СИСТЕМНИЙ АНАЛІЗ

УДК 519.8

Л.Ф. ГУЛЯНИЦЬКИЙ
Інститут кібернетики ім. В.М. Глушкова НАН України, Київ, Україна,
e-mail: *leonhul.icyb@gmail.com*.

ДИВЕРСИФІКАЦІЯ ПОШУКУ В АЛГОРИТМАХ ОМК ТА ЇЇ ЗАСТОСУВАННЯ

Анотація. Наведено підхід до розроблення диверсифікованих алгоритмів оптимізації мурашиними колоніями — одного з найпоширеніших методів комбінаторної оптимізації. Диверсифікація в алгоритмах ОМК основана на розгляді варіантів продовження побудови поточного фрагмента розв'язку, що враховують не одну, як зазвичай, а декілька вершин графу задачі, які можуть бути включені в цей маршрут. Можливість перегляду мурахами варіантів пошуку на декілька кроків вперед дає змогу підвищити ймовірність уникнення субоптимальних розв'язків та знаходити більш точні розв'язки. Запропонований підхід використовується для створення метаевристичних алгоритмів розв'язування різних задач комбінаторної оптимізації. Наведено результати проведених обчислювальних експериментів із розв'язування серії прикладних задач комбінаторної оптимізації з різних класів, які підтвердили можливість успішної модифікації відомих мурашиних алгоритмів.

Ключові слова: комбінаторна оптимізація, оптимізація мурашиними колоніями, маршрутизація, задача комівояжера, БпЛА, оптимізація авіаперельотів, обчислювальні експерименти.

ВСТУП

Більшість важливих задач комбінаторної оптимізації (ЗКО) належать класу NP-важких, що робить проблематичним застосування точних алгоритмів, тому поширення набули наближені алгоритми, головним чином, метаевристичні [1–3]. При цьому останнім часом розробники прикладних алгоритмів оптимізації все більше використовують парадигми, що навіяні природою [4–6], чільне місце серед яких посідають концепції штучного, насамперед ройового інтелекту.

Серед відомих алгоритмів ройового інтелекту в комбінаторній оптимізації найбільшого поширення набули алгоритми оптимізації мурашиною колонією (ОМК; ant colony optimization, ACO), запропоновані Марком Доріго [7, 8]. Мурашині алгоритми — це багатоагентні системи, де поведінка кожного агента, який називатимемо штучною мурахою (або далі — мурахою), базується на моделюванні поведінки справжніх мурашок. Мурашині алгоритми застосовують для розв'язання багатьох типів ЗКО, починаючи з класичної задачі комівояжера. Деякі дослідники під ОМК спочатку розуміли певний підклас мурашиних алгоритмів, однак, останнім часом зазначений термін у більшості публікацій використовують стосовно всіх алгоритмів цього типу, отже, далі вживатимемо його так само.

Алгоритми ОМК застосовують для розв'язування задач оптимізації, які можуть бути охарактеризовані загалом так: їхній розв'язок складається з компонентів (складових), з яких можна покроково будувати фрагменти розв'язків, а на завершальному етапі роботи алгоритмів — і повний розв'язок. Наприклад,

у задачі комівояжера, що стала першою задачею, до якої був застосований алгоритм ОМК, такими фрагментами є дуги, що з'єднують міста і з яких формується маршрут — Гамільтонів контур.

Далі пропонується підхід до диверсифікації пошуку в алгоритмах ОМК, оснований на розгляді варіантів продовження побудови фрагмента розв'язку, що враховують не одну, як зазвичай, а декілька вершин, які можуть бути включені в цей маршрут. Зазначений підхід є розвитком ідей, поданих у [9, 10]. Досвід його застосування ілюструється результатами обчислювальних експериментів із розв'язання ЗКО з різних класів.

ЗАГАЛЬНА СХЕМА АЛГОРИТМІВ ОМК

В алгоритмах ОМК формується спеціальна модель задачі, що розв'язується, тому вони належать класу моделеорієнтованих методів [1, 3, 7, 8]. Така модель задачі подається у вигляді певного зваженого графу $G(V, E)$, де $v_i \in V$, $i = 1, \ldots, n$, — вершини, що відповідають компонентам розв'язку, а $e_{ij} \in E$, $e_{ij} = (v_i, v_j)$, $v_i, v_j \in V$, — ребра, які відповідають можливим переходам між відповідними вершинами. Для кожного ребра може бути визначена функція вартості переходу, можливо, залежна від деякого параметра часу t.

Умови задачі, що розв'язується, можуть породжувати набір обмежень $\Pi = \Pi(V, E, t)$ для елементів V та E, які визначають допустимість зв'язків між компонентами та переходами, а в підсумку — і побудованого з них розв'язку.

Розв'язки ЗКО можуть бути подані як допустимі шляхи на графі G. Алгоритми ОМК можна використовувати для знаходження допустимих шляхів мінімальної вартості, що задовольняють обмеження задачі. Вартість як значення цільової функції ЗКО $f(x)$, що відповідає розв'язку x, є функцією всіх вартостей з'єднань, які належать цьому розв'язку.

Мурахою в таких алгоритмах фактично є параметричний рандомізований жадібний алгоритм, який покроково будує з множини компонентів (вершин чи ребер графу задачі) допустимий розв'язок задачі. У своїй роботі він використовує евристичну інформацію та феромонний слід, вони характеризують ребра (рідше — вершини) графу задачі.

Евристична інформація (зазвичай позначається η_{ij}) — це числове значення, що не залежить від знайдених на попередніх кроках розв'язків і відображає ступінь бажаності включення в побудований фрагмент розв'язку того чи іншого нового ребра графу моделі $e_{ij} \in E$. Евристичні значення η_{ij} базуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, яке відрізняється від мурах; ці значення можуть залежати від часу (ітерації) t. Рівень феромону (феромонний слід) τ_{ij}, що відповідає ребру $e_{ij} \in E$, — це додатне число, яке показує, наскільки часто мурахами використовувалося це ребро на попередніх кроках чи під час формування повного розв'язку. Феромонні сліди є для мурах довготривалою пам'яттю щодо всього процесу пошуку. Залежно від вибраного способу формулювання задачі феромонні сліди можуть відповідати всім дугам задачі або тільки деяким з них.

У мурашиних алгоритмах популяція агентів (або мурашок) спільно розв'язує сформульовану задачу оптимізації, використовуючи зазначене подання на графі моделі задачі.

Отже, основними компонентами обчислювальної схеми мурашиних алгоритмів є такі:

— модель задачі, що подається спеціальним графом;
— феромонні значення;
— евристична інформація;
— пам'ять (локальна та глобальна).

Хоча кожна мураха достатньо складна, щоб бути в змозі знайти (імовірно не найкращий) розв'язок цієї задачі, зрештою точніші розв'язки можуть з'явитися тільки внаслідок колективної взаємодії між мурахами.

В алгоритмах ОМК, як зазначалося, кожна штучна мураха — це послідовний жадібний алгоритм, що під час побудови шляху в графі задачі, який описується упорядкованою послідовністю вершин, використовує ймовірнісне правило для вибору чергового компонента розв'язку.

