УДК 004.932.4

Мазурук О.В., студент 2 курсу
спеціальності 122 «Комп'ютерні науки»
Федоров Є.Є., д.т.н., професор, професор
кафедри Інформаційних технологій

ПРИКЛАДНЕ ВИКОРИСТАННЯ МУРАШИНОГО АЛГОРИТМУ НА ПРИКЛАДІ ПОБУДОВИ МАРШРУТУ ВИКОРИСТОВУЮЧИ МОВУ ПРОГРАМУВАННЯ JAVA

Донецький національний університет імені Василя Стуса, м. Вінниця

Задачі із пошуку найкоротшого шляху є досить поширеними у багатьох сферах людської діяльності. Зазвичай, метою даної задачі є знаходження найкоротшого або найоптимальнішого маршруту між початковою та кінцевою точками. Проте існують і інші варіації даної задачі, наприклад коли для обчислення задається тільки вхідна чи вихідна точка або необхідно розрахувати маршрут для усіх пар точок.

Для вирішення було розроблено безліч алгоритмів, таких як: алгоритм Беллмана-Форда, алгоритм Дейкстри, алгоритм пошуку A*, мурашиний алгоритм та інші[1]. Кожний алгоритм має певні вимоги для вхідного набору даних, при якому швидкість пошуку найкоротшого маршруту найшвидша. Виходячи із цього правила, одним із найоптимальнішим варіантом пошуку маршруту за вказаними початковою та кінцевою точками є мурашиний алгоритм.

Мурашиний алгоритм – це алгоритм який дозволяє вирішувати задачі комбінаторної оптимізації, зокрема пошук найкоротшого шляху на графові[2]. Основою алгоритму стала поведінка мурашиної колонії при пошуку та перенесенні їжі.

Даний алгоритм застосовують для вирішення складних задач оптимізації. Типовим прикладом може виступати задача побудови логістичної системи, знаходження маршруту для певної мережі. Кількість задач до яких застосовується мурашиний алгоритм доволі велика і стосується кожної сфери людської діяльності. В нашому випадку алгоритм буде застосований для побудови туристичних маршрутів. Для проведення дослідження буде використано мову програмування Java, оскільки є одним із найпопулярніших інструментів для математичних досліджень[3].

В якості вхідних даних буде використаний граф, що містить 29 вершин, вихідною точкою вважати вершину №3, при цьому кожна вершина має зв'язок із кожною. Також, мурашиний алгоритм передбачає наявності додаткових параметрів, для яких було встановлено наступне: кількість агентів(штучних мурах) що буде створено за 1 ітерацію становить 150 штук, швидкість випаровування феромонів 0.1, вплив феромонів становить 1.

В результаті даних вхідних параметрів на 1 ітерації був отриманий результат зображений на рисунку 1.

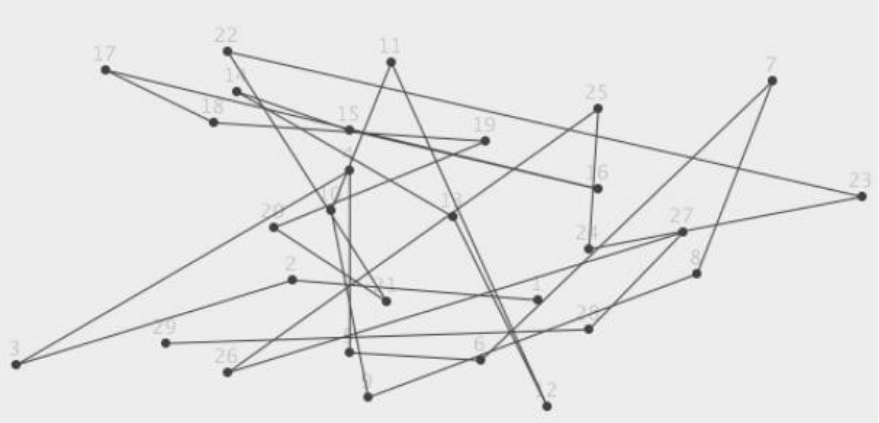

Рисунок 1 – візуалізація руху агентів після виконання першої ітерації алгоритму.

З рисунку 1 ми бачимо що агенти розпочали рух та здійснюють пошук найкоротшого маршруту. При виконанні останньої ітерації був отриманий фінальний результат, що відображено на рисунку 2.

