УДК 519.863+62-23 **doi: 10.20998/2079-0775.2023.1.01**

О. В. БОНДАРЕНКО, О. В. УСТИНЕНКО, В. І. СЄРИКОВ

МЕТАЕВРИСТИЧНІ АЛГОРИТМИ. МЕТАФОРИ-СТРАТЕГІЇ (ОГЛЯДОВА СТАТТЯ)

Описано актуальність освітлення сучасних метаевристичних алгоритмів, освітлено ряд термінів та взаємозв'язків між ними, необхідність проведення класифікації, а також метафор, що використовуються для опису алгоритмів. Це дає змогу зрозуміти необхідність висвітлення вказаної теми та проведення досліджень літературних джерел стосовно питання. Розглянуто ряд термінів і категорій, а також взаємозв'язків між ними, що дало змогу запропонувати класифікацію метаевристичних алгоритмів. Запропоновано підхід до класифікації метаевристичних алгоритмів, що базується на термінах та поділі категорій, взятих з природничих наук. Відповідно до назв класів відбувається і їх наповнення. Це дає змогу об'єднати певний сегмент знань у кластер з єдиною термінологією. Розглянуто категорію «метафора» та її функції при формуванні метаевристичних алгоритмів, це дало змогу глибше зрозуміти можливості використання метафор у науковій діяльності та сформувати перелік вимог до них при описі алгоритмів. Проведено огляд цікавих, оригінальних та різноманітних метаевристичних алгоритмів, що дало змогу зрозуміти сучасні тенденції стосовно цього питання, визначити переваги та недоліки алгоритмів, а також зрозуміти та сформувати роль метафори при їх створенні чи описі. Також, огляд дає можливість виділити два інтелектуальних напрямки використання метафор: допомога підвищення розуміння та інтенсифікація донесення ідеї до цільової аудиторії вже розробленого алгоритму чи стратегії та розробка нових алгоритмів чи стратегій пошуку оптимальних параметрів. Розглянуто систему формування та оформлення нового знання і ролі метафори у ній. Сформовано системний трикутник «ідея-алгоритм-метафора», а також можливі шляхи розвитку в цій системі, що дає можливість визначення чи вибору розробником певного шляху та наступних його етапів.

Ключові слова: метаевристичний алгоритм, метафора, оптимізація.

O. BONDARENKO, O. USTYNENKO, V. SIERYKOV

METAHEURISTIC ALGORITHMS. METAPHORS-STRATEGIES (REVIEW ARTICLE)

The relevance of modern metaheuristic algorithms, clarification of a number of terms and relationships between them, the need for classification, as well as metaphors used to describe algorithms are described. This makes it possible to understand the need to cover the specified topic and conduct research on literary sources related to the issue. A number of terms and categories and their interrelationships were considered, which made it possible to propose a classification of metaheuristic algorithms. An approach to the classification of metaheuristic algorithms based on the terms and division of categories taken from the natural sciences is proposed. According to the names of the classes, their filling takes place. This makes it possible to combine a certain segment of knowledge into a cluster with a single terminology. Considering the category of "metaphor" and its functions in the formation of metaheuristic algorithms, it made it possible to gain a deeper understanding of the possibilities of using metaphors in scientific activity and to form a list of requirements for them when describing algorithms. An overview of interesting, original and diverse metaheuristic algorithms was conducted, which made it possible to understand modern trends in this issue, to determine the advantages and disadvantages of algorithms, as well as to understand and shape the role of metaphor in their formation or description. Also, the review makes it possible to distinguish two intellectual directions of using metaphors: helping to increase understanding and intensifying the delivery of the idea to the target audience of an already developed algorithm or strategy and the development of new algorithms or strategies for finding optimal parameters. The system of formation and design of new knowledge and the role of metaphor in it are considered. The system triangle "idea-algorithm-metaphor" has been formed, as well as possible ways of development in this system, which allows the developer to define or choose a certain path and its subsequent stages.

Keywords: metaheuristic algorithm, metaphor, optimization.

Вступ. Актуальність задачі. Останні два десятиліття відзначились вибуховим сплеском розвитку метаевристичних алгоритмів, початок формування яких припадає на середину минулого сторіччя. На сьогодні вже існує біля 4–5 десятків широко розповсюджених апробованих алгоритмів та, можливо, декілька сотень алгоритмів, що знаходяться на початковій стадії свого «життя».

