УДК 519.8

Л.Ф. ГУЛЯНИЦЬКИЙ, А.І. ПАВЛЕНКО

АЛГОРИТМИ ОПТИМІЗАЦІЇ МУРАШИНИМИ КОЛОНІЯМИ З ДИВЕРСИФІКОВАНИМ ПОШУКОМ У ЗАДАЧІ ОПТИМІЗАЦІЇ АВІАПЕРЕЛЬОТІВ

Анотація. Сформульовано задачу пошуку оптимального шляху мандрівника в мережі авіаперельотів, яка враховує вартість побудованого маршруту та наявність користувацьких умов у випадку залежної від часу вартості сполучень. Запропоновано алгоритми систем мурашиних колоній для розв'язування залежної від часу задачі, поданої розширеним графом перельотів, які, на відміну від наявних мурашиних алгоритмів, враховують динамічність мережі (залежність наявності і вартості сполучення від часу) та користувацькі умови. Вдосконалено підхід до диверсифікації пошуку в мурашиних алгоритмах в умовах залежності від часу для щільного графу, що дало змогу підвищити якість побудованих маршрутів, які сполучають різні регіони. Ефективність запропонованих алгоритмів досліджено шляхом аналізу результатів обчислювального експерименту, виконаного з використанням реальних даних.

Ключові слова: оптимізація маршрутів, мережа авіаперельотів, мурашині алгоритми, диверсифікація пошуку, розширені мережі за часом, динамічне виявлення дискретизації.

ВСТУП

Планування маршрутів у транспортних системах набуває зростаючої популярності, шо стимулює розроблення нових і модифікацію наявних алгоритмів пошуку найкоротших шляхів [1]. Алгоритми мурашиних колоній застосовуються для дослідження способів маршрутизації через їхню мультиагентну і розподілену природу, а також здатність до гнучкої адаптації у разі зміни даних задачі. Протягом останніх 20–25 років розробники мурашиних алгоритмів продемонстрували їхнє успішне застосування в робототехніці, у розв'язуванні задач дослідження операцій і в галузі телекомунікацій [2, 3]. Мурашині алгоритми, запропоновані Марком Доріго [2] — це багатоагентні системи, де поведінка кожного агента (мурахи) ґрунтується на поведінці справжніх комах. У цій роботі запропоновано новий алгоритм оптимізації мурашиними колоніями (ОМК) із диверсифікацією пошуку шляхом розгляду варіантів продовження побудови фрагменту розв'язку, що враховують не одну, як зазвичай, а декілька вершин, які можуть бути включені в цей маршрут. Розроблення цього алгоритму є розвитком ідей, поданих у [4].

ПОСТАНОВКА ЗАДАЧІ

Задача полягає у пошуку оптимального за вартістю шляху в транспортній мережі із заданими користувацькими умовами: початковий і кінцевий пункт, 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. Інколи задачу пропонують розв'язувати за критеріями мінімальної тривалості подорожі (які включають час очікування) або мінімального транзитного часу (часу, проведеного у транспортних засобах) [5].

Мережу транзитних пунктів і сполучень подамо як направлений мультиграф $G = (V, A)$, який описує залежні від часу сполучення між містами. Нехай V — множина n вершин, що відповідають транспортним вузлам (аеропорти, порти,

залізничні зупинки тощо), A — множина m сполучень між транспортними вузлами. Кожна дуга $a_{ij}(t) \in A$ позначає наявність сполучення між пунктами i та j з відправленням у момент часу t.

Перехід дугою $a_{ij}(t) \in A$ в момент часу t спричиняє деяку часову затримку $c_{ij}(t)$ і матеріальні витрати $l_{ij}(t)$. Іншими словами, маємо $C: A \to \mathbb{R}^+ \cup \{0\}$ — функцію, яка визначає невід'ємне значення тривалості сполучення $c_{ij}(t)$ для кожної дуги $a_{ij}(t) \in A$. Аналогічно $L: A \to \mathbb{R}^+ \cup \{0\}$ — функція, яка визначає невід'ємне значення вартості сполучення $l_{ij}(t)$ для кожної дуги $a_{ij}(t) \in A$. Для спрощення час дискретизується (за годинами, днями тощо). У роботі вважається, що трійка (i, j, t) в єдиний спосіб задає рейс (дугу) $a_{ij}(t)$. Якщо в задачі фігурують кілька видів транспорту, то пропонується розглянути дві моделі подання мереж: множину графів мереж (кожен граф подає деякий вид транспорту, при цьому вершини можуть дублюватись у кількох графах) та граф множини мереж (задається кожна дуга графу $a_{ij}^r(t) \in A$, де r — вид транспорту).

