КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.8 DOI:10.34229/2707-451X.21.4.2

Л.Ф. ГУЛЯНИЦЬКИЙ, О.В. РИБАЛЬЧЕНКО

ФОРМАЛІЗАЦІЯ ПРОБЛЕМИ ОПТИМІЗАЦІЇ МІСЦЬ БАЗУВАННЯ ТА МАРШРУТІВ ГРУПИ БПЛА

Розглядається проблема планування місії команди гетерогенних БПЛА, що полягає у обстеженні чи/та обслуговуванні заданої множини цілей на місцевості. Запропоновано математичну модель проблеми і алгоритми розв'язування, що базуються на детермінованому локальному пошуку, а також оптимізації мурашиними колоніями. Ефективність алгоритмів досліджено на основі результатів розв'язування задач з реальними об'єктами на місцевості.

***Ключові слова**: маршрутизація, комбінаторна оптимізація, БПЛА, локальний пошук, мурашині алгоритми.*

Вступ. Безпілотні літальні апарати (БПЛА) набувають все більш широкого використання в різних сферах. Вони використовуються для контролю технічного стану, безпеки та процесу функціонування різних об'єктів і систем, зокрема, в екології, сільському чи лісовому господарстві, на залізничному транспорті, при організації морських пошуково-рятувальних операцій; суттєва їх роль і у військовій для ведення повітряної розвідки як тактичної, так і стратегічної. Не менш важливим є застосування БПЛА у поєднанні з автомобілями для доставки товарів чи інших вантажів.

Ефективним варіантом їх застосування є вирішення завдань у складі групи. Розглянуто спеціальні проблеми планування місій на основі оптимізації маршрутів групи БПЛА чи інших рухомих роботизованих систем, що діють як команда, перед якою стоїть завдання відвідати задану множину цілей (об'єктів). У певному аспекті досліджувані задачі є розвитком відомих задач маршрутизації транспортних засобів [1 – 3].

З точки зору питань практичного планування операцій із залученням БПЛА трьома ключовими проблемами, які слід вирішити при спільному плануванні багатьох місій БПЛА, є планування їх маршрутів, розподіл цілей та вибір платформ для базування (депо), які у низці пропонованих у літературі підходів породжують окремі задачі оптимізації [4]. На противагу цьому запропоновано підхід, який дозволяє об'єднати перші дві задачі в одну.

Далі запропоновано математичну модель і алгоритми розв'язування, що базуються на детермінованому локальному пошуку, а також оптимізації мурашиними колоніями: класичний алгоритм і спеціальний алгоритм із диверсифікацією пошуку. Виконано дослідження алгоритмів, зокрема, на основі результатів розв'язування задач з реальними об'єктами на місцевості. Отримано відносну похибку результатів роботи кожного алгоритму, що дозволило порівняти їх ефективність.

Математична модель задачі. Пропонована математична модель є розвитком постановок, запропонованих в [5, 6].

Введемо позначення:

$B = \{1,\ldots, b\}$ – множина пунктів (місць можливого базування), які потенційно можуть бути використаними як депо, b – кількість таких місць;

$N = \{1,\ldots, n\}$ – множина цілей, які слід відвідати, n – їх кількість;

$M = \{1,\ldots, m\}$ – множина наявних БПЛА, m – їх кількість;

d_{st} – відстань між цілями чи цілями і місцями можливого базування s, t, де $s, t \in B \cup N$;

c_i – вартість розміщення депо в пункті i, $i \in B$;

T_{ki} – оцінка часу обстеження k-м БПЛА цілі i, $i \in N$;

e_k – вартість затрат ресурсу k-го БПЛА на одиницю довжини шляху;

v_k – середня швидкість k-го БПЛА;

R_k – ресурс k-го БПЛА (вартість всього запасу палива чи заряду акумулятора).

Включення перельоту з точки (депо, цілі) i у точку (ціль, депо) j для БПЛА k задаватиметься змінними x_{ijk} :

$$x_{ijk} = \begin{cases} 0, \text{ переліт не виконується}, \\ 1, \text{ переліт виконується}, \end{cases}, \; i, j \in B \cup N, \; k \in M \; . \tag{1}$$

Задача полягає у мінімізації загальної вартості місії і може бути поданою так:
знайти

$$\min \sum_{k \in M} \sum_{i \in B \cup N} \sum_{j \in B \cup N} e_k d_{ij} x_{ijk} + \sum_{k \in M} \sum_{i \in N} \sum_{j \in B \cup N} e_k v_k T_{kj} x_{ijk} \tag{2}$$

за обмежень

$$\sum_{k \in M} \sum_{i \in B \cup N} x_{ijk} \le 1, \;\; j \in N; \tag{3}$$

$$\sum_{k \in M} \sum_{j \in B \cup N} x_{ijk} \le 1, \;\; i \in N; \tag{4}$$

$$\sum_{i \in B} \sum_{j \in N} x_{ijk} \le 1, \;\; k \in M; \tag{5}$$

$$\sum_{i \in B} \sum_{j \in N} x_{jik} \le 1, \;\; k \in M; \tag{6}$$

$$u_i - u_j + x_{ijk} \sum_{s \in B \cup N} \sum_{t \in B \cup N} x_{stk} \le \sum_{s \in B \cup N} \sum_{t \in B \cup N} x_{stk} - 1, \;\; k \in M; \tag{7}$$

$$\sum_{i \in B} \sum_{j \in B} x_{jik} = 0, \;\; k \in M; \tag{8}$$

$$d_{ij} = \infty, \;\; i, j \in B; \tag{9}$$

$$\sum_{i \in B \cup N} \sum_{j \in B \cup N} e_k d_{ij} x_{ijk} + \sum_{i \in N} \sum_{j \in B \cup N} e_k v_k T_{kj} x_{ijk} \le R_k, \;\; k \in M; \tag{10}$$

$$x_{ijk} \in \{0,1\}, i, j \in B \cup N, \;\; k \in M; \tag{11}$$

$$y_i \in \{0,1\}, \;\; i \in B. \tag{12}$$

Цільова функція (2) визначає сумарні затрати від планування вибору місць дислокації БПЛА разом із побудовою маршрутів для обльоту цілей та затраченого часу на їх обстеження.

