УДК 519.863

Царук В.В.
Демчук О.Ю.
Кулик Я.А.

АНАЛІЗ ТА ОПТИМІЗАЦІЯ МЕРЕЖЕВОГО МАРШРУТИЗАТОРА ЗА ДОПОМОГОЮ АЛГОРИТМІВ МУРАШИНОГО АНТУРАЖУ

Вінницький національний технічний університет

Анотація

Дослідження присвячено аналізу та оптимізації процесу мережевої маршрутизації за допомогою алгоритму мурахиного антуражу. У роботі виконано експериментальний аналіз впливу кількості вершин у графі на час та відстань маршрутизації. Для досягнення цієї мети були створені графи з різною кількістю вершин, і для кожного графа застосовано алгоритм мурахиного антуражу. Експериментальні результати були систематично зібрані, оброблені та проаналізовані.

В роботі розглянуто теоретичні аспекти використання мурашиних алгоритмів у контексті маршрутизації. Зокрема, розглянуті принципи вибору оптимального маршруту та адаптація алгоритмів до різноманітних графів. Статистичний аналіз та графічне представлення результатів дозволили визначити тенденції та залежності між кількістю вершин у графі та ефективністю маршрутизації. Отримані висновки становлять важливий внесок у розуміння можливостей застосування мурашиних алгоритмів у великих та складних мережах.

Ключові слова: маршрутизація, мурашині алгоритми, графи, оптимізація мережі, експериментальний аналіз, мережеві топології, відстань між вершинами, час виконання, алгоритмічна ефективність, мережеві додатки, графічне представлення результатів, адаптація алгоритмів, високопродуктивні мережі, теоретичний аналіз, перспективи досліджень.

Abstract

The study is devoted to the analysis and optimization of the network routing process using the ant entourage algorithm. The paper analyzes the effect of the number of vertices in a graph on the routing time and distance. To achieve this goal, graphs with different numbers of vertices were created, and the ant entourage algorithm was applied to each graph. The experimental results were systematically collected, processed, and analyzed.

The paper discusses the theoretical aspects of using ant algorithms in the context of routing. In particular, the principles of choosing the optimal route and the adaptation of algorithms to various graphs are considered.

Statistical analysis and graphical representation of the results allowed us to identify trends and dependencies between the number of vertices in a graph and routing efficiency. These findings make an important contribution to understanding the possibilities of using ant algorithms in large and complex networks.

Keywords: Routing, Ant algorithms, Graphs, Network optimization, Experimental analysis, Network topologies, Distance between vertices, Execution time, Algorithmic efficiency, Network applications, Graphical representation of results, Algorithm adaptation, High performance networks, Theoretical analysis, Research perspectives.

Вступ

Мурашині алгоритми - це методи оптимізації та пошуку рішень, засновані на поведінці колоній мурашок. Вони відносяться до класу природоподібних алгоритмів - алгоритмів, натхненних біологічними системами. Мурашині алгоритми є одними з найпопулярніших та ефективних біонічних алгоритмів, що застосовуються для вирішення складних комбінаторних та оптимізаційних задач [1].

У даній роботі буде розглянуто принцип роботи мурашиних алгоритмів, їх види та застосування. Мурашині алгоритми - це методи оптимізації та пошуку рішень, засновані на поведінці колоній мурашок. Вони відносяться до класу природоподібних алгоритмів - алгоритмів, натхненних біологічними системами. Мурашині алгоритми є одними з найпопулярніших та ефективних біонічних алгоритмів, що застосовуються для вирішення складних комбінаторних та оптимізаційних задач.

Такі алгоритми можуть бути застосовані у різних галузях, таких як маршрутизація в комп'ютерних мережах, оптимізація транспортних маршрутів, розташування об'єктів та інші завдання. Вони володіють властивістю адаптації до змінних умов та можуть знаходити оптимальні рішення у складних, динамічних

середовищах.

