УДК 004.891 DOI: 10.30748/soi.2019.158.08

О.С. Тимчук[1], Я.А. Проценко[2], А.І. Парамонов[2]

[1] *Університет економіки та права "КРОК", Київ*
[2] *Донецький національний університет ім. В. Стуса, Вінниця*

ЗАСТОСУВАННЯ АЛГОРИТМУ МУРАШИНОЇ КОЛОНІЇ ДО ВИРІШЕННЯ ЗАДАЧІ ДЕКІЛЬКОХ КОМІВОЯЖЕРІВ БЕЗ ДЕПО

У статті запропоновано алгоритм вирішення NP-повної задачі декількох комівояжерів без депо, яка є узагальненням "стандартної" задачі комівояжера. В основу алгоритму покладено метаевристику мурашиної колонії – мурахи, використовуючи різні типи феромонів, намагаються оптимально розбити граф на кластери і оптимізувати маршрут всередині кожного кластера. Наведено результати експерименту, який було проведено на графах з 15 та 30 вузлами для вирішення задачі з трьома комівояжерами. Розроблений алгоритм демонструє можливість узагальнення оптимізацій мурашиних колоній на задачі з додатковими умовами.

Ключові слова: *задача комівояжера, задача декількох комівояжерів без депо, мурашиний алгоритм, TSP, mTSP, ACO.*

Вступ

Постановка проблеми. Задача комівояжера (TSP) є однією з найвідоміших задач комбінаторної оптимізації, яка полягає у пошуку найвигіднішого замкнутого маршруту, що проходить через всі зазначені міста рівно один раз. Узагальненням задачі TSP є задача декількох комівояжерів (mTSP), в якій для пошуку самого вигідного маршруту використовується кілька комівояжерів [1]. Розрізняють mTSP з депо, в якій всі комівояжери завершують свій шлях в одному місті, і mTSP без депо – кожен комівояжер повертається в свій відправний пункт. Незважаючи на популярність TSP, mTSP більш наближена до реальності, так як більшість задач на підприємстві вирішуються більш ніж одним виконавцем. Можливі сфери застосування mTSP приведені у [1–2]. Наприклад, з огляду на сучасні тенденції в управлінні розробкою програмного забезпечення, mTSP без депо може бути використана для рівномірного "прозорого" розподілу пріоритетних незалежних задач між членами IT-команди. Кожен член команди буде асоційований з окремим комівояжером, а задачі – з містами. Рішення mTSP дозволить розбити задачі на кластери і визначити послідовність виконання задач кожним членом команди.

mTSP, як і TSP, відноситься до класу NP-повних задач, тому стандартні чіткі методи оптимізації можуть бути застосовані тільки для задач невеликої розмірності. Використання технологій обчислювального інтелекту дозволяє істотно скоротити розмірність задачі, але призводить до отримання наближеного рішення. З огляду на високий ступінь асоційованості між mTSP і метаевристичними алгоритмами ройового інтелекту, актуальним є розробка алгоритму розв'язання mTSP на базі алгоритму мурашиної колонії.

Аналіз останніх досліджень і публікацій. Аналіз публікацій, присвячених mTSP, показав, що увага дослідників більш зосередженим на вирішенні mTSP з депо. Роботи [1–2] є оглядовими, з докладним описом постановки задачі mTSP, областей застосування mTSP і методів вирішення mTSP, серед яких основними є еволюційні алгоритми (генетичні алгоритми) [3–4], нейронні мережі [5], евристичні алгоритми [6–8] і колективний інтелект (мурашині алгоритми) [9–11]. Незважаючи на наявність певної кількості публікацій, присвячених досліджуваному питанню, все ще залишається потреба в напрацюванні нових алгоритмів для вирішення mTSP без депо.

Мета статті – розробити алгоритм наближеного вирішення задачі mTSP без депо на базі метаевристики мурашиної колонії.

