ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ

УДК 004.8:519.85:656.7

Л.Ф. ГУЛЯНИЦЬКИЙ, А.І. ПАВЛЕНКО

ОПТИМІЗАЦІЯ ШЛЯХІВ У ДИНАМІЧНОМУ ГРАФІ ПЕРЕЛЬОТІВ МОДИФІКОВАНИМ АЛГОРИТМОМ МУРАШИНИХ СИСТЕМ

Анотація. *Розглянуто задачу пошуку оптимального маршруту авіалініями із заданими користувацькими умовами, критерієм у якій виступає мінімізація витрат на подороож. Використовуючи загальну схему алгоритму мурашиних систем, розроблено метаевристичний алгоритм для розв'язування поставленої задачі. Проведено порівняльний аналіз результатів застосування розробленого алгоритму та точного алгоритму міток на основі проведеного обчислювального експерименту.*

Ключові слова: *динамічна задача пошуку, найкоротший шлях, оптимальний маршрут, алгоритм мурашиних систем, алгоритм міток, оптимізація мурашиними колоніями.*

Вступ

Для вирішення низки проблем маршрутизації в транспортних мережах останніми роками сформульовано багато варіантів задач пошуку найкоротших шляхів, а також розроблено нові підходи до їх розв’язування і реалізовано відповідні навігаційні та логістичні системи. Кожен варіант постановки має певні особливості з прикладного середовища – умови і обмеження, а також повинен враховувати різноманітні аспекти задачі – динамічність, залежність від часу, стохастичність, необхідність адаптивності тощо. Складність описаного типу задач перетворило їх на поле інтенсивних досліджень [1-4].

Метою є розробка та аналіз алгоритмів розв’язування задачі пошуку оптимального за вартістю шляху в мережі авіаперельотів із заданими користувацькими умовами, серед яких: задані початковий і кінцевий пункт, часові вікна і тривалість подорожі. Розширена версія постановки задачі може включати умови обов’язкового відвідування певних проміжних пунктів у задані часові інтервали, врахування інформації про бажані авіалінії, загальну кількість пересадок і транзитний час тощо. В такому разі задачу можна розбивати на кілька підзадач пошуку проміжних шляхів між початковим і проміжним пунктом в заданому часовому вікні, пошуку шляхів між проміжними пунктами, а також проміжними і кінцевим. Іншим підходом є врахування проміжних пунктів на етапі побудови допустимих маршрутів.

Така задача актуальна для планування подорожі мандрівника авіалініями, коли вагомими критеріями є загальна вартість квитків для всього маршруту і відвідування множини пунктів у заданий часовий інтервал, але при цьому можна додавати проміжні туристичні пункти відвідування. Загальний транзитний час (проведений безпосередньо в транспорті) і кількість пересадок бажано мінімізувати.

Оскільки перельоти здійснюються за розкладом авіакомпаній, для задачі характерна змінна доступність і вартість квитків між пунктами в залежності від часу і рейсу. При цьому необхідно враховувати, що в реальних умовах алгоритм має бути адаптивним, тобто ефективно реагувати на зміну мінімальної вартості рейсів, наявність квитків тощо, а результати видавати в реальному часі.

Для подання задачі використовують графи, у яких вершини відображують пункти (міста або аеропорти), а дуги – сполучення між пунктами (рейси).

Існує два загальних підходи до подання розкладу у задачах за допомогою графів – побудова розширеного за часом графу (time-expanded graph) і побудова залежного від часу графу (time-dependent graph) [5]. Розширений за часом граф будується шляхом додавання допоміжних вершин для кожного дискретного моменту часу і рейсу. Зазначимо, що в такому випадку граф не містить кратних ребер, але має велику розмірність. Перевагою такого підходу є те, що для розширених графів можна застосовувати класичні методи пошуку шляху без значних модифікацій. При другому підході кожному пункту відповідає одна вершина, а дуга позначає наявність сполучення між пунктами. Вартість дуги визначається залежною від часу відправлення функцією (або номером рейсу).

1. Загальна схема оптимізації мурашиними колоніями

Оптимізація мурашиними колоніями (ОМК) [6, 7] широко застосовується для розв'язування широкого кола задач комбінаторної оптимізації. Вона базується на принципі використання великої кількості простих штучних агентів, які здатні будувати розв'язки через низькорівневу комунікацію, навіяну колаборативною поведінкою мурашиних колоній. В алгоритмах ОМК використовується механізм призначення рівня феромону кожній дузі, який оновлюється пропорційно якості знайденого маршруту. Штучні агенти використовують дані евристики і феромонів для побудови розв'язків.