У даному дослідженні звертається увага на концепцію мурашиних алгоритмів, їхню ефективність у різних сценаріях та вплив кількості вершин у графі на продуктивність алгоритму. Результати експериментів вказують на важливість розуміння та оптимізації мурашиних алгоритмів для розв'язання реальних завдань у різних галузях застосування.

Важливим аспектом вивчення мурашиних алгоритмів є їхня відповідність природній еволюції та взаємодії мурашок у природі. Алгоритми імітують принципи спільної роботи мурашок, такі як комунікація через феромони та взаємодія для досягнення спільної мети. Ця ідея призводить до виникнення ефективних методів пошуку та оптимізації в задачах зі складними просторами можливих рішень [2].

У рамках дослідження буде звернуто увагу на різноманітні види мурашиних алгоритмів, включаючи алгоритми на основі антуражу та інші модифікації. Окремий аналіз різних варіацій алгоритмів дозволить визначити їхню ефективність та застосування в конкретних завданнях.

Також у роботі буде вивчено вплив параметрів алгоритму на його результативність, зокрема, розглянуто вплив кількості мурашок, коефіцієнтів випаровування феромонів та інших параметрів на збіжність та якість отриманих рішень.

Отримані в ході експериментів дані та аналіз дадуть можливість глибше розуміти принципи роботи мурашиних алгоритмів та їхній потенціал у вирішенні реальних завдань оптимізації та пошуку рішень у різних галузях.

Результати дослідження

Мурашині алгоритми, які знаходять своє вдохновлення в стратегії мурашок при пошуку їжі, представляють собою важливий інструмент в області оптимізації. Основна ідея полягає в тому, що мурашки залишають феромони на шляху до їжі, що визначає оптимальний маршрут. Проста евристика полягає в тому, що інші мурашки вибирають шлях, слідуючи за сильнішим феромонним слідом.

Цей метод дозволяє вирішувати різноманітні завдання оптимізації та пошуку рішень, адже природні принципи, використовувані мурашками, можуть бути успішно застосовані і в комп'ютерних алгоритмах. Процес визначення найкращого маршруту через залишення феромонів стимулює співпрацю та обмін інформацією між мурашками, що призводить до знаходження оптимальних рішень у складних задачах.

Інші модифікації мурашиних алгоритмів, такі як алгоритм мурашиної колонії, елітний алгоритм мурашиної колонії, алгоритм макс-мін та алгоритм мурашиної системи рангів, доповнюють базовий підхід, роблячи його більш гнучким та ефективним у різних сценаріях використання [3].

Важливими областями застосування мурашиних алгоритмів є оптимізація маршрутів, розподіл ресурсів, кластеризація даних, пошук приблизних рішень складних обчислювальних задач та машинне навчання. Основні переваги цих алгоритмів включають їх простоту, масштабованість і здатність до знаходження глобально оптимальних рішень в різних областях застосування.

Існує кілька модифікацій мурашиного алгоритму, кожна з яких має свої особливості та вдосконалення:

- Алгоритм мурашиної колонії: Базовий алгоритм, що використовує позитивний зворотній зв'язок за рахунок феромонів для пошуку оптимального рішення. Мурашки залишають феромони на шляху, що дозволяє визначити найкращий маршрут для подальших мурашок.
- Елітний алгоритм мурашиної колонії: Цей підхід включає в себе "елітну" мурашку, яка відзначається здатністю посилювати найкращі рішення. Елітна мурашка може впливати на вибір шляху та сприяти збереженню та розповсюдженню оптимальних рішень.
- Алгоритм макс-мін: У цій модифікації феромонні значення обмежуються верхньою та нижньою межею для покращення збіжності. Такий підхід допомагає уникнути перевантаження феромонів і може покращити швидкість та стабільність алгоритму.
- Алгоритм мурашиної системи рангів: Цей підхід включає в себе ранжування рішень за якістю. Мурашки визначають оптимальні шляхи в залежності від їхнього рангу, що може призвести до кращої збалансованості розв'язків.