Кожен БПЛА відвідує певну кількість цілей, але прибуття та відправлення від кожної цілі має здійснитися лише один раз одним БПЛА. Одиничне значення у формулі (3) означає, що лише один БПЛА залишає ціль *j*, а (4) – що лише один БПЛА прибуває до цілі – або ж нулю, якщо БПЛА *k* не задіяний. Формула (5) задає умову вильоту БПЛА з одного із можливих місць базування, а формула (6) – умову повернення в одне із таких місць; знову ж, якщо БПЛА задіяний у варіанті розв'язку, то маємо рівність. Формула (7) задає умову уникнення підциклів у маршруті кожного БПЛА, що робить матрицю розв'язків несиметричною. Вимога не перелітати від одного можливого місця базування безпосередньо до іншого відображена в формулах (8) – (9). У формулі (10) подано урахування обмежень на ресурси БПЛА. Нарешті, формули (11) – (12) задають області визначення змінних задачі.

Розробка та дослідження алгоритмів розв'язування задачі маршрутизації з вибором альтернативних місць базування. Для розв'язування задач маршрутизації запропоновано три прикладні алгоритми: метода вектора спаду та два алгоритми на основі ідей оптимізації мурашиними колоніями (ОМК).

Метод вектора спаду (МВС), запропонований І.В. Сергієнком у 1964 р. для пошуку екстремумів у дискретних метричних просторах, належить до класу алгоритмів детермінованого локального пошуку (ДЛП) [7].

В цьому методі використовуються метричні околи довільної точки $x \in X$ заданого радіусу $\rho > 0$, які будемо позначати $L_\rho(x)$. Псевдокод алгоритму ДЛП з переходом до першого покращення подамо так.

---

procedure МВС (*x*)

Задання $\rho > 0$;
x^0 := деякий припустимий варіант розв'язку;
h := 0;
while *окіл* $L_\rho(x^h)$ *поточного варіанта не переглянутий повністю* **do**
    y := ГенераціяЧерговоїТочкиОколу $L_\rho(x^h)$;
    $\Delta = f(y) - f(x^h)$;
    if $\Delta < 0$ **then**
        $h := h + 1$; $x^h := y$;
    end if;
end while;
return $x = x^h$; { x^h – локально оптимальний розв'язок задачі}
end

В наведеній схемі *h* – кількість здійснених ітерацій, x^h – поточний розв'язок на ітерації *h*.

В цьому алгоритмі пропонується використати такі стратегії переходу в околі до першого покращення за значенням цільової функції, а при переході до нової ітерації – за схемою кільцевого генератора (точки в околі вважаються певним чином упорядкованими і після переходу до нової ітерації продовжується перебір точок, починаючи з наступної згідно зазначеного упорядкування) [8].

Критерієм завершення у алгоритмах локального пошуку часто виступають такі умови:

1. Відсутність покращуючої точки в околі поточного розв'язку, тобто перегляд всіх точок околу без знайденого покращення.

2. Проведення наперед заданого числа ітерацій.

3. Вичерпання заданого ліміту часу.

Як початковий варіант розв'язку найчастіше беруть довільний припустимий розв'язок. Для його знаходження часто використовується випадкове генерування варіантів розв'язку з наступною перевіркою їх на припустимість – інколи таких варіантів генерується декілька випадковим чином, а з них вибирається найкращий. Також використовуються розв'язки, знайдені послідовними евристичними алгоритмами, при цьому намагаються максимально врахувати специфіку задачі.

У реалізованому алгоритмі локального пошуку критерієм завершення є відсутність у околі поточного розв'язку варіантів, які покращують значення цільової функції.

Диверсифікований макс-мін алгоритм мурашиних систем. Алгоритми ОМК – одні із найпоширеніших алгоритмів комбінаторної оптимізації [9]. У них здійснюється інтенсивне використання кращих розв'язків у процесі пошуку, що призводить до підвищення їх продуктивності, але при цьому підвищується ризик передчасної збіжності. Тому доречно комбінувати використання кращих розв'язків та процедури запобігання передчасній збіжності. Для задоволення цих двох потреб розроблено спеціальний макс-мін алгоритм мурашиних систем як покращення алгоритму мурашиних систем [8, 10, 11]. Як і в класичному алгоритмі, для уникнення стагнації розв'язків введено інтервал значень для феромону, тобто введено поняття нижньої та верхньої межі.

У загальному випадку мурашині алгоритми використовують не просто дані задачі, а її модель, яка подається у вигляді повного зваженого графа $G(V,E)$, де $v_i \in V, i=1,...,n$, – вершини, що відповідають компонентам розв'язку задачі, n – їхня кількість, а $e_{ij} \in E, e_{ij}=(v_i,v_j), v_i,v_j \in V$, – множина можливих переходів між відповідними вершинами. Зазвичай у задачі є набір обмежень $\Pi=\Pi(V,E,t)$ для елементів V та E, де t – параметр часу. Обмеження визначають припустимість зв'язків між компонентами та переходами і побудованого з них розв'язку. У нашому випадку вершини – це цілі та депо, які і з'єднуються ребрами графа задачі.

Також введемо додаткові поняття. Евристична інформація η_{ij} – це числове значення, що не залежить від знайдених на попередніх кроках розв'язків і відображає ступінь бажаності включення у побудований фрагмент розв'язку того чи іншого нового ребра графа моделі $e_{ij} \in E$. Евристичні значення η_{ij} базуються на апріорній інформації, що відображає умови конкретної задачі та надається джерелом, відмінним від мурах.

