«Системні технології» 2 (139) 2022 «System technologies»
DOI 10.34185/1562-9945-2-139-2022-05
УДК 519.854

Л.Т. Бойко, І.С. Ляшенко

ПОРІВНЯННЯ МУРАШИНОГО АЛГОРИТМУ ОПТИМІЗАЦІЇ ТА ДВОХ ЙОГО МОДИФІКАЦІЙ

Анотація. Алгоритм мурашиної оптимізації є одним із ефективних сучасних алгоритмів для пошуку наближених розв'язків задачі комівояжера та подібних задач пошуку маршрутів на графах. Першу версію цього метаевристичного алгоритму оптимізації запропонував Марко Доріго в 1992 році [1]. Через деякий час в літературі було запропоновано декілька модифікацій цього алгоритму.

Метою дослідження є проведення порівняльного аналізу алгоритму оптимізації мурах (Ant Colony Optimization, ASO) [1] та його найбільш вдалих модифікацій: Ant Colony System (ACS) [2] і Max-Min Ant System (MMAS) [3]. Для цього проведено аналіз системних особливостей обміну інформацією в колонії мурах під час пошуку їжі. Представлено покроковий алгоритм, який моделює природну поведінку мурах-фуражирів у пошуку найкоротшого шляху доставки їжі до мурашника.

Розроблено програмну реалізацію трьох вказаних мурашиних алгоритмів для розв'язання задачі комівояжера. Через вікно інтерфейсу можна ввести кількість міст, кількість мурах, максимальну кількість ітерацій, виконати налаштування кожного алгоритму та вибрати будь-який із трьох алгоритмів. Результатами програми є: візуалізація найкоротшого знайденого маршруту, довжина цього маршруту та найменша кількість ітерацій, за допомогою яких досягається найкоротший маршрут.

Порівняльний аналіз результатів дозволив зробити такі висновки: 1) При вдало вибраних параметрах налаштування алгоритму ітераційні методи, зазвичай, дають результат, близький до оптимального, однак, кількість необхідних для цього ітерацій може істотно відрізнятися. 2) Вивчення проблеми комівояжера за допомогою мурашиних алгоритмів є швидше експериментальним, ніж теоретичним. Результат дуже сильно залежить від параметрів налаштування алгоритму, але теоретичне дослідження цих залежностей залишається актуальним і невирішеним питанням.

Ключові слова: задача комівояжера, оптимізація мурашиним алгоритмом, модифікації мурашиного алгоритму, програмна реалізація алгоритмів.

Вступ. Останні декілька десятиліть науковці все частіше звертаються до природних алгоритмів пошуку оптимальних розв'язків актуальних задач. При-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

рода відпрацювала ці алгоритми протягом мільйонів років в процесі адаптації флори і фауни до навколишнього середовища.

Мурашиний алгоритм оптимізації є одним із ефективних сучасних алгоритмів для знаходження наближених розв'язків задачі комівояжера, а також розв'язування аналогічних задач пошуку маршрутів на графах. Цей алгоритм моделює колективну поведінку мурашиної колонії під час пошуку їжі. Перша версія цього алгоритму метаевристичної оптимізації запропонована Марко Доріго у 1992 році [1]. Через деякий час в літературі було запропоновано декілька модифікацій ACO, серед яких найбільш успішними є: ACS [2] та MMAS [3].

Постановка задачі комівояжера. Дано множину міст, а також відстань між усіма можливими парами цих міст. Необхідно знайти маршрут, який пролягає через усі міста (лише по одному разу), та повертається у початкове місто, при цьому, сумарна довжина пройденого маршруту має бути мінімальною. Розглядаємо симетричний варіант цієї задачі.

Особливості поведінки мурах-фуражирів.

1. Спочатку мурахи в пошуках їжі безсистемно курсують по місцевості.

2. Як тільки якась мураха знайшла джерело їжі, вона бере частинку і несе її в мурашник, відмічаючи свій шлях феромоном.

3. Залишивши в мурашнику принесену їжу, мураха-фуражир повертається до джерела їжі, при цьому зразу знаходить потрібний шлях за запахом феромона. За нею цілеспрямовано, орієнтуючись на запах феромона, йдуть ще декілька мурах-фуражирів. Повертаючись із їжею до мурашника, кожна мураха буде мітити стежку феромоном, роблячи її ще більш привабливою.

4. Феромон має властивість з часом випаровуватися, тому, коли мурахи перенесуть всю їжу в мурашник, вони перестануть залишати феромон на стежці, і стежка поступово втратить свою привабливість для мурах-фуражирів.

5. Мурахи в пошуках нового джерела їжі почнуть знову безсистемно досліджувати місцевість.

Подивимося на мурашину сім'ю, як на систему, і зробимо такі висновки:

- система є *самоорганізованою,* тобто, без спільного центра керування;
- система є розподіленою, тобто, кожна мураха для прийняття рішення використовує лише доступну їй *локальну інформацію* (запах феромона на стежці);
- система успішно розв'язує спільну задачу *самооптимізації,* тобто, за допомогою феромона мурахи (елементи системи) знаходять найкоротший маршрут між джерелом їжі та мурашником;

- система є *стійкою*, тобто, якщо невелика кількість мурах-фуражирів з якихось причин поміняють свій шлях або заблукають, то це не вплине на вирішення загальної задачі.

По суті мурахи демонструють собою реалізацію живого алгоритму для швидкого відшукання найкоротшого шляху між двома точками. Цей алгоритм можна формалізувати і використати для пошуку оптимального маршруту між будь-якою кількістю міст.

Ітераційна схема розв'язування задачі комівояжера.

