Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики факультету інформатики

ПРИНЦИП БЕЛМАНА В ДИНАМІЧНОМУ ПРОГРАМУВАННІ

Курсова робота

Керівник курсової роботи
Доцент, кандидат ф.-м.н.
Щестюк Наталія Юріївна
_____
(підпис)
“____” _____ 2021 р.
Виконав студент
3-го року навчання спеціальності
113 «Прикладна математика»
Пархомчук Олександр Павлович

Київ 2021

Тема: ПРИНЦИП БЕЛМАНА В ДИНАМІЧНОМУ ПРОГРАМУВАННІ

Календарний план виконання роботи:

Номер етапу	Назва етапів курсової роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання з курсової роботи	23.10.2020	
2	Опрацювання матеріалів	28.12.2020	
3	Написання роботи	01.03.2021	
4	Перевірка роботи с навчальним керівником	11.05.2021	
5	Створення презентації та погодження доповіді с керівником	17.05.2021	
5	Захист курсової роботи	20.05.2021	

ЗМІСТ

ВСТУП

Проблеми оптимального управління займають дуже особливе положення в теорії оптимізації. Насправді вони являють собою природні приклади нескінченних розмірних задач оптимізації, навіть якщо це стосується моделей з кінцевою кількістю ступенів свободи. У теорії управління дається система, яка зазвичай описується диференціальними рівняннями, на яку може впливати зовнішня дія. В задачах оптимального управління така дія повинна здійснюватися для мінімізації заданого функціоналу витрат. Функції, що цікавлять витрати, можуть мати різний характер. Загалом, вони можуть залежати від стану системи, контролю та, можливо, від історії системи протягом заданого інтервалу часу. Елемент управління є оптимальним, якщо результатом розвитку системи є мінімізація витрат.

У реальному фінансовому світі з часом рішення повинні прийматися послідовно. Більш ранні рішення можуть вплинути на доцільність та ефективність пізнішого рішення. У таких середовищах короткозорі рішення, що оптимізують лише безпосередній вплив, зазвичай є неоптимальними для загального процесу. Щоб знайти оптимальні стратегії, потрібно одночасно враховувати поточні та майбутні рішення. Типи багатоступеневих проблем вирішення є типовими умовами, коли використовується динамічна програмування. Однак для задач оптимального управління існує також непрямий метод для отримання необхідних та достатніх умов оптимальності. Цей метод його винахідник Р. Беллман назвав динамічним програмуванням. Розробка правильної теоретичної основи для сортування фактичної застосовності методу динамічного програмування зайняла кілька років, але в

наш час використання динамічного програмування стало повністю суворим. У той же час світло поширюється на його недоліки. У будь-якому випадку динамічне програмування залишається вражаючи потужним мостом між двома, мабуть, не пов'язаними між собою галузями науки, а саме оптимальним управлінням та рівняннями з частковими процесами.

Метою даної курсової роботи є вивчення теоретичних основ динамічного програмування і вирішення задачі на побудову оптимального транспортного маршруту.

Об’єкт практичного дослідження - задача, ціль якої знаходження найкоротшого шляху з використанням динамічного програмування, а саме метод Беллмана, який забезпечує оптимальний маршрут.

Дана курсова робота складається зі вступу, 2 розділів, висновків, списку використаної літератури та додатку.

В першому розділі розглядаються основні поняття динамічного програмування, математична постановка задачі динамічного програмування, метод функціональних рівнянь Р. Белмана та як приклад, задача про ранець.

Другий розділ даної курсової роботи присвячено побудові оптимального транспортного маршруту та використання методу Белмана.

РОЗДІЛ 1. ОСНОВИ ДИНАМІЧНОГО ПРОГРАМУВАННЯ

1.1 Основні поняття динамічного програмування

Поняття динамічного програмування відносилось до розділу прикладної математики “дослідження операцій”. Наведемо декілька важливих теоретичних понять[1, 5].

Дослідження операцій - це побудова, розробка і додаток математичних моделей прийняття оптимальних рішень[1, 2, 5].