Рівень феромону τ_{ij}, що відповідає ребру $e_{ij} \in E$, – це додатне число, яке показує, наскільки часто мурахами використовувалося це ребро на попередніх кроках чи при формуванні повного розв'язку. Феромонні сліди є для мурах довготривалою пам'яттю щодо всього процесу пошуку. Залежно від вибраного способу подання задачі, феромонні сліди можуть відповідати всім дугам задачі або тільки деяким з них [10].

Обчислювальну схему алгоритма ОМК у загальному випадку можна подати таким псевдокодом.

```
procedure MMAS (x)
    ініціалізація_алоритму;
    while кількість_ітерацій_без_покращення_менша_заданої do
    формування_популяції_мурах;
        while не_всі_вершини_відвідані do
          for чергова_мураха_з_популяції do
            ініціалізація_поточного_кроку_мурахи;
            M = оновлення_пам'яті_мурахи;
            A = локальна_матриця_мурашиних_маршрутів;
            сформувати_множину_припустимих_вершин_з_урахуванням _наявного_ресурсу;
            p = обчислити_ймовірність_переходів (A, M, П);
            наступний_стан = правило_прийняття_рішення (p, П);
            вибравши_вершину, перейти_в_наступний_стан;
            M = оновити_внутрішній_стан;
            вилучити_вибрану_вершину_із_списку_припустимих;
          endfor
        endwhile
    завершити_діяльність;
    відкласти_феромон_на_найкращому_маршруті_на_групі_останніх_ітерацій;
    випаровування_феромону;
    оновлення_рекорду (x);
    endwhile
end
```

Тут П – предикат, що описує обмеження задачі (детальніший опис обчислювальної схеми алгоритмів оптимізації мурашиними колоніями подано, наприклад, в [8, 10, 11]).

На першому кроці відбувається визначення кількості мурах (у нашому випадку кількість мурах рівна кількості БПЛА) та відбувається початкова ініціалізація матриці феромонів. Для уникнення стагнації розв'язку вводиться інтервал значень для феромону, тобто вводиться поняття нижньої та верхньої межі. На початку роботи алгоритму матриця феромонів ініціалізується значеннями нижньої межі феромону.

Для розв'язування сформульованої задачі використовують модифікований макс-мін алгоритм мурашиних систем. Розв'язок будується покроково: поточна мураха вибирає наступну вершину графа задачі, після переходу в яку ця вершина стає недоступна для відвідання мурахами, які роблять свій крок пізніше.

Для всіх мурах на кожному кроці відбувається наступна послідовність дій [8]:

a) формування підмножини припустимих вершин, які може відвідати мураха;

b) обчислення ймовірності переходів від поточної вершини i до всіх допустимих вершин; c) вибір допустимої вершини і перехід до неї.

Кожна мураха формує лише частковий розв'язок задачі, а їх сукупність дає повний розв'язок. Через це проводиться заздалегідь визначена кількість ітерацій, формується набір часткових розв'язків і з них обирається найкращий, на ньому відкладається феромон. На наступних етапах відбуються такі дії:

- випаровування феромонів;
- оновлення допустимих нижньої та верхньої межі феромону;
- оновлення матриці феромонів відповідно нижньої та верхньої межі;
- перевірка стагнації.

Детальніше розглянемо правила переходу до наступної вершини та процес обчислення ймовірностей переходів. Стани задачі визначаються в термінах скінченних послідовностей $y=(v_{s_1},v_{s_2},...)$, $v_{s_r}\in V$, елементів V (або, що рівносильно, E), які на всіх проміжних кроках мурахи є фрагментами розв'язку задачі оптимізації. Якщо Y – множина всіх можливих послідовностей, то множина Y усіх (під)послідовностей, які задовольняють обмеження $\Pi=\Pi(V,E,t)$, є підмножиною $Y:Y\subseteq Y$, а її елементи визначають припустимі стани задачі. Нехай на певному кроці мураха k побудувала фрагмент розв'язку y, останньою компонентою якого є вершина $i\in V$, тобто вона перебуває в цій вершині: $y=(...,i)$. Тоді мураха може переміститися до будь-якої вершини j із множини припустимих сусідніх вершин N_i^k, що визначаються як $N_i^k=\{j: j\in N_i\wedge(y,j)\in Y\}$, де N_i – множина всіх сусідніх до i вершин графа задачі [8, 12]. Вибір наступної вершини відбувається за псевдовипадковим пропорційним правилом. Введемо новий параметр $p_0\in[0,1]$, кожна мураха переходить з вершини $i\in V$ до вершини $j\in N_i^k$ з імовірністю p_0, j визначається наступним чином: $j=\arg\max\{\tau_{ir}^{\alpha}(t)\eta_{ir}^{\beta}(t): r\in N_i^k\}$, а з імовірністю $1-p_0$ обирається вершина за правилом колеса рулетки з використанням імовірності p_{ij}^k:

$$p_{ij}^{\ k}=\frac{\tau_{ij}^{\ \alpha}(t)\eta_{ij}^{\ \beta}(t)}{\sum\limits_{r\in N_i^k}\tau_{ir}^{\ \alpha}(t)\eta_{ir}^{\ \beta}(t)}. \tag{13}$$

Відкладання та випаровування феромонів відбувається за такою формулою:

$$\tau_{ij}(t+1)=\rho\tau_{ir}(t)+\frac{(1-\rho)}{f_{\min}^0}\},$$

де $\rho-$ коефіцієнт випаровування, що лежить у межах від 0 до 1, а $f_{\min}^0-$ найкраще значення цільової функції на початковій популяції мурах.

Нижня та верхня межа феромонів визначаються за наступними формулами [10]:

$$\tau_{\max}=1/(\rho f_{\min}^0),$$

$$\tau_{\min}=[\tau_{\max}(1-\sqrt[n]{0.05})]/((\frac{n}{2}-1)\sqrt[n]{0.05})),$$

де $n=|V|$.