Розробка та використання метаевристичних алгоритмів є не дуже розповсюдженими в країнах Східної Європи. Але протилежне спостерігається в країнах Західної Європи, США, країнах Близького Сходу та Китаї, де відбуваються інтенсивні апробація та використання, а також активізація розробок нових метаевристичних алгоритмів. По-перше, така тенденція спостерігається завдяки наявності певної «наукової моди», тобто зацікавленості певних верств наукової спільноти деяких географічних регіонів певними тематиками в окремий проміжок часу. Хвилеподібний розвиток зацікавленості якоюсь темою є нормальним для наукових досліджень, на це вказує філософія науки. Ось саме зараз і відбувається така хвиля у вказаних країнах. По-друге, для опису та подання переважної більшості метаевристичних алгоритмів автори використовують метафори та навіть іноді популярний стиль, що є нетиповим для класично-консервативної наукової спільноти Східно-Європейського регіону.

Освітлення сучасних метаевристичних алгоритмів, термінів і категорій та проведення взаємозв'язків між ними для класифікації дасть змогу глибше зрозуміти сутність питання та провести актуалізацію цього наукового напряму. Широкий розгляд категорії «метафора», що використовується для опису алгоритмів, дасть змогу глибше зрозуміти можливості використання та акцентувати її ролі у ході формування чи донесення наукової думки.

Проведення огляду цікавих, оригінальних та різноманітних метаевристичних алгоритмів дасть змогу зрозуміти сучасні тенденції стосовно цього питання, визначити переваги та недоліки алгоритмів, а також провести їх популяризацію.

Таким чином, освітлення вказаних питань у межах тематики є актуальною науково-практичною задачею.

Основна частина.

Метаевристичні алгоритми. Щоб зрозуміти, що саме є метаевристичним алгоритмом, потрібно розглянути ряд визначень деяких категорій та взаємозв'язки між ними.

Алгоритм – чіткий опис послідовності дій, які необхідно виконати при розв'язуванні задачі.

Загалом алгоритми поділяються на детерміновані та недетерміновані. *Детермінований* алгоритм передбачає єдиний шлях від вхідних даних до виходу. *Недетермінований* алгоритм відрізняється можливістю

отримання результату декількома різними шляхами, тобто деякі шляхи виконання алгоритму можуть привести до однакового результату, а деякі – до особливих результатів.

Саме недетерміновані алгоритми цікавлять нас у межах питання, що досліджується. Тому доцільно вказати, що недетерміновані алгоритми розділяються на наступні: *апроксимації*, *ймовірнісні* та *стохастичні*.

Стохастичні алгоритми оптимізації використовують випадковість у процесі пошуку оптимуму. Зазвичай їх застосовують, коли цільова функція складна, багатоекстремальна, з розривами та перешкодами.

Стохастичні алгоритми у свою чергу умовно можна поділити на *евристичні* та *метаевристичні*.

Евристичні алгоритми – спроможні видати прийнятне розв'язання серед багатьох розв'язань, але неспроможні гарантувати, що воно буде найкращим. Зазвичай такі алгоритми знаходять розв'язок, близький до найкращого, та роблять це швидко.

Метаевристичні алгоритми – комбінують методи пошуку локальних та глобальних розв'язків у абстрактні стратегії евристичної оптимізації задач. Саме префікс «мета» вказує на складність, тобто на «високий рівень» або «понад». Більшість таких алгоритмів базується на методах пошуку локальних розв'язків, тобто вони обчислюють деякий початковий розв'язок та покращують його іншими методами. Використовуючи комбінації та поєднання різних стратегій, такі алгоритми намагаються уникнути глухих кутів під час пошуку локальних мінімумів. Велика частина метаевристичних алгоритмів використовують для свого опису природні процеси.

У літературі існує декілька класифікацій метаевристичних алгоритмів в залежності від використаних метафор.

Наприклад, автори в [1] поділили метаевристичні алгоритми на дві категорії (еволюційний інтелект та ройовий інтелект), а в [2] автори розділили їх на три різні класи (ройовий інтелект, еволюційний інтелект та фізико-хімічні алгоритми). У [3] автор класифікував їх на чотири групи (ройовий інтелект, алгоритми на основі біологічних явищ, природничо-наукові алгоритми та алгоритми на основі природних явищ). У іншій роботі [4] автор пропонує наступні чотири категорії: еволюційні алгоритми, ройовий інтелект, алгоритми на основі фізики, алгоритми, натхненні людиною.