1. Позначимо загальну кількість міст буквою n, а мурах в колонії – буквою m. Вважаємо значення n, m та розташування міст незмінними на весь час розв'язування задачі. Фіксуємо максимальну кількість ітерацій.

2. Робота алгоритму починається з випадкового розміщення n міст (вершин графа) та випадкового вибору початкового міста, з якого починає свій маршрут окрема мураха. На початковій ітерації кількість феромона на всіх ребрах графа приймаємо за малу константу $\tau_0 \geq 0$.

3. Перехід мурахи з міста i в місто j на t-ій ітерації алгоритму залежить від трьох складових: табу-списку, видимості та віртуального сліду феромона. Табу-список заповнюємо нулями на початку кожної ітерації алгоритму. На протязі здійснення маршруту до табу-списку вносимо перелік тих міст, які вже відвідані мурахою, і в які заборонено заходити ще раз. Позначимо через J_i^k список міст, які ще може відвідати мураха з номером k, що перебуває у місті i. *Видимість* – це величина $\eta_{ij} = 1/d_{ij}$, де d_{ij} — відстань між містами i та j. Видимість базується лише на локальній інформації. Чим ближче міста i та j одне до одного, тим більшою буде видимість η_{ij}. *Віртуальний слід феромона* на ребрі (i, j) – це величина $\tau_{ij}(t)$, яка дорівнює кількості феромона на ребрі (i, j) на *t*-ій ітерації.

4. Ймовірність переходу k-ої мурахи з міста i до міста j на t-ій ітерації позначимо $P_{ij}^k(t)$ і розраховуємо за правилом:

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

$$P_{ij}^k(t)=\begin{cases}\dfrac{\left(\tau_{ij}(t)\right)^{\alpha}\left(\eta_{ij}\right)^{\beta}}{\sum\limits_{l\in J_i^k}\left(\tau_{il}(t)\right)^{\alpha}\left(\eta_{il}\right)^{\beta}}, & якщо\ \ j\in J_i^k;\\ 0, & якщо\ \ j\notin J_i^k.\end{cases}$$

Тут $\tau_{ij}(t)$ – віртуальний слід феромона на ребрі $(i,\ j)$ на t-ій ітерації; η_{ij} – видимість міста j з міста i; α, β – два *регульованих* параметри (параметри налаштування алгоритму), які є вагами інтенсивності сліду феромона та видимості. Чим більшим буде число $P_{ij}^k(t)$, тим більш привабливим буде перехід k-ої мурахи з міста i до міста j на t-ій ітерації.

5. В процесі виконання t-ої ітерації кожна k-та мураха знаходить свій маршрут $T^k(t)$ довжини $L^k(t)$. Вибираємо найкращий варіант $L_{best}(t)=\min\limits_{k=1,m}L^k(t)$. Якщо він кращий за найкоротший маршрут усіх попередніх ітерацій, то вибираємо його за поточний розв'язок задачі.

6. Готуємося до $(t+1)$-ої ітерації. Для цього кожна k-та мураха залишає на ребрі $(i,\ j)$ таку додаткову кількість феромона:

$$\Delta\tau_{ij}^k(t)=\begin{cases}\dfrac{Q}{L^k(t)}, & якщо\ (i,j)\in T^k(t)\\ 0, & якщо\ (i,j)\notin T^k(t)\end{cases},$$

де Q — *регульований* параметр (параметр налаштування алгоритму), значення якого обирають одного порядку з довжиною оптимального маршруту. Знаходимо сумарну додаткову кількість феромона

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^k(t).$$

Випаровування феромона реалізуємо за допомогою *регульованого* параметра $p\in[0;1]$. Правило оновлення феромона стосується всіх ребер і має вигляд

$$\tau_{ij}(t+1)=p\cdot\tau_{ij}(t)+\Delta\tau_{ij}(t).$$

Далі переходимо на п. 3.

Мурашиний алгоритм ACS відрізняється від алгоритму ACO тим, що рівень феромона змінюють при проходженні маршруту кожної мурахи на поточ-

ній ітерації. При глобальному відновленні феромона додавання відбувається лише до ребер, які або глобально кращі, або локально кращі.

В алгоритмі MMAS добавляють обмеження на кількість феромона $(\tau_{\min}, \tau_{\max})$. Феромони залишають лише на глобально кращих маршрутах або кращих в ітерації. Ініціалізуємо всі ребра значенням $\tau_{\max}$.

Комп'ютерні експерименти. Програмна реалізація алгоритмів виконана мовою програмування C# у середовищі Microsoft Visual Studio Community 2019. Програмний продукт є багатопотоковим, тобто, під час виконання розрахунків інтерфейс не блокується, що надає можливість керувати процесом виконання алгоритмів.

За допомогою програми розраховано три приклади трьома, розглянутими вище, алгоритмами. Приклади відрізняються між собою лише кількістю міст та кількістю мурах, які фіксуємо випадково в заданій області для кожного із прикладів, і які в кожному окремому прикладі залишаємо незмінними для всіх алгоритмів. Результати роботи програми виводяться у вікно інтерфейсу одразу ж після закінчення потрібної кількості ітерацій, або після кожної ітерації, якщо так зручно.

У вікні інтерфейсу розташовані такі елементи керування роботою програми (див. рис. 1 – 3; 5 – 7; 9 –11):

- поля для введення кількості мурах та міст;
- поля для введення параметрів алгоритму;
- поле завдання максимальної кількості ітерацій;
- radiobutton для вибору одного з трьох алгоритмів;
- кнопка генерації нових міст;
- кнопки керування потоком виконання програми, а саме: запусканням, призупиненням, зупиненням, відновленням.