Можливість простою в проміжних пунктах можна подати як додаткові вершини очікування, проте недоліком цього подання є значне зростання графу залежно від інтервалу подорожі та кількості часових інтервалів, у межах яких будується граф. Перевагою такого подання є спрощення адаптації базових алгоритмів на графах. Тому запропоновано використовувати залежні від часу функції, а не розширення графу за часом. Нехай $\zeta(i_u, i_{u+1}, t_u)$ — функція, що описує час простою в пункті i_u перед подорожжю сполученням (i_u, i_{u+1}, t_u), а $\delta(i_u, i_{u+1}, t_u)$ — вартість простою в пункті i_u перед подорожжю сполученням (i_u, i_{u+1}, t_u). У випадку побудови туристичного маршруту умови обов'язкових для відвідування пунктів можуть включати часові інтервали для відвідування кожного пункту; мінімальний $\zeta_{\min}(i_u)$ і максимальний $\zeta_{\max}(i_u)$ час простою; межі вартості простою (мінімальне допустиме значення вартості більш актуальне для багатокритерійних задач, коли вагу намагаються максимізувати): $\zeta_{\min}(i_u) \le \zeta(i_u, i_{u+1}, t_u) \le \zeta_{\max}(i_u)$, $t_{u_{\min}} \le \overline{t_u} \le t_{u_{\max}}$, $\delta_{\min}(i_u) \le \delta(i_u, i_{u+1}, t_u) \le \delta_{\max}(i_u)$.

За відсутності обов'язкових для відвідування проміжних пунктів і від'ємних значень ваг дуг очевидно, що в оптимальному маршруті через кожну вершину можна пройти не більше одного разу, тому умова проходження через кожну проміжну вершину для задачі без обов'язкових проміжних пунктів має такий вигляд:

$$0 \le \sum_{i \ne j} x_{ij} \le 1, \tag{1}$$

де $x_{ij} = 1$, якщо рейс з пункту i в пункт j належить y, інакше $x_{ij} = 0$, $i, j \in V$.

Окрім початкового s і кінцевого d пунктів користувач може задавати проміжні обов'язкові для відвідування пункти $v_{\text{mandatory}} \in V_{\text{mandatory}} \subset V$, що впливає на кількість входжень вершин до маршруту. Тоді умови проходження через проміжні вершини мають такий вигляд:

$$\sum_{i \in V} x_{si} \ge 1, \ \sum_{i \in V} x_{id} \ge 1, \ \sum_{\substack{i \in V \setminus \{j\} \\ j \in V_{\text{mandatory}} \setminus \{i\}}} x_{ij} \ge 1, \ \sum_{\substack{i \in V_{\text{mandatory}} \setminus \{j\} \\ j \in V \setminus \{i\}}} x_{ij} \ge 1. \tag{2}$$

Якщо користувач задає заборонений для відвідування пункт з множини $V_{\text{prohibited}}$, то

$$\sum_{\substack{i \in V \setminus \{j\} \\ j \in V_{\text{prohibited}}}} x_{ij} = 0, \ \sum_{\substack{i \in V_{\text{prohibited}} \\ j \in V \setminus \{i\}}} x_{ij} = 0. \tag{3}$$

Користувач може визначити бажані, але необов'язкові для відвідування проміжні пункти $v_{\text{desired}} \in V_{\text{desired}} \subset V$. Така інформація є евристичними даними,

що можуть надаватись алгоритму для пріоритизації дослідження деяких пунктів або під час здійснення багатокритерійного пошуку.

Назвемо маршрутом (шляхом) $y(i_0, i_u, t_0)$ з пункту i_0 в пункт i_u і відправленням у момент часу t_0 таку впорядковану послідовність дуг, для якої виконується

$$y(i_0, i_u, t_0) = ((i_0, i_1, t_0), (i_1, i_2, t_1), \ldots, (i_{u-2}, i_{u-1}, t_{u-2}), (i_{u-1}, i_u, t_{u-1})),$$

$$t_u > t_{u-1} + c(i_{u-1}, i_u, t_{u-1}) > t_{u-2} + c(i_{u-2}, i_{u-1}, t_{u-2}) > \ldots > t_0 + c(i_0, i_1, t_0),$$

тобто відправлення з наступного пункту відбувається завжди пізніше ніж прибуття, $t_u > t_{u-1} > \ldots > t_0$.

Позначимо y довільний маршрут із початкової вершини $i_0 = s$ до кінцевої $i_w = d$, який визначається послідовністю w фрагментарних маршрутів $\widetilde{y}$:

$$y = (\widetilde{y}(i_0, i_1, t_0), \widetilde{y}(i_1, i_2, t_1), \ldots, \widetilde{y}(i_{w-1}, i_w, t_{w-1})).$$

Тоді загальний час подорожі маршрутом y з кількістю сегментів *arcs* визначимо так:

$$\mathrm{C}(y) = \sum_{u=1}^{arcs} c(i_u, i_{u+1}, t_u) + \zeta(i_u, i_{u+1}, t_u),$$

де $\zeta(i_u, i_{u+1}, t_u)$ — час простою в пункті i_u перед подорожжю сполученням (i_u, i_{u+1}, t_u).

Аналогічно вартість переходу з вершини відправлення i_0 у кінцеву $i_{\widetilde{u}}$ з відправленням у момент часу t_0 позначимо $l(i_0, i_{\widetilde{u}}, t_0)$. Тоді загальну вартість маршруту $L(y)$ задамо формулою $L(y) = \sum_{u=1}^{arcs} [l(i_u, i_{u+1}, t_u) + \delta(i_u, i_{u+1}, t_u)]$, де $\delta(i_u, i_{u+1}, t_u)$ — вартість простою в пункті i_u перед подорожжю сполученням (i_u, i_{u+1}, t_u).

Якщо користувачем визначено мінімальну $y_{\min}$ і максимальну $y_{\max}$ кількості обов'язкових проміжних пунктів у маршруті, тоді $y_{\min} \le \| y \| \le y_{\max}$, де $\| y \|$ — кількість пунктів у маршруті y.