ОМК є ітеративним методом пошуку, в якому на кожній ітерації агентами (штучними мурахами) генерується множина розв'язків на основі поточного розподілу рівня феромону у мережі та даних задачі, а потім розподіл феромону оновлюється в залежності від новознайдених розв'язків [6, 7]. Існує багато варіацій реалізації ОМК [8], серед яких: мурашині системи [9], системи мурашиних колоній [10], MAX-MIN мурашині системи [11] – ці алгоритми різняться механізмами оновлення феромону і способами генерування нових розв'язків.

Необхідно відмітити, що агенти діють незалежно і конкурентно. Хоча кожен агент є достатньо комплексною структурою, здатною знайти розв'язок (ймовірно неоптимальний), проте більш точні розв'язки отримуються в

результаті колективної взаємодії між агентами. Взаємодія відбувається через запис та використання інформації про якість сегментів шляху у так звану матрицю феромонних слідів.

Процедура оновлення феромону може збільшувати його значення на пройдених сегментах шляху після кожної ітерації або зменшувати в результаті випаровування.

Загальна схема алгоритму у випадку онлайнового відтермінованого оновлення феромону наведена на рис. 1 [7].

```
1       procedure ACO ()
2               while (критерій_завершення_не_задоволений)
3                       планування_дій
4                               генерація_і_дії_мурашок();
5                               випаровування_феромона();
6                               дії_демона(); {необов'язково}
7                       end планування_дій
8               end while
9       end procedure

1       procedure генерація_і_дії_мурашок()
2               while (доступні_ресурси)
3                       запланувати_створення_нової_мурашки();
4                       нова_активна_мурашка();
5               end while
6 end procedure

1       procedure нова_активна_мурашка() {життєвий цикл мурашки}
2               ініціалізація_мурашки();
3               M = оновлення_пам`яті_мурашки();
4               while (поточний_стан ≠ шуканий_стан)
5                       A = зчитати_локальну_таблицю_мурашиних_маршрутів();
6                       P = обчислити_ймовірність_переходів(A,M,Ω);
7                       наступний_стан = правило_прийняття_рішення(P,Ω);
8                       перейти_в_наступний_стан(наступний_стан);
                        if (онлайнове_покрокове_оновлення_феромона)
9                               відкласти_феромон_на_відвіданій_дузі();
10                              поновити_таблицю_мурашиних_маршрутів();
                        end if
11                      M = оновити_внутрішній_стан();
12              end while
                if (онлайнове_відстрочене_оновлення_феромона)
13                      foreach відвіданої_дуги ∈ ψ  do
14                              відкласти_феромон_на_відвіданій_дузі();
15                              оновити_таблицю_мурашиних_маршрутів();
16                      end foreach
                end if
17              завершити діяльність();
18      end procedure
```

Рисунок 1 – Загальна схема алгоритму мурашиних колоній

Процедура дії_демона() в рядку 6 ACO_meta_heuristic() необов'язкова і відноситься до централізованих дій, які виконуються демоном, що володіє глобальними знаннями. В процедурі нова_активна_мурашка() шуканий_стан (рядок 4) відноситься до повного знайденого мурашкою розв'язку, тоді як процедури покрокового і відстроченого оновлення феромона в рядках 9-10 і 14-15 часто взаємно виключають одна одну. Коли вони обидві відсутні, феромон відкладає демон.

2. Математична постановка задачі пошуку оптимального шляху за наявності додаткових умов

Нехай дано направлений граф $G=(V,A)$, де $V=\{v_1,\ldots,v_N\}$ – множина вершин, що позначають аеропорти, N – їх кількість, A – множина дуг, що відповідають сполученням між аеропортами. Між кожною парою вершин може існувати кілька дуг. Нехай (v_i,v_j) – множина дуг з вершини v_i до v_j, $a_{ij}^t=(v_i,v_j,t_k)$ – конкретна дуга, $a_{ij}^t\in(v_i,v_j)$, t_k – дискретний час відправлення літака з пункту v_i в пункт v_j. Для деяких вершин (напрямків) можлива ситуація, що $(v_i,v_j)=\varnothing$, тобто граф може бути не повним. Кожній дузі a_{ij}^t відповідає значення вартості перельоту $c(a_{ij}^t)$ і час перельоту $\theta(a_{ij}^t)$.

Метою задачі є знайти оптимальний шлях з початкового пункту (вершини) $v_s\in V$ до цільового пункту $v_d\in V$, причому момент $t_s=t_0$ відправлення з v_s повинен належати заданому користувачем інтервалу: $t\in[t_{0min},\ t_{0max}]$.