Автори цієї роботи пропонують дещо інший підхід до класифікації метаевристичних алгоритмів, що базується на термінах та поділі категорій, взятих з природничих наук. Пропонується виділити два базових класа:

1. Метаевристичні алгоритми *живої природи*.
2. Метаевристичні алгоритми *неживої природи*.

Відповідно з назвою класу, до нього будемо відносити метаевристичні алгоритми, що побудовані на відповідних діях та процесах. У свою чергу, метаевристичні алгоритми живої природи пропонується поділити на три підкласи:

1. *Еволюційні* алгоритми, які пов'язані з еволюційними процесами та діями.

2. *Біо-асоційовані* алгоритми, які пов'язані з діями та поведінковими особливостями тварин, рослин, бактерій, грибів, вірусів.

3. *Людино-асоційовані* алгоритми, які пов'язані з діями та поведінковими особливостями людини.

Надану класифікацію зображено на рис. 1.

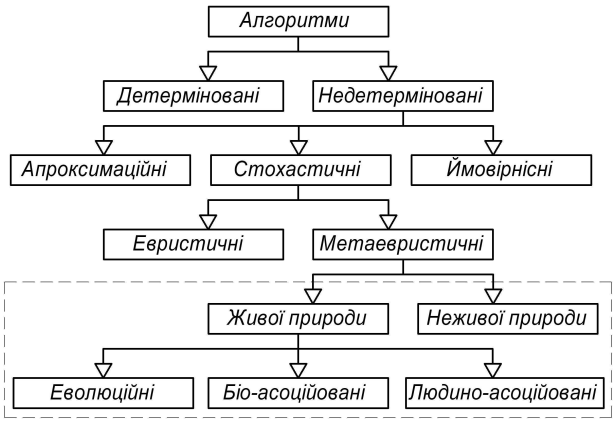

Рисунок 1 – Запропонована класифікація метаевристичних алгоритмів

До класу алгоритмів *неживої* природи можна віднести: алгоритм «великого вибуху–великого стиснення»; гравітаційний пошук; алгоритм інтелектуальних крапель води; електромагнітний пошук; стохастичний дифузійний пошук; імітація відпалу; променевий пошук; алгоритм оптимізації центральною силою; оптимізація на основі генерації плазми та інші.

До підкласу *еволюційних* алгоритмів класу *живої природи* можна віднести: диференціальна еволюція; генетичні алгоритми; еволюційні алгоритми; стратегія еволюції; генетичне програмування та інші.

До підкласу *біо-асоційованих* алгоритмів класу *живої природи* можна віднести: алгоритм перемішаних стрибаючих жаб; алгоритм оптимізації на основі розповсюдження бур'яну; алгоритм кажанів; алгоритм світляків; алгоритм рою часток; оптимізація роєм світляків; меметичні алгоритми; алгоритм штучної бджолиної колонії; алгоритм дерев, що ростуть; алгоритм рою бджіл; пошук на основі поведінки косяка штучних риб; зозулиний пошук; алгоритм наслідування поведінки мавп; мавпячий пошук; бактеріальна оптимізація; алгоритм наслідування поведінки мурашиної колонії; оптимізація на основі поведінки котячої зграї; алгоритм оптимізації на основі поведінки китів; алгоритм оптимізації на основі поведінки горобців; оптимізація на основі поведінки зграї сірих вовків; алгоритм оптимізації на основі поведінки ворон; оптимізація на основі поведінки беркута; алгоритм оптимізації на основі поведінки фламінго та інші.

До підкласу *людино-асоційованих* алгоритмів класу *живої природи* можна віднести: алгоритм штучних імунних систем; алгоритм пошуку гармонії; алгоритм еволюції розуму; оптимізація на основі навчання; табу пошук; оптимізатор групового пошуку; соціально-емоційна оптимізація; оптимізація мозковим штурмом; алгоритм волейбольної першої ліги; алгоритм отримання-обмін знаннями та інші.

Метафора. Що і навіщо? *Метафора* – мовне і мисленнєве явище, що полягає в перенесенні властивостей одного предмета (явища, дії) та його мовного знаку на інший предмет (явище, дію) за принципом аналогії або контрасту.