Виклад основного матеріалу

Постановка задачі

mTSP, аналогічно TSP, визначається на зваженому графі $G(V,A)$, де безліч вершин графа $V=\{1,...,n\}$ відповідає містам, а безліч ребер між вершинами $A=\{(i,j):i,j\in V\}$ – шляхами сполучення між цими містами. З A асоціюється матриця $D=\|d_{ij}\|$, де d_{ij} є вартістю переходу з i-ї вершини в j-ту. Під вартістю зазвичай розуміють відстань, час або вартість поїздки між містами i та j.

Сформулюємо задачу mTSP без депо в термінах цілочисельного лінійного програмування [1]:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ijk} \to \min, k=\overline{1,m}, \qquad (1)$$

$$x_{ijk} = \begin{cases} 1, (i, j) \in \textit{маршруту k-го комівояжера}, \\ 0, \textit{інакше}, \end{cases} \quad (2)$$

$$\forall (i, j) \in A, k \in M,$$

$$\forall k \in M : \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij1} \geq \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ijk}\ , \quad (3)$$

$$\forall i \in V : \sum_{k=1}^{m} u_{ik} = 1, \quad (4)$$

$$\forall k \in M, i \in N : \sum_{j=1}^{n} u_{ijk} = 2u_{ik}\ , \quad (5)$$

+ обмеження, які забороняють підшляхи. (6)

де x_{ijk} – бінарна ознака включення переходу (i, j) в маршрут k-го комівояжера,

m – кількість комівояжерів,

u_{ik} – бінарна ознака включення i-ї вершини в маршрут k-го комівояжера.

Також допускаємо, не обмежуючи загальність формулювання, що найдовший шлях пройшов комівояжер з індексом 1. (3–6) є обмеженнями задачі:

(3) – обмеження симетричності задачі; (4) гарантують, що шлях першого комівояжера не є коротшим за шлях інших комівояжерів (3); (5) – кожне місто повинне бути відвідано рівно одним комівояжером; (6) гарантують, що всі шляхи не можуть розгалужуватись – кожен комівояжер рівно один раз входить і виходить у міста, які належать його шляху, та жодного разу не входить і не виходить у всі інші міста, при цьому немає обмеження, щоб кожний комівояжер відвідував хоча б одне місто. Обмеження (4) та (5) разом не дозволяють комівояжерам відвідувати міста, які належать шляхам інших комівояжерів.

Опис алгоритму

Розроблений алгоритм вирішення mTSP заснований на базових принципах алгоритму ACO [12] з деякими модифікаціями. Розглянемо основні принципи запропонованого алгоритму.

Маршрут. Як і в класичному ACO алгоритмі, один мураха – це простий агент, який вирішує всю задачу від початку до кінця. Такий самий принцип ми збережемо і зараз – один мураха послідовно проходить шлях спочатку одного комівояжера, потім "перестрибує" у невідвідане місто і починає проходити шлях другого комівояжера, і так далі, доки не будуть пройдені всі міста. Таким чином, маршрут мурахи складається з "піших переходів" та "стрибків", причому стрибків рівно m–1, тобто, шлях мурахи складається з m фрагментів, кожен з яких є шляхом одного комівояжера. Рис. 1 наводить приклад шляху, який утворюється мурахою.

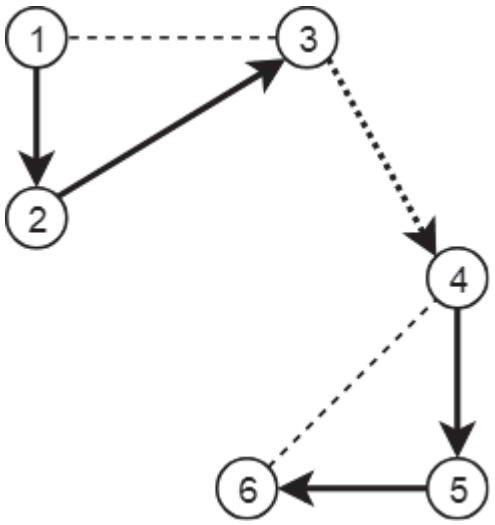


Рис. 1. Приклад побудови шляху мурахою

Переходи 1-2, 2-3, 4-5 та 5-6 – "звичайні", а перехід 3-4 – означає завершення шляху першого комівояжера та початок шляху наступного. Дуги 3-1 та 6-4 з'єднують начало та кінець сегментів безперервного пішого шляху мурахи та замикають шляхи комівояжерів. Здійснивши таку послідовність переходів, мураха утворив замкнені шляхи 1-2-3 та 4-5-6, які належать двом різним комівояжерам.