Коригування матриці феромонів відбувається наступним чином:

$$a'=\min\{a,\tau_{\max}\},\ a'=\max\{a',\tau_{\min}\},$$

де a – елемент матриці феромонів.

Для боротьби зі стагнацією здійснюється скиданням значень матриці феромонів у початковий стан, якщо за визначену кількість ітерацій не відбулося покращення найкращого розв'язку.

Для диверсифікації пошуку розроблено алгоритм ОМК, який базується на використанні двокрокового переходу [12]. Його принциповою відмінністю є те, що мурахи можуть робити перехід як на один крок вперед, так і на два. Для одного кроку використовується формула (13), а ймовірність переходу k-ої мурахи з вершини i в j через вершину s на поточній ітерації t визначимо так:

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

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

Розглядався також мультиплікативний варіант цієї формули:

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

Оскільки розглядаються одночасно переходи як на один, так і на два кроки, то виникає необхідність їх нормування, щоб у сумі була одиниця. Імовірність переходу 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, $r \in N_i^{\,k}$, $j \in N_r^{\,k} \setminus i$, для адитивного варіанта можна визначити так:

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

Мультиплікативну версію можна подати наступним чином:

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

Експериментальним шляхом було підтверджено, що мультиплікативний варіант показує кращі результати. Це можна пояснити тим, що при використанні адитивного варіанта сумарна імовірність двокрокових переходів значно перевищує сумарну імовірність однокрокових, а це призводить до того, що майже завжди відбуваються двокрокові переходи. Використання лише двокрокових переходів робить алгоритм більш жадібним, що забезпечує лише миттєву вигоду, водночас як в цілому результат може виявитися несприятливим.

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

Обчислювальний експеримент. При розв'язуванні задач комбінаторної оптимізації (ЗКО) дуже рідко вдається отримати теоретичні результати, зокрема, стосовно ефективності застосування тих чи інших алгоритмів, які б були корисні на практиці. Тому для дослідження практичної

застосовності та ефективності розроблених алгоритмів, як це і робиться у всьому світі, проведено обчислювальний експеримент із розв'язування серії задач. Три задачі було сформовано на реальних геоданих, а дві – шляхом використання відомих задач із бібліотеки TSPLIB [13]:

– задача з 1 базою та 3 цілями, 1 БПЛА має запас ходу, достатній для відвідування лише однієї цілі за один виліт. Перевіряє простий план операції на виконання обмежень (задача 1);

– задача з 3 депо, 15 цілями та 3 БПЛА. Запасу ходу жодного БПЛА не вистачає на повний обліт усіх цілей без заміни блоків живлення (задача 2);

– задача з 4 депо, 23 цілями та 3 БПЛА. Запасу ходу жодного БПЛА не вистачає на повний обліт усіх цілей без заміни блоків живлення, можна візуально виділити три кластери цілей (задача 3);

– модифікація задачі att48 з TSPLIB, одна з вершин визначена як депо. 47 цілей, 1 депо та 1 БПЛА (задача 4);

– модифікація задачі berlin52 з TSPLIB, одна з вершин визначена як депо. 51 ціль, 1 депо та 1 БПЛА (задача 5).

Для кожної задачі, окрім першої, було виконано попередній підбір параметрів мурашиного алгоритму за допомогою пришвидшених запусків з меншою кількістю ітерацій. З обраними параметрами виконана зазначена кількість запусків з різними ініціалізаторами генератора псевдовипадкових чисел. Для алгоритму детермінованого локального пошуку як початковий розв’язок обирався результат роботи жадібного алгоритму, а $\rho = 1$. В усіх випадках ДЛП запускався лише один раз. Час роботи кожного алгоритму в усіх запусках обмежено 20 секундами.

Для обчислення відносної похибки ε використовується f^* – найменше відоме значення цільової функції для задачі, а відносна похибка розв’язку обчислюється за формулою $\varepsilon = 100 \cdot (f - f^*) / f^*$.

Для отримання результатів на якісному рівні значення c_i, T_{ki}, e_k та v_k у формулах (2)–(10) було покладено рівними 1.

Особливістю *задачі* 1 є те, що запасу ходу вистачає лише на переліт до будь-якої з цілей та назад у депо. Таким чином, треба здійснити дві заміни блоків живлення у депо.

Найкращий відомий план показано на рис. 1.

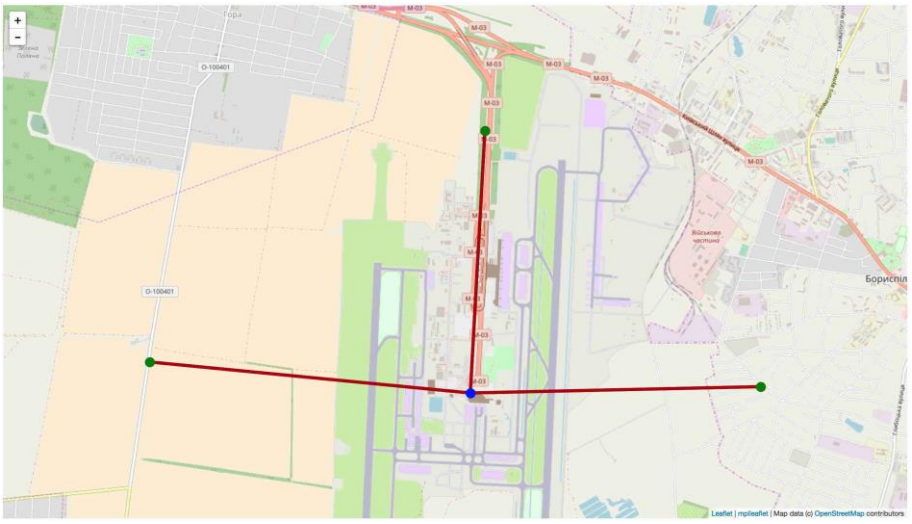

РИС. 1. Отриманий план операції

Тут всі алгоритми повернули однакові результати. Як і очікувалося, при реалізації місії відбуваються дві заміни блоків живлення.