Оптимальне рішення задачі - це аспект дослідження операцій, в який включається аналіз і рішення математичних задач вибору на заданій множині допустимих рішень X елемента, що задовольняє певним критеріям оптимальності. Такі завдання, які шукають оптимальне рішення, називаються оптимізаційними. Прикладний аспект дослідження операцій полягає в складанні і реалізації оптимізаційних задач[1, 2, 5].

Математичне програмування - математична дисципліна, присвячена теорії і методам вирішення завдань про знаходження екстремумів функцій на множинах в скінченновимірних векторних просторах, що визначаються лінійними і нелінійними обмеженнями (рівностями і нерівностями). Математичне програмування – розділ науки про дослідження операцій, що охоплює широкий клас завдань управління, математичними моделями яких є скінченномірні екстремальні завдання. Найменування «математичне програмування» пов'язане з тим, що метою вирішення завдань є вибір програми дій[1, 2, 5].

У математичному програмуванні прийнято виділяти наступні розділи:

Лінійне програмування – цільова функція і обмеження лінійні;

Квадратичне програмування - цільова функція квадратична і опукла, допустима множина визначається лінійними рівностями і нерівностями;

Опукле програмування - цільова функція і допустима множина опуклі;

Дискретне програмування - рішення шукається лише в дискретних, наприклад цілочисельних, точках допустимої множини значень змінних;

Стохастичне програмування - на відміну від детермінованих завдань тут вхідна інформація носить елемент невизначеності.

Метод динамічного програмування, його специфіка полягає в тому, що для цілого класу задач керуючих впливом (планування), можна розбити (розділити) на ряд послідовних кроків (етапів), тоді планування стає багатокроковим, багатоетапним, що розвиваються послідовно, і при цьому керуючі оптимальні (найкращі) впливи приймаються на кожному кроці. Це не означає незалежність в ухваленні рішень на кожному кроці. Поточні рішення залежать від рішень, прийнятих на попередніх кроках планування, і, звичайно, від початкових умов, але, ще раз підкреслюю, оптимізація йде на кожному кроці[1, 2, 5].

Динамічне програмування математики і теорії обчислювальних систем (dynamic program) - метод вирішення завдань з оптимальною підструктурою і підзадачами, що перекриваються, який набагато ефективніший, ніж повний перебір[1, 2, 5].

В ході подальшого розвитку, динамічне програмування вже перестав вважатися як розділ прикладної математики. Воно стало відноситись до інформатики в цілому і знайшло свою назву[1, 2, 5].

Динамічне програмування (dynamicprogram) - це спосіб вирішення складних завдань шляхом розбиття їх на більш прості підзадачі, де на кожен крок рішення завдання виділяється фіксований час. Завдання, до якої застосовується метод динамічного програмування повинно містити оптимальну підструктуру що виглядає, як набір підзадач, що перекриваються, складність якої менше вихідної[1, 2, 5].

Динамічне програмування має справу з послідовними процесами прийняття рішень, якими є моделі динамічних систем під контролем особи, що приймає рішення. На кожній стадії або в кожному періоді певного планування, особа, яка приймає рішення вибирає дію з набору доступних альтернатив, який, як правило, залежить від поточного стану системи. Ця дія викликає витрати (або винагороду) та перехід у новий стан системи залежно від обраної дії і попереднього стану. Мета - визначити послідовність дій (а так звана політика), яка оптимізує продуктивність системи (наприклад, мінімізує загальні витрати) протягом планування. Ключовий аспект такої проблеми полягає в тому, що рішення не можна розглядати окремо, тому що потрібно збалансувати бажання низьких поточних витрат з можливістю високих майбутніх витрат[1, 2, 5].

Тепер наведемо основні поняття динамічного програмування.