Описані умови (1)–(3) є опціональними, тобто можуть бути включені або відкинуті користувачем у момент запиту, а пошуковий алгоритм має гнучко адаптуватися.

Задача полягає в пошуку оптимального шляху з початкового пункту (вершини) $s \in V$ до цільового пункту $d \in V$. Критерієм оптимальності може бути час, вартість, кількість зупинок тощо залежно від умов користувача. Якщо критеріїв декілька, тоді визначається інтегральна оцінка розв'язку залежно від ваг критеріїв. Часовий інтервал, у якому необхідно побудувати маршрут подорожі, задається найранішим і найпізнішим часом відправлення з початкової вершини $[t_{0\min}, t_{0\max}]$, а також найранішим і найпізнішим часом прибуття в кінцеву вершину $[t_{d\min}, t_{d\max}]$. Отже, критерієм є мінімімальна сумарна вартість подорожі, яка визначається функцією $f(y) = \sum_{u=1}^{w} l(y(i_u, i_{u+1}, t_{i_u}))$, де w — кількість фрагментарних маршрутів, що утворюють маршрут y, $i_1 = s$, $i_{w+1} = d$, а t_{i_u} — час відправлення з пункту i_u. Задача полягає у знаходженні такого допустимого маршруту $y_* \in Y$, який мінімізує цільову функцію $y_* = \arg \min_{y \in \widetilde{Y} \subseteq Y} f(y)$, де $\widetilde{Y}$ — множина допустимих розв'язків.

Складність задачі зумовлена додатковими обмеженнями за часом (перельоти повинні відбуватися згідно з графіком), тоді як цільова функція залежить від змінної вартості квитків на рейс [6]. Оскільки знаходження точних розв'язків цієї задачі за прийнятний час не видається можливим, запропоновано наближений алгоритм, що ґрунтується на схемі оптимізації мурашиними колоніями.

АЛГОРИТМ ОМК З ДИВЕРСИФІКОВАНИМ ПОШУКОМ

Мурашині алгоритми — це багатоагентні системи, де поведінка кожного агента, який називається штучною мурахою або просто мурахою, ґрунтується на поведінці справжніх мурашок. Вони успішно застосовуються для розв'язування багатьох типів задач комбінаторної оптимізації, починаючи з класичної задачі комівояжера.

В алгоритмах ОМК (рис. 1) формується спеціальна модель розв'язуваної задачі, тому вони належать до класу моделе-орієнтованих методів. Така модель задачі подається у вигляді повного зваженого графу $G(V, A)$, де $v_i \in V$, $i = 1, \ldots, n$, — вершини, що відповідають компонентам розв'язку, а $a_{ij} \in A$, $a_{ij} = (v_i, v_j)$, $v_i, v_j \in V$ — ребра, які відповідають можливим з'єднанням (переходам) між відповідними вершинами. Для кожного ребра може бути визначена функція вартості з'єднання, що може залежати від деякого параметра часу t.

Умови розв'язуваної задачі можуть визначати набір обмежень $\Pi = \Pi(V, A, t)$ для елементів V та A, які визначають допустимість зв'язків між компонентами та з'єднаннями, а в підсумку — побудованого з них розв'язку.

```
    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();
        оновити_матрицю_мурашиних_маршрутів_A();
        endif
        M = оновити_внутрішній_стан();
    endwhile
    if онлайнове_відстрочене_оновлення_феромона then
        foreach відвіданої_дуги_побудованого_розв'язку do
        відкласти_феромон_на_відвіданій_дузі();
        оновити_матрицю_мурашиних_маршрутів_(A);
    endforeach
endif
завершити_діяльність();
endforeach
випаровування_феромона();
оновлення_рекорду(x);
дії_Демона();    {необов'язково}
endwhile
end
```

Рис. 1. Загальна схема алгоритмів мурашиних колоній

Розв'язки задачі оптимізації можуть бути подані як допустимі шляхи на графі G. Алгоритми ОМК можна застосовувати для знаходження допустимих шляхів з мінімальною вартістю, що задовольняють обмеження задачі. Вартість (значення цільової функції $f(x)$), яка 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 компонент (вершин чи ребер графу задачі) допустимий розв'язок задачі. У ньому використовується евристична інформація та феромонний слід, які 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 $a_{ij} \in A$. Евристичні значення η_{ij} ґрунтуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, відмінним від мурах; вони можуть залежати від часу (ітерації) t.

Рівень феромона (феромонний слід) τ_{ij}, що відповідає ребру $a_{ij} \in A$ — 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 обраного способу подання 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 наведене раніше подання на графі моделі задачі. Отже, основні компоненти обчислювальної схеми мурашиних алгоритмів є такими: модель задачі, що подається спеціальним графом; феромонні значення; евристична інформація; пам'ять (локальна та глобальна).

В усіх відомих модифікаціях ОМК на кожному кроці для вершини, що є останньою в поточному фрагменті розв'язку (маршруту в графі задачі), формується множина допустимих сусідніх вершин і обчислюється ймовірність переходу до кожної з цих вершин. На основі цих імовірностей обирається чергова вершина для продовження наявного фрагмента маршруту [2, 3].