Назвемо *фрагментарним* маршрутом (шляхом) $P(v_i,v_j,t_k)$ з точки v_i в точку v_j і відправленням в момент часу t_k таку упорядковану послідовність дуг ($a_{i_1i_2}^{t_{k_1}}$, $a_{i_2i_3}^{t_{k_2}},\ldots,a_{i_{w-1}i_w}^{t_{k_w}}$), для якої:

$$t_{k_w}>t_{k_{w-1}}+\theta(a_{i_{w-1}i_w}^{t_{w-1}})>t_{k_{w-2}}+\theta(a_{i_{w-2}i_{w-1}}^{t_{w-2}})>\ldots>t_{k_1}+\theta(a_{i_1i_2}^{t_{k_1}}),$$

тобто відправлення з наступного пункту відбувається завжди пізніше, ніж прибуття, $t_{k_w}\ge t_{k_{w-1}}\ge\ldots\ge t_{k_1}$. $t_{k_w}\ge t_{k_{w-1}}\ge\ldots\ge t_{k_1}$.

Позначимо y довільний *повний* маршрут із початкової вершини v_s до кінцевої вершини v_d, який визначається послідовністю w фрагментарних маршрутів, тобто, $y=(P(v_{i_1},v_{i_2},t_{k_1}),P(v_{i_2},v_{i_3},t_{k_2}),\ldots,P(v_{i_w},v_d,t_{k_w}))$, де $i_1=s,\ t_{k_1}=t_s$; множину всіх таких можливих маршрутів позначимо Y.

Загальний час подорожі маршрутом *y* визначимо як

$$\Xi(y)=\sum_{m=1}^{p}\theta(a_{i_m i_{m+1}}^{t_{k_m}})+\zeta(a_{i_m i_{m+1}}^{t_{k_m}}),$$

де $\zeta(a_{i_m i_{m+1}}^{t_{k_m}})$ – час простою в пункті v_{i_m}, p – кількість дуг у маршруті y. В деяких постановках доцільно максимізувати час простою (якщо метою є побудова туристичного маршруту з зупинками в містах) або задавати межі простою.

Аналогічно, вартість переходу з вершини відправлення v_i у кінцеву вершину v_j в момент часу t позначимо $c\left(a_{ij}^{t}\right)$.

Вартість фрагментарного маршруту $P\left(v_i, v_j, t_k\right)$ задамо формулою

$$c\left(P\left(v_i, v_j, t_k\right)\right)=\sum_{m=1}^{p}c(a_{i_m i_{m+1}}^{t_{k_m}}),$$

де $a_{i_m i_{m+1}}^{t_{k_m}} \in A$ – дуги, що входять у цей маршрут.

Часовий інтервал, в якому необхідно побудувати маршрут подорожі, задається найранішим і найпізнішим часом відправлення з початкової вершини $\left[t_{0min},\ t_{0max}\right]$, а також найранішим і найпізнішим часом прибуття в кінцеву вершину $\left[t_{d\,min},\ t_{d\,max}\right]$

$$t_{0min} \le t_0 \le t_{0max},$$
$$t_{d\,min} \le t_d \le t_{d\,max},$$
$$t_{k_w} > t_{k_{w-1}} + \theta(a_{i_{w-1}i_w}^{t_{k_{w-1}}}) > t_{k_{w-2}} + \theta(a_{i_{w-2}i_{w-1}}^{t_{k_{w-2}}}) > \ldots > t_{k_1} + \theta(a_{i_1 i_2}^{t_{k_1}})).$$

Оскільки в заданій постановці не включаються обов'язкові для відвідування проміжні пункти, вважаємо, що в кожний пункт, що належить допустимому маршруту y, можна потрапити тільки один раз. Введемо позначення:

$$x_{ijt}=\begin{cases}1, & \textit{якщо рейс з пункту } i \textit{ в пункт } j \textit{ з відправленням у час } t \textit{ належить } y,\\ 0, & \textit{якщо рейс з пункту } i \textit{ в пункт } j \textit{ з відправленням у час } t \textit{ не належить } y.\end{cases}$$

Подамо умову, яка гарантує, що в кожний пункт (вершину), належний допустимому повному маршруту *y*, можна потрапити тільки один раз при допустимому очікуванні в декілька днів в одному пункті:

$$\sum_{\substack{i,j\\ i\neq j}} x_{ij*}=1,\ (v_i, v_j, *)\in y.$$

Метою є мінімізація сумарної вартості подорожі, яка визначається цільовою функцією

$$f(y) = \sum_{j=1}^{w} c(P(v_{i_j}, v_{i_{j+1}}, t_{i_j}),$$

де w – кількість фрагментарних маршрутів, що утворюють маршрут y, $v_{i_1} = v_s, v_{w+1} = v_d$, а t_{i_j} – час відправлення з пункту v_{i_j}, тобто маємо залежність вартості подорожі від часу [9].