Нехай m — кількість мурах в одному поколінні; кожна мураха k, $k = 1, \ldots, m$, має пам'ять M^k, яку використовує для зберігання інформації про пройдений шлях. Мурахи починають із початкового фрагмента (вершини графу) і рухаються в допустимі сусідні вершини, поступово формуючи розв'язок. Процедура пошуку завершується, якщо для мурахи k виконано принаймні одну з її завершальних умов.

Стани задачі визначаються в термінах скінченних послідовностей $y = (v_{s_1}, v_{s_2}, \ldots), v_{s_r} \in V$, елементів V (або, що рівнозначно, елементів E), які на всіх проміжних кроках мурахи є фрагментами розв'язку задачі оптимізації. Якщо Y — множина всіх можливих послідовностей, то множина $\widetilde{Y}$ усіх (під)послідовностей, які задовольняють обмеження задачі $\Pi = \Pi(V, E, t)$, є підмножиною Y, $\widetilde{Y} \subseteq Y$, а її елементи визначають допустимі стани задачі.

Нехай на певному кроці мураха k побудувала фрагмент розв'язку y, останнім компонентом якого є вершина $i \in V$, тобто вона перебуває в цій вершині, $y = (\ldots, i)$. Тоді мураха може переміститися до будь-якої вершини j із множини допустимих сусідніх вершин N_i^k, визначуваних як $N_i^k = \{j : j \in N_i \wedge (y, i) \in \widetilde{Y}\}$, де N_i — множина всіх сусідніх до i вершин графу задачі. Перехід здійснюється на основі ймовірнісного правила рішення.

Імовірнісне правило рішення для мурах — це функція:

— значень, що зберігаються в локальній для вершини структурі даних — матриці мурашиних маршрутів $A_i = (a_{ij})$, значення елемента якої отримують функціональною композицією локально доступних для вершини значень феромонних слідів і евристичних значень;

— пам'яті мурахи, що зберігає передісторію її дій;

— обмежень задачі.

Під час переміщення з вершини i в сусідню вершину j мураха може змінити феромонний слід τ_{ij} на дузі (i, j). Цей процес називається онлайновим покроковим оновленням феромону. Завершивши формування розв'язку, мураха може пройти той самий шлях назад і поновити феромонні сліди на пройдених дугах. Цей процес має назву онлайнове відстрочене оновлення феромону.

Після того, як мураха, знайшовши розв'язок, повернулася до початкової вершини, вона завершує роботу, звільняючи всі зайняті ресурси.

Крім дій мурашок, ці алгоритми можуть включати ще дві процедури:

1) випаровування феромону — це процес, за допомогою якого інтенсивність феромонного сліду на ребрах графу задачі автоматично зменшується з часом; використовується коефіцієнт випаровування ρ, $\rho \in (0, 1)$. Випаровування феромону потрібне для уникнення дуже швидкої збіжності алгоритму до субоптимальної області. Воно здійснює корисну форму забування, сприяючи дослідженню нових областей у просторі пошуку;

2) дії Демона, що можуть використовуватися для здійснення централізованих дій, які не можуть бути виконані окремими мурахами. Прикладом такої роботи є активація та виконання процедури локальної оптимізації, в якій як по-

чаткове наближення застосовується один чи декілька знайдених мурахами розв'язків, або збирання глобальної інформації, яка може бути потрібною для прийняття рішення щодо корисності відкладання додаткового феромону для відхилення процесу пошуку від зони знайденого локального розв'язку. Зазначимо, що включення процедури дій Демона в алгоритм необов'язкове.

Наприклад, Демон може проглядати шляхи, знайдені кожною мурахою в колонії, і відкладати додатковий феромон на дугах, використаних однією чи декількома мурахами, що знайшли найкоротші шляхи. Оновлення феромону, виконані Демоном, називаються офлайновими оновленнями феромону.

Наведемо загальну обчислювальну схему алгоритмів ОМК [8, 9, 11]:

procedure $ACO(x)$
ініціалізація_алоритму;
while критерій_завершення_не_задоволений **do**
формування_популяції_мурах; {поточне покоління}
foreach мураха_з_популяції **do** {життєвий цикл мурахи}
ініціалізація_мурахи;
M = *оновлення_пам'яті_мурахи*;
while поточний_стан ≠ повний_розв'язок **do**
A = *локальна_ матриця_мурашиних_маршрутів*;
сформувати_множину допустимих_вершин;
p = *обчислити_ймовірність_переходів* (A, M, П);
наступний_стан = *правило_прийняття_рішення* (p, П);
перейти_в_наступний_стан (наступний_стан);
if онлайнове_покрокове_оновлення_феромону **then**
відкласти_феромон_на_відвіданій_дузі;
поновити_матрицю_мурашиних_маршрутів_A;
endif
M = *оновити_внутрішній_стан*;
endwhile
if онлайнове_відстрочене_оновлення_феромону **then**
foreach відвіданої_дуги_побудованого_розв'язку **do**
відкласти_феромон_на_відвіданій_дузі;
оновити_матрицю_мурашиних_маршрутів_A;
endforeach
endif
завершити діяльність;
endforeach {завершення життєвого циклу мурахи}
випаровування_феромону;
оновлення_рекорду (x);
дії_Демона; {необов'язково}
endwhile
end

На етапі ініціалізації здійснюються початкові налаштування: вибір значень параметрів алгоритму (зокрема, кількості мурах m у колонії), задання початкових значень феромонного сліду. Після утворення нового покоління мурах їх розміщують довільно у вершинах V графу G, які є початковими фрагментами маршрутів.

Як зазначалося раніше, на кожному кроці конкретна мураха досліджує найближчих сусідів тієї вершини графу задачі, якою закінчується побудований нею

на поточний момент фрагмент маршруту. Отже, для кожної мурахи k із популяції здійснюється:

— формування підмножини допустимих вершин $N_i^k \subseteq N_i$ із множини вершин, сусідніх для тієї вершини $i \in V$, яка є останньою в поточному фрагменті маршруту;

— обчислення ймовірності p_{ij}^k переходу від вершини i до довільної допустимої вершини $j \in N_i^k$ як функції від значень феромону τ_{ij} на ребрі $(i, j) \in E$ та евристичної інформації η_{ij};

— імовірнісний вибір чергової допустимої вершини $s \in N_i^k$, включення її в кінець фрагмента маршруту, що будується;

— модифікація значень τ_{ij}, якщо вибрано онлайнове оновлення.

Коли всі мурахи завершують роботу, може здійснюватися модифікація феромонних значень для певних дуг графу G (офлайнове оновлення), а також випаровування феромону з коефіцієнтом ρ (зниження значень τ_{ij}) для всіх його ребер, якщо такі дії не поєднані з оновленням феромону. Оскільки алгоритм не має релаксаційного характеру, то потрібне запам'ятовування найкращого із знайдених розв'язків (рекорду).

Найчастіше використовуються такі критерії завершення обчислень у мурашиних алгоритмах:

— відсутність (або невелика зміна) поліпшення наявного рекорду чи середньої пристосованості популяції мурашок на кількох останніх ітераціях;

— перевищення заданої кількості $H_{\max}$ ітерацій (змін поколінь / популяцій), де $H_{\max}$ — параметр алгоритму;

— вичерпання заданого часу на обчислення.