Феромон. У класичному ACO алгоритмі реалізується ідея феромону як спосіб непрямої взаємодії між мурахами колонії. У TSP мурахи залишають феромони на переходах (i, j) і коли черговий мураха колонії виявляється в i-й вершині, він вибирає один з наявних переходів з ймовірністю, пропорційною кількості феромону на доступних переходах. У запропонованому алгоритмі ідея феромону має такий же зміст – формування ступеня "привабливості" переходу для мурашки, але при ухваленні рішення здійснювати стрибок чи ні, значення рівня феромона використовується дещо в інший спосіб. Перебуваючи в певній вершині, мураха може продовжити шлях комівояжера, або замкнути його в петлю і перейти до формування маршруту наступного комівояжера. Кількість феромону між поточною вершиною і вершиною, в якій почався шлях комівояжера, і визначає "привабливість" рішення завершити шлях комівояжера і зробити стрибок в іншу вершину. Вершина, до якої здійснюється "стрибок", вибирається без участі феромону, але з використанням евристики, яка буде описана нижче.

Процедура генерації рішень. Як і у класичному алгоритмі, рішення генерується стохастично і поступово – на кожній ітерації випадково обирається наступне місто на шляху поточного комівояжера або перше місто на шляху наступного. Таким чином, якщо на певній ітерації маємо N невідвіданих міст, потрібно обрати одну з 2N дій. Ймовірнісний розподіл описується наступними формулами:

$$P_{move}(j) = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta} + \sum_{k=1}^{n} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}}, \quad (7)$$

$$P_{jump}(j) = \frac{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta}}{(\eta_{ig})^{\alpha} \cdot (\tau_{ig})^{\beta} + \sum_{k=1}^{n} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \times \times \frac{(d_{ij})^{\gamma}}{\sum_{k=1}^{n} (d_{ik})^{\gamma}}, \quad (8)$$

де i – індекс поточного міста; j – індекс невідвіданого міста; g – індекс міста, у якому розпочався шлях поточного комівояжера; d_{ij} – відстань між містами; τ_{ij} – кількість феромону між містами; α, β та γ – параметри алгоритму.

Нескладно переконатись, що сума всіх ймовірностей дорівнює одиниці, а сума ймовірностей "стрибків" дорівнює ймовірності переходу у місто-початок шляху комівояжера.

Застосування такого розподілу ніяк не гарантує, що стрибків буде рівно m–1. Тому необхідно розглянути два граничних випадки: 1) коли міст залишається не більше ніж залишилося комівояжерів – потрібно завжди здійснювати стрибок; 2) коли вже здійснено m–1 стрибків – потрібно завжди здійснювати піший перехід. Перший випадок гарантує, що у згенерованому рішенні буде не менш ніж m шляхів. При цьому місто для стрибка обирається довільно, оскільки це не впливає на якість рішення. Другий випадок гарантує, що у згенерованому рішенні буде не більше ніж m шляхів. При цьому місто для наступного переходу обирається за зміненим розподілом:

$$P_{jump}(j) = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\sum_{k=1}^{n} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}}. \quad (9)$$

Повну процедуру генерації рішення наведено на рис. 2.

Оновлення феромону. Феромон оновлюється періодично після здійснення декількох генерацій рішення – кількість генерацій між оновленнями є параметром алгоритму і зазвичай називається "кількістю мурах" за аналогією з природними мурашиними колоніями, де декілька мурах блукають у пошуках їжі одночасно. Оновлення феромону здійснюється за класичною формулою:

$$\Delta\tau = K\frac{1}{L},$$

де K – константа; L – довжина найдовшого шляху у згенерованому рішенні. Експериментально встановлено, що у якості константи K ефективно використовувати величину $1/Q$, де Q – кількість мурах на одну ітерацію алгоритму.