Мурашині алгоритми успішно використовуються у різних сферах, таких як оптимізація маршрутів, розподіл ресурсів, кластеризація даних, пошук приблизних рішень складних обчислювальних задач та машинне навчання. Їхні переваги включають простоту, масштабованість та здатність знаходження глобально оптимальних рішень в різноманітних викликах.

Основні переваги мурашиних алгоритмів полягають у їхній ефективності та універсальності в різноманітних областях застосування:

- Простота: Мурашині алгоритми відзначаються простотою концепцій та зрозумілістю в реалізації. Це

робить їх доступними для використання без глибоких математичних або технічних знань.

- Масштабованість: Мурашині алгоритми легко масштабуються для вирішення проблем різної складності та розміру. Вони можуть ефективно працювати як з невеликими, так і з великими наборами даних та складними задачами оптимізації.
- Стійкість до змін даних: Здатність мурашиних алгоритмів пристосовуватися до змін в навколишньому середовищі та динаміці даних робить їх стійкими та надійними в реальних умовах використання.
- Здатність знаходження глобально оптимального рішення: Мурашині алгоритми мають потенціал знаходити глобально оптимальні рішення завдяки механізму взаємодії мурашок та використанню феромонів. Це робить їх важливим інструментом для розв'язання складних оптимізаційних задач.

У сукупності ці переваги роблять мурашині алгоритми популярними серед інженерів та дослідників у різних галузях, де необхідно знаходити оптимальні рішення для різноманітних завдань.

Гібридні мурашині алгоритми

Одним з перспективних напрямків удосконалення мурашиних алгоритмів є розробка гібридних підходів, що поєднують їх з іншими методами оптимізації та штучного інтелекту.

Наприклад, поєднання мурашиних алгоритмів з генетичними алгоритмами дає гібридні мурашино-генетичні методи. Вони використовують синергію випадкового пошуку рішень за допомогою популяції особин (як в генетичних алгоритмах) та спрямованого інтелектуального пошуку за "феромонним слідом" (як в мурашиних).Іншим прикладом є поєднання з методами рою частинок. Тут використовуються ідеї кооперативної поведінки рою частинок при пошуку оптимальних рішень.

Досліджуються також гібриди з нейронними мережами, нечіткою логікою, методами локального пошуку (градієнтним спуском, пошуком з табу тощо).

Переваги гібридних підходів:

- Компенсація недоліків окремих методів
- Синергетичний ефект від поєднання різних підходів
- Підвищення точності та швидкості пошуку рішень

Гібридні мурашині алгоритми активно застосовуються для:

- Оптимізації складних інженерних систем
- Машинного навчання та штучного інтелекту
- Фінансової оптимізації та прогнозування
- Обробки зображень, сигналів, даних
- Біоінформатики

Отже, гібридизація мурашиних алгоритмів відкриває широкі можливості для побудови високоефективних оптимізаційних методів, здатних розв’язувати складні прикладні задачі з великою кількістю змінних. Існують різні схеми гібридизації: послідовна, паралельна, ієрархічна. Вибір схеми залежить від конкретної задачі. Для налаштування гібридних алгоритмів використовуються методи математичного моделювання, статистичної оптимізації параметрів, машинного навчання.

Актуальними напрямками досліджень є розробка адаптивних гібридних алгоритмів, що можуть підлаштовуватися під задачу, а також застосування гібридних підходів в хмарних та розподілених обчисленнях. Подальші дослідження в цьому напрямку мають велике практичне значення для створення ефективних методів вирішення складних оптимізаційних проблем в науці, інженерії, економіці, медицині.

Практичне застосування мурашиних алгоритмів

Мурашині алгоритми можуть бути ефективно застосовані для розв'язання багатьох практичних задач. Одним з ключових напрямків їх використання є оптимізація транспортних маршрутів. Наприклад, алгоритм мурашиної колонії може бути використаний для побудови оптимального маршруту доставки товарів, що дозволяє мінімізувати час та витрати на логістику [4].