Далі запропоновано підхід до формування маршруту, згідно з яким мурахи на кожному кроці використовують інформацію не лише з одного ребра, що може бути включене у фрагмент розв'язку, а також інформацію з більшої кількості можливих ребер [4]. Отже, на кожній ітерації мураха може додати до шляху одразу декілька вершин або, іншими словами, може зробити декілька «кроків», диверсифікуючи у такий спосіб процес пошуку.

На етапі ініціалізації здійснюються початкові налаштування: вибір значень параметрів алгоритму (зокрема кількості мурах у колонії), задавання значень феромонного сліду. Після створення нового покоління мурах їх розміщують у вершинах графу G.

На кожному кроці алгоритму мураха досліджує найближчих сусідів тієї вершини графу задачі, якою закінчується фрагмент маршруту, побудований нею на цей момент. 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. Пам'ять кожної мурахи містить інформацію про вже відвідані вершини, поточний стан, глобальний кращий розв'язок (за всіма поколіннями мурах).

Для двокрокової версії алгоритму ОМК для кожної мурахи k за наявності вже побудованого фрагмента розв'язку $y = (\ldots, i)$ формується множина ще не

відвіданих 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 наявного фрагмента розв'язку додано вершину s, $y^+ = (\ldots, i, s)$, та обчислюється ймовірність переходу від вершини i до вершини j через вершину s з урахуванням евристичної інформації та поточних значень феромона. У цьому разі перехід k-ої мурахи з вершини i у вершину j через вершину s на поточній ітерації t здійснюється з імовірністю, яку можна розрахувати так:

$$p_{ij}^k = \frac{[\tau_{ist} + \tau_{sjt}]^\alpha [\eta_{ist} + \eta_{sjt}]^\beta}{\sum\limits_{r \in N_{irj}^k} [\tau_{irt} + \tau_{rjt}]^\alpha [\eta_{irt} + \eta_{rjt}]^\beta}, \tag{4}$$

де η_{ijt} — евристична оцінка, що визначає ступінь бажаності переходу і є обернено пропорційною до вартості перельоту з пункту v_i в пункт v_j з відправленням у момент часу t, тобто $\eta_{ijt} = 1/l(v_i, v_j, t)$;

$N_{irj}^k = \{z: z \in N_i^k, j \in N_z^k\}$ — допустимий окіл для мурахи k, що знаходиться в пункті v_i у момент часу t, тобто множина допустимих невідвіданих вершин (міст);

α і β — параметри, що визначають відносний вплив феромонного сліду та евристичної інформації (стадність і жадібність алгоритму);

τ_{ijt} — феромонний слід, який відображає бажаність перельоту з пункту v_i в пункт v_j з відправленням у момент часу t (добуток феромонів, які відповідають кожній дузі маршруту).

Якщо є переходи і на один крок і на два, то слід ці ймовірності пронормувати так, щоб результат їхнього підсумовування становив одиницю. 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:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ijt}^\alpha \eta_{ijt}^\beta}{\sum\limits_{r \in N_i^k} \tau_{irt}^\alpha \eta_{irt}^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{irt} + \tau_{rjt}]^\alpha [\eta_{irt} + \eta_{rjt}]^\beta}, & \text{якщо } j \in N_i^k, \\ \dfrac{[\tau_{ist} + \tau_{sjt}]^\alpha [\eta_{ist} + \eta_{sjt}]^\beta}{\sum\limits_{r \in N_i^k} \tau_{irt}^\alpha \eta_{irt}^\beta + \sum\limits_{r \in N_{irj}^k} [\tau_{irt} + \tau_{rjt}]^\alpha [\eta_{irt} + \eta_{rjt}]^\beta}, & \text{якщо } j \in N_r^k. \end{cases} \tag{5}$$

Отже, у пропонованому алгоритмі здійснюється аналіз усіх можливих переходів з поточного пункту на K кроків вперед (у цьому дослідженні розглядається $K = 1 \div 3$). Для розгалуженого графу (яким є граф авіаперельотів) простір пошуку суттєво зростає, тому було застосовано оптимізацію у вигляді табу-списків вже розглянутих шляхів, кешування знайдених сусідів, а також збільшення кількості феромонів на дугах, що ведуть до пунктів прибуття, і процедура як параметри приймає значення поточного рекорду і поточного стану мурахи.

На кроці *ініціалізація_алгоритму* елементи матриці феромонів заповнюються значенням τ_0, а *правило_прийняття_рішення* повертає скомпонований підмаршрут, що складається з кількох кроків мурахи. *Поточний стан* мурахи — це пам'ять мурахи, що містить інформацію про вже відвідані вершини і поточний підмаршрут.

На рис. 2 подано схему процедури правила прийняття рішення, де *gb* — глобально кращий на цей момент розв'язок.

```
procedure правило_прийняття_рішення(c, gb)
    ng = множина_допустимих_вершин(c, gb);
    w = визначити_спосіб_пошуку_наступного_сегмента();
    return знайти_наступний_сегмент(w, ng);
end
```

Рис. 2. Схема процедури правила прийняття рішення

Процедура *множина_допустимих_вершин* повертає множину вершин, яких може досягти мураха за одну ітерацію залежно від кількості видимих кроків K. Отже, для однокрокового алгоритму процедура поверне множину $N_{it}^{k}(1)$, тобто усі суміжні і досяжні за часом вершини i; для двокрокового — множину $N_{it}^{k}(1) \cup N_{it}^{k}(2)$, тобто всі доступні вершини за один крок і два кроки. Аналогічно, для K-крокового алгоритму результатом процедури є $N_{it}^{k}(1) \cup N_{it}^{k}(2) \cup ... \ldots \cup N_{it}^{k}(K)$. Під час формування околів береться до уваги множина умов і відкидаються недопустимі варіанти, а також варіанти, частковий шлях яких перевищує значення кращого розв'язку на етапі побудови.