Система S (завдання). Система, яка визначає стан значень багатьох параметрів x (описується багатьма властивостями змінних - дискретних). В простійшому випадку x є з одного параметра (однієї змінної). Нехай буде так для простоти викладу. Параметр x може приймати значення з деякої множини X (область допустимих значень, $x \in X$). Стан системи S може змінитися в результаті впливу різних факторів - керуючих діяльністю u (керуючих змінних). Знову припустимо, що є тільки одна керуюча змінна. Керуюча змінна u приймає значення з деякої множини U допустимих значень ($u \in U$). Система знаходиться в деяком початковому стані x_0. В результаті впливу на S керує ще одним впливом u_1 система переходить в стан x_1 – перший крок (етап). Опишемо перехід як $x_1 = f(x_0, u_1)$, де функція (залежність) f відображає закон (логіку) зміни стану системи S. Другий крок – $x_2 = f(x_1, u_2)$, третій – $x_3 = f(x_2, u_3)$ і т. д., поки не буде досягнуто деякий кінцевий стан x_n, де n - кількість кроків (етапів), що визначаються особливостями функціонування S. Ефективність (оптимальнаність) функціонування S на кожному кроці визначаються через значення w_i , якщо так можна виразитися, значення часткового критерію ефективності. Так, $w_1 = w_1(x_0, u_1), w_2 = w_2(x_1, u_2), \dots, w_n = w_n(x_{n-1}, u_n)$[1, 2, 5].

Сформулюємо перше ключове положення методу динамічного програмування. Значення критерію W оцінки функціонування системи в цілому є сума значень часткових критеріїв, т. е. $W = \sum_{i=1}^{n} w_i$ - адитивність критерію (цільової функції і т. Д.). Скажімо по-іншому, тільки для систем (завдань, процесів), що володіють властивость адитивності критерію, застосуємо метод динамічного програмування як апарат для її аналізу (рішення, опису) [1, 2, 5].

Друге ключове положення методу відображено в наведених функціональних залежностях: $x_i = f(x_{i-1}, u_i)$ та $w_i = w_i(x_{i-1}, u_i)$. Стан S на кроці i (x_i) Безпосередньо визначається тільки станом на попередньому кроці(x_{i-1}) і керуючим впливом (u_i). Залежність від x_{i-2}, x_{i-3} і т. д. непряма, через x_{i-1}. Аналогічно для w_i[1, 2, 5].

Отже, в чому полягає проблема? Слід так вибрати сукупність (послідовність) впливів u_i, щоб система S перейшла з початкового стану x_0 в кінцевий стан x досягалося екстремальне значення критерію ефективності W її функціонування[1, 2, 5].

Уточнимо принцип оптимальності. Який не був би стан системи перед черговим кроком, потрібно обрати управління на цьому кроці так, щоб виграш на даному кроці плюс оптимальний виграш на всіх наступних кроках був максимальним. Доцільно шукати рішення завдання динамічного програмування, що розглядається, з останнього, n-го кроку. Потім двох останніх, трьох і т. д., аж до першого кроку. Для того щоб прорахувати n-й крок, слід зробити різні припущення про те, як закінчиться передостанній крок, тобто

прорахувати $w_n(x_{n-1}, u_n)$. для всіх значень x_{n-1}., вибираючи найкраще u_n^*. Аналогічно для наступних кроків. Керуючі впливи u_i^*, Обрані за певних припущеннях про закінчення попереднього кроку, зазвичай називають умовно оптимальними. Доходимо до першого кроку, початковий стан системи відомо, тому припущень про допустимі стани системи не потрібно робити. Слід тільки вибрати найкращий керуючий вплив з урахуванням умовно оптимальних управлінь, вже прийнятих на всіх наступних кроках[1, 2, 5].

1.2 Математична постановка задачі динамічного програмування. Метод функціональних рівнянь Р. Белмана

Припустимо, що всі мінімуми, що з'являються в подальшому, існують. Це випадок, наприклад, якщо всі простори стану та дій є (непустими) скінченними множинами[9].