Задача полягає у знаходженні такого допустимого маршруту $y_* \in S \subseteq Y$, який мінімізує наведену цільову функцію:

$$y_* = \arg\min_{y \in S \subseteq Y} f(y),$$

де S – множина допустимих розв'язків – тих маршрутів, які задовольняють умови (1)-(4).

3. Алгоритм ОМК для залежної від часу задачі пошуку оптимального шляху з користувацькими умовами

Побудова графу роботи ОМК. Граф роботи ОМК ідентичний графу задачі: множина вершин і сполучень графу ОМК відповідає графу задачі $G = (V, A)$. Кожне сполучення має вагу, яка відповідає вартості рейсу на момент відправлення. Наявність сполучення і вага є залежними від часу. Множиною станів задачі вважатимемо множину усіх можливих фрагментарних шляхів.

Умови задачі вимагають побудову такого розв'язку, при якому маршрути є допустимими за часом (послідовність рейсів повинна відповідати часовому інтервалу), кожний пункт у маршруті відвідується лише один раз (хоча допустимо очікування у пунктах). На кожному кроці агент обирає наступний пункт серед тих, які задовольняють умовам (1)-(4), тобто: 1) ще не відвідувались; 2) задовольняє умовам за часом; 3) є досяжним в момент часу, більший ніж прибуття в поточний пункт. Для спрощення час можна дискретизувати, але слід мати на увазі можливість очікування в проміжних пунктах, що дозволяє включати усі пункти з наступних часових інтервалів.

Феромонний слід і евристична інформація. Феромонний слід τ_{ijt} відображує бажаність перельоту з пункту v_i в пункт v_j з відправленням в момент часу t. Евристична інформація η_{ijt} є обернено пропорційною вартістю перельоту з пункту v_i в пункт v_j з відправленням в момент часу t, тобто

$$\eta_{ijt} = 1 / c\left(v_i, v_j, t\right).$$

Конкретизація алгоритму. Позаяк величини феромонного сліду та евристична інформація для довільної пари ребер *i, j* графу задачі залежать і від часу *t*, відповідні величини τ_{ijt} і η_{ijt} подаються тривимірними матрицями.

Ініціалізація феромонної матриці. Феромонні сліди ініціалізуються таким значенням: $\forall(i,j,t), \tau_{ijt} = \tau_0 = m / C^{nn}$, де m – кількість мурах, C^{nn} – довжина туру, згенерована евристикою найближчого сусіда. В результаті початкове значення феромону повинно бути трохи вищим, ніж очікується залишати після кожної ітерації, що сприяє уникненню обмеження простору пошуку на перших ітераціях.

Побудова розв'язку. Кожен агент при ініціалізації поміщується в стартовий пункт v_0. На кожному кроці до часткового маршруту з використанням значень τ_{ijt}, η_{ijt} і ймовірнісної процедури додається досяжний за часом черговий пункт. Побудова маршруту завершується, якщо досягнуто кінцевий пункт v_d або більше не існує допустимих кроків (за часом).

Ймовірність переходу мурахи k, яка знаходиться в пункті i, в пункт j в момент часу t визначається так:

$$p_{ijt}^k = \frac{\left[\tau_{ijt}\right]^\alpha \left[\eta_{ijt}\right]^\beta}{\sum_{im N_{it}^k} \left[\tau_{ijt}\right]^\alpha \left[\eta_{ijt}\right]^\beta}, \quad jm N_{it}^k,$$

де $\eta_{ijt} = 1/\ c\left(v_i, v_j, t\right)$ – евристична оцінка, що визначає ступінь бажаності переходу; α і β – параметри, що визначають відносний вплив феромонного сліду і евристичної інформації (стадність і жадібність алгоритму); N_{it}^k – допустимий окіл для мурахи k, що знаходиться в пункті v_i в час *t*, тобто множина допустимих невідвіданих вершин (міст). Пам'ять мурахи M^k використовується для виключення вже відвіданих вершин. Ймовірність вибору дуги (i,j,t) зростає зі збільшенням значень відповідного феромонного сліду τ_{ijt} і евристичної інформації η_{ijt}.