Спостереження за функціонуванням природних елементів та системами значно вплинуло на розвиток і оформлення метаевристики. Автори, що надихнулися природою, створили ефективні механізми оптимізації, які вилилися у ряд ефективних алгоритмів.

При використанні метафор для опису процесу пошуку оптимальних параметрів проводиться паралель між природнім процесом-метафорою і математичним описом та формальною термінологією. Це дає змогу краще та легше зрозуміти запропоновані алгоритми. Використання метафори є хорошим способом для представлення, опису та донесення ідеї, принципів функціонування та послідовностей алгоритмів. А широкий спектр застосування та високоякісні результати призвели до наукового визнання таких алгоритмів та до стрімкого їх розповсюдження у практичному використанні й визнання ідеї метафори при формуванні та описі алгоритму загалом.

Метафори, що використовуються для створення метаевристичних алгоритмів, повинні відповідати ряду вимог:

1. Нова.

2. Нова ідея – новий процес або заново ідеалізований.

3. Цікава та нетривіальна (суб'єктивно).

4. Допомагати розумінню та сприйняттю.

Огляд. Еволюційні та генетичні алгоритми були докладно розглянуті в [5–6], тому у межах цієї роботи їх торкатися не будемо.

При проведенні огляду метаевристичних алгоритмів автори цієї роботи намагалися винести в нього цікаві, оригінальні та різноманітні алгоритми. Огляд кожного алгоритму побудовано за єдиним принципом та складається з двох частин. Першою частиною є опис метафори, її ролі та вплив при формуванні чи описі алгоритму. У другій частині дається псевдокод або послідовність алгоритму. Така зв'язка дає можливість не тільки ознайомитись з сутністю та послідовністю того чи іншого алгоритму, але також зрозуміти хід думки автора при його генеруванні, формуванні та описі.

Бактеріальна оптимізація. Цей алгоритм базується на соціальній та кооперативній поведінці, що зустрічається в природі. Поведінка бактерій, що шукають області з високим рівнем поживних речовин, закладена у якості процесу оптимізації. Кожна з бактерій намагається максимізувати отриману енергію за кожну одиницю часу, витраченого на пошук їжі та уникнення шкідливих речовин. Крім того, такий пошук передбачає спілкування між особинами. Основні положення поведінки бактеріального скупчення можна надати наступним чином [7]:

1) бактерії випадковим чином розподілені на карті поживних речовин;

2) бактерії можуть рухатися до регіонів з високим вмістом поживних речовин. Ті, що знаходяться в регіонах зі шкідливими речовинами або регіонах з низьким вмістом поживних речовин, загинуть та розсіються відповідно. Бактерії у зручних регіонах можуть розмножуватися;

3) бактерії, що розташовані в перспективних регіонах карти поживних речовин, намагаються залучити інші бактерії, генеруючи хімічні атрактанти;

4) бактерії прагнуть розташуватися в області з найбільшим вмістом поживних речовин;

5) бактерії можуть розповсюджуються в пошуку нових регіонів на карті з поживними речовинами.

Поведінка бактерій [8] у пошуку їжі складається з основних етапів: 1 – хемотаксис, 2 – роїння, 3 – розмноження та 4 – ліквідація та розосередження. На основі цих кроків авторами методу було запропоновано наступну послідовність алгоритму, де агентом, тобто пробною точкою, виступає бактерія:

1. Ініціалізація початкових вхідних параметрів.

2. Фаза усунення-розсіювання. При зростанні щільності бактерій на невеликій території температура цієї області стає високою, а також може бути недостатньо кількості поживних речовин для всіх бактерій. Щоб цього уникнути, популяція бактерій може випадково змінювати своє положення. Цей процес розподіляє бактерії по простору для уникнення ситуації потрапляння скупчення у зону локального оптимального значення.

3. Фаза відтворення більш здорових бактерій. До відтворення допускаються бактерії з найвищими показниками якості. Якість оцінюється «станом здоров'я» бактерії – сумою ступенів пристосованості та придатності протягом життя. При відтворенні бактерія робить свою копію та розташовує її на своє ж місце.

