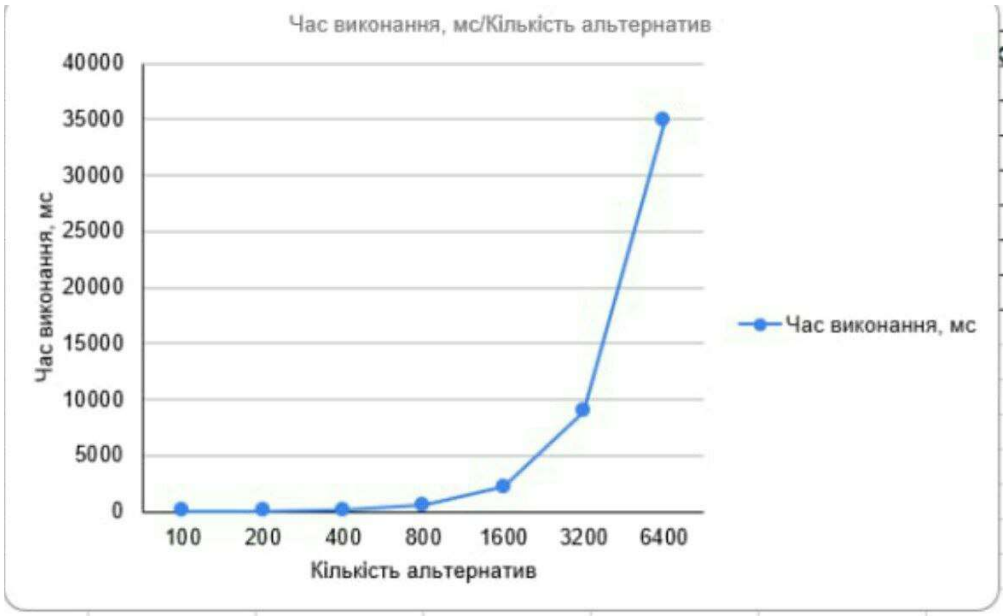
Розглянемо пряму задачу ранжування альтернатив. Дослідимо складність розв’язання прямої задачі.

Встановимо вхідні дані випадково, з наступними параметрами:

- Кількість критеріїв – 6;
- Оцінки від 0 до 29.

Проведемо дослідження часу роботи прямого ранжування альтернатив у залежності від кількості альтернатив.

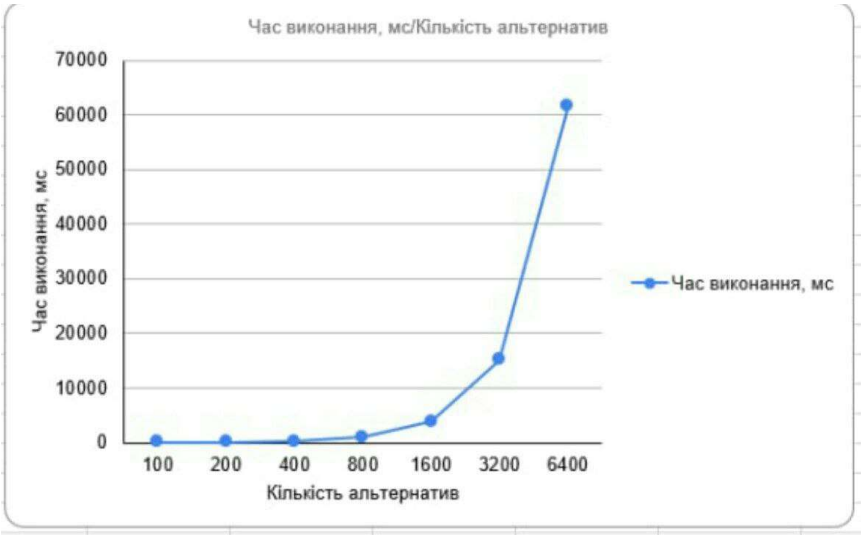
Кількість альтернатив	Час виконання, мс
100	8
200	33
400	131
800	560
1600	2158
3200	8904
6400	34847



Змінимо вхідні параметри. Встановимо:

- Кількість критеріїв – 12;
- Оцінки від 0 до 29.