Іншим важливим застосуванням є розподіл ресурсів в складних системах. Мурашині алгоритми здатні ефективно розподіляти обмежені ресурси (пам'ять комп'ютера, пропускну здатність мережі тощо) між конкуруючими процесами.

В галузі телекомунікацій мурашині алгоритми використовуються для маршрутизації трафіку в мережах, пошуку оптимальних шляхів передачі даних. Вони дозволяють швидко адаптуватися до змін топології мережі.

Мурашині алгоритми також ефективні для кластерного аналізу та класифікації даних. Вони можуть виявляти приховані закономірності та групувати схожі об'єкти в кластери.

Крім того, мурашині алгоритми застосовуються в задачах комп'ютерного зору для розпізнавання образів, в системах штучного інтелекту для навчання агентів, в теорії управління для оптимізації складних систем.

Вони також можуть використовуватися в задачах проектування інтегральних схем, для верифікації програмного забезпечення, оптимізації інвестиційних портфелів та інших фінансових застосувань.

Мурашині алгоритми ефективні в біоінформатиці для аналізу послідовностей ДНК і білків. Вони допомагають знаходити приховані закономірності в геномних та протеомних даних.

У промисловості мурашині алгоритми застосовуються для оптимізації складних технологічних процесів, розкладів роботи устаткування, ланцюжків постачання.

В медицині вони можуть використовуватися для аналізу зображень в радіології, планування лікування в онкології, прогнозування поширення епідемій.

Отже, сфера застосування мурашиних алгоритмів надзвичайно широка і охоплює практично всі галузі науки і техніки, де потрібно знаходити оптимальні рішення в умовах комбінаторного вибуху. Їх гнучкість, масштабованість та здатність до самоорганізації робить мурашині алгоритми потужним інструментом для вирішення найскладніших реальних задач.

Комп’ютерні експерименти

Перед початком експериментів, ми написали власний код для відтворення тих чи інших ситуацій. Використовували програмне середовище Visual Studio, мову програмування Python та такі бібліотеки як: matplotlib, networkx та інші. Написаний алгоритм зображений нижче:

```
import random  # Імпорт модуля для генерації випадкових чисел
import time # Імпорт модуля для вимірювання часу виконання
import matplotlib.pyplot as plt # Імпорт модуля для візуалізації даних
import networkx as nx # Імпорт модуля для роботи з графами
from aco_routing import ACO  # Імпорт класу ACO з бібліотеки aco_routing

def create_graph(n):        #Функція для створення графу з випадковими вагами між вершинами.  |  :param n: Кількість вершин у графі | :return: Граф
    G = nx.Graph()
    for i in range(n):
        for j in range(i+1, n):
            G.add_edge(str(i), str(j), cost=random.randint(1, 10))
    return G

times = []          # Список для зберігання часу виконання
distances = []      # Список для зберігання відстаней
vertices = list(range(5, 105, 5))        # Список кількостей вершин для аналізу

for v in vertices:
    print(f"Будуємо граф із {v} вершинами...")
    G = create_graph(v)
    source = "0"
    destination = str(v-1)

    start_time = time.time()
    aco = ACO(G, ant_max_steps=1000, num_iterations=50)
    aco_path, aco_cost = aco.find_shortest_path(source, destination, num_ants=100)
    end_time = time.time()

    times.append(end_time - start_time)
    distances.append(aco_cost)

plt.figure(figsize=(12, 6))         # Створення нового графіку розміром 12x6 дюймів

plt.subplot(1, 2, 1)
plt.plot(vertices, times, marker='o')
plt.title('Час виконання від кількості вершин')
plt.xlabel('Кількість вершин')
plt.ylabel('Час виконання (сек)')

plt.subplot(1, 2, 2)
plt.plot(vertices, distances, marker='o')
plt.title('Відстань від кількості вершин')
plt.xlabel('Кількість вершин')
plt.ylabel('Відстань')
```

```
plt.tight_layout()
plt.show()
```

