# КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 004.94 DOI:10.34229/2707-451X.22.1.2

В.В. АНДРІЙЧУК, В.В. ТРЕТИНИК

## ДОСЛІДЖЕННЯ ПРИРОДНИХ АЛГОРИТМІВ ТА ЇХ КОМПЛЕКСНЕ ЗАСТОСУВАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЗАВДАНЬ ЛОГІСТИКИ

*Розглянуто вирішення логістичних проблем за допомогою сучасних методів машинного навчання, а саме природних алгоритмів на прикладі алгоритму мурашиної колонії. Також було запропоновано використання алгоритмів оптимізації для пошуку оптимальних гіпер-параметрів алгоритму мурашиної колонії для прискорення обчислень.*

***Ключові слова:*** *природні алгоритми, логістика, мурашиний алгоритм, мета-алгоритм.*



На сьогоднішній день лінійні алгоритми перестали задовольняти потреби людства у багатьох прикладних сферах діяльності, тому їм на зміну прийшло багато інших методів. Частина з таких методів базуються на встановленні подібності між світом природи і машинним інтелектом – наприклад, це стосується областей нейронних мереж, генетичних алгоритмів, природніх алгоритмів та інших.

Мета роботи – підвищення ефективності використання природних алгоритмів у задачах логістики. Загалом, основними задачами логістики є прогнозування кількості витрат, продукції та ресурсів за певних обставин. Проте не завжди в логістичних задачах наявна потрібна кількість даних матеріальних, інформаційних, фінансових, сервісних та інших потоків, багато змінних є невідомими. Тоді на допомогу приходять евристичні методи алгоритмів для вирішення таких задач прикладного характеру.

Також відома проблема пошуку оптимальних параметрів для природних алгоритмів. Тому основна мета цієї роботи – розробити мета-алгоритм, який автоматизує пошуки оптимальних параметрів для алгоритму мурашиної колонії (як одного з природних алгоритмів).

**Алгоритм мурашиної колонії**

Алгоритм оптимізації колоній мурах [1] — це імовірнісний метод розв'язування обчислювальних задач, який може зменшити їх розмірність, щоб знайти оптимальні шляхи за допомогою графів.

У реальному світі мурахи спочатку блукають випадковим чином, а знайшовши їжу, повертаються в свою колонію, прокладаючи феромонні сліди. Якщо інші мурахи знайдуть такий шлях, вони, швидше за все, не будуть продовжувати подорожувати навмання, а натомість підуть слідом; повертаючи та підкріплюючи його, якщо врешті знайдуть їжу.

Оригінальна ідея походить від спостереження за експлуатацією харчових ресурсів мурахами, коли індивідуально обмежені когнітивні здібності мурах разом змогли знайти найкоротший шлях між джерелом їжі та гніздом.

Опишемо процес вибору оптимального маршруту мурахою (рис.1):

1. Перша мураха знаходить джерело їжі (F) будь-яким способом (a), а потім повертається до гнізда (N), залишивши за собою стежку з феромонів (b).

2. Потім мурахи вибирають один із чотирьох можливих шляхів, зміцнюють його і роблять привабливим.

3. Мурахи вибирають найкоротший маршрут, тому що феромони з довших шляхів швидше випаровуються.

У серії експериментів на колонії мурах з вибором між двома шляхами неоднакової довжини, що ведуть до джерела їжі, біологи помітили, що мурахи, як правило, використовують найкоротший шлях. Модель, що пояснює таку поведінку, виглядає так:

– мураха (так звана «бліц») бігає більш-менш навмання навколо колонії;
– якщо вона виявляє джерело їжі, то повертається більш-менш прямо в гніздо, залишаючи на своєму шляху слід феромонів;
– ці феромони привабливі, мурахи, що знаходяться поблизу, будуть схильні йти більш-менш прямо, слідом;
– повернувшись до колонії, ці мурахи зміцнять маршрут;
– якщо до одного джерела їжі можна дістатися двома шляхами, то коротшим за той же час пройде більше мурах, ніж довгим;
– короткий маршрут буде все більше розширюватися, а отже ставати більш привабливим;
– довгий шлях з часом зникне, феромони нестабільні;
– зрештою, всі мурахи визначилися і тому «обрали» найкоротший шлях.