Враховуючи функції f_j та g_j, а також простори стану та дії X_{j+1}. Та U_j для $\mathrm{j} = 1, \dots, \mathrm{n}$ та її рішення залежать лише від початкового стану x_1. Позначимо цю задачу оптимізації через $P_1(x_1)$. Відповідна задача, яка включає лише періоди $\mathrm{j}, \mathrm{j}+1, \dots, \mathrm{n}$ і залежно від початкового стану x_j позначається $P_j(x_j)$. Нехай, $(u_j^*, u_{j+1}^*, \dots, u_n^*)$ оптимальна політика і $v_j^*(x_j)$ - мінімальна вартість для задачі $P_j(x_j)$. Тоді $(u_{j+1}^*, \dots, u_n^*)$ - це оптимальна політика для задачі $P_{j+1}(x_{j+1})$ з початковим станом $x_{j+1}^* := f_j(x_j, u_j^*)$ і вартістю $v_{j+1}^*(x_{j+1}^*)$. Якби існувала "краща" політика $(u_{j+1}^+, \dots, u_n^+)$ для задачі $P_{j+1}(x_{j+1}^*)$ з меншими витратами $v_{j+1}^*(x_{j+1}^*)$, тоді $(u_j^+, u_{j+1}^+, \dots, u_n^+)$ була б "кращою" політикою для $P_{j+1}(x_{j+1}^*)$ із вартістю

$$g_j(x_j, u_j^*) + v_j^+(x_{j+1}^+) < g_j(x_j, u_j^*) + v_{j+1}^*(x_{j+1}) = v_j^*(x_j)$$

всупереч оптимальності $v_j^*(x_j)$. Більше того,

$$v_j^*(x_j) = g_j(x_j, u_j^*) + v_{j+1}^*(x_{j+1}^*) = \min\{g_j(x_j, u_j) + v_{j+1}^*(f_j(x_j, u_j))\}$$

(1.2) [9].

Той факт, що частина оптимальної політики (щодо фіксованого початкового стану) представляє оптимальну політику для відповідної часткової

проблеми, відому як принцип оптимальності Беллмана. Цей принцип сформульовано для задач $P_1(x_1)$ та $P_j(x_j)$ [9].

Принцип оптимальності Беллмана. Нехай $(u_1^*, \dots, u_j^*, \dots, u_n^*)$ - мінімальна політика для задачі $P_1(x_1)$, а x_j - стан на початку періоду j. Тоді $(u_j^*, \dots, u_n^*)$ є оптимальною політикою для задачі $P_j(x_j^*)$. Іншими словами: Рішення в періодах $\mathrm{j}, \dots, \mathrm{n}$ задачі n-періоду $P_1(x_1)$ не залежать від рішень у періодах, $1, \dots, \mathrm{j}-1$ з урахуванням стану x_j на початку періоду j. Функція v_j^*, яка визначена у просторі станів X_j, називається функцією значення $(1 \le \mathrm{j} \le \mathrm{n})$. Для $\mathrm{j} = \mathrm{n}+1$ покладемо

$$v_{n+1}^*(x_{n+1}) \coloneqq 0 \text{ для } x_{n+1} \in X_{n+1} \text{ (1.3)[9].}$$

Для $X_j \subset R$ доцільно визначити v_j на всій R і встановити $v_j^*(x_j) \coloneqq \infty$ для $x_j \in R \setminus X_j (1 \le j \le n+1)$. Співвідношення (1.2), яке справедливе для $\mathrm{j} = 1, \dots, \mathrm{n}$ є називається рівнянням Беллмана:

$$v_j^*(x_j) = \min\left\{g_j(x_j, u_j) + v_{j+1}^*\left(f_j(x_j, u_j)\right)\right\} \quad (x_j \in X_j, 1 \le j \le n) \text{ (1.4)}$$

Рівняння Беллмана пов'язує дві послідовні функції значення v_j^* і v_{j+1}^* та дозволяє нам обчислювати функцію v_j^* коли функція v_{j+1}^* відома[9].