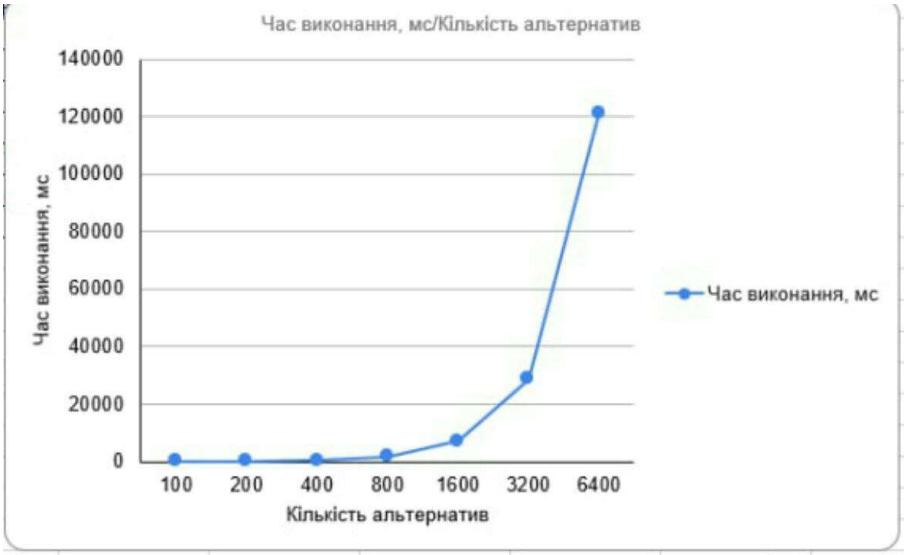
Кількість альтернатив	Час виконання, мс
100	14
200	58
400	234
800	974
1600	3818
3200	15108
6400	61619



Встановимо:

- Кількість критеріїв – 24;
- Оцінки від 0 до 29.

Кількість альтернатив	Час виконання, мс
100	28
200	118
400	477
800	1799
1600	7098
3200	28560
6400	121071



Таким чином, час розв’язання прямої задачі пропорційний квадрату кількості альтернатив.

Також, можна помітити, що час роботи алгоритму прямої задачі прямопропорційний до кількості критеріїв.

Порівняльний аналіз розв’язку оберненої задачі ранжування методом прямого перебору та застосуванням фільтрації

Розглянемо, яких результатів дозволить досягти застосування фільтрації, у порівнянні з елементарним прямим перебором.

Розглянемо обернену задачу ранжування з наступними вхідними даними, встановленими випадково:

- Кількість критеріїв – 6;
- Кількість альтернатив – 30;
- Оцінки від 0 до 19.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
1917	3743	1,95

1231	32131	26,1
77116	189031	2,45
10798	11901	1,10
936	23547	25,2
6689	14269	2,13
7195	33933	4,72
16451	22988	1,39
285	1823	6,39
22839	24633	1,08
3291	100597	30,5
3719	41776	11,2
9631	25212	2,62
32143	32198	1
840	4142	4,93
397	10021	25,24
18880	69121	3,66
109005	108901	1
889	4357	4,9
40138	55509	1,38

Середнє прискорення – у 7,947 раз.

Змінимо вхідні дані. Встановимо:

- Кількість критеріїв – 6;
- Кількість альтернатив – 70;
- Оцінки від 0 до 19.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
1183	14586	12,3
1771	48027	27,1
35829	161003	4,49
2439	89683	36,7
10204	118125	11,6
2176	45653	20,9
8117	11502	1,42
7746	24924	3,22
1541	8008	5,19
32014	58504	1,83
54458	158644	4,95
22228	146384	6,59
12731	67288	5,28
13390	34172	2,6
111275	141440	1,3
36821	135346	3,68
64202	328400	5,12
8798	48999	5,57
6717	186814	27,8
6646	31258	4,7

Середнє прискорення – 9,6.

Встановимо:

- Кількість критеріїв – 6;

- Кількість альтернатив – 150;
- Оцінки від 0 до 19.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
77684	475659	6,1
1244	195153	156,9
19616	165538	8,43
433047	486890	1,12
1758	296334	168,6
96881	691007	7,13
15388	91336	5,94
20126	243409	12,1
1643	43042	26,2
2884	220038	76,3
2154	217949	101,18
9589	35960	3,75
175007	545915	3,12
39591	89731	2,27
14223	77382	5,44
16812	54686	3,25
47868	178835	3,73
30001	154302	5,14
21979	194602	8,85
1853	284592	153,6

Середнє прискорення – 37,96.

Можемо зробити висновок, що середня ефективність застосування фільтрації зростає зі збільшенням кількості альтернатив.