Існує два підходи реалізації побудови маршруту: паралельний і послідовний. В паралельній реалізації на кожному кроці алгоритму всі мурахи одночасно пересуваються з поточного пункту в наступний, в послідовній – одна мураха будує повний маршрут перед початком роботи наступної мурахи.

Розроблений алгоритм базується на схемі алгоритму мурашиних систем (АМС) – Ant Colony System, тому вибір наступного пункту і час відправлення в нього здійснюється за псевдовипадковим пропорційним правилом:

$$\left(v_j, t_{v_i}\right) = \begin{cases} \arg max_{lm N_{it}^k} \left\{\tau_{ijt} \times \eta_{ijt}{}^\beta\right\}, якщо\, q \le q_0 \\ J, в\, іншому\, випадку \end{cases},$$

де q – випадкова змінна, рівномірно розподілена в $[0,1]$,

$q_0 (0 \le q_0 \le 1)$ – заданий параметр алгоритму,

J – випадкова змінна, обрана відповідно до заданого розподілу ймовірності.

Отже, з ймовірністю q_0 мураха здійснює найкращий перехід за даними феромонних слідів і евристичною інформацією (тобто використовує отримані знання) і з ймовірністю $(1-q_0)$ досліджує інші дуги. Налаштування параметра q_0 дозволяє балансувати між пошуком навколо поточного кращого розв'язку (інтенсифікація пошуку) і дослідженням нових шляхів (диверсифікація пошуку).

Оновлення феромонної матриці. Після того як всі мурахи завершили побудову маршрутів, здійснюється оновлення феромонних слідів: спочатку на всіх дугах рівень феромону знижується на деяке константне значення, а потім на дугах, пройдених мурахами – підвищується.

Випаровування феромонів в загальній схемі ОМК відбувається згідно з формулою:

$$\tau_{v_i,v_j,t_{v_i}} \leftarrow (1-\rho)\times \tau_{v_i,v_j,t_{v_i}}, \forall (v_i, v_i) \in A,$$

де ρ – параметр, що задає рівень випаровування феромону, $0 < \rho \le 1$. Цей параметр використовується для уникнення необмеженого накопичення феромону і дозволяє алгоритму «забути» попередні розв'язки, створюючи тим умови для диверсифікації розв'язків. Тобто, якщо дуга не належить побудованим мурахами маршрутам, то її феромонний слід зменшуватиметься експоненційно кількості здійснених ітерацій.

Після випаровування мурахи залишають феромонний слід на дугах, що входять в побудований маршрут, наступним чином:

$$\tau_{v_i,v_j,t_{v_i}} \leftarrow \tau_{v_i,v_j,t_{v_i}} + \sum_{k=1}^{m} \Delta \tau^k_{v_i,v_j,t_{v_i}}, \forall (v_i, v_i) \in A,$$

де $\tau^k_{v_i,v_j,t_{v_i}}$ – кількість феромону, залишеного мурахою k на відвіданих дугах; це значення визначається за формулою:

$$\Delta \tau^k_{v_i,v_j,t_{v_i}} = \begin{cases} \dfrac{1}{C^k}, \text{ якщо } (v_i, v_j, t_{v_i}) \in y^k, \\ 0, \text{ в іншому разі}, \end{cases}$$

де C^k – вартість маршруту y^k, побудованого мурахою k, яка є сумою ваг всіх дуг маршруту y^k.

Оновлення феромону в модифікації Ant Colony System здійснюється у два етапи – локальне оновлення і глобальне.

Глобальне оновлення феромону. Після кожної ітерації алгоритму феромон додається тільки на тих дугах, що входять в кращий маршрут T^{bs} із знайдених в поточному поколінні мурах. Тобто, оновлення здійснюється за формулою:

$$\tau_{v_i,v_j,t_{v_i}} \leftarrow (1-\rho)\times\tau_{v_i,v_j,t_{v_i}} + \rho\Delta\tau^{bs}_{v_i,v_j,t_{v_i}}, \forall\left(v_i,v_i,t_{v_i}\right)\in y^{bs},$$

де $\Delta\tau^{bs}_{v_i,v_j,t_{v_i}} = 1/C^{bs}$.

В такій модифікації алгоритму оновлення феромонних слідів (випаровування і додавання) стосується тільки дуг маршруту y^{bs} (а не всіх дуг графу). В цьому разі обчислювальна складність процедури оновлення феромонів на кожній ітерації зменшується з $O(n^2)$ до $O(n)$, де n – розмірність задачі. Розмірністю задачі вважаємо кількість рейсів, тобто дуг в графі.