На рис. 1 показано схему знаходження мурахами найкоротшого шляху між гніздом та їжею.

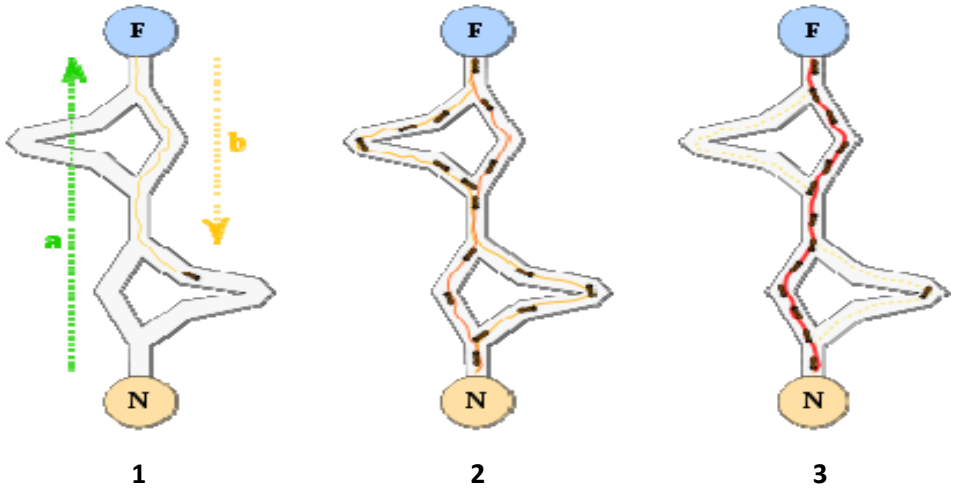


РИС. 1. Схема пошуку найкоротшого шляху між гніздом та їжею

**Пошук оптимальних параметрів алгоритму мурашиної колонії**

Загалом, алгоритм оптимізації колоній мурах – вже досить відомий алгоритм з досить відомою перевагою роботи над алгоритмом повного перебору. Далі наведена порівняльна таблиця складності цих двох алгоритмів, де $n$ – кількість міст, $m$ – чисельність популяції, $t$ – кількість здійснених ітерацій.

ТАБЛИЦЯ

| **Алгоритм** | **Складність** |
|---|---|
| Повний перебір | O(n!) |
| Мурашина колонія | $O(t * m * n^2)$ |

З формули вище ми бачимо, що складність алгоритму мурашиної колонії залежить від трьох факторів (назвемо їх гіпер-параметрами) – кількості міст, чисельності популяції та кількості здійснених ітерацій. На фактор кількості міст ми впливати не можемо, тому що це вхідний заданий параметр, а от кількість мурах у популяції та кількість здійсненних ітерацій дослідник визначає самостійно. Як видно з формули, зменшуючи кількість мурах та ітерацій, ми прискорюємо отримання результату (робимо алгоритм «швидшим»). Ми розуміємо, що параметри не можуть бути однаковими для різних наборів даних, тому дослідник кожного разу у ручному режимі витрачає час на пошуки оптимальних гіпер-параметрів. Автоматизація цього процесу є перспективним напрямком, над якими працюють дослідники [2].

У даній роботі розглядаються вже відомі методи пошуку гіпер-параметрів з «класичного» машинного навчання - GridSearch, RandomSearch. Bayesian optimization – проаналізуємо їх переваги і застосуємо до алгоритму мурашиної колонії для автоматизації пошуку оптимальних гіпер-параметрів.

У GridSearch (рис.2) ми пробуємо кожну комбінацію заданого списку значень гіпер-параметрів та оцінюємо модель кожної комбінації. Рисунок схожий на сітку, де всі значення розташовані у вигляді матриці. Кожен набір параметрів береться до уваги і фіксується точність. Після оцінки всіх комбінацій модель з набором параметрів, що забезпечують максимальну точність, вважається найкращою.