Результатами роботи програми є:

- візуалізація найкоротшого по довжині знайденого маршруту за останньою ітерацією (ліва частина вікна інтерфейсу);
- графічна залежність найкоротшого маршруту від номера ітерації (права частина вікна інтерфейсу);
- довжина найкоротшого знайденого маршруту (праворуч знизу);
- найменший номер ітерації, на якій досягається найкоротший маршрут (праворуч знизу).

Далі наведені результати роботи програми для кожного із прикладів.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Приклад 1. Кількість міст – 25; кількість мурах – 5. Результати показані на рис. 1, 2, 3, 4.

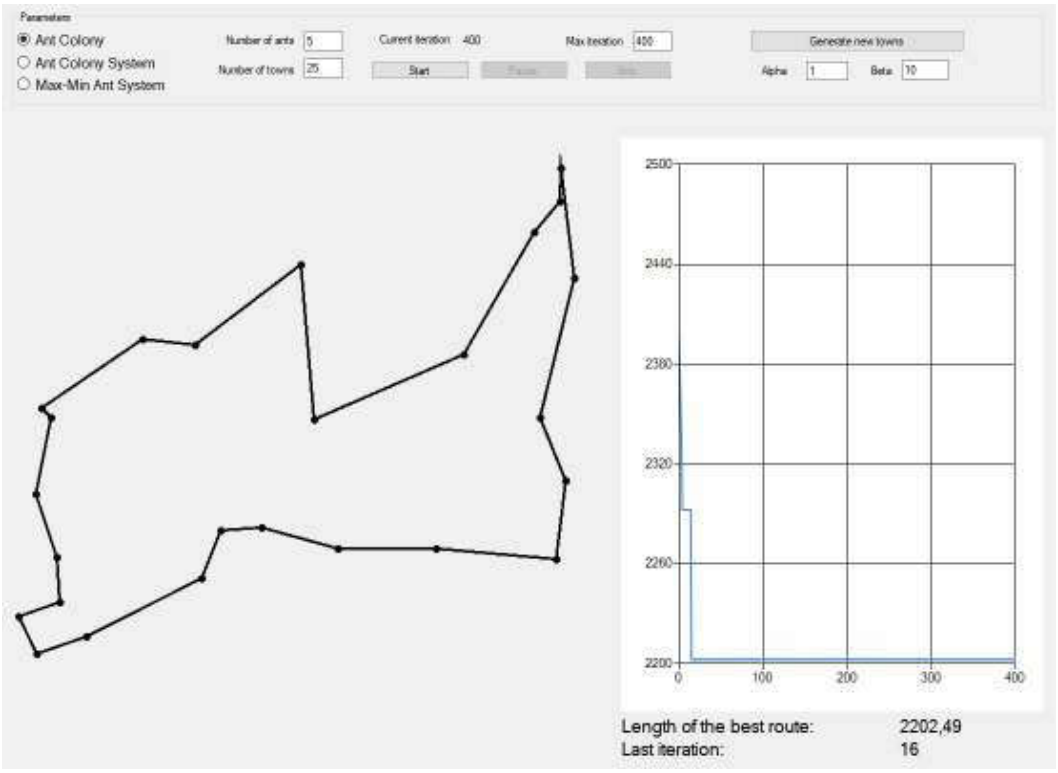

Рисунок 1 – Візуалізація результатів прикладу 1, добутих з використанням AC алгоритму

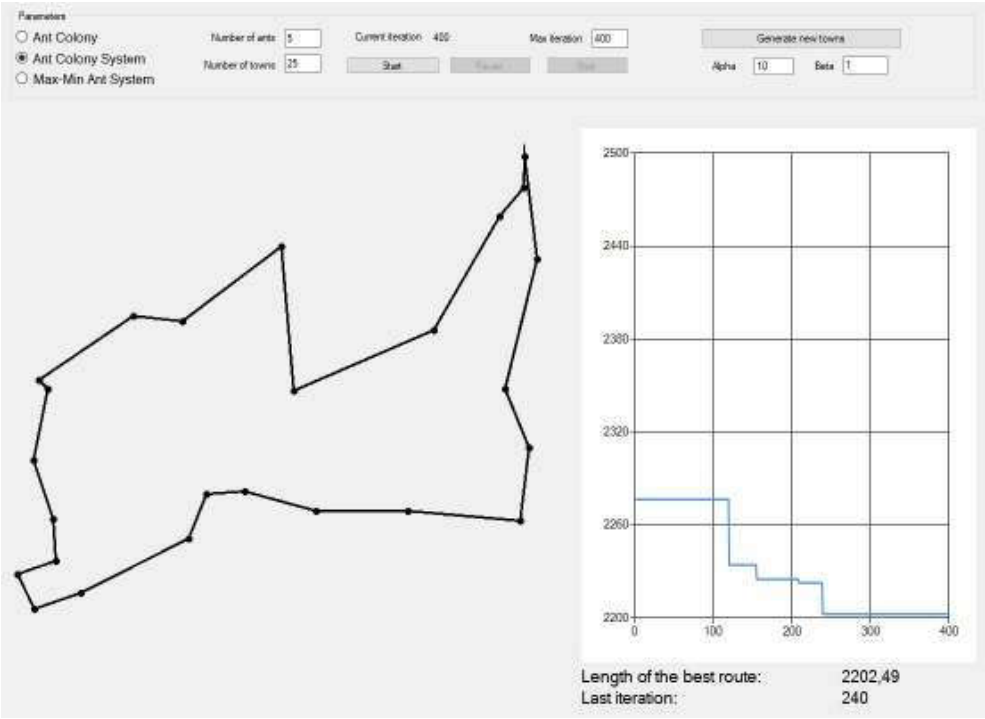

Рисунок 2 – Візуалізація результатів прикладу 1, добутих з використанням ACS алгоритму