Особливістю *задачі 2* є те, що точки явно скупчені у два кластери: один біля депо 1 та 2, інший біля депо 3. У найкращому відомому плані операції виліт здійснюється з одного депо, повернення – в інше. Це демонструє описану можливість будувати шляхи між різними депо. Слід зазначити, що це також актуально для запуску БПЛА з динамічних позицій, тобто, коли повернення не може бути здійснене у ту саму точку.

Найкращий відомий план показано на рис. 2. План, отриманий класичним ОМК, – на рис. 3, а ДЛП – на рис. 4.

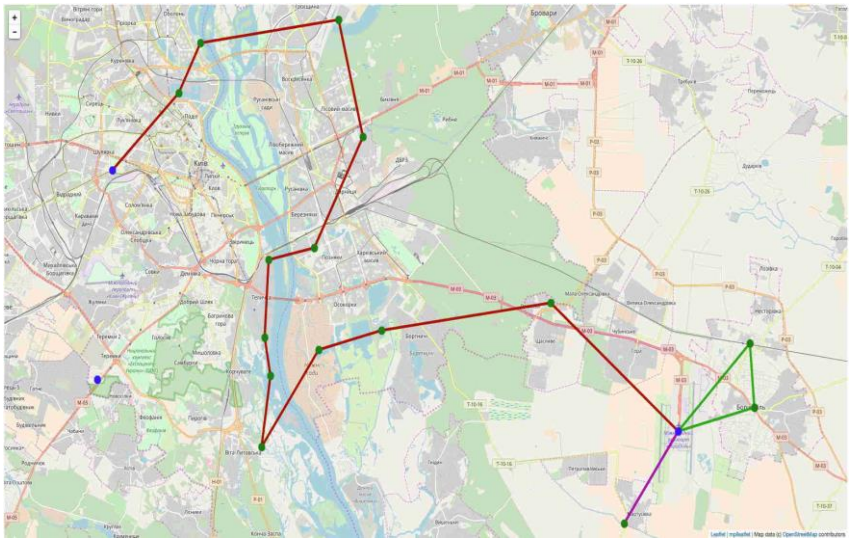

РИС. 2. План операції, отриманий модифікованим алгоритмом ОМК

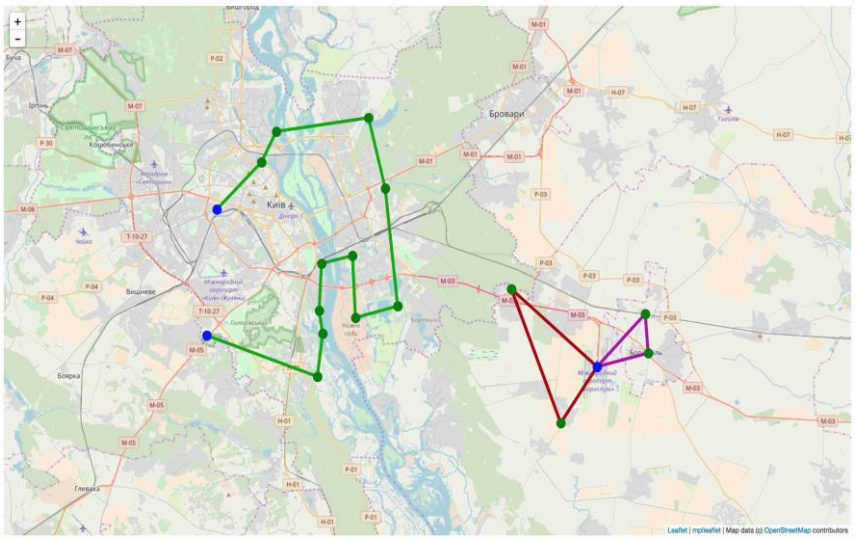

РИС. 3. План операції, отриманий класичним ОМК

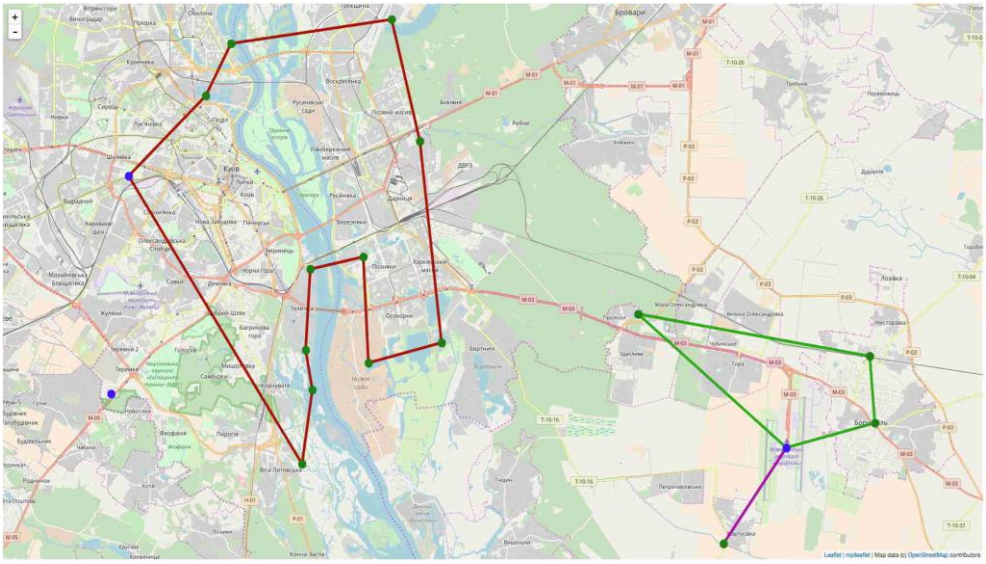

РИС. 4. План операції, отриманий ДЛП

Класичний та модифікований алгоритми ОМК знайшли близькі до найкращого із відомих розв’язків. Модифікований алгоритм ОМК знайшов найкращий відомий на даний момент розв’язок, водночас як відхилення ДЛП від оптимуму виявилося досить значним – на 10 % гіршим за найкращий розв'язок.