Один з основних недоліків пошуку по сітці полягає у тому, що коли справа доходить до розмірності, вона страждає, коли кількість гіпер-параметрів зростає експоненційно. За наявності лише чотирьох параметрів ця проблема може стати непрактичною, оскільки кількість оцінок, необхідних для цієї стратегії, збільшується експоненційно з кожним додатковим параметром через «прокляття» розмірності.

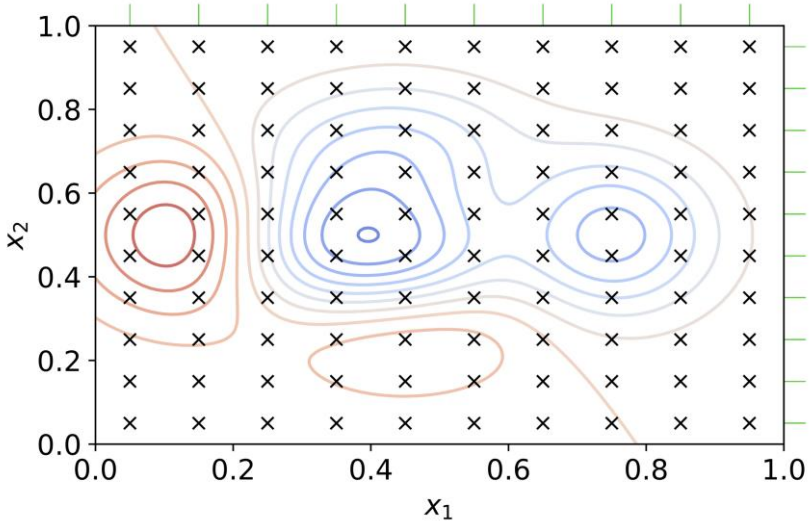


РИС. 2. GridSearch

RandomSearch – це метод, у якому випадкові комбінації гіпер-параметрів використовуються для пошуку найкращого рішення для побудованої моделі. Він пробує випадкові комбінації діапазону значень. Для оптимізації за допомогою випадкового пошуку функція обчислюється за певної кількості випадкових конфігурацій у просторі параметрів.

RandomSearch (рис. 3) найкраще працює для даних нижчої розмірності, тому що час, необхідний для знаходження правильної множини, менше за меншої кількості ітерацій. RandomSearch є найкращим методом пошуку параметрів за меншої кількості вимірювань. У статті «Random Search for Hyper-Parameter Optimization» за авторством Bergstra and Bengio емпірично і теоретично показується, що RandomSearch ефективніший для оптимізації параметрів, ніж пошук по решітці.