Ми розглянемо деякі модифікації стандартної задачі (1.1). Якщо цільовою функцією є максимізація, а не мінімізація, ми просто замінюємо "min" на "max" у рівнянні Беллмана (1.4). Коли цільова функція має вигляд

$$\sum_{j=1}^{n} g_j(x_j, u_j) + g_j(x_{n+1}),$$

де $g_{n+1}(x_{n+1})$ являє собою кінцеву вартість, формулу (1.3) слід замінити на

$$v_{n+1}^*(x_{n+1}) \coloneqq g_{n+1}(x_{n+1}) \text{ для } x_{n+1} \in X_{n+1} \text{ [9].}$$

Якщо цільова функція має вигляд

$$\prod_{j=1}^{n} g_j(x_j, u_j) \text{ (1.5)}$$

де всі функції $g_j(j = 1, \dots, n)$ мають бути позитивними, тоді у рівнянні Беллмана (1.4) додавання замінюється множенням, а для $j = n + 1, v^*_{n+1}(x_{n+1}) \coloneqq 0$ замінюється на $v^*_{n+1}(x_{n+1}) \coloneqq 1$. Цільова функція типу (1.5) виникає, наприклад, якщо функції g_j представляють надійність компонентів послідовної системи, надійність якої повинна бути мінімізована[9].

Типові задачі з використанням методу динамічного програмування:

1.Задача про обчислення чисел Фібоначчі;

2.Задача про пошук найбільшої загальної підпослідовності;

3.Задача пошуку найбільшої зростающої підпослідовності;

4.Задача про відстань Левенштейна;

5.Задача про порядок множення матриць;

6.Задачи про вибір траєкторії;

7.Задача послідовного прийняття рішення;

8.Задача про використання робочої сили;

9.Задача управління запасами;

10.Задача про ранці;

11.Алгоритм Флойда-Уоршелла: знайти найкоротші відстані між усіма вершинами зваженого орієнтованого графа;

12.Алгоритм Беллмана-Форда: знайти найкоротший шлях в зваженому графі між двома заданими вершинами;

13.Максимальна незалежна множина вершин в дереві.

1.3 Приклад. Задача про ранець

Задача про "ранець" - відома задача, яка ілюструє послідовне прийняття рішень. Розглянемо ранець, який має кінцевий об’єм K одиниць. Ми можемо наповнити рюкзак цілим числом предметів x_i, де є N типів предметів. Кожен предмет має одиницю об'єму v_i та одиницю вартості c_i. Наша мета - визначити кількість предметів x_i для розміщення в рюкзаку, щоб максимізувати загальну вартість. Приклад на рис. 1.1[3, 5, 6, 7, 8].