Параметри: $n, m, \|d_{ij}\|, \|\tau_{ij}\|, \alpha, \beta, \gamma$
$S \leftarrow \varnothing$
p ← [1]
C ← {2, ..., n}
WHILE C not empty:
  i ← p[length(p)]
  IF |S| < m:
    IF |C|+|S| > m:
      g ← p[1]
      jump, j ← обрати за розподілом (7, 8)
    ELSE:
      jump ← True
      j ← обрати будь-яке з C
    ENDIF
  ELSE:
    jump ← False
    j ← обрати за розподілом (9)
  IF jump:
    $S \leftarrow S \cup \{p\}$
    p ← [j]
  ELSE:
    append(p, j)
  ENDIF
  C ← C/{j}
ENDWHILE
$S \leftarrow S \cup \{p\}$
RETURN S

Рис. 2. Алгоритм генерації евристично-випадкового рішення

Матриця феромону оновлюється аналогічно до класичного алгоритму – приросщення отримують феромони на всіх переходах між суміжними містами всіх шляхів рішення, включаючи перехід між першим та останнім містом кожного шляху.

Оскільки під час виконання алгоритму генерація рішень виконується послідовно, потрібно накопичувати оновлення феромону в окремих буферних матрицях, та оновлювати матриці феромонів лише тоді, коли буде здійснено всі Q генерацій.

Повний алгоритм. Об'єднуючи ідеї, викладені у попередніх підрозділах, наведемо повний алгоритм вирішення mTSP без депо (рис. 3). У якості критерію припинення обрано виконання заданої кількості ітерацій. Результатом роботи алгоритму є найкраще евристично-випадкове рішення, яке було згенероване за весь час роботи алгоритму.

Параметри: $n, m, \|d_{ij}\|, \alpha, \beta, \gamma, \varepsilon, K, Q, n_{iter}$

$\|\tau_{ij}\| \leftarrow \dfrac{1}{n \cdot \text{mean}(\|d_{ij}\|)}$

$L_{best} \leftarrow \infty$

$S_{best} \leftarrow \varnothing$

REPEAT n_{iter} times:

  $\|\Delta\tau_{ij}\| \leftarrow$ нульова матриця n x n

  REPEAT Q times:

    S ← згенерувати рішення за Алгоритмом 1

    L ← довжина найдовшого шляху у S

    IF $L < L_{best}$:

      $S_{best} \leftarrow S$

      $L_{best} \leftarrow L$

    ENDIF

    FOR p in S:

      FOR i, j in усі послідовні пари міст у p:

        $\Delta\tau_{ij} \leftarrow \Delta\tau_{ij} + \dfrac{K}{L}$

        $\Delta\tau_{ji} \leftarrow \Delta\tau_{ji} + \dfrac{K}{L}$

      ENDFOR

    ENDFOR

  ENDREPEAT

  FOR i in {1,...,n}:

    FOR j in {1,...,n}/{i}:

      $\tau_{ij} \leftarrow (1-\varepsilon)\tau_{ij} + \Delta\tau_{ij}$

    ENDFOR

  ENDFOR

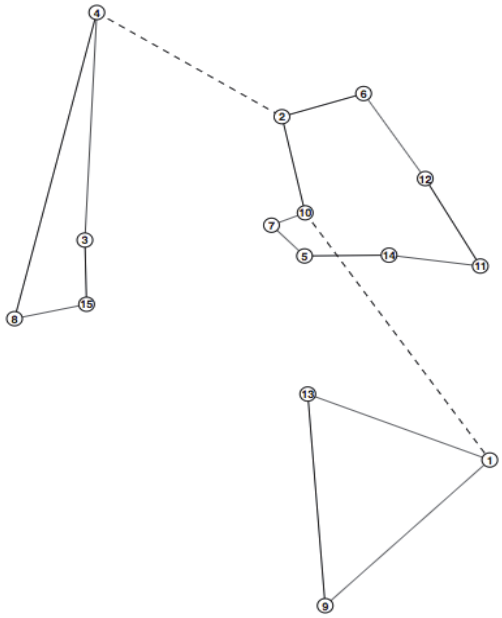
ENDREPEAT

RETURN S_{best}

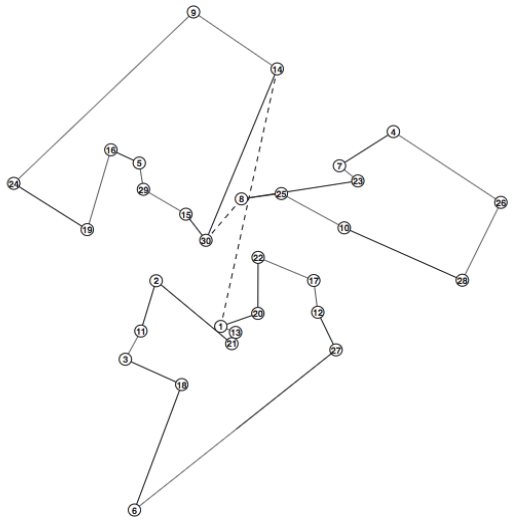
Рис. 3. Алгоритм вирішення задачі mTSP без депо методом оптимізації мурашиної колонії

З метою експериментальної оцінки якості алгоритму та підбору ефективних параметрів було згенеровано декілька випадкових наборів точок на двовимірній площині. Координати точок згенеровані за допомогою нормального розподілу з нульовим середнім та одиничною дисперсією. Матрицю відстаней між вершинами графу визначено як матрицю Евклідових відстаней між точками. Алгоритм тестувався на графах з 15 та 30 вузлами для вирішення задачі з трьома комівояжерами. На рис. 4 зображені рішення, знайдені для графів з 15 та 30 вузлами.

На рис. 4 видно, що розроблений алгоритм виконав розбиття вершин графа на кластери і побудував маршрут переміщення комівояжерів для кожного кластера. Рішення, близьке до оптимального, можна отримати шляхом настройки параметрів алгоритму $\alpha, \beta, \gamma, \varepsilon, n_{iter}, Q$ з урахуванням специфікації розв'язуваної задачі.



а



б

Рис. 4. Результат розв'язання mTSP без депо з 3-ма комівояжерами (а – граф з 15 вершинами, б – граф з 30 вершинами)

Висновки

У статті розглянута симетрична задача декількох комівояжерів без депо. Для її наближеного рішення запропонований алгоритм на базі метаевристики мурашиної колонії. Проведена серія експериментів на випадкових графах даних з 15-ю і 30-ю вершинами у задачі з трьома комівояжерами показала задовільні результати.

Розроблений алгоритм демонструє, яким чином можна узагальнювати ідею оптимізації мурашиних колоній на задачі з іншими/додатковими умовами. Це може бути основою для подальшого узагальнення та систематизації мурашиних алгоритмів, та навіть створення формального методу генерації алгоритмів мурашиних колоній для наперед заданої задачі комбінаторної оптимізації.

Список літератури

1. Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures / T. Bektas // Omega. – 2006. – № 34. – P. 209-219. https://doi.org/10.1016/j.omega.2004.10.004.

2. Matai R. Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches / R. Matai, S. Singh, M.L. Mittal // Traveling Salesman Problem, Theory and Applications. – 2010. – P. 1-24. https://doi.org/10.5772/12909.

3. Sofge D. Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema / D. Sofge, A. Schultz, K. Jong // Lecture Notes in Computer Science. – 2002. – № 2279. – P. 153-162.

4. Király A. Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm / A. Király, J. Abonyi // Intelligent Computational Optimization in Engineering. – 2011. – P. 241-269. https://doi.org/10.1007/978-3-642-21705-0_9.

5. Kaempfer Y. Learning the Multiple Traveling Salesmen Problem with Permutation Invariant Pooling Networks / Y. Kaempfer, L. Wolf // arXiv preprint arXiv:1803.09621 – 2018.

6. Solving Multiple Traveling Salesman Problem using the Gravitational Emulation Local Search Algorithm / A.S. Rostami, F. Mohanna, H. Keshavarz, A.A.R. Hosseinabadi // Applied Mathematics & Information Sciences. – 2015. – № 9. – P. 1-11.