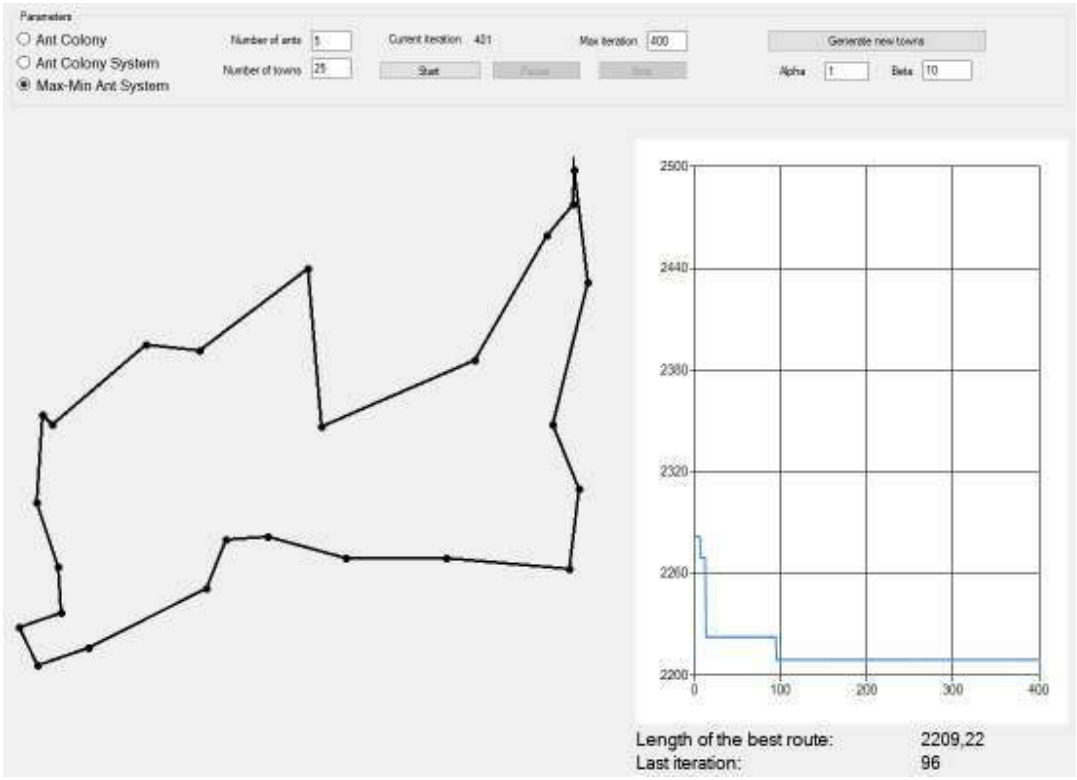

Рисунок 3 – Візуалізація результатів прикладу 1,
добутих з використанням MMAS алгоритму

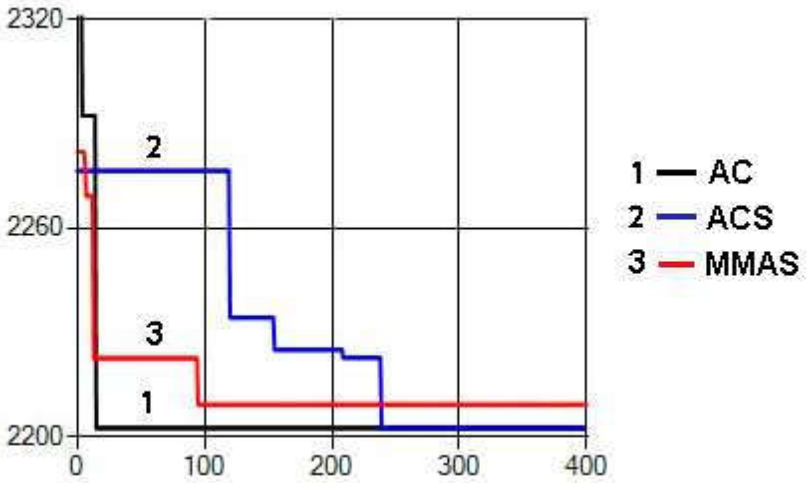

Рисунок 4 – Порівняння результатів прикладу 1,
добутих трьома варіантами мурашиного алгоритму

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Приклад 2. Кількість міст – 35; кількість мурах – 5. Результати роботи програми показані на рисунках 5, 6, 7, 8.