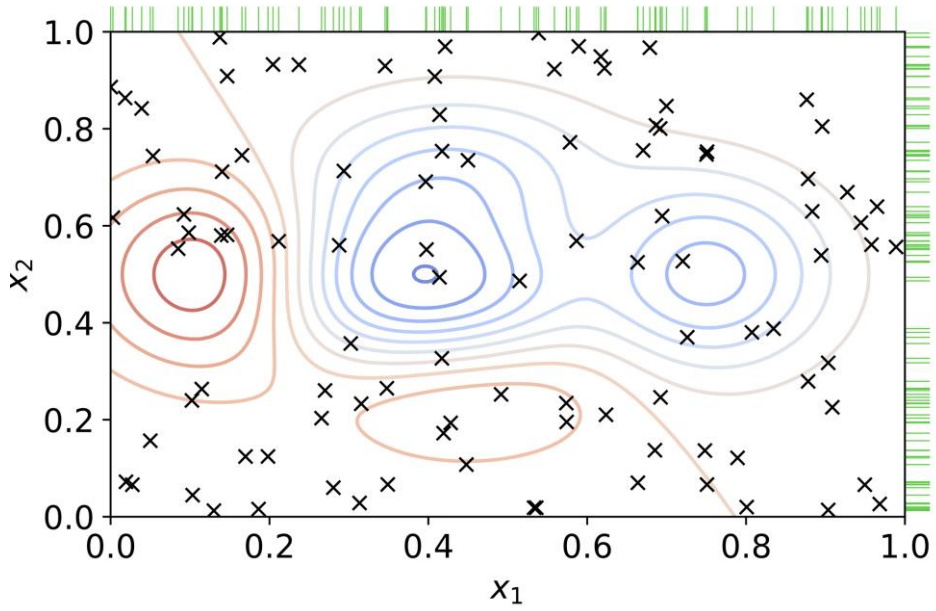


РИС. 3. RandomSearch

Bayesian optimization (баєсівська оптимізація) – це метод глобальної оптимізації для невідомої функції (чорної скриньки) з шумом (рис. 4). Застосована до гіперпараметричної оптимізації байєсівська оптимізація будує стохастичну модель функції відображення значень гіпер-параметра в цільову функцію, застосовану на множині перевірок. Шляхом ітеративного застосування перспективної конфігурації гіпер-параметрів, заснованої на поточній моделі, а потім її оновлення, байєсівська оптимізація прагне зібрати як найбільше інформації про цю функцію і, зокрема, місце оптимуму. Метод намагається збалансувати зондування (гіпер-параметри, для яких зміна найменш достовірно відома) та використання (гіпер-параметри, які, як очікується, найбільш близькі до оптимуму). Якщо коротко, то методи подібні до байєсівської оптимізації намагаються аналізувати раніше отримані дані, щоб оцінити, яку комбінацію гіперпараметрів краще досліджувати наступною. Насправді байесівська оптимізація показала [3 – 6] кращі результати з меншими обчисленнями у порівнянні з GridSearch та RandomSearch зважаючи на можливості судження якості експериментів ще до виконання.

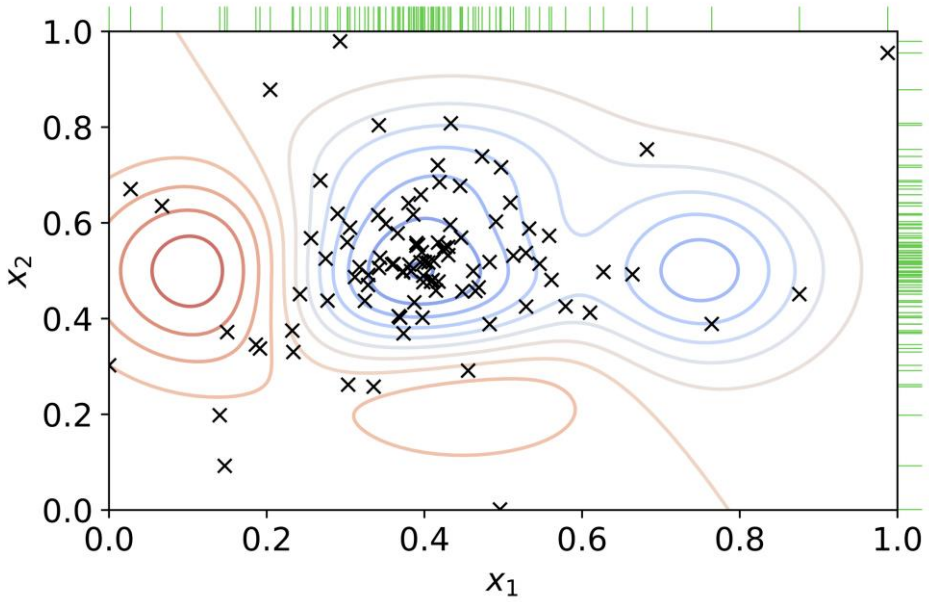


РИС. 4. Баєсівська оптимізація

**Мета-алгоритм**

У даній роботі пропонується поєднати два алгоритми – алгоритм мурашиної колонії та один з алгоритмів пошуку оптимальних параметрів – у один, створивши мета-алгоритм, який автоматично буде знаходити оптимальні параметри для алгоритму мурашиної колонії кожного разу, коли на вхід подаються параметри міст (рис. 5). Тобто, ми виключаємо дорогоцінний час дослідника на пошуки параметрів кожного разу, а також прискорюємо отримання результату за рахунок використання сучасних підходів пошуку цих параметрів.