Найважливішими аспектами мурашиних алгоритмів є: спосіб оновлення феромону, ймовірності переходів до наступної вершини та правила завершення роботи алгоритму [7, 8].

Зазначимо, що найбільш поширеними варіантами обчислення ймовірностей p_{ij}^k під час переходу мурахи k від поточної вершини $i \in V$ до нової вершини $j \subset N_i^k$ у типових мурашиних алгоритмах є мультиплікативна

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum\limits_{r \in N_i^k} [\tau_{ir}(t)]^\alpha [\eta_{ir}(t)]^\beta} \tag{1}$$

чи адитивна форма

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha + [\eta_{ij}(t)]^\beta}{\sum\limits_{r \in N_i^k} \{[\tau_{ir}(t)]^\alpha + [\eta_{ir}(t)]^\beta\}}, \tag{2}$$

де α, β — параметри алгоритму, які задають ступінь значущості феромонного сліду та евристичної інформації ($\alpha, \beta > 0$), а параметр t відображає можливу зміну значень відповідних параметрів залежно від зміни поколінь.

НОВА МОДИФІКАЦІЯ ПОШУКУ В АЛГОРИТМАХ ОМК

Пропонується підхід для формування маршруту мурах, за яким вони на кожному кроці використовують інформацію не тільки з одного ребра, що може бути включене у фрагмент розв'язку, а також й інформацію з більшої кількості можливих ребер. Отже, на кожній ітерації мураха може додати до шляху одразу декілька вершин або, інакше кажучи, може зробити як один, так і декілька «кроків» [9, 10]. Розглянемо детальніше варіант, коли враховується можливість здійснення одного чи двох кроків.

Для одного кроку використовують стандартні формули (1) чи (2). А для двокрокової версії модифікованого алгоритму ОМК для кожної мурахи k за наявності уже побудованого фрагмента розв'язку $y = (\dots, i)$ формується множина ще невідвіданих ребер (i, s), (s, j), $s \in N_i^k$, $j \in N_s^k$, де N_s^k — множина допустимих для мурахи k вершин графу задачі за умови, що до існуючого фрагмента розв'язку додана вершина s, $y^+ = (\dots, i, s)$, та обчислюється ймовірність переходу від вершини i до вершини j через вершину s з урахуванням евристичної інформації та поточних значень феромону. Для двокрокового алгоритму ОМК перехід k-ї мурахи з вершини i в j через вершину s на поточній ітерації t здійснюється з ймовірністю, що може обчислюватися за такою формулою (мультиплікативна форма):

$$p_{ij}^k = \frac{[\tau_{is}(t) \cdot \tau_{sj}(t)]^\alpha [\eta_{is}(t) \cdot \eta_{sj}(t)]^\beta}{\sum\limits_{r \in N_{irj}^k} [\tau_{ir}(t) \cdot \tau_{rj}(t)]^\alpha [\eta_{ir}(t) \cdot \eta_{rj}(t)]^\beta},$$

де $N_{irj}^k = \{r : l \in N_i^k, j \in N_r^k\}$.

Якщо допустити переходи і на один, і на два кроки, то потрібно ці ймовірності пронормувати так, щоб у сумі була одиниця. Тому ймовірність переходів на один крок (від вершини i до вершини j, $j \in N_i^k$) та на два кроки (від вершини i до іншої вершини j через вершину r, $r \in N_i^k$, $j \in N_r^k \setminus i$) буде визначатися так:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum\limits_{r \in N_i^k} [\tau_{ir}(t)]^\alpha [\eta_{ir}(t)]^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{ir}(t) \cdot \tau_{rj}(t)]^\alpha [\eta_{ir}(t) \cdot \eta_{rj}(t)]^\beta}, & \text{якщо } j \in N_i^k, \\ \dfrac{[\tau_{is}(t) \cdot \tau_{sj}(t)]^\alpha [\eta_{is}(t) \cdot \eta_{sj}(t)]^\beta}{\sum\limits_{r \in N_i^k} [\tau_{ir}(t)]^\alpha [\eta_{ir}(t)]^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{ir}(t) \cdot \tau_{rj}(t)]^\alpha [\eta_{ir}(t) \cdot \eta_{rj}(t)]^\beta}, & \text{якщо } j \in N_r^k. \end{cases} \tag{3}$$

У разі базування на адитивній формулі (2) така ймовірність буде мати вигляд:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha + [\eta_{ij}(t)]^\beta}{\sum\limits_{r \in N_i^k} [\tau_{ir}(t)]^\alpha [\eta_{ir}(t)]^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{ir}(t) + \tau_{rj}(t)]^\alpha [\eta_{ir}(t) + \eta_{rj}(t)]^\beta}, & \text{якщо } j \in N_i^k, \\ \dfrac{[\tau_{is}(t) + \tau_{sj}(t)]^\alpha [\eta_{is}(t) + \eta_{sj}(t)]^\beta}{\sum\limits_{r \in N_i^k} \tau_{ir}(t)^\alpha \eta_{ir}(t)^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{ir}(t) + \tau_{rj}(t)]^\alpha [\eta_{ir}(t) + \eta_{rj}(t)]^\beta}, & \text{якщо } j \in N_r^k. \end{cases} \tag{4}$$

Експериментально було підтверджено, що мультиплікативний варіант обчислення ймовірностей за формулою (3) під час розв'язування практичних ЗКО показує кращі результати, ніж за (4). Це можна пояснити тим, що у разі використання адитивного варіанта сумарна ймовірність двокрокових переходів значно перевищує сумарну ймовірність однокрокових, а тому майже завжди здійснюються двокрокові переходи. Використання лише двокрокових переходів робить алгоритм більш жадібним, що забезпечує лише миттєву вигоду, тоді як загалом результат може виявитися несприятливим.

Одночасна можливість як однокрокових, так і двокрокових переходів за один такт дає змогу отримувати кращий розв'язок і при цьому уникати локальних оптимумів.

Аналогічно виводяться формули для обчислення ймовірностей переходів на три кроки; використання алгоритмів з більш ніж трьома переходами потребує суттєвих обчислювальних затрат.

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО ПІДХОДУ ДО ДИВЕРСИФІКАЦІЇ

Зауважимо, що в галузі прикладної комбінаторної оптимізації вкрай рідко вдається отримувати теоретичні результати, застосовні на практиці, тому для порівняння ефективності алгоритмів використовують обчислювальні експерименти. Зокрема, апробація більшості нових алгоритмів комбінаторної оптимізації проводиться розв'язуванням низки задач комівояжера, сформованих на базі використання реальних даних із відомої бібліотеки TSPLIB [12].

Всюди далі розглядаються задачі пошуку мінімуму додатної цільової функції.