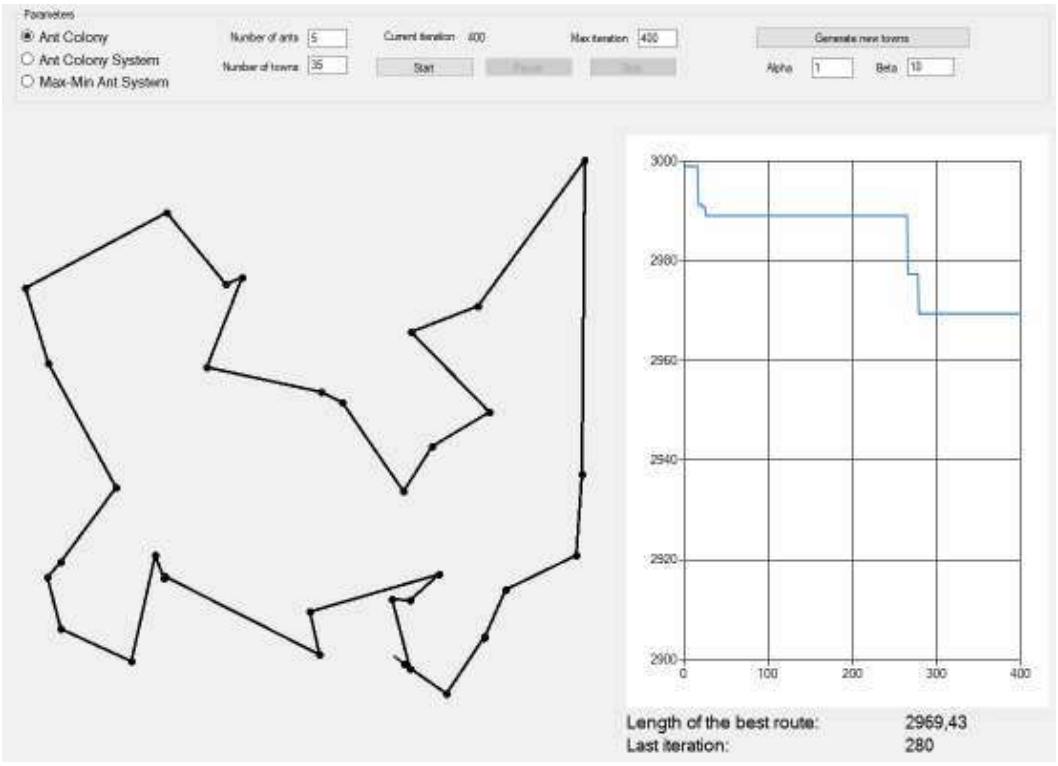

Рисунок 5 – Візуалізація результатів прикладу 2,
добутих з використанням АС алгоритму

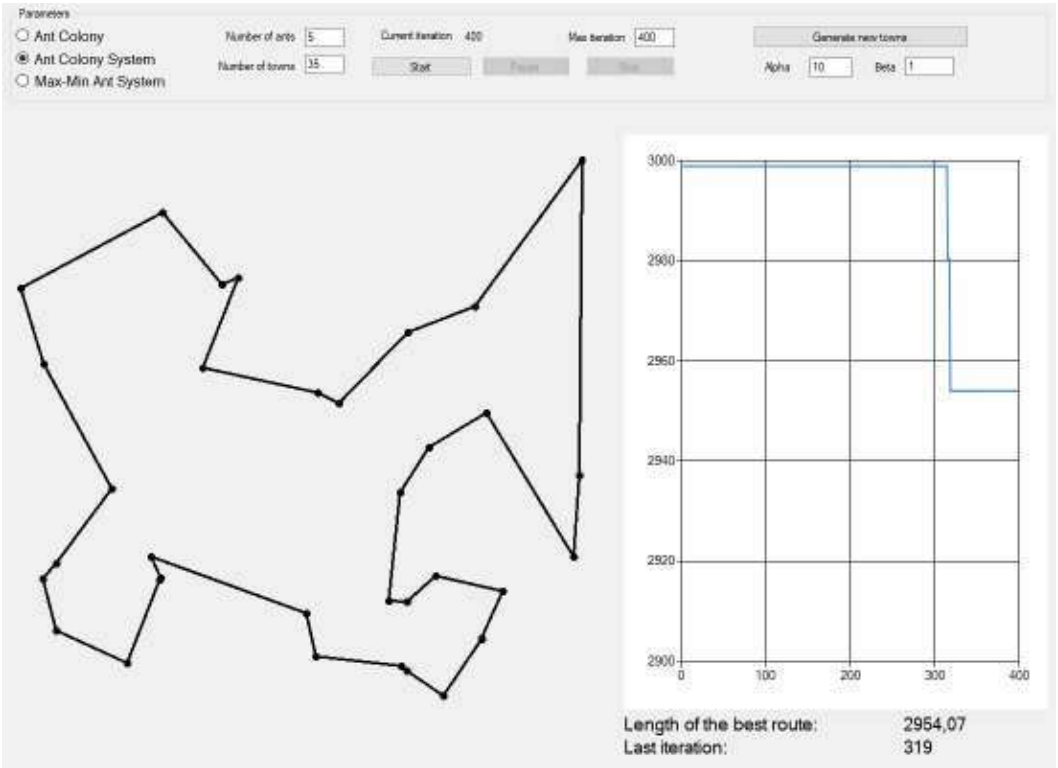

Рисунок 6 – Візуалізація результатів прикладу 2,
добутих з використанням ACS алгоритму