У межах реалізації алгоритму формуються нові дуги між початковою вершиною на ітерації і вершиною, якої може досягти мураха залежно від K. Для кожної нової дуги розраховуються час подорожі і вартість (з урахуванням простоїв), а також ймовірності (4) чи (5).

Для оптимізації обчислень у процедурах пошуку множини допустимих вершин і формуванні комплексних дуг використовується додаткова пам'ять для збереження обчислювальних ресурсів і пришвидшення роботи алгоритму. Також використовуються табу-списки для запису вже розглянутих пунктів. Фрагментарні маршрути, які перевищують значення рекорду, відкидаються за принципом алгоритму гілок і меж. Також, якщо фрагментарний маршрут заходить у глухий кут, оновлюються пов'язані дуги у феромонній матриці з присвоєнням значення нуль. Підґрунтям такої логіки є те, що немає сенсу розглядати маршрути, підмаршрути яких ведуть у глухий кут.

Для нових дуг (i, j, t_i), сформованих на основі проміжних дуг $(i, i+1, t_i), (i+1, i+2, t_{i+1}), \ldots, (j-1, j, t_{j-1})$, у матрицю феромонів додаються нові значення і ініціалізуються у такий спосіб: $\tau_{ijt_i} = \tau_{i,i+1,t_i} \times \tau_{i+1,i+2,t_{i+1}} \times ... \ldots \times \tau_{j-1,j,t_{j-1}}$, $l_{ijt_i} = l_{i,i+1,t_i} + l_{i+1,i+2,t_{i+1}} + \ldots + l_{j-1,j,t_{j-1}}$, $c_{ijt_i} = t_{j-1} + c_{j-1,j,t_{j-1}} - t_i$, де $\{i, i+1, i+2, \ldots, j-1, j\}$ — послідовність вершин, з якої формується нова дуга, l_{ijt_i} — вартість перельоту для сполучення через послідовність $\{i, i+1, i+2, \ldots, j-1, j\}$, тобто вага вартості нової дуги, c_{ijt_i} — час перельоту для сполучення з використанням послідовності $\{i, i+1, i+2, \ldots, j-1, j\}$, тобто вага часу нової дуги.

Розглянемо крок *визначити_спосіб_пошуку_наступного_сегмента*. Розроблений алгоритм ґрунтується на схемі алгоритму мурашиних систем (АМС) Ant Colony System [2, 3], тому вибір наступного пункту і часу відправлення в нього здійснюється за псевдовипадковим пропорційним правилом

$$(v_j, t_{v_i}) = \begin{cases} \arg\max_{l \in N_{it}^{k}} \{\tau_{ijt} \times \eta_{ijt}^{\beta}\}, & \text{якщо } q \le q_0, \\ J \text{ в іншому випадку,} \end{cases} \qquad (6)$$

де q — випадкова змінна, рівномірно розподілена на відрізку [0, 1], q_0 $(0 \le q_0 \le 1)$ — заданий параметр алгоритму, J — випадкова змінна, обрана відповідно до розподілу ймовірності з (4) чи (5).

Формула (6) описує логіку процедури *знайти_наступний_сегмент()*.

Процедура *перейти_в_наступний_стан* включає оновлення стану мурахи та аналіз ситуації. Якщо на поточній ітерації мураха потрапила в глухий кут, то для оптимізації виконується операція бектрекінгу, яка передбачає видалення попереднього стану мурахи, додавання пункту як розглянутого в пам'ять мурахи і продовження роботи. Феромонна матриця для дуг, які є інцидентними вершині глухого кута, оновлюється значенням нуль.

Якщо наступний стан є допустимим, тоді виконується *локальне оновлення феромона* , яке здійснюється мурахами після кожного переміщення вздовж дуги протягом побудови маршруту: $\tau_{v_i,v_j,t_{v_i}} \leftarrow (1-\varepsilon)\times\tau_{v_i,v_j,t_{v_i}} + \varepsilon\tau_0$, де ε, $0<\varepsilon<1$, і τ_0 — параметри алгоритму. Значення τ_0 зазвичай покладають рівним початковому значенню феромонних слідів, $\tau_0 = 1/nC^{nn}$, C^{nn} — вартість маршруту за методом найближчого сусіда. Ідея такого оновлення полягає в тому, що після кожного проходу мурахи дугою (v_i, v_j, t_{v_i}), відповідний феромонний слід зменшується, тому дуга стає менш привабливою для наступних мурах.

Якщо мураха не змогла знайти маршрут у межах заданої кількості ітерацій, здійснюється добудова маршруту, який буде допустимим. Підґрунтям такої логіки є евристичні знання графу. Зазвичай мураха в такій ситуації будує довгий і дешевий маршрут, який є ціллю мандрівника, але не замикає його. В алгоритмі додається прямий рейс з попередньої вершини поточного стану мурахи до цільової вершини. Описана логіка виконується в процедурі *повернення_мурах*.