У *задачі 3* точки явно скупчені у два кластери: один біля депо A та B, інший біля депо C. Також є точки неподалік від депо 4 та 3, проте розкидані на великій відстані одна від одної. У найкращому відомому плані операції є такий маршрут, що виліт здійснюється з одного депо, повернення – в інше. Це демонструє описану можливість будувати шляхи між різними депо. Також двічі використовується процедура заміни блоків живлення.

Найкращий відомий план показано на рис. 5, а отриманий класичним ОМК – на рис. 6.

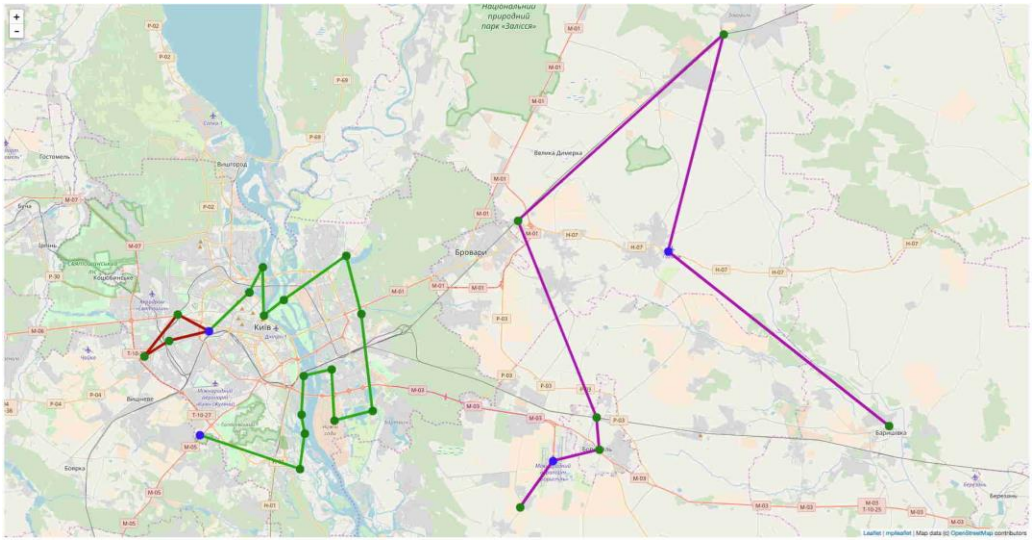

РИС. 5. План операції, отриманий диверсифікованим алгоритмом ОМК

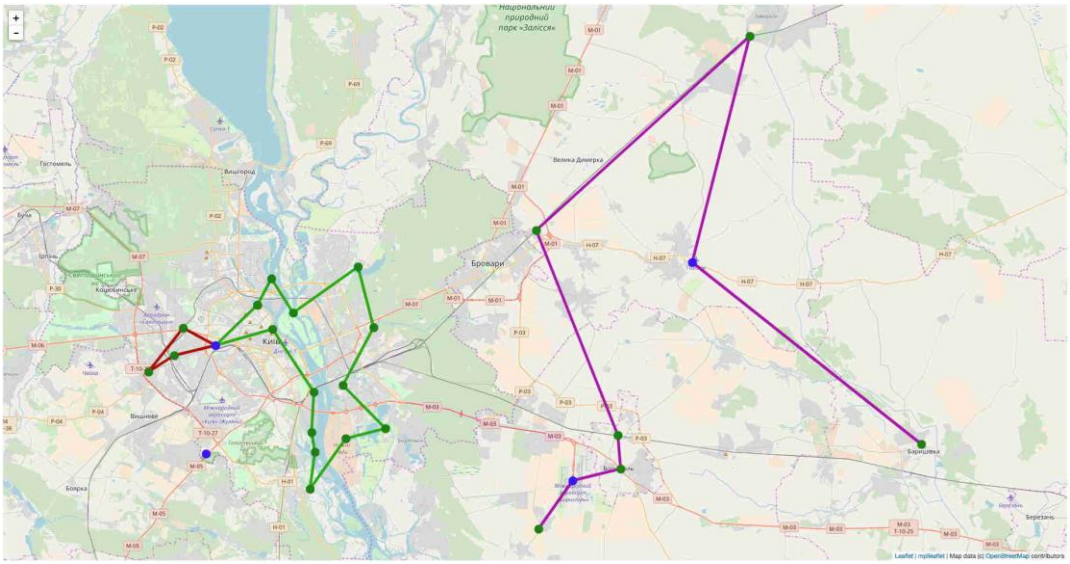

РИС. 6. План операції, отриманий класичним ОМК

Класичний та диверсифікований ОМК знайшли близькі до найкращого відомого розв’язки. Обидва плани операції відрізняються лише одним маршрутом, різниця у похибці майже відсутня, проте, візуально відмінність – очевидна і значна. Диверсифікований ОМК знайшов найкращий відомий на даний момент розв’язок, водночас як відхилення ДЛП від оптимуму склало 11 %.

План операції такий (маршрут описується депо та цілями).

БПЛ a: *вилітає з депо B*; *19*; *20*; *18*; *здійснює посадку в депо B.*

БПЛ b: *вилітає з депо C*; *23*; *заміна блоків живлення в депо C*; *24*; *22*; *25*; *26*; *заміна блоків живлення в депо D*; *27*; *D.*

БПЛ c: *вилітає з депо B*; *17*; *5*; *6*; *16*; *15*; *7*; *8*; *14*; *21*; *12*; *13*; *9*; *11*; *10*; *здійснює посадку в депо A.*

На рис. 7 показано отриманий ДЛП план операції, що виявився на 11 % гіршим за найкращий.