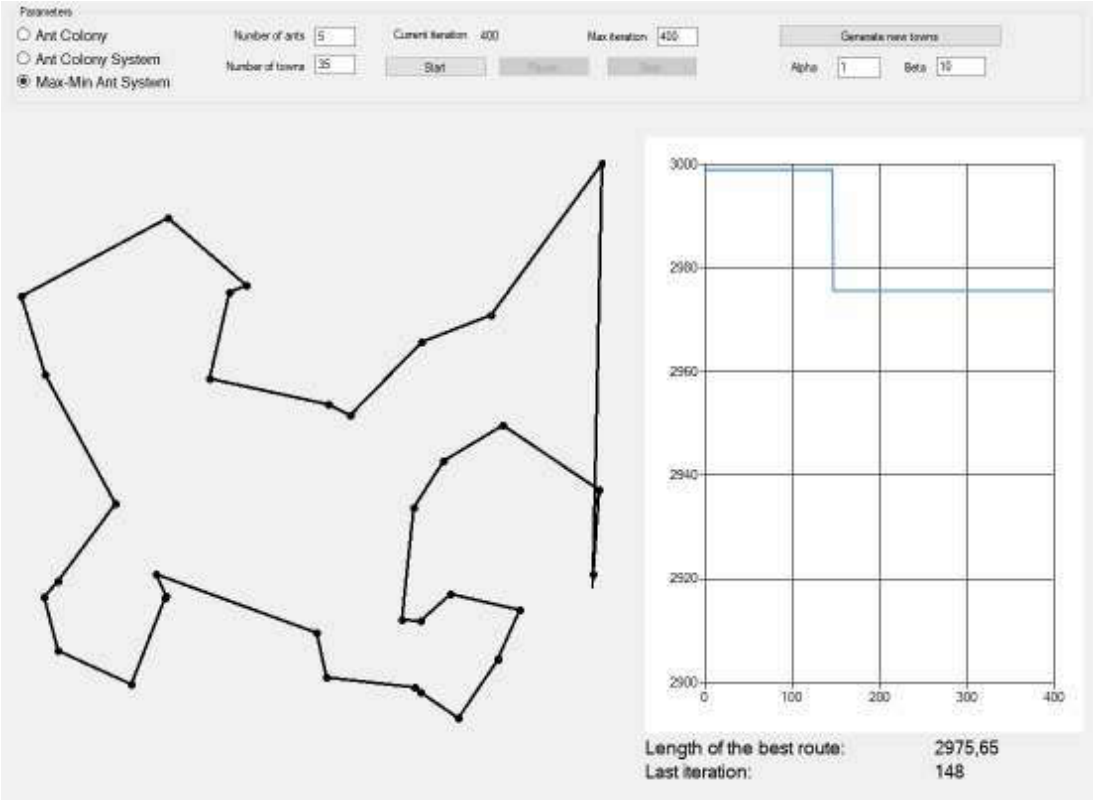

Рисунок 7 – Візуалізація результатів прикладу 2,
добутих з використанням MMAS алгоритму

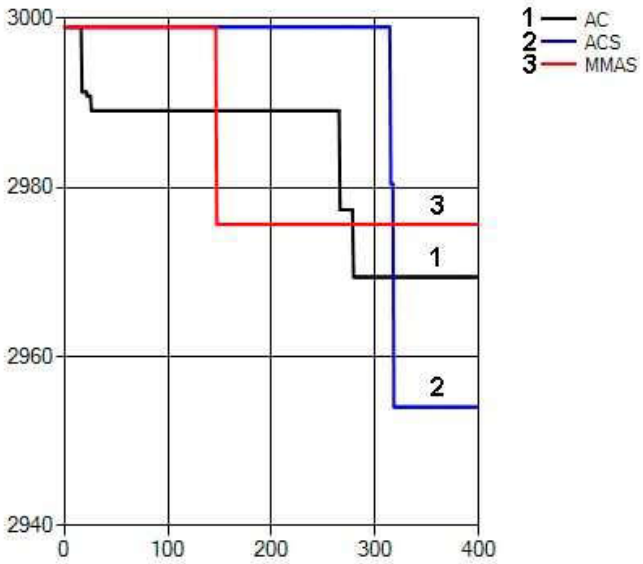

Рисунок 8 – Порівняння результатів прикладу 2,
добутих трьома варіантами мурашиного алгоритму

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Приклад 3. Кількість міст – 45; кількість мурах – 15. Результати показані на рис. 9, 10, 11, 12.

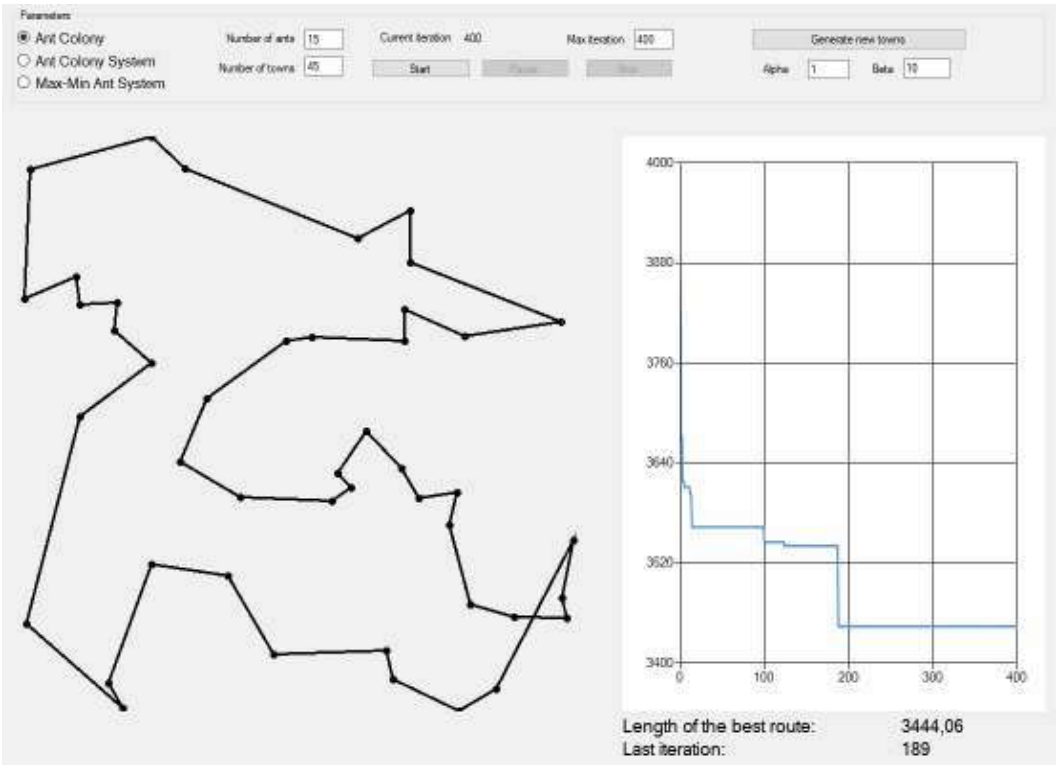

Рисунок 9 – Візуалізація результатів прикладу 3,
добутих з використанням AC алгоритму