Локальне оновлення феромону 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 методом найближчого сусіда. Ідея такого оновлення полягає в тому, що після кожного проходу мурахи по дузі $\left(v_i,v_j,t_{v_i}\right)$ відповідний феромонний слід зменшується, тому дуга стає менш привабливою для наступних мурах. Така процедура сприяє залученню наступних мурах до досліджень ще не відвіданих чи мало відвідуваних дуг.

4. Реалізація АМС

Основними процедурами ОМК є побудова розв'язків, управління феромонними значеннями, додаткові опціональні процедури [8]. Розглянемо детальніше кроки загальної схеми алгоритму ОМК, наведеної на рис. 1.

Ініціалізація алгоритму. На етапі ініціалізації завантажується розклад авіаперельотів, ініціалізується таблиця феромонів, розраховуються параметри алгоритму, сутності мурах та статистичні дані (наприклад, найкращий розв'язок).

Умова припинення роботи алгоритму. Алгоритм завершує свою роботу, якщо виконується щонайменше одна умова припинення. Можливими умовами припинення є:

1) алгоритм знайшов розв'язок на заданій відстані від відомої нижньої межі якості оптимального розв'язку (тобто задається допустима похибка);

2) була досягнута максимальна кількість побудованих маршрутів або ітерацій алгоритму;

3) алгоритм показує ознаки застою (стагнації).

Правило прийняття рішення. Правило прийняття рішення реалізується так:

1) Випадковим чином визначити значення змінної q, рівномірно розподіленої на відрізку $[0,1]$. Значення є у подальшому експерименті було рівним 0.1.

2) Визначити наступний перехід за формулою (6).

Деталізована схема роботи алгоритму наведена на рисунку 2. Побудова маршруту здійснюється мурахами паралельно.

```
   procedure ACS (y)
   ініціалізація_алоритму;
    while критерій_завершення_не_задоволений do
формування_популяції_мурах
foreach мураха_з_популяції do          {життєвий цикл мурахи}
                ініціалізація_мурахи;
        M = оновлення_пам'яті_мурахи;
                while умова_припинення_пошуку_не_виконана do
                        A = локальна_ матриця_мурашиних_маршрутів;
                П = сформувати_множину припустимих_вершин;
                        p = обчислити_ймовірність_переходів(A, M, П);
                наступний_стан = правило_прийняття_рішення(p, П);
                        перейти_в_наступний_стан(наступний_стан);
                        локальне_оновлення_феромону;
                        поновити_ матрицю_мурашиних_маршрутів_A;
                        M = оновити_внутрішній_стан;
                endwhile
                завершити діяльність;
        endforeach
    глобальне_оновлення_феромону;
    оновлення_рекорду_і_статистичної_інформації (y);
    endwhile
   end
```

Рисунок 2 – Обчислювальна схема АМС

Оновлення таблиці феромонів. Оскільки в роботі реалізовано підхід на основі АМС, оновлення феромонної матриці виконується у два етапи: зниження рівня феромонів після кожного розглянутого переходу (процедура локальне_оновлення_феромону на рис. 2) і підвищення після визначення рекорду в поточному поколінні (процедура глобальне_оновлення_феромону), що реалізуються за формулами (7)-(8).

Статистична інформація АМС. Останнім кроком в реалізації АМС є збереження та повернення статистичних даних алгоритму (прикладами можуть бути найкращий знайдений розв'язок y з початку запуску алгоритму або номер ітерації, на якій було знайдений найкращий розв'язок).

5. Порівняльне дослідження ефективності розробленого алгоритму АМС і алгоритму міток

Оскільки при розв'язуванні практичних задач комбінаторної оптимізації досить рідко вдається отримати прийнятні теоретичні оцінки точності чи трудомісткості алгоритмів, основним інструментом сучасних дослідників алгоритмів є обчислювальний експеримент. Важливою складовою таких експериментів виступає розв'язування тестових задач, але найбільший інтерес викликає аналіз результатів розв'язування задач з реальними даними.

Дані для обчислень (розклад авіаперельотів) було отримано за допомогою API пошукових клієнтів SkyScanner [12] і QpxExpress [13] по аеропортах Європи за перший тиждень жовтня 2017 року. Час був дискретизований по днях для спрощення. В різних прикладах були обрані різні частини завантаженого розкладу для пришвидшення обчислень і наочності результатів.