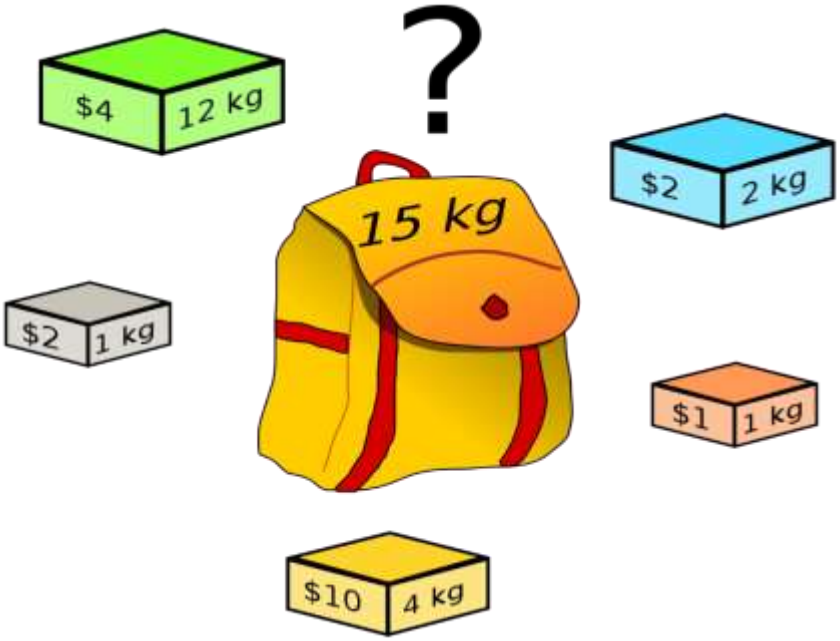

Рис. 1.1 – Іллюстрація задачі про ранець

Ця задача є ідеальним кандидатом для такого методу, як динамічне програмування. А саме, якщо ми розглядаємо можливість послідовного заповнення рюкзака по одному предмету, то легко зрозуміти, що рішення, який предмет додати, зараз впливає на те, які предмети ми можемо додати пізніше. Щоб сформулювати проблему динамічного програмування, ми визначаємо "стан" для системи. Зокрема, визначимо y як залишок місця в рюкзаку. На початку процесу заповнення залишковий об'єм дорівнює $y = K$. Стан еволюціонує відповідно до $y - v_i$, якщо ми додаємо одну одиницю i-типу. Очевидно, ми не можемо включати одиниці, обсяг яких перевищує решту обсягу, тобто $v_i \leq y$. Більше того, значення рюкзака збільшується на c_i при додаванні однієї одиниці предмету i-типу[3, 5, 6, 7, 8]. Підсумовуємо математично,

- Стан y представляє залишок вільного місця в рюкзаку
- Динаміка стану $y - v_i$
- Нарахована вартість становить c_i
- Початковий стан - K
- Елементи, які ми можемо додати, обмежені за обсягом, тобто $v_i \leq y$.

Тепер ми ретельно можемо визначити функцію значення. Нехай $V(y)$ представляє максимально можливу вартість ранцю для залишку вільного місця, який дорівнює y. Наприклад, коли залишок вільного місця дорівнює нулю, тоді максимальнеа можлива вартість цього залишку дорівнює нулю. Тепер ми можемо написати принцип оптимальності та граничні умови:

$$V(y) = max_{v_i \leq y, i \in \{1,..,N\}} \{c_i + V(y - v_i)\}$$
$$V(0) = 0$$

Приведемо конкретний приклад. Маємо ранець, який ми можемо заповнити або їжею, або обладнанням. Ми повинні самі вирішити, скільки чого нам потрібно. Обсяг ранцю обмеженний[3, 5, 6, 7, 8].

Однак, ми повинні зібрати максимально “вигідний” для нас ранець. З математичного боку задача виглядає ось так:

$$\max 2x_1 + x_2 \text{ – рівняння вартості ранцю}$$
$$x_0 + 2x_1 + 3x_2 = 9 \text{ – рівняння об’єму ранця}$$
$$x_i \geq 0 \ \in Z$$

Де x_i - число товарів; $i = 1$ - їжа, $i = 2$ - обладнання, і припустимо, і = 0 це буде порожній простір; об'єм ранцю $K = 9$; значення вартості елементу складають $\mathrm{c}_0 = 0, \mathrm{c}_1 = 2, \mathrm{c}_2 = 1$; значення обэму одиниці товару складають $v_0 = 1, v_1 = 2, v_2 = 3$.

Використаємо рекурсію:

$V(0) = 0$

$V(1) = max_{v_i \leq 1} \{c_i + V(1 - v_i)\}$

$= \max\{0 + V(1 - 1)\} = 0$

$V(2) = max_{v_i \leq 2} \{c_i + V(2 - v_i)\}$

$= \max\{0 + V(2 - 1).2 + V(2 - 2)\} = 0$

$V(3) = max_{v_i \leq 3} \{c_i + V(3 - v_i)\}$

$= \max\{0 + V(3 - 1), 2 + V(3 - 2), 1 + V(3 - 3)\} = 2$

$V(7) = max_{v_i \leq 7} \{c_i + V(7 - v_i)\}$

$= \max\{0 + V(7 - 1), 2 + V(7 - 2), 1 + V(7 - 3)\}$

$= \max\{6, 2 + 4, 1 + 4\} = 6$

$V(8) = max_{v_i \leq 8} \{c_i + V(8 - v_i)\}$

$= \max\{0 + V(8 - 1), 2 + V(8 - 2), 1 + V(8 - 3)\}$

$= \max\{6, 2 + 6, 1 + 4\} = 8$

$V(9) = max_{v_i \leq 9} \{c_i + V(9 - v_i)\}$

$= \max\{0 + V(9 - 1), 2 + V(9 - 2), 1 + V(9 - 3)\}$

$= \max\{8, 2 + 6\ , 1 + 6\} = 8$

$V(9) = 8$

Тому оптимальним рішенням поставленної задачі буде взяти 4 одиниці їжі, 0 одиниць обладнання та в нас ще залишиться одне вільне місце.