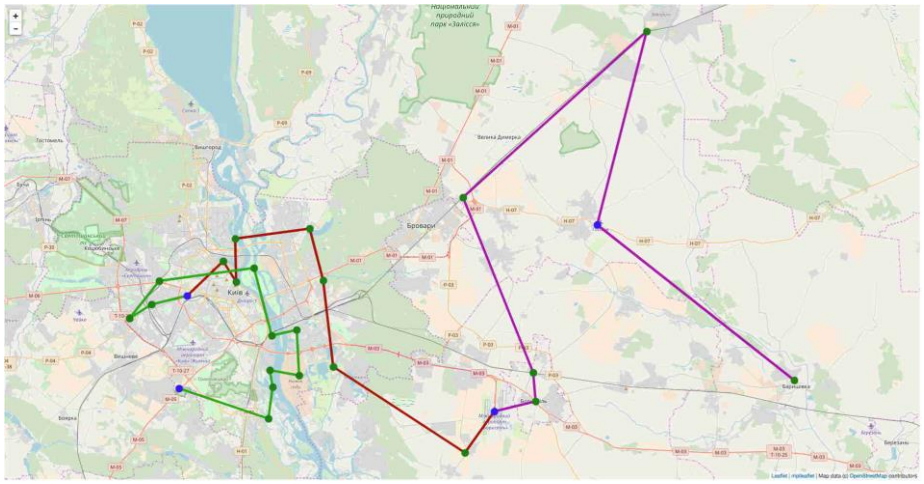

РИС. 7. План операції, отриманий ДЛП

Особливістю задач 4, 5 є те, що взяті з бібліотеки задачі комівояжера з відомим оптимумом перетворюються на задачу маршрутизації БПЛА шляхом заміни однієї вершини на депо. Оптимальний шлях залишається незмінним, а отже, є можливість порівнювати отриманий розв'язок з найкращим.

Задача 4 базується на даних задачі att48 з TSPLIB. Найкращий відомий план показано на рис. 8.

РИС. 8. Оптимальний розв'язок задачі att48

Класичний та модифікований алгоритми ОМК знайшли оптимальні розв'язки, водночас як відхилення ДЛП від оптимуму виявилося досить значним.

У *задачі 5* використано вхідні дані для задачі комівояжера berlin52 з TSPLIB, одна з цілей була розглянута як депо. Найкращий відомий план показано на рис. 9.

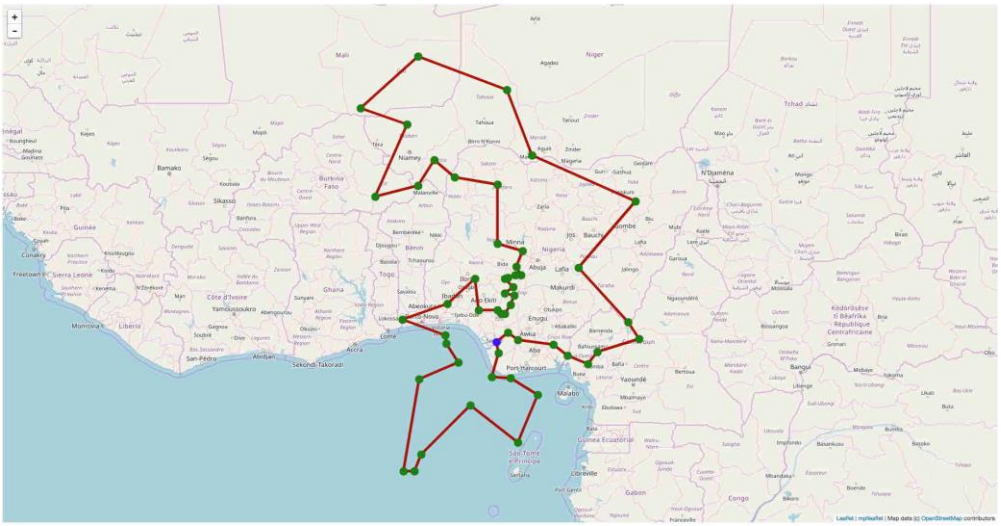

РИС. 9. Оптимальний розв'язок задачі berlin52

Знову класичний та модифікований алгоритми ОМК знайшли оптимальні розв’язки, водночас як відхилення ДЛП від оптимуму виявилося досить значним. Отже, оптимальний план операції, отриманий обома алгоритмами, співпадає з доведеним точним методом оптимумом для задачі att48.

Найважливіші результати обчислювального експерименту із розв’язування вищезгаданих задач наведено у таблиці.

ТАБЛИЦЯ. Зведені результати дослідження ефективності алгоритмів

№	Алгоритм ОМК			Диверсифікований алгоритм ОМК			Метод вектора спаду (ДЛП)		
	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

Різниця між диверсифікованим та класичним алгоритмом ОМК на задачах комівояжера не проявлялася, проте, диверсифікований алгоритм забезпечує значно більшу варіативність розглянутих розв’язків. Особливо суттєво це впливає на вибір початкової розстановки БПЛА по депо.

Як підсумкову рекомендацію можна зазначити, що диверсифікований алгоритм доречно використовувати у випадку, коли немає жорстких обмежень на час побудови плану операції. На задачах невеликої розмірності (до 52, згідно тестових запусків), різниця у часі є нехтовно малою.

Висновки. Наведено змістовну постановку сформульованих проблем маршрутизації для виконання обстеження та/або обслуговування заданої множини цілей групою БПЛА з умовою завершення маршруту в певних зонах приймання (депо) та обмеженнях на ресурси БПЛА як спеціальної задачі комбінаторної оптимізації.