На основі запропонованого підходу до диверсифікації пошуку розроблено низку алгоритмів розв'язання ЗКО із різних класів, більшість з яких базується на схемі MMAS (Max-Min Ant System) [13] — однієї із найефективніших схем обчислень в алгоритмах ОМК [9]. Розглянемо деякі результати їхнього застосування для розв'язання прикладних задач, у яких похибка розраховувалася за формулою

$$\varepsilon = \frac{f - f^{\,\mathrm{opt}}}{f^{\,\mathrm{opt}}} \times 100\ (\%), \tag{5}$$

де f — значення цільової функції задачі у знайденому відповідним алгоритмом розв'язку, $f^{\,\mathrm{opt}}$ — її значення у точці оптимального розв'язку (якщо він відомий) чи у найкращому із знайдених розв'язків.

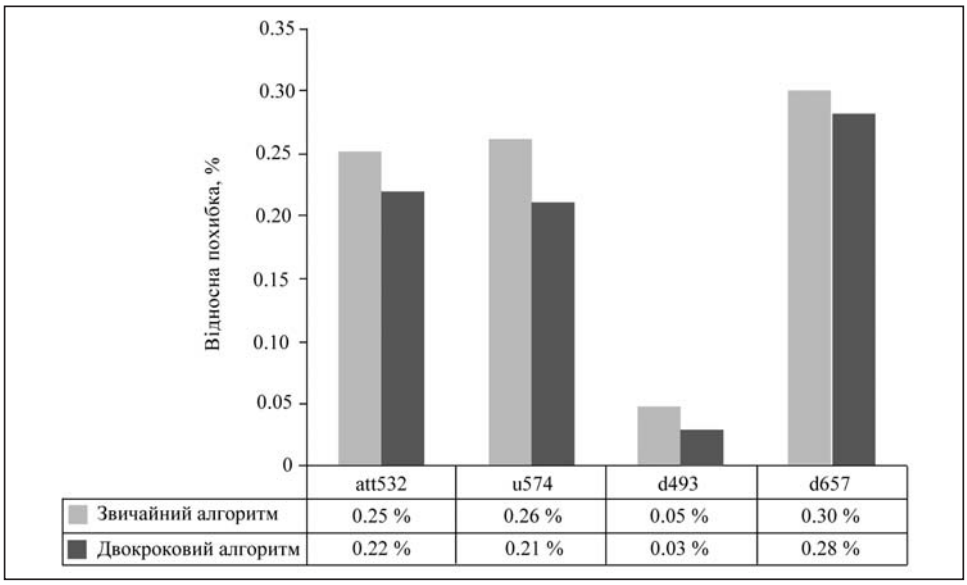
Задача комівояжера [3, 11, 12, 14, 15]. Для порівняння ефективності запропонованої модифікації алгоритму ОМК із бібліотеки TSPLIB було вибрано низку задач комівояжера з відомими оптимальними розв'язками [12]. Кожна задача була розв'язана стандартним та модифікованим алгоритмами MMAS 20 разів, порівнювалися середні значення відносної похибки (відхилення довжини знайденого маршруту від довжини оптимального). У формулі (5) $f^{\,\mathrm{opt}}$ — довжина оптимального розв'язку із бібліотеки TSPLIB.

Для розв'язування задач застосовані такі параметри: $\alpha = 1$; $\beta = 2$; $\rho = 0.5$. Умовою завершення роботи алгоритму було виконання 2000 ітерацій, а кількість мурах дорівнює кількості міст у задачі.

Результати обчислювального експерименту наведено на рис. 1, де показано середню відносну похибку стандартного («звичайного») і модифікованого («двокрокового») алгоритмів MMAS та розмірність задач, які розв'язувалися.

Відповідно до результатів обчислювального експерименту використання двокрокової версії дало змогу збільшити точність результатів на 0.02–0.05 % залежно від задачі. Таким чином, експериментальні дані підтверджують доцільність застосування запропонованої модифікації алгоритмів ОМК у випадках, коли точність отриманих результатів важливіша за швидкість знаходження розв'язку і навіть найменше її підвищення дає суттєвий ефект.

Планування логістичних місій групи БпЛА за наявності альтернативних депо. Розглянуто проблеми планування логістичних місій на основі оптимізації маршрутів групи БпЛА, що діють як команда, перед якою стоїть за-



	att532	u574	d493	d657
Звичайний алгоритм	0.25 %	0.26 %	0.05 %	0.30 %
Двокроковий алгоритм	0.22 %	0.21 %	0.03 %	0.28 %

Рис. 1. Порівняння відносної похибки стандартного та модифікованого алгоритму MMAS

вдання відвідати задану множину цілей (клієнтів, об'єктів), допускаючи можливості старту та приземлення БпЛА в альтернативних депо та враховуючи його обмежений польотний ресурс, з мінімізацією сумарної довжини маршрутів чи тривалості польотів за умов, що кожна ціль відвідується одним і лише одним БпЛА. При цьому можливе використання заданої множини чи її частини потенціальних місць для базування депо для досягнення компромісу між максимізацією загальної вигоди та мінімізацією споживання ресурсів за наявності додаткових обмежень. Увага акцентується на ситуаціях, коли вантажопідйомність БпЛА не є обмежувальним чинником [16, 17].

З огляду на практичне планування подібних місій із залученням БпЛА трьома ключовими проблемами, які потрібно розв'язати під час спільного планування місій команди БпЛА, є розподіл цілей по депо, оптимізація маршрутів та вибір платформ для базування (депо) [18], які у низці наведених у публікаціях підходів породжують окремі задачі оптимізації [19, 20]. На противагу цьому запропоновано підхід, що дає змогу об'єднати всі ці три задачі в одну задачу комбінаторної оптимізації [16, 17].

У проведеному обчислювальному експерименті три задачі було сформовано на реальних геоданих, а дві — з використанням відомих задач із бібліотеки TSPLIB.

Задача 1: одна база і три цілі, один БпЛА має запас ходу, достатній для відвідування лише однієї цілі за один виліт.

Задача 2: три депо, 15 цілей та три БпЛА, запасу ходу жодного з них не вистачає на повний обліт усіх цілей без заміни блоків живлення.

Задача 3: чотири депо, 23 цілі та три БпЛА, запасу ходу жодного з них не вистачає на повний обліт усіх цілей без заміни блоків живлення, можна візуально виділити три кластери цілей.

Задача 4: модифікація задачі att48 з TSPLIB, одна з вершин визначена як депо, 47 цілей, одне депо та один БпЛА.

Задача 5: модифікація задачі berlin52 з TSPLIB, одна з вершин визначена як депо, 51 ціль, одне депо та один БпЛА.

Для кожної задачі, окрім першої, було виконано попередній підбір параметрів мурашиного алгоритму за допомогою пришвидшених запусків з меншою кількістю ітерацій. З вибраними параметрами виконана зазначена кількість запусків з різними ініціалізаторами генератора псевдовипадкових чисел. Було реалізовано також алгоритм методу вектора спаду — одного із алгоритмів детермінованого локального пошуку [21]. Для нього як початковий розв'язок вибирався результат роботи жадібного алгоритму та покладалося $\rho = 1$ [16]; в усіх випадках алгоритм запускався лише один раз. Час роботи кожного алгоритму в усіх запусках обмежено 20 с.

Для обчислення відносної похибки ε у (5) f^{opt} — це найменше із відомих значень цільової функції для відповідної задачі.