Існує багато різновидів проблеми рюкзака, які виникли внаслідок величезної кількості застосувань основної проблеми. Основні варіації відбуваються шляхом зміни кількості деяких параметрів проблеми, таких як кількість предметів, кількість цілей або навіть кількість рюкзаків[3, 5, 6, 7, 8].

РОЗДІЛ 2. ПРАКТИЧНЕ ЗАСТОСУВАННЯ МЕТОДУ ДИНАМІЧНОГО ПРОГРАМУВАННЯ

2.1. Використання методу динамічного програмування для розв’язання задачі про знаходження найкоротшого маршруту

З метою практичного вивчення методу динамічного програмування була поставлена задача на знаходження найкоротшого маршруту від будинку автора до могилянскої Академії. Якщо розглядати завдання з математичної точки зору, то задача зводиться до задачи пошуку найкоротшого шляху у зваженому графі.

Для побудови графу було використано сервіс GOOGLE MAPS. Отриманий граф (рис 2.1).

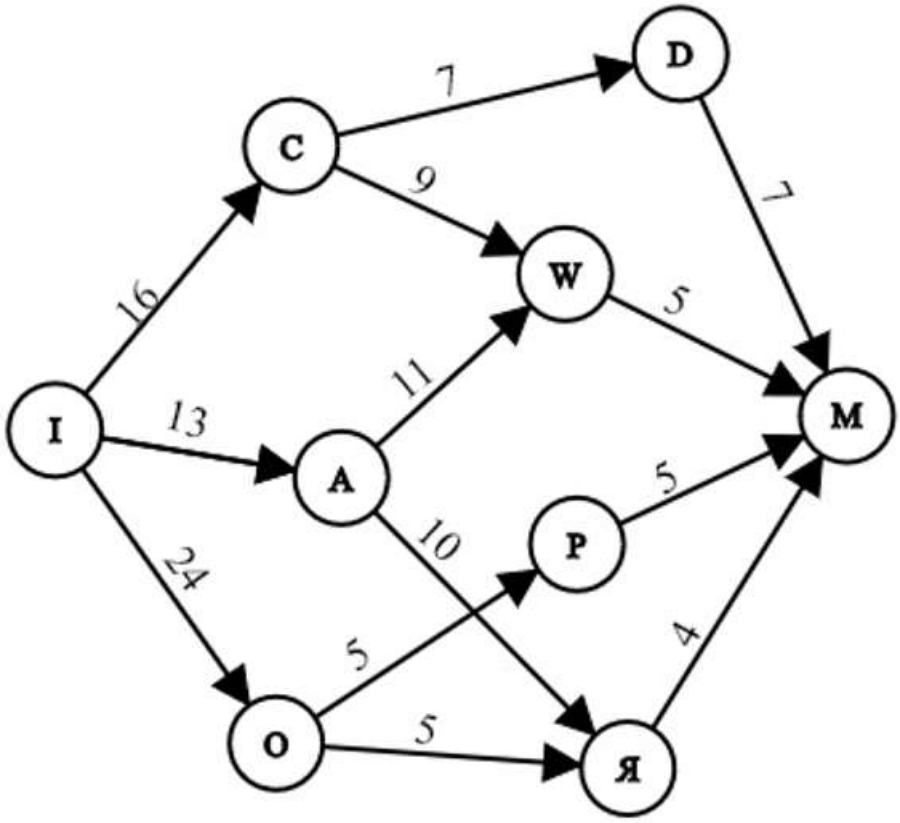

Рис. 2.1 - Граф можливих маршрутів

Отримані вершини: I – м. Ірпінь, C – Святошин, D – Дорогожичі, A – Академістечко, O – Оболонь, P – Рибальскій острів, Я – Репьяхів Яр, W – проспект Перемоги та M – Києво-Могилянська академія. Вага ребер представлена у вигляді відстані(у км.) від одної вершини до іншої.