Розглянемо глобальне оновлення феромона. Після кожної ітерації алгоритму феромон додається тільки на тих дугах, які входять у кращий маршрут y^{bs} із знайдених в поточному поколінні мурах. Іншими словами, оновлення здійснюється за формулою $\tau_{v_i,v_j,t_{v_i}} \leftarrow (1-\rho)\times\tau_{v_i,v_j,t_{v_i}} + \rho\Delta\tau^{bs}_{v_i,v_j,t_{v_i}} \quad \forall(v_i, v_j, t_{v_i}) \in y^{bs}$, де $\Delta\tau^{bs}_{v_i,v_j,t_{v_i}} = 1/C^{bs}$.

У такій модифікації алгоритму оновлення феромонних слідів стосується тільки дуг маршруту y^{bs} (а не всіх дуг графу). Процедура *оновлення_рекорду_і_статистичної_інформації*(x) виконує оновлення глобального рекорду.

Отже, розроблений алгоритм ґрунтується на класичному АМС з урахуванням часових обмежень і додає «скорочені» (комплексні) часткові маршрути, збільшуючи простір пошуку.

ЗБІЖНІСТЬ РОЗРОБЛЕНОГО АМС

Доведемо, що розроблений АМС гарантує знаходження оптимального розв'язку з імовірністю, близькою до одиниці, якщо дано достатньо часу для виконання алгоритму (доведення виконується аналогічно [2, 7]).

Ймовірнісне правило для побудови розв'язку є таким:

$$P_T(v_{h+1} = j | x_h) = \begin{cases} \dfrac{\tau^{\alpha}_{ijt}}{\sum_{(i,j,t)\in N^k_{it}} \tau^{\alpha}_{ijt}}, & \text{якщо } (i,j,t) \in N^k_{it}, \\ 0 \text{ в іншому випадку,} \end{cases} \qquad (7)$$

де $(i, j, t) \in N^k_{it}$, якщо послідовність $x_{h+1} = <(v_1, t_1), (v_2, t_2), \ldots, (v_h, t_h)>$, побудована мурахою, задовольняє множину обмежень поставленої задачі [6].

В описаному алгоритмі використовується псевдовипадкове пропорційне правило (6): на кожному кроці побудови розв'язку мураха з імовірністю q_0 обирає сполучення з найбільшим значенням феромона та з імовірністю $(1-q_0)$ — сполучення за формулою (4) чи (5). В АМС локальне значення феромонів випа-

ровується одразу після проходження мурахи дугою, а потім знов оновлюється феромонна матриця після побудови маршрутів всіма мурахами, додаючи значення феромона для приваблення мурах на найкращі шляхи. Можна помітити, що правила оновлення феромона є ізоморфними, але мають різне призначення: локальне оновлення робить відвідані на поточній ітерації дуги менш привабливими для мурах у популяції, що сприяє дослідженню ще не розглянутих вершин і перешкоджає збіжності до єдиного розв'язку; глобальне оновлення феромона сприяє дослідженню в околі найкращих розв'язків. Таким чином, незважаючи на однакову структуру правил оновлення феромона, вони зазвичай працюють з різними коефіцієнтами випаровування у випадках локального чи глобального оновлення.

Лема 1. На кожній ітерації h алгоритму виконується умова

$$\lim_{h\to\infty} \tau_{ijt}(h) \le \tau_{\max} = b(y_*).$$

Доведення. Правило оновлення феромонів можна записати в такому загальному вигляді:

$$a_{h+1} = (1-\psi)a_h + \psi b,$$

де $a_{h+1} = \tau_{ijt}^{h+1}$, $a_h = \tau_{ijt}^{h}$, тобто феромонний слід між вершинами v_i і v_j у момент відправлення t на ітерації h, b — порція феромона, що додається при оновленні феромонної матриці, ψ — порція феромона, що випаровується при оновленні феромонної матриці, тобто ε або ρ.

Запишемо значення феромонного сліду на кожній ітерації:

$$a_1 = (1-\psi)a_0 + \psi b,$$

$$a_2 = (1-\psi)a_1 + \psi b = (1-\psi)^2 a_0 + (1-\psi)\psi b + \psi b,$$

$$a_3 = (1-\psi)a_2 + \psi b = (1-\psi)^3 a_0 + (1-\psi)^2 \psi b + (1-\psi)\psi b + \psi b,$$

$$\dots .$$

Можна скористатися формулою суми геометричної прогресії для розрахунку значення a_{h+1}:

$$S_n = \begin{cases} \dfrac{\beta_1(1-\omega^n)}{1-\omega}, & \text{якщо } \omega \ne 1, \\ \beta_1, & \text{якщо } \omega = 1, \end{cases}$$

де β_1 — перший член прогресії, $\beta_1 = \psi b$, ω — знаменник прогресії, $\omega = (1-\psi)$.