Обчислення проводилися на персональному комп'ютері Macbook Pro з такими параметрами: Macbook Pro 16-inch 2019, процесор 2.6 GHz 6-Core Intel Core i7. Графічна підтримка: AMD Radeon Pro 5300M 4 GB, Intel UHD Graphics 630 1536 MB. Операційна система Mac OS: 13.2.1 (22D68), RAM: 16 GB 2667 MHz DDR4.

Найважливіші результати обчислювального експерименту із розв'язування згаданих задач наведено у табл. 1 [16].

Розглянуто також дещо іншу проблему пошуку оптимальних маршрутів для групи БпЛА з альтернативними депо з таким уточненням: прийом БпЛА може здійснюватися в декількох територіально заданих зонах, характеристики яких вибираються так, що у конкретній зоні можливе приймання не більше заданої кількості БпЛА; у такій зоні можуть завершувати свій маршрут всі чи частина БпЛА.

Задача полягала у визначенні як маршрутів БпЛА з мінімізацією сумарної довжини (тривалості польотів) та їхньої задіяної кількості, так і місць (зон) їхнього приймання. Ця постановка є близькою до проблеми планування місії групи БпЛА, які транспортуються літаком [18] (про подібні задачі див. далі).

У [22, 23] для підвищення ефективності генетичних алгоритмів запропоновано так звану острівну модель. Її суть полягає у розгляді не однієї, а відразу декількох популяцій, які через певну кількість поколінь обмінюються отриманими розв'язками для корекції подальшого пошуку. Ця ідея була використана для модифікації алгоритмів ОМК у двох напрямах: введення до розгляду декількох колоній та застосування процедури диверсифікації пошуку (двокрокового алгоритму).

На початку острівного алгоритму на островах незалежно відбувається ініціалізація мурашиних колоній, а далі починає роботу алгоритм MMAS. Єдина відмінність у тому, що через задану кількість ітерацій відбувається обмін інформацією про «хороші» розв'язки між островами; називатимемо цей процес міграцією.

Таблиця 1. Зведені результати дослідження ефективності алгоритмів

Номер задачі	Стандартний алгоритм ОМК			Диверсифікований алгоритм ОМК			Метод вектора спаду		
	f	ε	t	f	ε	t	f	ε	t
1	22.6741	0	15	22.6741	0	15	22.6741	0	0.01
2	113.809	3	15	109.738	0	15	134.853	10	0.01
3	253.098	0.5	15	252.021	0	15	280.777	7.7	0.01
4	10628	0	15	10628	0	19	11523	8.4	0.01
5	7542	0	15	7542	0	18	8127	7.7	0.01

Важливу роль у роботі острівної моделі відіграє топологія островів, тобто те, як вони між собою зв'язані. Лише взаємозв'язані острови можуть обмінюватися інформацією. Оскільки алгоритм MMAS зберігає лише найкращий глобальний розв'язок, то саме він буде відігравати роль мігранта, тобто буде відправлений на «сусідні» острови під час процесу міграції. Нехай K_i — множина сусідніх островів для острова i. Тоді мігрант для острова i визначається так:

$$x_i = \arg\min \{f(x_r) : r \in K_i\},$$

де x_r — найкращий розв'язок, знайдений на острові r.

Після потрапляння мігранта на острів відбувається відкладання феромону на цьому розв'язку, за умови що розв'язок-мігрант є кращим за поточний найкращий глобальний розв'язок на острові.

Робота острівного алгоритму продовжується доти, поки є хоч один острів, на якому кількість зроблених ітерацій без покращення менше заданої.

Було проведено аналіз чотирьох топологій: повна, кільце, ґратка та гіперкуб. Повна топологія: кожен острів зв'язаний з кожним, а приклади решти топологій подано на рис. 2.

Оскільки у відкритому доступі немає даних для цієї задачі, то був згенерований набір задач, які мають розмірність від 10 до 70 об'єктів, що необхідно відвідати, у середньому 4–7 місць старту та 2–5 зон приймання.

План експерименту включав порівняння реалізацій детермінованого локального пошуку — алгоритму табу-пошуку TabuSearch [24], стандартного алгоритму MMAS [11, 13], а також запропонованого острівного диверсифікованого алгоритму MMAS, який використовує різні топології і має назву IMMAS.

Після застосування модуля автоматичного налаштування параметрів отримано такі значення: $\alpha = 1.57$, $\beta = 0.55$, $\rho = 0.85$.

Було проведемо 10 запусків зазначених алгоритмів для семи задач різної розмірності із згенерованого набору, пошук розв'язків здійснювався на персональному комп'ютері під керуванням Windows із 16 Гб оперативної пам'яті та восьмиядерним процесором з тактовою частотою 3.6 ГГц.

У табл. 2 для кожного алгоритму розв'язування задач наведено отримані його найкращі результати у серії із 10 запусків, причому в алгоритмі IMMAS використовувалася повна топологія для зв'язків між островами. Тут n — кількість об'єктів для обстеження, f — розв'язок задачі, отриманий алгоритмом, t — час виконання алгоритму (у наносекундах).

Наведемо середні значення ε для найкращих значень отриманих розв'язків кожним алгоритмом за серію запусків: TabuSearch — 12.06, MMAS — 4.08, IMMAS — 0.42.

Далі розглянемо середні значення отриманих розв'язків для кожного алгоритму (табл. 3).

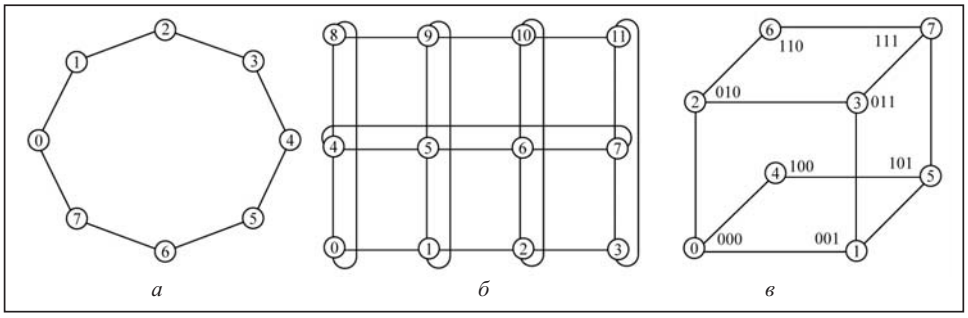


Рис. 2. Топології: кільце (*а*), ґратка (*б*) та гіперкуб (*в*)

Таблиця 2. Найкращі значення, отримані алгоритмами

Номер задачі	*n*	TabuSearch			MMAS			IMMAS		
		f	ε	*t*	*f*	ε	*t*	*f*	ε	*t*
1	16	3657	26.9	2.3E6	**2880**	0	7.0E9	**2880**	0	8.9E9
2	30	3906	6.6	2.6E6	3770	2.9	4.5E10	**3662**	0	5.5E10
3	36	4962	16.6	6.8E6	4395	3.3	7.8E10	**4253**	0	8.6E10
4	43	4858	10.5	7.4E7	4529	3	1.5E11	**4396**	0	1.7E11
5	57	7035	13.7	1.4E7	6361	2.8	2.9E11	**6185**	0	3.2E11
6	65	8414	10.1	1.5E7	7925	3.7	8.5E11	**7641**	0	9.8E11
7	70	**7912**	0	1.5E7	8940	12.9	8.7E11	8179	3	1E12