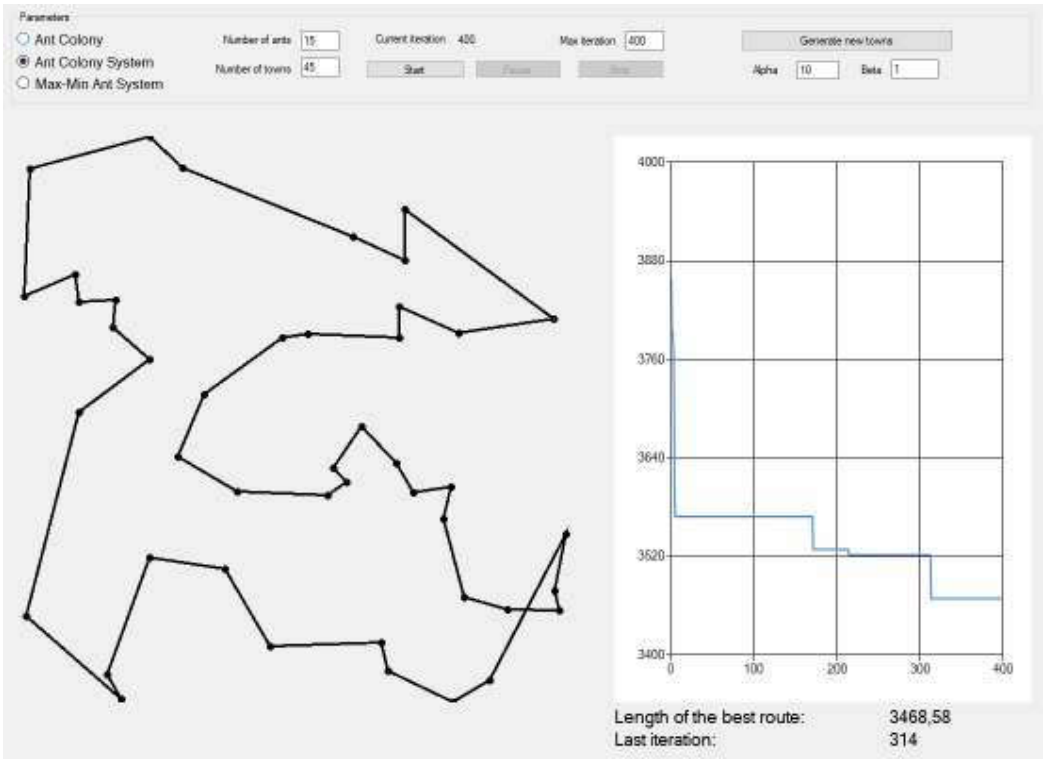

Рисунок 10 – Візуалізація результатів прикладу 3,
добутих з використанням ACS алгоритму

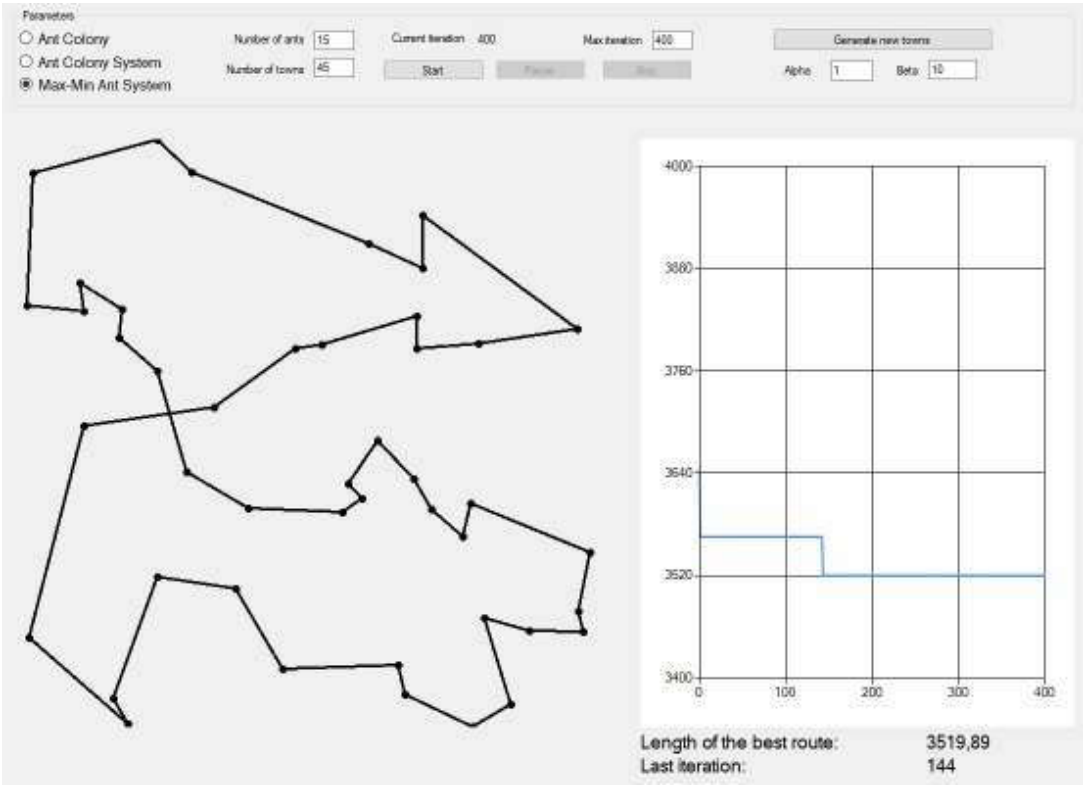

Рисунок 11 – Візуалізація результатів прикладу 3,
добутих з використанням MMAS алгоритму