7. Baranwal M. Multiple traveling salesmen and related problems: A maximum-entropy principle based approach / M. Baranwal, B. Roehl, S.M. Salapaka // 2017 American Control Conference (ACC). – 2017. – P. 3944-3949. https://doi.org/10.23919/ACC.2017.7963559.

8. Hou M.S. A novel method for solving the multiple traveling salesmen problem with multiple depots / M.S. Hou, D.B. Liu // Chin Sci Bull. – 2012. – № 57. – P. 1886-1892. https://doi.org/10.1007/s11434-012-5162-7.

9. Multi-type ant colony system for solving the multiple traveling salesman problem / Y.J. Costa, R.A. Ledón, N.I.C. Machado, A. Nowe // Revista Técnica de la Facultad de Ingeniería Universidad del Zulia. – 2012. – № 35. – P. 311-320.

10. Junjie P. An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem / P. Junjie, W. Dingwei // Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06). – 2006. https://doi.org/10.1109/ICICIC.2006.40.

11. An Ant Colony Optimization Algorithm for the Multiple Traveling Salesmen Problem / W. Liu, S. Li, F. Zhao, A. Zheng // Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications. – 2009. – P. 1533-1537. https://doi.org/10.1109/ICIEA.2009.5138451.

12. Dorigo M. Ant colony optimization theory: A survey / M. Dorigo, C. Blum // Theor. Comput. Sci. – 2005. – № 344. – P. 243-278. https://doi.org/10.1016/j.tcs.2005.05.020.

References

1. Bektas, T. (2006), The multiple traveling salesman problem: an overview of formulations and solution procedures, *Omega*, No. 34, pp. 209-219. https://doi.org/10.1016/j.omega.2004.10.004.

2. Matai, R., Singh, S. and Mittal, M.L. (2010), Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches, *Traveling Salesman Problem, Theory and Applications*, pp. 1-24. https://doi.org/10.5772/12909.

3. Sofge, D., Schultz, A. and Jong, K. (2002), Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema, *Lecture Notes in Computer Science*, No. 2279, pp. 153-162.

4. Király, A. and Abonyi, J. (2011), Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm, *Intelligent Computational Optimization in Engineering*, pp. 241-269. https://doi.org/10.1007/978-3-642-21705-0_9.

5. Kaempfer, Y. and Wolf, L. (2018), Learning the Multiple Traveling Salesmen Problem with Permutation Invariant Pooling Networks, *arXiv preprint arXiv:1803.09621*.

6. Rostami, A.S., Mohanna, F., Keshavarz, H. and Hosseinabadi, A.A.R. (2015), Solving Multiple Traveling Salesman Problem using the Gravitational Emulation Local Search Algorithm, *Applied Mathematics & Information Sciences*, No. 9, pp. 1-11.

7. Baranwal, M., Roehl, B. and Salapaka, S.M. (2017), Multiple traveling salesmen and related problems: A maximum-entropy principle based approach, *2017 American Control Conference (ACC)*, pp. 3944-3949. https://doi.org/10.23919/ACC.2017.7963559.

8. Hou, M.S. and Liu, D.B. (2012), A novel method for solving the multiple traveling salesmen problem with multiple depots, *Chin Sci Bull*, No. 57, pp. 1886-1892. https://doi.org/10.1007/s11434-012-5162-7.

9. Costa, Y.J., Ledón, R.A., Machado, N.I.C. and Nowe, A. (2012), Multi-type ant colony system for solving the multiple traveling salesman problem, *Revista Técnica de la Facultad de Ingeniería Universidad del Zulia*, No. 35, pp. 311-320.

10. Junjie, P. and Dingwei, W. (2006), An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem, *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)*, https://doi.org/10.1109/ICICIC.2006.40.

11. Liu, W., Li, S., Zhao, F. and Zheng, A. (2009), An Ant Colony Optimization Algorithm for the Multiple Traveling Salesmen Problem, *Proceedings of the 4th IEEE Conference on Industrial Electronics and Applications*, pp. 1533-1537. http://doi.org/10.1109/ICIEA.2009.5138451.

12. Dorigo, M. and Blum, C. (2005), Ant colony optimization theory: A survey, *Theor. Comput. Sci.*, No. 344, pp. 243-278. https://doi.org/10.1016/j.tcs.2005.05.020.