Таблиця 3. Середні значення отриманих розв'язків

Номер задачі	*n*	TabuSearch			MMAS			IMMAS		
		f	ε	*t*	*f*	ε	*t*	*f*	ε	*t*
1	16	3657	25.9	3.9E6	2949	1.6	6.0E9	**2903**	0	8.8E9
2	30	3906	2.8	2.6E6	3881	2.2	4.5E10	**3797**	0	5.5E10
3	36	4962	12.9	6.2E6	4601	4.7	7.8E10	**4393**	0	8.8E10
4	43	4858	5.4	4.3E7	4922	6.8	1.5E11	**4606**	0	1.7E11
5	57	7035	8.6	1.4E7	6774	4.6	2.9E11	**6472**	0	3.3E11
6	65	8414	5	1.5E7	8424	5.1	8.5E11	**8010**	0	9.8E11
7	70	**7912**	0	2E7	9084	14.8	8.7E11	8472	7	1E12

Наведемо усереднені значення ε для середніх значень розв'язків кожним алгоритмом за серію запусків: TabuSearch — 8.65, MMAS — 5.68, IMMAS — 1.

Як випливає із наведених результатів, диверсифікований алгоритм IMMAS із застосуванням острівної моделі показує найкращі результати як у знаходженні абсолютно найкращого розв'язку, так і серед усереднених.

Оскільки у наведеному порівнянні використовувався алгоритм IMMAS з повною топологією, проведено серію експериментів для визначення, яка з топологій дає змогу отримувати найкращі результати.

Аналогічно спочатку розглянемо найкращі отримані значення за використання різних топологій (табл. 4).

Далі розглянемо середні значення отриманих розв'язків за використання різних топологій (табл. 5).

Якщо брати до уваги середні значення, то яскраво вираженого лідера немає, найкраще себе показує кільцева топологія, а найгірше — топологія гіперкуб. Якщо брати до уваги найкращий отриманий розв'язок, то кільцева топологія виграє з великим відривом, гіперкуб при цьому, як і в попередньому випадку, показує найгірші результати. Час виконання для всіх топологій майже однаковий, тому можна не брати його до уваги під час порівняння.

Отже, на основі цих експериментів можна дійти висновку:

— серед наведених трьох алгоритмів, які використані в експериментах, беззаперечним лідером є запропонований острівний диверсифікований алгоритм IMMAS;

— топологія острівної моделі має вплив на точність отримуваного розв'язку.

Таблиця 4. Найкращі результати розв'язку задач за різних топологій

Номер задачі	*n*	Complete		Hypercube		Ring		Torus	
		f	*t*	*f*	*t*	*f*	*t*	*f*	*t*
1	16	**2880**	8.9E9	**2880**	9.0E9	**2880**	8.5E9	**2880**	8.3E9
2	30	3662	5.5E10	3646	5.0E10	**3617**	5.1E10	**3617**	5.3E10
3	36	4253	8.6E10	4379	9.0E10	**4184**	8.9E10	4237	8.6E10
4	43	4396	1.7E11	4447	1.7E11	**4236**	1.7E11	4537	1.7E11
5	57	6185	3.2E11	6277	3.3E11	**6077**	3.4E11	6147	3.3E11
6	65	**7641**	9.8E11	7734	9.8E11	7898	9.7E11	7872	9.7E11
7	70	**8179**	1E12	8204	1.0E12	8243	1.0E12	8189	1.1E12

Таблиця 5. Середні значення отриманих розв'язків за різних топологій

Номер задачі	*n*	Complete		Hypercube		Ring		Torus	
		f	*t*	*f*	*t*	*f*	*t*	*f*	*t*
1	16	2903	8.8E9	2886	8.8E9	**2883**	8.6E9	**2883**	8.7E9
2	30	3797	5.5E10	**3731**	5.2E10	3761	5.2E10	3736	5.2E10
3	36	4393	8.8E10	4492	8.8E10	**4391**	8.8E10	4421	8.7E10
4	43	4606	1.7E11	4679	1.7E11	**4551**	1.7E11	4670	1.7E11
5	57	6472	3.3E11	6466	3.3E11	6417	3.3E11	**6329**	3.3E11
6	65	**8010**	9.8E11	8057	9.9E11	8132	9.8E11	8057	9.8E11
7	70	**8472**	1E12	8568	1E12	8589	1E12	8537	1E12

Найкраще себе зарекомендувала топологія кільце, але при цьому вона поступилася повній топології на двох останніх задачах найбільшої розмірності. Отже, доходимо висновку, що кожна топологія має свої переваги і недоліки, тому потрібно підбирати її індивідуально під задачу.

Пошук оптимального шляху мандрівника в мережі авіаперельотів. Задача полягає у пошуку оптимального за вартістю шляху в транспортній мережі із заданими користувацькими умовами: початковий і кінцевий пункти, часове вікно і тривалість подорожі, проміжні пункти, кількість трансферів. Розширена версія задачі може включати обов'язкове відвідування проміжних пунктів у межах заданих часових інтервалів, тип авіаліній, загальну кількість пересадок і транзитний час.

Часовий інтервал, в якому потрібно побудувати маршрут подорожі, задається найранішим і найпізнішим часом відправлення з початкової вершини, а також найранішим і найпізнішим часом прибуття в кінцеву вершину. Розглянута задача полягає у знаходженні такого допустимого маршруту, який мінімізує сумарну вартість подорожі [25].

Складність задачі зумовлена додатковими обмеженнями за часом (перельоти мають відбуватися згідно з графіком), тоді як цільова функція залежить від змінної вартості квитків на рейс.

Важливою складовою експериментів із дослідження ефективності алгоритмів комбінаторної оптимізації є розв'язування тестових задач, але найбільшу зацікавленість викликає аналіз результатів розв'язання задач з реальними даними.

На початку досліджень [26] результати, отримані розробленим алгоритмом ОМК без диверсифікації пошуку, порівнювались із відповідними розв'язками, знайденими відомим точним алгоритмом міток [27], тобто з глобальними оптимумами. Дані для обчислень (розклад авіаперельотів) було отримано за допомогою API пошукових клієнтів SkyScanner [28] і QpxExpress [29] по аеропортах Європи за перший тиждень жовтня 2017 р. Час був дискретизований за днями для спрощення.

Результати проведених досліджень проілюстрували, що для розкладу малих розмірів алгоритм міток працює швидше, ніж ОМК, проте для розкладів з кількістю елементів більше ніж 400 час роботи ОМК збільшується несуттєво, на відміну від алгоритму міток. Для розкладу з 746 рейсами алгоритм міток у середньому працює півгодини, тоді як ОМК — одну хвилину. При цьому обидва знайшли точні розв'язки. Наявний тренд зростання часу роботи показав, що для розв'язування задач більшої розмірності в реальному часі алгоритм міток є незадовільним.

Для подальших досліджень було здійснено розроблення та реалізацію диверсифікованого алгоритму ОМК з двома та трьома кроками [25].