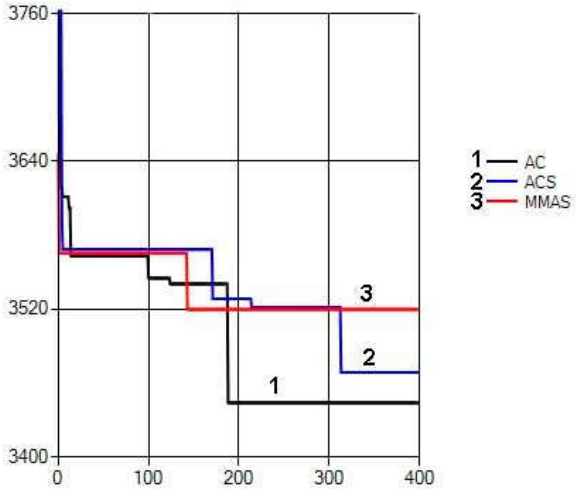

Рисунок 12 – Порівняння результатів прикладу 3, добутих
трьома варіантами мурашиного алгоритму

Аналіз результатів. За результатами розрахунків складено таблицю, в якій наведені довжини найкоротших маршрутів для кожного з алгоритмів на трьох прикладах з різною кількістю міст та мурах. Як видно з таблиці, всі три алгоритми знаходять дуже близькі довжини найкоротших маршрутів, однак, мінімальна кількість потрібних для цього ітерацій може суттєво відрізнятися.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Таблиця 1

Довжини найкоротших маршрутів та необхідна кількість ітерацій

	Приклад №	1	2	3
	Кількість міст	25	35	45
	Кількість мурах	5	5	15
AC алгоритм	Довжина маршруту	2202	2969	3444
	Кількість ітерацій	16	280	189
ACS алгоритм	Довжина маршруту	2202	2954	3468
	Кількість ітерацій	240	319	314
MMAS алгоритм	Довжина маршруту	2209	2975	3519
	Кількість ітерацій	96	148	144

Висновки. Мурашині алгоритми мають позитивну властивість налаштовуватися на конкретний приклад задачі комівояжера. Іншими словами, при вдало вибраних параметрах налаштування алгоритмів α, β, Q, p, τ_0 ітераційні методи, зазвичай, достатньо швидко дають результат, близький до оптимального.

Дослідження задачі комівояжера мурашиними алгоритмами є швидше експериментальним, ніж теоретичним. Результат дуже залежить від параметрів налаштування алгоритмів, але теоретичне дослідження цих залежностей залишається актуальним і поки невирішеним питанням.

REFERENCES

1. Dorigo, M., Stützle, T., Ant Colony Optimization. MIT Press, Cambridge, MA, 2004, 305 p.

2. Dorigo M., Gambardella L. M., Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, v. 1, 1997, p. 53–66.

3. Stützle T., H. H. Hoos., MAX–MIN Ant System. Future Generation Computer Systems, v. 16, 2000, p. 889-914.

Received 09.02.2022.
Accepted 12.02.2022.

Comparison of the ant colony optimization algorithm and its two modifications

The ant optimization algorithm is one of the effective modern algorithms for finding approximate solutions of the salesman problem and similar problems of finding routes on graphs. The first version of this metaheuristic optimization algorithm was proposed by Marco Dorigo in 1992 [1]. After some time, several modifications of this algorithm have been proposed in the literature.

The aim of the study is to conduct a comparative analysis of the ant optimization algorithm (Ant Colony Optimization, ASO) [1] and its most successful modifications: Ant Colony System (ACS) [2] and Max-Min Ant System (MMAS) [3]. To do this, the system features of information exchange in the ant colony during the search for food are analyzed. A step-by-step algorithm that simulates the natural behavior of forage ants in finding the shortest path to deliver food to the anthill is presented.

A software implementation of the three listed ant algorithms for solving the travelling salesman problem has been developed. Through the interface window, you can enter the number of cities, the number of ants, and the maximum number of iterations, fix the settings of the algorithm and select any of the three algorithms. The program randomly locates cities and selects the starting city for each ant. The software product is multi-threaded, i.e. during the calculations the interface is not blocked, which allows you to control the process of program execution, namely: start, pause, stop, resume work. The results of the program are: visualization of the shortest route found, the length of this route and the smallest iteration number, which achieves the shortest route.

Comparative analysis of the results allowed us to draw the following conclusions: 1) With well-chosen algorithm settings, iterative methods usually give a result close to optimal, however, the number of iterations required for this may differ significantly. 2) The study of the travelling salesman problem by ant algorithms is experimental rather than theoretical. The result very much depends on the parameters of the algorithm settings, but the theoretical study of these dependencies remains relevant and unresolved.

Бойко Лідія Трохимівна – кандидат фізико-математичних наук, доцент по кафедрі обчислювальної математики.

Ляшенко Ілля Сергійович – магістр за спеціальністю Прикладна математика, Дніпровський національний університет імені Олеся Гончара.

Boiko Lidiia – Candidate of Physical and Mathematical Sciences, Associate Professor of Computational Mathematics.

Liashenko Illia - Master's degree in Applied Mathematics, Oles Honchar Dnipro National University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)