Результати, отримані розробленим АМС, порівнювались із відповідними розв'язками, знайденими відомим точним алгоритмом міток [5], тобто з глобальними оптимумами.

Обчислювальні експерименти нами 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.

На рис. 3 наведено порівняння часу роботи алгоритму міток і розробленого АМС в залежності від розміру таблиці розкладу. Для проведення експерименту алгоритм запускався з параметрами, наведеними в табл. 1.

Таблиця 1 – Визначення параметрів АМС для обчислювального експерименту

Параметр	**Значення/умова**
Кількість елементів розкладу	34-746
α	0.1
β	2
Умова припинення роботи алгоритму	Не відбувалось покращення рекорду протягом 10 поколінь
Максимальний час очікування в пункті	7 днів
Максимальна кількість кроків мурахи	20
τ_0	0.1
Кількість мурах в колонії	25
ε	0.1
q_0	0.5

У табл. 2 наведено середні значення часу роботи (мс) по 10 запусках алгоритму АМС для кожної величини розміру розкладу.

Результати проведених досліджень проілюстрували, що для розкладу малих розмірів алгоритм міток працює швидше, ніж АМС, проте для розкладів з кількістю елементів більше ніж 400, час роботи АМС

збільшується несуттєво, на відміну від алгоритму міток. Для розкладу з 746 рейсами алгоритм міток в середньому працює півгодини, у той час коли АМС – одну хвилину. Очевидно, що для розв'язування задач більшої розмірності в реальному часі алгоритм міток є незадовільним за часом. Алгоритм міток і АМС обидва знайшли точні розв'язки, тому похибки не аналізувались.

Таблиця 2 – Середні значення часу (мс) роботи алгоритмів у залежності від розмірності задачі

Розмір таблиці розкладу	Кількість пунктів	Кількість днів	Час роботи алгоритма міток	Час роботи АМС	Розмір таблиці розкладу	Кількість пунктів	Кількість днів	Час роботи алгоритму міток	Час роботи АМС
34	5	4	21	1676	256	11	4	410	5868
40	5	5	67	1575	318	11	5	3398	9675
50	5	6	160	1643	384	11	6	37815	16400
57	5	7	337	1729	445	11	7	325711	24035
65	6	4	56	1982	293	12	4	518	6529
78	6	5	236	2176	364	12	5	4895	10373
96	6	6	982	2432	439	12	6	62245	17823
111	6	7	3369	2954	507	12	7	518219	31075
97	7	4	92	2537	297	13	4	536	9053
118	7	5	511	3066	370	13	5	4841	10593
144	7	6	2472	4857	447	13	6	59407	18175
167	7	7	15017	5428	515	13	7	526261	28623
123	8	4	133	3011	359	14	4	703	9169
152	8	5	701	3335	446	14	5	7493	14944
183	8	6	4842	4825	540	14	6	116210	25626
213	8	7	24245	6752	625	14	7	1188713	40303
163	9	4	205	3278	395	15	4	731	11177
202	9	5	1302	4664	494	15	5	8480	17892
243	9	6	10127	7301	598	15	6	144635	35026
281	9	7	68781	10615	692	15	7	1502587	51272
187	10	4	277	5028	423	16	4	809	12312
232	10	5	1886	5641	530	16	5	9353	19945
280	10	6	13912	9165	640	16	6	161413	36610
321	10	7	106464	13783	746	16	7	1735794	58271