Тоді

$$a_{h+1} = (1-\psi)^{h+1} a_0 + \frac{\psi b(1-(1-\psi)^h)}{1-(1-\psi)} =$$

$$= (1-\psi)^{h+1} a_0 + \frac{\psi b(1-(1-\psi)^h)}{\psi} = (1-\psi)^{h+1} a_0 + b(1-(1-\psi)^h).$$

Оскільки $0 < \psi < 1$, то для $h \to \infty$ описана вище рівність асимптотично прямує до b. Припустимо, що на кожному кроці алгоритму значення феромона оновлювалося з врахуванням максимально можливого значення $\Delta\tau_{\max} = b(y_*)$, що відповідає випадку оптимального розв'язку, і без урахування локального оновлення, яке зменшує рівень феромонів на розглянутих дугах. Тоді для будь-якого $\tau_{ijt}(h)$ справедлива нерівність

$$\lim_{h\to\infty} \tau_{ijt}(h) \le \tau_{\max} = b(y_*).$$

Теорема 1. Нехай $P^*(h)$ — ймовірність того, що алгоритм знайде оптимальний розв'язок за перші h ітерацій. Тоді для будь-якого як завгодно малого $\varepsilon > 0$ буде виконуватись умова $P^*(h) \geq 1 - \varepsilon$.

Доведення. Покажемо, що будь-який допустимий розв'язок може бути побудований з ненульовою ймовірністю. Припустимо, що сполученню (i, j, t) відповідає не найбільший рівень феромона. Розглянемо такий випадок: феромонний слід бажаного вибору становить $\tau_{\min}$, тоді як усім іншим $|N_{it}^k| - 1$ кандидатам відповідає максимальне значення феромона $\tau_{\max} = b(y_*)$. Враховуючи, що значення феромонної матриці знаходяться в межах $[\tau_{\min}, b(y_*)]$, будь-який допустимий вибір з рівності (4) для будь-якого часткового розв'язку виконується з імовірністю $(1 - q_0) \times P_{\min} > 0$, при цьому $P_{\min}$ — це ймовірність вибору сполучення (i, j, t),

$$P_{\min} = \frac{\tau_{ijt}^{\alpha}}{\sum_{(i,g,t) \in N_{it}^k} \tau_{igt}^{\alpha}} = \frac{\tau_{\min}^{\alpha}}{(|N_{it}^k| - 1) \times \tau_{\max}^{\alpha} + \tau_{\min}^{\alpha}},$$

де $(i, j, t) \in N_{it}^k$, якщо послідовність $x_{h+1} = <(v_1, t_1), (v_2, t_2), \ldots, (v_h, t_h)>$ побудована мурахою, задовольняє множину обмежень, N_{it}^k — потужність околу вершини i в момент часу t. Тоді будь-який допустимий розв'язок y, включаючи оптимальний $y_* \in Y_*$, може бути побудований з імовірністю $\hat{P} \geq (1 - q_0) \hat{P}_{\min}^n > 0$, де $n < \infty$ — максимальна довжина послідовності. Оскільки достатньо, щоб оптимальний розв'язок знайшла лише одна мураха, нижня межа для $P^*(h)$ задається у такий спосіб: $\hat{P}^*(h) = 1 - (1 - \hat{P})^h$. Якщо задати h достатньо великим, ця ймовірність може перевищити будь-яке значення $1 - \varepsilon$. Отже, отримуємо $\lim\limits_{h \to \infty} \hat{P}^*(h) = 1$. Теорему доведено.

АНАЛІЗ РЕЗУЛЬТАТІВ ОБЧИСЛЮВАЛЬНОГО ЕКСПЕРИМЕНТУ

У табл. 1 наведено результати експериментів з дослідження ефективності алгоритму мурашиних систем з багатокроковим зором для $K = 1, 2, 3$. Експерименти проведено для графу з 21144 дугами, що подають авіарейси в Європі та кілька маршрутів до США. Дослідження показали значний рівень збільшення часу виконання при додаванні кожного кроку. Для маршрутів у межах Європи, де вартість авіаквитків порівняно однакова для різних напрямів, зрос-

Таблиця 1

Маршрут	Класичний алгоритм			Двокроковий алгоритм			Трикроковий алгоритм		
	Середній час роботи, мс	Середня похибка, %	Кращий розв'язок	Середній час роботи, мс	Середня похибка, %	Кращий розв'язок	Середній час роботи, мс	Середня похибка, %	Кращий розв'язок
Київ–Портланд	6437	35	410	23396	27	385	620585	22	382
Київ–Сієтл	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

Таблиця 2

Параметр	Значення
α	0.1
β	2
Умова припинення роботи алгоритму	Не відбувалося покращення рекорду протягом 10 поколінь
Максимальний час очікування в пункті	7 днів
Максимальна кількість кроків мурахи	7
τ_0	0.1
Кількість мурах в колонії	100
ξ	0.1
q_0	0.5

тання кількості кроків не дало очікуваного результату, беручи до уваги 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. Дослідження проведено з однаковими параметрами алгоритму (табл. 2) і кількістю запусків 100.

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

ВИСНОВКИ

Диверсифікований алгоритм мурашиних систем показав суттєві покращення середніх похибок для складних (трансатлантичних) маршрутів. Водночас через структуру графу перельотів кожний крок диверсифікації пошуку мурахи зумовлює суттєве збільшення часу роботи і пам'яті алгоритму. Наступним кроком дослідження може стати дослідження впливу збільшення рівня феромона на ключових дугах графу (для рейсів з найбільших хабів) на точність отримуваних розв'язків. Важливим напрямом є також розроблення кооперативних алгоритмів, які ґрунтуються на ОМК [8, 9], а також на використанні спеціальних комбінаторних конфігурацій [10, 11].