Як алгоритм для пошуку оптимальних параметрів було використано Bayesian optimization (ми вже знаємо, що він дає кращі результати з меншими обчисленнями).

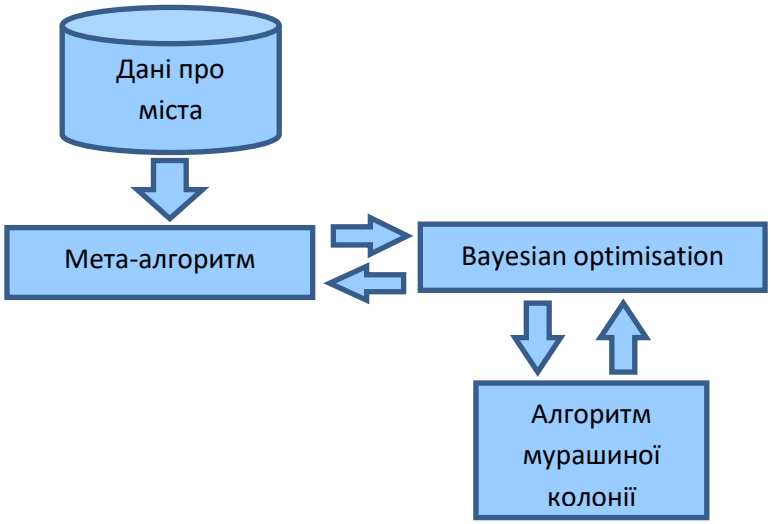


РИС. 5. Блок-схема роботи мета-алгоритму

Кожного разу, коли в мета-алгоритм надходять дані про міста, для яких потрібно побудувати оптимальних шлях переміщення (під оптимальним наразі ми розглядаємо найкоротший шлях, але взагалі ми можемо використовувати будь-яку цільову функцію), алгоритм Bayesian optimization бере параметри з заданої дослідником множини параметрів і ініціює запуск алгоритму мурашиної колонії, а потім отриманий результат (довжину шляху) порівнює з попереднім результатом, намагаючись мінімізувати довжину (отримати найкоротший шлях).

Код розроблено на мові програмування Python. Для роботи алгоритму мурашиної колонії був розроблений власний код на Python, а для Bayesian optimization було використану відому і вже протестовану бібліотеку hyperopt.

Для експерименту були згенеровані координати міста:

```
7.87549553010200 6869e-01,4.77880665114260 4167e-01
4.83182407710820 4385e-01,8.22276056348386 2972e-01
5.83963730336016 7842e-01,2.93581993224323 7673e-01
4.50175380722013 6849e-01,9.63900878949615 2918e-01
9.23602127207787 2268e-01,3.49164529337956 4974e-01
9.93521214030928 9675e-01,2.83915797245934 7423e-01
1.97929471978056 8812e-01,5.70574712995810 3070e-01
1.38383655145387 0856e-01,7.28438250368207 9735e-01
2.22150355118101 5501e-01,6.60852343169721 7454e-01
8.75750105650544 6746e-01,6.91441975084124 3051e-01
```

Між ними були знайдені відстані і ці відстані подані у мета-алгоритм.

Результат роботи мета-алгоритму для пошуку оптимальних параметрів для алгоритму мурашиної колонії:

| | loss | params | status | max_iter | size_pop |
|---|---|---|---|---|---|
| 0 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 38 | 2.549817 | {'max_iter': 11, 'size_pop': 16} | ok | 11 | 16 |
| 22 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 23 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 24 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 25 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 29 | 2.549817 | {'max_iter': 31, 'size_pop': 16} | ok | 31 | 16 |
| 18 | 2.549817 | {'max_iter': 21, 'size_pop': 36} | ok | 21 | 36 |
| 30 | 2.549817 | {'max_iter': 26, 'size_pop': 46} | ok | 26 | 46 |
| 32 | 2.549817 | {'max_iter': 11, 'size_pop': 16} | ok | 11 | 16 |
| 33 | 2.549817 | {'max_iter': 26, 'size_pop': 46} | ok | 26 | 46 |
| 34 | 2.549817 | {'max_iter': 36, 'size_pop': 21} | ok | 36 | 21 |
| 35 | 2.549817 | {'max_iter': 11, 'size_pop': 16} | ok | 11 | 16 |
| 36 | 2.549817 | {'max_iter': 26, 'size_pop': 46} | ok | 26 | 46 |
| 37 | 2.549817 | {'max_iter': 36, 'size_pop': 21} | ok | 36 | 21 |
| 31 | 2.549817 | {'max_iter': 31, 'size_pop': 11} | ok | 31 | 11 |
| 17 | 2.549817 | {'max_iter': 21, 'size_pop': 41} | ok | 21 | 41 |

Підставивши отримані параметри у алгоритм мурашиної колонії, отримуємо оптимальний маршрут між точками (рис. 6).

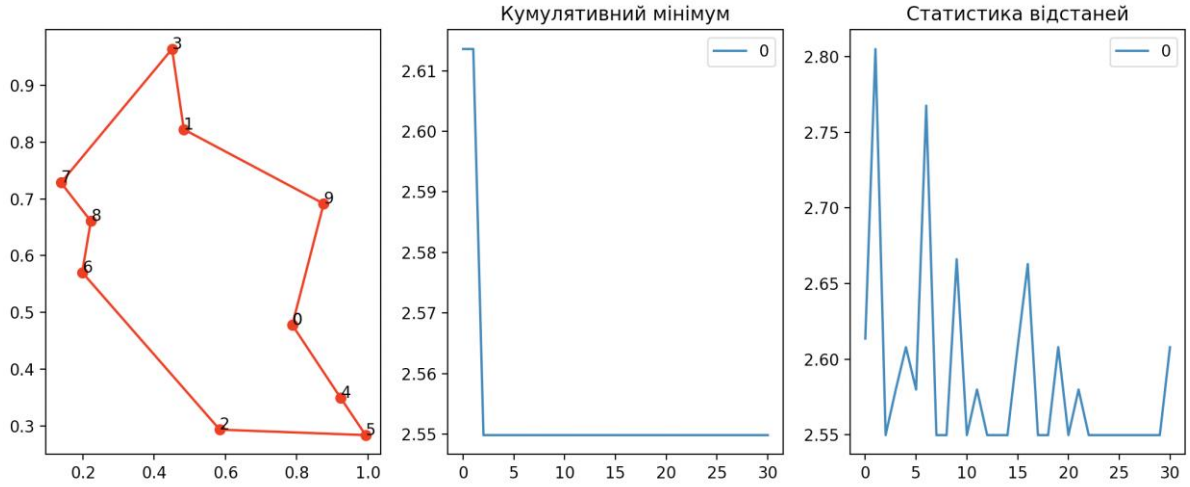


РИС. 6. Результат роботи мета-алгоритму для пошуку оптимальних параметрів для алгоритму мурашиної колонії

Як ми бачимо з результатів мета-алгоритму, ми отримали декілька варіантів параметрів, коли наш алгоритм отримує оптимальний маршрут. Підставивши значення кількості ітерацій 31, у розмір популяції – 11 (тобто самий верхній з параметрів), ми бачимо на рис. 6, що оптимальний результат був отриманий вже на 4 ітерації, а тому не було сенсу перебирати інші. Це значить, що ми можемо ще оптимізувати наш мета-алгоритм і навчити його обирати мінімальні значення параметрів для тих випадків, коли ми отримали однакові відстані. Або можемо навчити його опиратися на час, відведений на роботу алгоритму мурашиної колонії. І якщо оптимізація по відстані буде незначна, то можемо брати параметри, які дадуть результат за більш короткий проміжок часу. Тобто, є горизонт для подальших покращень мета-алгоритму, що є предметом наших майбутніх досліджень.