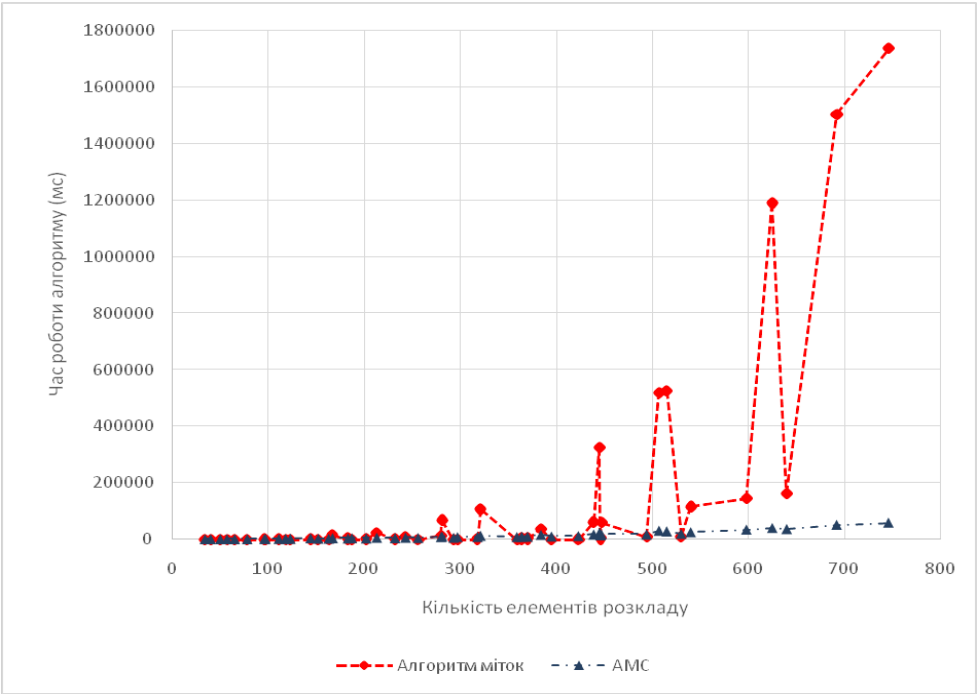

Рисунок 3 – Час роботи алгоритму міток і розробленого АМС в залежності від розміру таблиці розкладу

Висновки

Наведено постановку і математичну модель задачі пошуку оптимального шляху в мережі авіаперельотів із рядом користувацьких умов. Подано опис класичного АМС та розробленого модифікованого алгоритму для розв'язування поставленої задачі. Виконано порівняльне дослідження розробленого наближеного АМС і точного алгоритму міток. Результати дослідження показали, що АМС досягає кращих результатів за часом для задач великої розмірності, в той час як обидва алгоритми знаходять оптимальні розв'язки, тому похибка близька до нуля для АМС і нульова – для алгоритму міток. Метою наступних досліджень може стати з'ясування показників точності отримуваних результатів та часу роботи АМС у залежності від параметрів алгоритму, розмірності графу задачі, значень похибки та характеристик обмежувальних умов, а також дослідження підходу багатокрокового зору мурахи.

СПИСОК ЛІТЕРАТУРИ

1. Pajor T. Algorithm Engineering for Realistic Journey Planning in Transportation Networks [Text]: Dissertation. Doktors der Naturwissenschaften / Thomas Pajor. – Potsdam, 2013. – 266 p.
2. Гуляницький Л. Ф. Моделювання залежних від часу проблем пошуку оптимальних маршрутів: огляд / Л. Ф. Гуляницький, А. І. Павленко. // Математичне моделювання в економіці. – 2017. – № 1-2(8). №1. – С. 102–116.

3. Omer J. Time-dependent shortest path with discounted waits / J. Omer, P. Michael. // hal.archives-ouvertes.fr/hal01836007. – 2018.
4. Du K. Search and Optimization by Metaheuristics. Techniques and Algorithms Inspired by Nature / K. Du, M. Swamy. – Basel: Birkhäuser, 2016. – 434 c.
5. Павленко А.І. Розв'язування багатокритеріальної задачі пошуку оптимального шляху в динамічних мережах алгоритмом міток // Теорія оптимальних рішень. – 2017. – С. 58–63.
6. Dorigo M. Ant Colony Optimization / M. Dorigo, T. Stutzle. – Cambridge, MA: MIT Press, 2004. – 305 p.
7. López-Ibáñez M., Stützle T., Dorigo M. Ant Colony Optimization: A Component-Wise Overview. In: Handbook of Heuristics (eds. Martí R., Panos P., Resende M.). – Cham: Springer, 2016. – P. 1-37.
8. Гуляницький Л. Ф. Прикладні методи комбінаторної оптимізації: навч. посіб. / Л. Ф. Гуляницький, О. Ю. Мулеса. – К.: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с.
9. Mei Y. Efficient meta-heuristics for the multi-objective time-dependent orienteering problem / Y. Mei, F. Salim, X. Li. // European Journal of Operational Research. – 2016. – №2. – С. 443–457.
10. Dorigo M. Ant colony system: a cooperative learning approach to the traveling salesman problem / M. Dorigo, L. Gambardella. // IEEE Transactions on evolutionary computation. – 1997. – №1. – С. 53–66.
11. Stützle T. MAX–MIN ant system / T. Stützle, H. Hoos. // Future generation computer systems. – 2000. – №8. – С. 889–914.
12. Travel APIs - Skyscanner [Електронний ресурс] – Режим доступу до ресурсу: https://partners.skyscanner.net/affiliates/travel-apis.
13. QPX Express API | Google Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/qpx-express.

Стаття надійшла до редакції 11.06.2018.