Цей код генерує графи з випадковими вагами між вершинами. Для кожної кількості вершин від 5 до 100 створюється граф з відповідною кількістю вершин. Потім для кожного графа знаходиться найкоротший шлях від першої вершини до останньої за допомогою алгоритму мурах (ACO). Час виконання алгоритму ACO та довжина знайденого шляху зберігаються в двох списках. Кожен раз, коли ми запускаємо код, створюється новий граф з різними вагами між вершинами. Це призводить до того, що алгоритм оптимізації колонії мурахів (ACO) знаходить різні шляхи, що відображаються на графіках. Випадкові числа використовуються для симуляції реального світу, де ваги між вершинами можуть варіюватися. Це допомагає перевірити ефективність алгоритму в різних умовах і забезпечує більш загальне розуміння його роботи.

Графіки, які будуються на основі цих списків, показують залежність часу виконання та довжини шляху від кількості вершин у графі (рисунок 1.1).

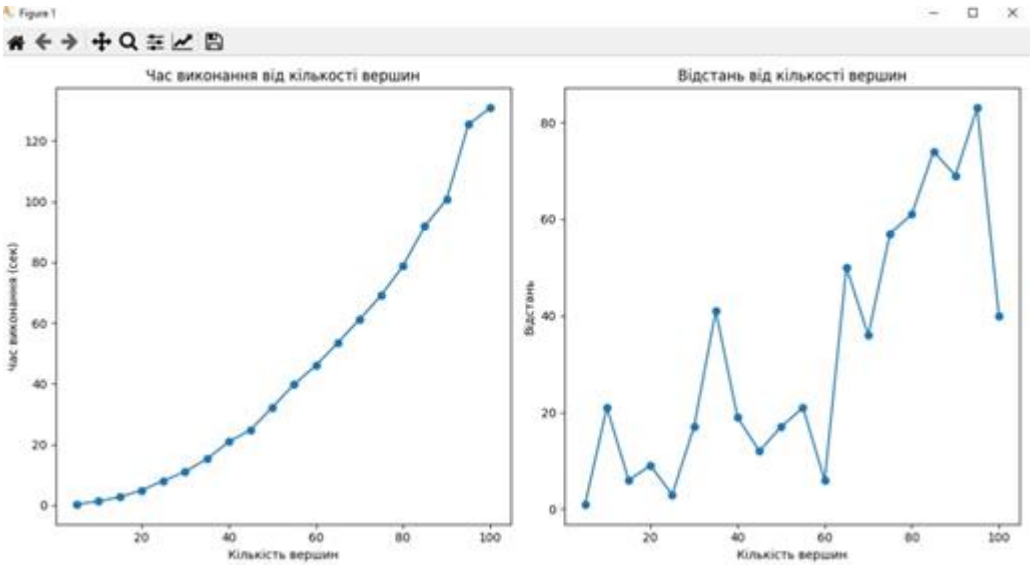

Рисунок 1.1 – Результат виконання програми та графічна побудова графіків

З ростом кількості вершин час виконання алгоритму ACO зростає. Це пов'язано з тим, що алгоритм ACO виконує декілька ітерацій, а на кожній ітерації він має обчислити всі можливі шляхи від першої вершини до останньої. Кількість можливих шляхів зростає експоненційно з ростом кількості вершин, тому час виконання алгоритму ACO також зростає експоненційно. Довжина знайденого шляху від першої вершини до останньої варіюється від графіка до графіка. Однак, у середньому, довжина шляху зростає зі збільшенням кількості вершин. Це пов'язано з тим, що з ростом кількості вершин зростає кількість можливих шляхів, і середня довжина шляху серед усіх можливих шляхів зростає.

Ці графіки можуть бути використані для оцінки складності алгоритму ACO. Вони показують, що час виконання алгоритму ACO зростає експоненційно з ростом кількості вершин. Це означає, що алгоритм ACO не може бути використаний для роботи з великими графами. Крім того, ці графіки можуть бути використані для порівняння ефективності алгоритму ACO з іншими алгоритмами пошуку найкоротшого шляху. Наприклад, якщо для графів з невеликою кількістю вершин алгоритм ACO знаходить шляхи з меншою довжиною, ніж інші алгоритми, то це означає, що алгоритм ACO може бути більш ефективним для роботи з такими графами.