**Висновок.** У роботі досліджено ряд природніх алгоритмів, зокрема під час вирішення задач логістичних поставок, здійснено аналіз роботи мурашиного алгоритму, проведено його комплексний аналіз, реалізовано мета-алгоритм і побудовано відповідні графіки.

**Список літератури**

1. Dorigo M. “Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimization, Learning, and Natural Algorithms)”, дисертація на здобуття вченого ступеня “Doctorate in Systems and Information Electronic Engineering”, Politecnico di Milano, 1992.
2. Stützle T., López-Ibáñez M., Pellegrini P., Maur M., M. de Oca, Birattari M., Maur M., Dorigo M., “Parameter Adaptation in Ant Colony Optimization”. Technical Report, IRIDIA, Université Libre de Bruxelles, 2010.
3. Hutter F., Hoos H.H., Leyton-Brown K. Sequential Model-Based Optimization for General Algorithm Configuration. Proceedings of the conference on Learning and Intelligent OptimizatioN (LION 5). Rome, Italy: Springer-Verlag, 2011.
4. Bergstra J., Bardenet R., Bengio Y., Kegl B. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*. 2011.
5. Snoek J., Larochelle H., Adams R. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*. 2012. Bibcode: 2012arXiv1206.2944S. arXiv:1206.2944.

6. Thornton C., Hutter F., Hoos H., Leyton-Brown K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Knowledge Discovery and Data Mining*. 2013. Bibcode: 2012arXiv1208.3719T. arXiv:1208.3719.

Одержано 04.05.2022

**Андрійчук Віктор Васильович**,
магістрант кафедри прикладної математики
НТУУ «КПІ ім. Ігоря Сікорського»

**Третиник Віолета Вікентіївна**,
кандидат фізико-математичних наук, доцент кафедри прикладної математики
НТУУ «КПІ ім. Ігоря Сікорського»
https://orcid.org/0000-0002-3538-8207
viola.tret@gmail.com

UDC 004.94

**Victor Andriichuk, Violeta Tretynyk** *

## On Investigation of Natural Algorithms and Their Complex Application for Optimization of Logistics Tasks

*The National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"*
* *Correspondence: viola.tret@gmail.com*

**Introduction.** At present, with the increase in the number of data and the complexity of various processes in many branches of the national economy, the classical methods of answering have ceased to meet the needs. So, for example, if you consider the logistics industry (this and transportation, and various cartographic services for consumers), we see how dynamically developing these areas. To construct optimal routes, if we continued to use old algorithms, we would have to spend a huge amount of computer resources and time to get results. Instead, the study finds new methods that make it possible to speed up and simplify the solution of problems. So, for example, for logistics problems, instead of classical combinatorial problems, algorithms are being replaced by environmental scientists. They were called natural algorithms. For example, scientists have noticed that ants find the shortest route to food using the iteration and intensity of a pheromone trace that leaves previous ants who are moving along the path. This idea formed the basis of the algorithm of the ant colony. Even if in reality ants find the optimal way more difficult than the simplified idea of researchers, it turned out that this simplification was enough to find an optimal route between some points of the type like ants looking for an optimal route food.

To develop a complex meta-algorithm that would solve the problem of logistics network, the following methods are taken into account: natural algorithms, salesman problem, dynamic programming methods, combinatorial approaches, algorithms for complex data analysis.

This paper considers the application of methods of natural algorithms to solve the problem of coordinated logistics.

**The purpose of the paper**. The aim of the work is to increase the efficiency of using natural algorithms in logistics problems. In general, the main tasks of logistics are to forecast the amount of costs, products and resources under certain circumstances. However, the required amount of material, information, financial, service and other data flows is not always available in logistics tasks, many variables are unknown. Then heuristic methods of algorithms come to the rescue to solve such problems of an applied nature.

**Results**. The software implementation of the model in the Python programming language was performed. Some methods of dynamic programming, fuzzy logic were used and the hyperopt library was used to implement the script.

**Keywords:** natural algorithms, logistics, ant algorithm, meta-algorithm.