Надійшла до редколегії 05.06.2019
Схвалена до друку 13.08.2019

Відомості про авторів:

Тимчук Олег Сергійович
кандидат технічних наук доцент
завідувач кафедри
Університету економіки та права "КРОК",
Київ, Україна
https://orcid.org/0000-0002-9046-8015

Проценко Ярослав Андрійович
магістр Донецького національного університету
ім. В. Стуса,
Вінниця, Україна
https://orcid.org/0000-0001-6188-3298

Парамонов Антон Іванович
кандидат технічних наук доцент
доцент кафедри Донецького національного
університету ім. В. Стуса,
Вінниця, Україна
https://orcid.org/0000-0001-6616-248

Information about the authors:

Oleg Tymchuk
Ph.D. Associate Professor
Head of Department
of University "KROK",
Kyiv, Ukraine
https://orcid.org/0000-0002-9046-8015

Yaroslav Protsenko
Postgraduate Student of Vasyl' Stus
Donetsk National University,
Vinnitsa, Ukraine
https://orcid.org/0000-0001-6188-3298

Anton Paramonov
Ph.D. Associate Professor
Senior Lecturer of Vasyl' Stus
Donetsk National University,
Vinnitsa, Ukraine
https://orcid.org/0000-0001-6616-248

ПРИМЕНЕНИЕ АЛГОРИТМА МУРАВЬИНОЙ КОЛОНИИ К РЕШЕНИЮ ЗАДАЧИ НЕСКОЛЬКИХ КОММИВОЯЖЕРОВ БЕЗ ДЕПО

О.С. Тимчук, Я.А. Проценко, А.И. Парамонов

В статье предложен алгоритм решения NP-полной задачи нескольких коммивояжеров без депо, которая является обобщением "стандартной" задачи коммивояжера. В основу алгоритма положена метаэвристика муравьиной колонии – муравьи, используя различные типы феромонов, пытаются оптимально разбить граф на кластеры и оптимизировать маршрут внутри каждого кластера. Приведены результаты эксперимента, проведенного на графах с 15 и 30 узлами для решения задачи с тремя коммивояжерами. Разработанный алгоритм демонстрирует возможность обобщения оптимизации муравьиных колоний на задачи с дополнительными условиями.

Ключевые слова: *задача коммивояжера, задача нескольких коммивояжеров без депо, муравьиный алгоритм, TSP, mTSP, ACO.*

APPLICATION OF ANT COLONY OPTIMIZATION TO SOLVING MULTIPLE TRAVELLING SALESMAN PROBLEM WITHOUT DEPOT

O. Tymchuk, Y. Protsenko, A. Paramonov

The paper presents the algorithm for solving the NP-complete problem of multiple travelling salesman problem without depot, which is a generalization of the "standard" traveling salesman problem and consists in finding disjoint closed routes on the graph, with each vertex belonging to exactly one route, and the cost of the longest path should be minimal. The proposed algorithm is based on metaheuristics of an ant colony - ants, using various types of pheromones, try to optimally split the graph into clusters using a pheromone of the first type, and optimize the route inside each cluster using a pheromone of the second type. Dividing the ant route into clusters is ensured by introducing two types of ant movement between the vertices of the graph: "foot transitions" and "jumps". Being at a certain vertex of the graph, the ant can continue its path using the "foot transition", or close it into a loop and proceed to the formation of a new cluster using the "jump". The balance between the two types of pheromones is used as a heuristic choice of which type of movement the ant uses. The description of the proposed algorithm in the paper is performed using pseudocode. The algorithm is tested on the data of locations of most populous cities of Ukraine. Tests show that the algorithm is able to find quite good suboptimal solutions for problems of small sizes ($n \leq 12$). The algorithm demonstrates a way how to generalize ant colony optimization approach while keeping spirit of classic algorithm. It may be the basis for further systematization of ant colony optimization approach and even for development of generic algorithm of generation ACO algorithm for given combinatorial optimization problem.

Keywords: *the traveling salesmen problem, the multiple traveling salesmen problem without depot, ant colony optimization algorithm, ant colony optimization metaheuristic.*