Для вибору значень основних параметрів алгоритму попередньо застосовувався блок їхньої оптимізації, внаслідок отримано такі значення: $\alpha = 0.1$, $\beta = 2$, $\rho = 0.1$. Кількість мурах становила 100. Алгоритм завершував обчислення, коли покращення рекорду не відбувалось протягом 10 останніх поколінь.

Експерименти проводились для графу з 21144 дугами, що відображають авіарейси в Європі та кілька маршрутів до США. Дослідження показали значний рівень збільшення часу виконання в процесі додавання кожного кроку. Для маршрутів у межах Європи, де вартість авіаквитків була порівняно однаковою для різних напрямків, зростання кількості кроків не дало очікуваного результату, попри збільшення часу виконання. Кожний крок виконання зумовлював значне розширення простору пошуку відносно рівноцінними маршрутами. Однак для складніших маршрутів (Європа–США), де не кожен аеропорт має пряме сполучення з цільовими аеропортами, багатокроковий алгоритм показав набагато кращі результати.

Обчислювальні експерименти проводились на персональному комп'ютері з процесором Intel® Core (TM) i7-4790 CPU @ 3.60GHz, RAM 32.0 GB, 64-бітною операційною системою Windows 10, x64 процесором.

У табл. 6 наведено результати експериментів з дослідження ефективності диверсифікованого алгоритму ОМК з кроками $K = 1, 2, 3$, тобто для $K = 1$ це була реалізація, наближена до стандартного алгоритму ОМК. Тут t — середній час роботи (мс), ε — середня похибка (%), f_* — значення для кращого із знайдених розв'язків.

Таблиця 6. Порівняльні результати застосування алгоритмів

Маршрут	Класичний алгоритм			Двокроковий алгоритм			Трикроковий алгоритм		
	t	ε	f_*	t	ε	f_*	t	ε	f_*
Київ–Портланд	6437	35	410	23396	27	385	620585	22	382
Київ–Сіетл	3672	24	478	16408	23	478	561565	22	311
Київ–Вашингтон	11108	50	352	21987.57	20	385	730514	17	347
Київ–Роттердам	3270	0.22	147	18.96996	1.69	152	619892	3	152
Київ–Лісабон	2742	3.3	141	16293	0.8	137	577225	2	137

Зазначимо, що експерименти порівняння ефективності алгоритмів для $K = 1$ і $K = 2$ за умови збільшення кількості мурах для однокрокового варіанта були проведені у такий спосіб, щоб час виконання був однаковим. Для складніших маршрутів (Київ–Вашингтон), однокроковий алгоритм показав похибку в 25 % і середній час виконання 18812 с, тобто якщо продовжувати пошук однокроковим алгоритмом протягом такого самого часу, як у випадку застосування двокрокового алгоритму, то середня похибка на 5 % перевищує похибку, що виникає в разі застосування останнього.

ВИСНОВКИ

У багатьох прикладних задачах комбінаторної оптимізації навіть незначне покращення точності має дуже важливе значення, що і зумовлює потребу в розробці алгоритмів, які дають змогу досягти цього. Запропонований підхід дає змогу розширити сферу пошуку в алгоритмах ОМК, створюючи умови для знаходження покращених розв'язків шляхом уникнення передчасної збіжності — однієї із найгостріших проблем для більшості прикладних алгоритмів комбінаторної оптимізації.

Можливість перегляду мурахами варіантів пошуку на декілька кроків вперед дає змогу підвищити ймовірність уникнення субоптимальних розв'язків у багатьох задачах, а отже, отримувати більш якісний підсумковий розв'язок. Ця стратегія може бути використана у будь-якій модифікації мурашиного алгоритму, що робить її універсальним засобом покращення ефективності алгоритмів ОМК, створюючи умови для диверсифікації пошуку.

Перспективним є розроблення алгоритмів ОМК з диверсифікованим пошуком для розв'язування й інших класів ЗКО, насамперед актуальних нині задач маршрутизації транспортних засобів з використанням БпЛА [30–32] або задач евклідової комбінаторної оптимізації [33].

Напрямами подальших досліджень може стати порівняльний аналіз практичної ефективності алгоритмів, розроблених на основі запропонованого підходу, та інших відомих прикладних алгоритмів комбінаторної оптимізації. Перспективним також є використання модифікованих алгоритмів ОМК, як представників моделеорієнтованих методів у гібридних чи кооперативних метаевристиках [34–38] або у технології автоматичного проєктування метаевристик [39].

СПИСОК ЛІТЕРАТУРИ

1. Gendreau M., Potvin J.Y. (Eds.). Handbook of metaheuristics. International Series in Operations Research & Management Science. Vol. 272. Cham: Springer, 2019.
2. Stegherr H., Heider M., Hähner J. Classifying metaheuristics: Towards a unified multi-level classification system. *Natural Computing*. 2022. Vol. 21, N 2. P. 155–171. https://doi.org/10.1007/s11047-020-09824-0.
3. Sergienko I.V., Hulianytskyi L.F., Sirenko S.I. Classification of applied methods of combinatorial optimization. *Cybernetics and Systems Analysis*. 2009. Vol. 45, N 5. P. 732–741. https://doi.org/10.1007/s10559-009-9134-0.
4. Pintea C. Advances in bio-inspired computing for combinatorial optimization problems. Springer, 2014. 188 p.
5. Yang X.S. Nature-inspired optimization algorithms. 2nd ed. Academic Press, 2021. 290 p.
6. Kumar A., Nadeem M., Banka H. Nature inspired optimization algorithms: A comprehensive overview. *Evolving Systems*. 2023. Vol. 14, N 1. P. 141–156.
7. Dorigo M., Stützle T. Ant colony optimization. Cambridge (MA): MIT Press, 2004. 348 p. https://doi.org/10.7551/mitpress/1290.001.0001.

8. Dorigo M., Stützle T. Ant colony optimization: Overview and recent advances. Handbook of metaheuristics. Cham: Springer, 2019. P. 311–352. https://doi.org/10.1007/978-3-319-91086-4_10.

9. Гуляницький Л.Ф. Диверсифікація пошуку в алгоритмах ОМК. *Abstracts of Int. Conf «Problems of Decision Making under Uncertainties (PDMU–2011)»* (Sept. 19–23, 2011, Yalta, Ukraine). Київ, 2011. P.66–67.

10. Гуляницький Л.Ф. Диверсифікація пошуку в алгоритмах оптимізації мурашиними колоніями. *Теорія оптимальних рішень*. 2017. С. 47–57.

11. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. Київ: Видавничо-поліграфічний центр «Київський університет», 2016. 142 с.

12. Reinelt G. TSPLIB 95. Technical report. Universität Heidelberg, 1995. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf (Accessed 12 Jun 2024).

13. Stützle T., Hoos H.H. MAX-MIN ant system. *Future Gen. Comput. Systems*. 2000. Vol. 16, N 8. P. 889–914.

14. Applegate D.L., Bixby R.E., Chvatál V., Cook W.J. The traveling salesman problem: A computational study. Princeton Series in Applied Mathematics. Princeton University Press, 2006. 606 p.

15. Toaza B., Esztergár-Kiss D. A review of metaheuristic algorithms for solving TSP-based scheduling optimization problems. *Applied Soft Computing*. 2023. Vol. 148, 110908. https://doi.org/10.1016/j.asoc.2023.110908.