СПИСОК ЛІТЕРАТУРИ

1. Foschini L., Hershberger J., Suri S. On the complexity of time-dependent shortest paths. *Algorithmica*. 2014. Vol. 68, N 4. P. 1075–1097.
2. Dorigo M., Stützle T. Ant Colony Optimization. Cambridge, MA: MIT Press, 2004. 319 p.
3. Dorigo M., Stützle T. Ant colony optimization: overview and recent advances. In: Handbook of Metaheuristics. Third Edition. Gendreau V., Potvin J.-Y. (Eds). Cham: Springer, 2019. P. 311–351.
4. Гуляницький Л.Ф. Новий алгоритм оптимізації мурашиними колоніями. *Сучасна інформатика: проблеми, досягнення та перспективи розвитку.* Пр. Міжн. конф., присвяченої 60-річчю заснування ІК ім. В.М. Глушкова НАН України (Київ, 13–15 грудня 2017 р.) Київ, 2017. С. 41–43.
5. He E., Boland N., Nemhauser G., Savelsbergh M. A dynamic discretization discovery algorithm for the minimum duration time-dependent shortest path problem. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Proc. of 15th Int. Conf. CPAIOR 2018 (Delft, The Netherlands, June 26–29, 2018). Delft, 2018. P. 289–297.
6. 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.
7. Thomas S., Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactionson Evolutionary Computation*. 2002. Vol. 6, N 4. P. 358–365.

8. Гуляницкий Л.Ф., Сиренко С.И. Разработка и исследование кооперативных моделе-ориентированных метаэвристик. *Кибернетика и системный анализ*. 2010. Т. 46, № 5. С. 31–39.
9. Hulianytskyi L.F., Sirenko S.I. Cooperative model-based metaheuristics. *Electronic Notes in Discrete Mathematics*. 2010. Vol. 36. P. 33–40.
10. Yakovlev S., Kartashov O., Yarovaya O. On class of genetic algorithms in optimization problems on combinatorial configurations. *Proc. of IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies* (*CSIT 2018*). 2018. Vol. 1. P. 374–377. https://doi.org/10.1109/STC-CSIT.2018.8526746.
11. Hulianytskyi L.F., Riasna I.I. Formalization and classification of combinatorial optimization problems. In: Optimization Methods and Applications. Butenko S., Pardalos P. M., Shylo V. (Eds). Cham: Springer International Publishing AG, 2017. P. 239–250.

Надійшла до редакції 21.06.2019

Л.Ф. Гуляницкий, А.И. Павленко

АЛГОРИТМЫ ОПТИМИЗАЦИИ МУРАВЬИНЫМИ КОЛОНИЯМИ С ДИВЕРСИФИЦИРОВАННЫМ ПОИСКОМ В ЗАДАЧЕ ОПТИМИЗАЦИИ АВИАПЕРЕЛЕТОВ

Аннотация. Сформулирована задача поиска оптимального пути путешественника в сети авиаперелетов, которая учитывает стоимость построенного маршрута и наличие пользовательских условий при зависимой от времени стоимости соединений. Предложены алгоритмы системы муравьиных колоний для решения зависимой от времени задачи, представленной расширенным графом перелетов, которые, в отличие от существующих муравьиных алгоритмов, учитывают динамичность сети (зависимость наличия и стоимости сообщения от времени) и пользовательские условия. Усовершенствован подход к диверсификации поиска в муравьиных алгоритмах в условиях зависимости от времени для плотного графа, что позволило повысить качество построенных маршрутов, связывающих различные регионы. Эффективность предложенных алгоритмов исследована на основе анализа результатов вычислительного эксперимента, выполненного с использованием реальных данных.

Ключевые слова: оптимизация маршрутов, сеть авиаперелетов, муравьиные алгоритмы, диверсификация поиска, расширенные сети по времени, динамическое обнаружение дискретизации.

L. Hulianytskyi, A. Pavlenko

ANT COLONY OPTIMIZATION ALGORITHMS WITH DIVERSIFIED SEARCH IN THE PROBLEM OF OPTIMIZATION OF AIRTRAVEL ITINERARY

Abstract. The formulated problem is to find optimal traveler's path in airline networks, which takes into account cost of the constructed route and user conditions with time-dependent cost of connections. Ant colony system algorithms are proposed to solve the time-dependent problem represented by an extended flight graph. Unlike the available ant algorithm implementations, the developed algorithms take into account the properties of dynamic networks (time-dependent availability and connections cost) and user conditions. The improved approach to the diversification of search in ant colony system algorithms in terms of time dependence for a dense graph increased the quality of the constructed routes from different regions. The proposed algorithms are analyzed for efficiency based on the analysis of the results of a computational experiment from real data.

Keywords: route optimization, flight network, ant algorithms, search diversification, time-expanded networks, dynamic discretization discovery.

Гуляницький Леонід Федорович,
доктор техн. наук, завідувач відділу Інституту кібернетики ім. В.М. Глушкова НАН України, Київ, e-mail: lh_dar@hotmail.com.

Павленко Анна Ігорівна,
кандидат техн. наук, інженер з розроблення програмного забезпечення компанії «Амазон», США, e-mail: dmitrieva.anya@gmail.com.