Висновок

Мурашині алгоритми є частиною широкого спектру природоінспірованих методів, що використовуються для розв'язання проблем оптимізації та пошуку рішень. Їх успіх у різних галузях свідчить про потужний потенціал використання природних явищ при розробці алгоритмів. Особливу важливість має їхнє застосування в сучасних технологіях, таких як розподілені системи, де вони можуть слугувати для розв'язання задач оптимізації та координації між вузлами мережі. Здатність мурашиних алгоритмів адаптуватися до змін середовища робить їх відмінними для розв'язання завдань у динамічних умовах.

Завершуючи, важливо відзначити, що мурашині алгоритми залишаються об'єктом інтенсивних досліджень та розвитку. Подальші вдосконалення та інтеграція їхніх концепцій у сучасні технології можуть відкрити нові перспективи та підняти їх до рівня передових методів оптимізації.

Отож, ці графіки допомагають нам зрозуміти, як кількість вершин впливає на час виконання алгоритму

ACO та відстань між вершинами. Це важливо для оптимізації алгоритму та покращення його ефективності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Carabaza S.P., Besada E. and Lopez-Orozco J.A., “Ant Colony Optimization for Multi-UAV Minimum Time Search in Uncertain Domains”, Applied Soft Computing, Volume 62, 2018, Pp. 789-806 [Online]. Available: DOI, https://doi.org/10.1016/j.asoc.2017.09.009. [Accessed: 8 Sept. 2022].
2. Shtovba S.D., “Ant Algorithms: Theory and Applications”, Program Comput Soft, Volume 31, 2005, Pp. 167-178 [Online]. Available: DOI, https://doi.org/10.1007/s11086-005-0029-1. [Accessed: 5 Oct. 2022].
3. Kvetny R.N., Kulyk Y.A., Knysh B.P., Ivanov Yu.Yu., Smolars A., Mamyrbaev O. and Burlibayer A., “Modelling the one channel systems of a delivery of goods provided by unmanned aerial vehicles”, INTL Journal of electronics and telecommunications, Volume 2020, No 3, Pp. 487-492 [Online]. Available: DOI, https://doi.org/10.24425/ijet.2020.134003. [Accessed: 16 Sept. 2022].
4. Yaseen M., Razia J. and Rahman Md.T, “Experimental Comparison between Genetic Algorithm and AntColony Optimization on Traveling Salesman Problem”, International Journal of Scientific Research in Science, Engineering and Technology, Volume 8, Issue 1, Pp. 155-162 [Online]. Available: DOI, https://doi.org/10.32628/IJSRSET218135. [Accessed: 10 Sept. 2022].

Царук Вадим Віталійович – студент групи 1ІСТ-20б, факультету інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: tsarukvadik@gmail.com

Демчук Олександр Юрійович – студент групи 1ІСТ-20б, факультету інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: alex.2003.demchuk@gmail.com

Науковий керівник: **Кулик Ярослав Анатолійович** – доцент кафедри Автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця , e-mail: Yaroslav_Kulik@i.ua

Tsaruk Vadym Vitaliyovych – student of group 1IST-20b, faculty of intellectual information technologies and automation, Vinnytsia National Technical University, Vinnytsia, e-mail: tsarukvadik@gmail.com

Demchuk Oleksandr Yuriyovych - student of group 1IST-20b, faculty of intellectual information technologies and automation, Vinnytsia National Technical University, Vinnytsia, e-mail: alex.2003.demchuk@gmail.com

Scientific supervisor: **Kulyk Yaroslav Anatoliyovych** - associate professor of the Department of Automation and Intelligent Information Technologies, Vinnytsia National Technical University, Vinnytsia, e-mail: Yaroslav_Kulik@i.ua