16. Гуляницький Л.Ф., Рибальченко О.В. Формалізація проблеми оптимізації місць базування та маршрутів групи БПЛА. *Кібернетика та комп'ютерні технології*. 2021. Вип. 4. С. 12–26. https://doi.org/10.34229/2707-451X.21.4.2.

17. Hulianytskyi L., Rybalchenko O. Optimization of decisions when planning a UAV group mission with alternative depots. *Selected Papers of the III International Scientific Symposium "Intelligent Solutions" (IntSol-2023). Symposium Proc.* (Sept. 27-28, 2023, Kyiv, Ukraine). *CEUR Workshop Proceedings.* 2023. Vol. 3538. P. 245–256. https://ceur-ws.org/Vol-3538/Paper_22.pdf.

18. Horbulin V.P., Hulianytskyi L.F., Sergienko I.V. Optimization of UAV team routes in the presence of alternative and dynamic depots. *Cybernetics and Systems Analysis*. 2020. Vol. 56, N 2. P. 195–203. https://doi.org/10.1007/s10559-020-00235-8.

19. Kuo RJ, Lu SH, Lai PY, Mara STW. Vehicle routing problem with drones considering time windows. *Expert Systems with Applications.* 2022. Vol. 191, 116264. https://doi.org/10.1016/j.eswa.2021.116264.

20. Li J., Xiong Y., She, J. UAV path planning for target coverage task in dynamic environment. *IEEE Internet of Things Journal.* 2023. Vol. 10, Iss. 20. P. 17734–17745. https://doi.org/10.1109/jiot.2023.3277850.

21. Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации. Киев.: Наук. думка, 1998. 472 с.

22. Lässig J., Sudholt D. The benefit of migration in parallel evolutionary algorithms. *Proc. of the 12th Annual Conference on Genetic and Evolutionary Computation*. New York: ACM, 2010. P.1105–1112.

23. Lässig J., Sudholt D. Experimental supplements to the theoretical analysis of migration in the island model. *Intern. Conf. on Parallel Problem Solving from Nature*. Berlin; Heidelberg: Springer, 2010. P. 224–233. https://doi.org/10.1007/978-3-642-15844-5_23.

24. Martí R., Martínez-Gavara A., Glover F. Tabu search. In: Discrete Diversity and Dispersion Maximization. Martí R., Martínez-Gavara A. (Eds.). Springer Optimization and Its Applications. Cham: Springer, 2023. Vol. 204. P. 137–149. https://doi.org/10.1007/978-3-031-38310-6_7.

25. Hulianytskyi L., Pavlenko A. Ant colony optimization algorithms with diversified search in the problem of optimization of airtravel itinerary. *Cybernetics and Systems Analysis.* 2019. Vol. 55, N 6. P. 978–987. https://doi.org/10.1007/s10559-019-00208-6.

26. Гуляницький Л.Ф., Павленко А.І. Оптимізація шляхів у динамічному графі перельотів модифікованим алгоритмом мурашиних систем. *Математичне моделювання в економіці.* 2018. № 2. С. 26–39.

27. Zografos K.G., Androutsopoulos K.N. Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems*. 2008. Vol. 9, N 1. P. 175–184.

28. Travel APIs. URL: https://partners.skyscanner.net/affiliates/travel-apis.

29. QPX Express API. URL: https://developers.google.com/qpx-express.

30. Poikonen S., Golden B. Multi-visit drone routing problem. *Computers & Operations Research.* 2020. Vol. 113, 104802. https://doi.org/10.1016/j.cor.2019.104802.

31. Horbulin V.P., Hulianytskyi L.F. Sergienko I.V. Planning of logistics missions of the "UAV+vehicle" hybrid systems. *Cybernetics and Systems Analysis*. 2023. Vol. 59, N 5. P. 733–742. https://doi.org/10.1007/s10559-023-00609-8.

32. Гуляницький Л.Ф., Рибальченко О.В. Оптимізація маршрутів при плануванні місій гібридних транспортних систем "Дрон+Транспортний засіб". C*ybernetics and Computer Technologies.* 2023. Iss. 3. C. 44–58. https://doi.org/10.34229/2707-451X.23.3.4.

33. Stoyan Y.G., Yakovlev S.V. Theory and methods of euclidian combinatorial optimization: current status and prospects. *Cybernetics and Systems Analysis.* 2020. Vol. 56, N 3. P. 366–379. https://doi.org/10.1007/s10559-020-00253-6.

34. Talbi E.G. Metaheuristics: from design to implementation. Hoboken, N.J.: John Wiley & Sons, Inc., 2009. 593 p.

35. Raidl G.R., Puchinger J., Blum C. Metaheuristic hybrids. In: Handbook of Metaheuristics. Gendreau M., Potvin J.Y. (Eds.). International Series in Operations Research & Management Science. 2019. Vol. 272. P. 385–417.

36. Tezel B.T., Mert A. A cooperative system for metaheuristic algorithms. *Expert Systems with Applications*. 2021. Vol. 165, 113976. https://doi.org/10.1016/j.eswa.2020.113976.

37. Hulianytskyi L.F., Sirenko S.I. Hybrid metaheuristic combining ant colony optimization and H-method. Swarm intelligence. *Proc. 7th International Conference ANTS 2010* (Brussels, Belgium, Sept. 8–10, 2010). M. Dorigo et. al. (Eds.). *Lecture Notes in Computer Science.* Berlin; Heidelberg: Springer-Verlag, 2010. Vol. 6234. P. 568–569. https://doi.org/10.1007/978-3-642-15461-4_64.

38. Hulianytskyi L.F., Sirenko S.I. Cooperative model-based metaheuristics. *Electronic Notes in Discrete Mathematics*. 2010. Vol. 36. P. 33–40. https://doi.org/10.1016/j.endm.2010.05.005.

39. Martín-Santamaría R., López-Ibáñez M., Stützle T., Colmenar J.M. On the automatic generation of metaheuristic algorithms for combinatorial optimization problems. *European Journal of Operational Research.* 2024. Vol. 318, Iss. 3. P. 740–751. https://doi.org/10.1016/j.ejor.2024.06.001.

L.F. Hulianytskyi

SEARCH DIVERSIFICATION IN ACO ALGORITHMS AND ITS APPLICATION

Abstract. This paper presents a novel approach to the development of diversified ant colony optimization (ACO) algorithms, which are among the most widely used methods in combinatorial optimization. The proposed diversification in ACO algorithms is based on considering multiple options for extending the current solution fragment by incorporating several vertices of the problem graph into the route instead of just one, as is typically done. The ability of ants to foresee multiple search steps ahead increases the likelihood of avoiding suboptimal solutions and finding more accurate ones. This approach is applied to create metaheuristic algorithms for solving various combinatorial optimization problems. The results of computational experiments on a series of applied combinatorial optimization problems across different classes demonstrate the successful modification of the known ACO algorithms.

Keywords: ccombinatorial optimization, Ant Colony Optimization, routing, traveling salesman problem, UAV, optimization of flight routes, computational experiment.

Надійшла до редакції 19.07.2024