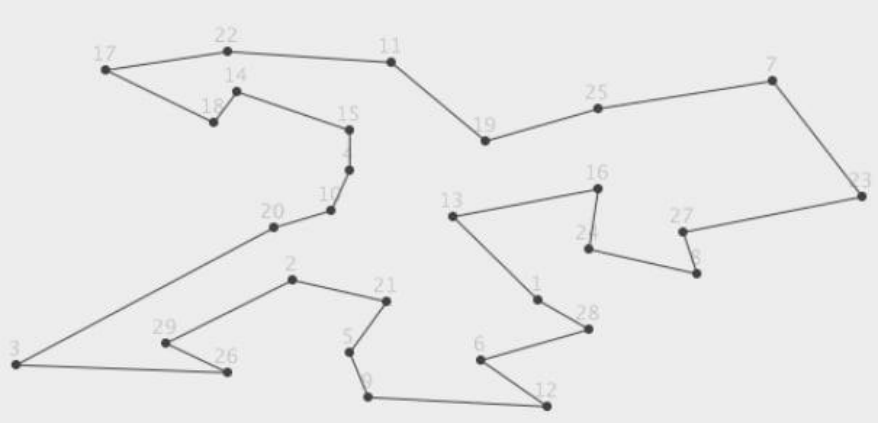

Рисунок 2 – візуалізація фінального результату

Фінальний маршрут є найкоротшим і містить вирішення для даної задачі. Також, швидкість роботи алгоритму можна вдосконалити за рахунок зміни вхідних даних, а саме кількості агентів які створюють для однієї ітерації, швидкість випаровування феромонів та вплив феромонів. Для визначення оптимального співвідношення проведено тестування із різними вхідними параметрами, які відображені у таблиці 1.

К-сть агентів за ітерацію	Швидкість випаровування феромонів	Вплив феромонів	Швидкість роботи, мілісекунд
100	0.1	1	2271
100	0.2	1	2149
100	0.1	2	2100
100	0.5	2	2019
100	1	2	2031

Таблиця 1- результати тестування роботи алгоритму із різними вхідними параметрами

Тестування проводилось на комп'ютері із наступними параметрами: процесор AMD Ryzen 4700U 7(8 ядер, 8 потоків, 2.00-4.10 ГГц, кеш 8 Мб), оперативна пам'ять DDR4(2400MHz) 16 гб, периферійна пам'ять M.2 PCIe SSD: 512 Гб. Отже, найточніший маршрут і найшвидша робота алгоритму була досягнута коли швидкість випаровування феромонів становить 0.5, а вплив феромонів 2.

При проведенні дослідження було встановлено, що різні вхідні параметри можуть погіршувати чи покращувати роботу алгоритму. Також, на основі проведеного дослідження, можна стверджувати що мурашиний алгоритм є оптимальним для вирішення задач маршрутизації.

Список використаної літератури

1. *Shortest Path Problem – [Електронний ресурс]. Режим доступу: https://www.sciencedirect.com/topics/computer-science/shortest-path-problem*
2. *Ant colony optimization theory: A survey – [електронний ресурс]. Режим доступу: https://www.sciencedirect.com/science/article/pii/S0304397505003798*
3. *Performance Comparison - C++ / Java / Python / Ruby/ Jython / JRuby / Groovy – [Електронний ресурс]. Режим доступу: https://www.cxyzjd.com/article/pj1258/17527257*

УДК 004.01

Бралатан Р. А.
Студент 2-го курсу СО «Магістр»
Спеціальності 122 «Комп'ютерні науки»
Баркалов О.О.
професор кафедри інформаційних технологій

МОДИФІКАЦІЯ РЕКОМЕНДАЦІЙНОГО АЛГОРИТМУ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ З ВИКОРИСТАННЯМ АЛГОРИТМУ TOMPSON SAMPLING

Донецький національний університет імені Василя Стуса, Україна

Рекомендаційна система – це алгоритми, що пропонують різні товари та послуги на основі отриманих та проаналізованих даних про користувача. Найчастіше вони рекомендаційні алгоритми використовуються у різних комерційних проектах, інтернет магазинах, музикальних та відео сервісах і т. д.

В основі роботи item-item алгоритму колаборативної фільтрації лежить принцип використання user-item матриці. Ця матриця формується за таким принципом: у її рядках записуються користувачі системи, а у стовпцях продукти та категорії продуктів. Пересічення рядків та стовпців матриці містить оцінку,