Для розв’язування сформульованої задачі маршрутизації групи БПЛА розроблено макс-мін алгоритм мурашиних систем, особливістю якого є покрокова взаємодія мурах для формування розв'язків, та гібридний алгоритм табу пошуку та алгоритм детермінованого локального пошуку – метод вектора спаду.

Розроблені алгоритми апробовано як на відомих задачах комівояжера, так і на спеціально сформованих на місцевості задачах з багатьма депо і наявними обмеженнями. Запропоновані алгоритми на базі ОМК у більшості випадків продемонстрували кращі результати за точністю, хоча при цьому збільшувався час обчислень.

Напрямом подальших досліджень може бути покращення мурашиного алгоритму за рахунок використання кооперативних алгоритмів пошуку розв’язків [14], а також паралельної реалізації острівної моделі [15].

Список літератури

1. Ramos T.R.P, Gomes M.I., Povoa, A.P.B. Multi-depot vehicle routing problem: a comparative study of alternative formulations. *International Journal of Logistics Research and Applications*. Jun, 2019. P. 103–120. https://doi.org/10.1080/13675567.2019.1630374

2. Bezerra S.N., Souza S.R., Souza M.J.F. A GVNS Algorithm for Solving the Multi-depot Vehicle Routing Problem. *Electronic Notes in Discrete Mathematics*. 2018. **66**. P. 167–174. https://doi.org/10.1016/j.endm.2018.03.022
3. Nucamendi-Guillen S., Padilla A.G., Olivares-Benitez E. et al. The multi-depot open location routing problem with a heterogeneous fixed fleet. *Expert Systems with Applications*. 2020. **165**. 113846. https://doi.org/10.1016/j.eswa.2020.113846
4. 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. **56** (2). P. 195–203. https://doi.org/10.1007/s10559-020-00235-8
5. Гуляницький Л.Ф., Рибальченко О.В. Формалізація та розв'язування одного типу задач маршрутизації БПЛА. *Теорія оптимальних рішень*. 2018. 17. С. 107–114. http://dspace.nbuv.gov.ua/handle/123456789/144979
6. Liu Y., Liu Z., Shi J., Wu G., Chen C. Optimization of Base Location and Patrol Routes for Unmanned Aerial Vehicles in Border Intelligence, Surveillance, and Reconnaissance. *Hindawi Journal of Advanced Transportation*. Volume 2019. Article ID 9063232. 13 p. https://doi.org/10.1155/2019/9063232
7. Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации. К.: Наук. думка, 1988.
8. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. К.: Видавничо-поліграфічний центр "Київський університет", 2016. 142 с.
9. 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
10. Dorigo M., Stutzle T. Ant Colony Optimization. MIT Press, Cambridge. 2004. https://doi.org/10.7551/mitpress/1290.001.0001
11. Stützle T., Hoos H.H. MAX-MIN ant system. *Future Generation Computer Systems*. 2000. P. 889 – 914. https://doi.org/10.1016/S0167-739X(00)00043-1
12. Гуляницький Л.Ф. Диверсифікація пошуку в алгоритмах оптимізації мурашиними колоніями. *Теорія оптимальних рішень*. 2017. 16. С. 47–57. http://dspace.nbuv.gov.ua/handle/123456789/131437
13. Бібліотека TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (звернення: 14.12.2021)
14. Гуляницький Л.Ф. Алгоритми ОМК з диверсифікованим пошуком. Міжн. наук. симпозіум *"Інтелектуальні рішення". Теорія прийняття рішень:* мат. IX Міжн. шк.-сем. (Ужгород, 15-20 квітня 2019 р.). Ужгород: УжНУ, 2019. С. 17–21.
15. Mora A.M., García-Sánchez P., Merelo J.J. Pareto-based multi-colony multi-objective ant colony optimization algorithms: an island model proposal. *Soft Computing*. 2013. **17**. P. 1175. https://doi.org/10.1007/s00500-013-0993-y

Одержано 11.12.2021

Гуляницький Леонід Федорович,
доктор технічних наук, завідувач відділу
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-1379-4132
leonhul.icyb@gmail.com

Рибальченко Олег Валерійович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-5716-030X
rv.oleg.ua@gmail.com

UDC 519.8

Leonid Hulianytskyi [*], Oleg Rybalchenko

Formalization of the Problem of Optimization of Base Places and Routes of the UAV Group

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
[*] Correspondence: leonhul.icyb@gmail.com

Introduction. The problem of planning the mission of a set of heterogeneous unmanned aerial vehicles (UAVs)is considered, which is to survey and/or service a given set of targets in the field. A mathematical model of the problem and algorithms for its solving that is based on deterministic local search, as well as

optimization by ant colonies are proposed. The efficiency of algorithms is investigated based on the results of solving problems with real objects in the field. The relative error of the results of each algorithm was obtained, which allowed to compare their efficiency.

The purpose of the paper is to solve a routing problem in different ways to reduce overall mission cost and compare the efficiency. The problem statement considers multiple starting points and destinations (depots) for UAVs with determined capacity, so algorithms proposed in the paper are designed to optimize the initial placement. Each UAV has a maximum flight distance because of an energy limit, though vehicles can be recharged by visiting one of previously placed depots. The mission goal is to visit all the given targets while minimizing the overall cost, so fuel consumption over distance, depot placement, and resources needed to survey and/or service of the target by each UAV are considered as components of the final cost metric to be minimized considering a set of specific constraints.

Results. To solve the given UAV routing problem, a max-min algorithm of ant systems was developed, which features step-by-step interaction of ants to form solutions, a hybrid taboo search algorithm and a deterministic local search algorithm - the decay vector method. The developed algorithms were tested both on the known travelling salesman problems, and on specially developed problems with multiple depots and additional restrictions.

Conclusions. The proposed algorithms which are based on ant colony optimization are compared both in terms of accuracy and computation time. A hybrid algorithm achieved slightly better score, though computation time has increased.

Keywords: routing, combinatorial optimization, UAV, local search, ant colony optimization algorithms.