Встановимо:

- Кількість критеріїв – 5;
- Кількість альтернатив – 20;
- Оцінки від 0 до 19.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
491	1934	3,94
2364	3092	1,31
44	167	3,8
197	1757	8,92
1922	4467	2,32
2488	7358	2,96
1731	2464	1,42
232	967	4,17
558	1124	2
49	1534	31,3
4280	4445	1,04
612	1258	2
2451	3956	1,61
1537	1708	1,11
906	1659	1,83
518	1003	1,94
1827	2576	1,41
1399	1666	1,19

257	333	1,3
50	2510	50,2

Середнє прискорення – 6,29.

Встановимо:

- Кількість критеріїв – 7;
- Кількість альтернатив – 20;
- Оцінки від 0 до 19.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
34261	38971	1,14
174281	174110	1
14881	21090	1,41
271584	441825	1,63
62323	73819	1,18
14506	21813	1,5
104278	112905	1,08
1745	15725	9
28454	84715	2,98
826	29571	35,8
8751	42697	4,88
54522	59066	1,08
62487	62261	1
28763	64097	2,23
8792	15094	1,72
25234	515515	20,43

137298	137012	1
227621	273338	1,2
37055	86547	2,34
196012	245392	1,25

Середнє прискорення – 4,69.

Встановимо:

- Кількість критеріїв – 6;
- Кількість альтернатив – 20;
- Оцінки від 0 до 29.

Час роботи з фільтрацією, мс	Час роботи прямого перебору, мс	Прискорення
58135	146944	2,53
11716	13868	1,18
14098	14957	1,06
219616	304995	1.39
622094	623584	1
22196	59776	2.69
170872	294084	1.72
92860	121797	1.31
75652	145275	1.92
95085	153516	1.61
35344	54083	1,53
51799	68169	1,32
102684	244539	2,38
4891	136292	27,87

116098	127191	1,1
12987	23457	1,81
121485	384845	3,17
125479	164633	1,31
104630	241856	2,31
128518	257247	2

Середнє прискорення – 3,13.

Аналізуючи отримані результати, можемо зробити висновки:

- зі збільшенням кількості альтернатив середнє прискорення при застосуванні алгоритму фільтрації зростає;
- ефективність застосування фільтрації залежить від характеру можливих модифікацій;
- на випадкових даних найкращих результатів досягається при структурі задачі, яка включає порівняння альтернатив за 6 критеріям;
- у окремих випадках застосування фільтрації може як не приносити жодної користі, так і приносити значний виграш. Ефективність застосування алгоритму, таким чином, сильно залежить від вхідних даних.

Висновки

У розділі описано деталі реалізації алгоритмів розв’язання прямої та оберненої задач. Розглянуто влаштування програмної частини. Проаналізовано ефективність роботи програми під час розв’язання прямої та оберненої задач ранжування альтернатив. Знайдено сильну залежність роботи програми розв’язання оберненої задачі ранжування альтернатив при застосуванні методу фільтрації, однак доведено доцільність застосування методу фільтрації загалом.

Висновки

У роботі сформульовано та розв’язано пряму та обернену задачу ранжування альтернатив.

Розв’язання прямої задачі здійснене модифікованим методом ELECTRE III. Причини вибору, постановку задачі, та відмінність модифікацій наведено у розділі першому роботи.

Другий розділ присвячений оберненій задачі ранжування альтернатив. Сформульовано та розв’язано обернену задачу.

Третій розділ присвячено аналізові ефективності отриманих рішень. Досліджено поведінку обох задач. Доведено ефективність застосування запропонованого розв’язку оберненої задачі.

Список використаних джерел

1) ТЕХНОЛОГІЧНІ ЗАСОБИ ОНТОЛОГІЧНОГО СУПРОВОДУ РОЗВ'ЯЗАННЯ ЗАДАЧ РАНЖУВАННЯ АЛЬТЕРНАТИВ, Горборуков В. В.
2) An Improved Ranking Method for ELECTRE III, Hui-Fen Li, Jian-Jun Wang
3) ELECTRE III and IV Decision Aids in an Environmental Problem, Joonas Hokkanen, Pekka Salminen
4) Application of ELECTRE III for the integrated management of municipal solid wastes in the Greater Athens Area, Avraam Karagiannidis, Nicolas Moussiopoulos
5) Decision-making in energy planning. Application of the Electre method at regional level for the diffusion of renewable energy technology, M. Beccali, M. Cellura, M. Mistretta