Нам потрібно знайти найкоротший шлях з вершини I до вершини M. Давайте визначимо функцію вартості витрат як $V(i)$. Тобто $V(i)$ – найкоротший шляху від вершини i до вершини M. Наприклад $V(M) = 0$. Нехай $c(i,j)$ позначає вартість поїздки з вершини i до вершини j. Наприклад, (I, A) = 13. Тоді $c(i,j) + V(j)$ представляє вартість подорожі з вершини i до вершини j, а потім з вершини j до вершини M найкоротшим шляхом. Це дозволяє нам записати рівняння принципу оптимальності та граничних умов:

$$V(i) = \min\{c(i,j) + V(j)\}$$

$$V(M) = 0$$

Використавши рекурсію ми можемо вирішити ці рівняння, починаючи з вершини M, рухаючись у зворотному порядку до вершини I наступним чином:

$V(D) = c(D, M) + V(M) = 7$

$V(W) = c(W, M) + V(M) = 5$

$V(P) = c(P, M) + V(M) = 5$

$V(Я) = c(Я, M) + V(M) = 4$

$V(P) = c(P, M) + V(M) = 5$

$V(C) = \min\{c(C, D) + V(D), c(C, W) + V(E)\} = \{7 + 7{,}9 + 5\} = 14$

$V(A) = \min\{c(A, W) + V(W), c(A, Я) + V(Я)\} = \{11 + 5{,}10 + 4\} = 14$

$V(O) = \min\{c(O, P) + V(P), c(O, Я) + V(Я)\} = \{5 + 5{,}5 + 4\} = 14$

$V(I) = \min\{c(I, C) + V(C), c(I, A) + V(A), c(I, O) + V(O)\}$

$= \{16 + 14, 13 + 14, 24 + 9\} = 27$

Отже, ми знайшли оптимальный шлях $I - A - Я - M$.

Таким чином, ми розв’язали задачу, використавши метод динамічного програмування на практиці.

ВИСНОВКИ

Підведемо підсумки курсової роботи:

Наше дослідження було направлено на опанування методу динамічного програмування та методу функціональних рівнянь Беллмана. Для закріплення вивчення методу динамічного програмування у курсовій роботі була розв’язана задача задача пошуку найкоротшого маршруту на зваженому графі. Для конкретного прикладу був знайдений найкоротший шлях від дому до академії. За допомогою сервісу Google Maps були створені різні маршрути, були вставновлені проміжні вершини та побудований граф. За допомогою методу Беллмана, був знайдений найкоротший шлях.

Також був реалізований метод Беллмана-Форда на мові програмування Java

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Динамічне програмування [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Динамічне_програмування

2. Беллман Р. Динамическое програмирование. / Ричард Беллман [пер. с английского И.М. Андреезой, А.А Корбута, И. В. Романовского, И. Н. Соколовой]. – Москва: 1960. – 400с.

3. Трунов О. М. Приклади розв’язку практичних та ситуаційних задач з курсу «Дослідження операцій» / О. Трунов, С. Волкова – К: Миколаївський державний гуманітарний університет ім. Петра Могили, 2008 – 116с.

4. Квітко О. С. Дослідження методів динамічного програмування у корпоративних мережах / О. Квітко, К. Дорошенко, В. Полторак

5. Окулов С.М., Пестов О.А. Динамическое програмирование

6. H. Hotelling, “Stability in competition,” The Economic Journal, vol. 39, no. 153, pp. pp. 41–57, 1929

7. E. V. Denardo,Dynamic programming models and applications. Courier Dover Publications, 2003.

8. D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas,Dynamic programming and optimal control. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

9. Karl Frauendorfer, Hans Glavitsch, Optimization in Planning and Operation of Electric Power Systems