4. Фаза хемотаксису. У біології хемотаксис – це процес руху бактерій для отримання поживних речовин. Цей процес імітує рух бактерії шляхом плавання та перекидання через джгутики. При плаванні бактерія пливе в тому самому напрямку, що і на попередньому кроку, а при перекиданні – бактерія змінює напрямок на інший. При плаванні пропонується використовувати функцію атрактанта, який генерує кожна бактерія залежно від своєї поточної якості. Зазвичай при перекиданні пропонується рухатись у випадковому напрямку на випадковий крок.

5. Перехід до фази 2.

Таким чином, при впорядкованому русі агентів за декілька ітерацій популяція стягується до найбільш придатної зони простору. Аналізуючи показники якості агентів, тобто значення цільової функції, обирається кращий розв'язок.

Оптимізація на основі поведінки беркута. Алгоритм заснований на спіральному русі беркутів при пошуку здобичі. Беркут запам'ятовує найкраще місце, яке він відвідав на поточний момент. Беркут одночасно має тяжіння до нападу на здобич та до подорожі в пошуках кращої їжі. У кожній ітерації кожен беркут випадковим чином вибирає здобич іншого беркута та кружляє навколо найкращого місця, яке на поточний момент він відвідав. У кожній ітерації кожен беркут повинен вибрати жертву для виконання операцій польоту та атаки. У алгоритмі здобич моделюється як найкращі розв'язки, знайдені до цього моменту зграєю беркутів. Кожен беркут здатний запам'ятати найкращий розв'язок, який він знайшов.

У кожній ітерації кожен пошуковий агент вибирає цільову жертву з пам'яті всієї зграї. Потім обчислюються вектори атаки та маршруту для кожного беркута відносно вибраної здобичі. Якщо нова позиція краща, ніж попередня позиція в пам'яті, пам'ять оновлюється. Стратегія вибору здобичі відіграє важливу роль у алгоритмі. Відбір може відбуватися простим способом,

коли кожен беркут вибирає здобич лише у власній пам'яті. Щоб змусити беркутів краще досліджувати простір, автори пропонують випадкову схему, де кожен беркут випадковим чином вибирає свою здобич у поточній ітерації з пам'яті будь-якого іншого члена зграї. Примітно, що вибрана здобич не обов'язково є найближчою чи найдальшою. У цій схемі кожна здобич у пам'яті зіставляється лише з одним беркутом. Потім кожен беркут виконує атаку або крейсерську операцію на обрану жертву. Кожен пошуковий агент може атакувати лише одну з позицій у пам'яті, яка належить іншому пошуковому агенту. Атака моделюється за допомогою вектору, починаючи від поточної позиції беркута і закінчуючи місцем розташування жертви в пам'яті беркута. Крейсерський вектор розраховується на основі вектору атаки.

Згідно з наданими положеннями псевдокод реалізації алгоритму може бути представлений наступним чином:

1. Ініціалізація популяції беркутів.
2. Оцінка функції пристосованості.
3. Ініціалізація пам'яті популяції.
4. Ініціалізація коефіцієнтів атаки та крейсерського польоту.
5. Оновити значення коефіцієнтів атаки та крейсерського польоту.
6. Для кожного беркута випадково обрати жертву з пам'яті популяції.
7. Обчислити вектор атаки.
8. Якщо довжина вектору атаки не дорівнює нулю:
8.1. Обчислити крейсерський вектор.
8.2. Обчислити вектор кроку.
8.3. Оновити позицію.
8.4. Оцініть функцію придатності для нової позиції.
8.5. Якщо придатність краща, ніж придатність позиції в пам'яті беркута, то замінити її.
9. Перейти до пункту 5.

Алгоритм оптимізації на основі розповсюдження бур'яну. Оптимізація алгоритмом інвазійних бур'янів є числовий стохастичний алгоритм пошуку та базується на популяційному інтелекті, який імітує поведінку бур'янів під час колонізації певних територій у пошуку місць для зростання. Цей алгоритм намагається імітувати агресивність, стійкість, адаптивність та випадковість популяції бур'янів.

Як стверджують автори, алгоритм є простим, але ефективним у збіжності до оптимальних розв'язків із використанням основних функцій, таких як посів, ріст та конкуренція в колонії бур'янів. Щоб імітувати поведінку бур'янів у середовищі існування, розглядаються деякі основні особливості процесу:

1. Первинна ініціалізація популяції: обмежена кількість насіння розподіляється випадковим чином в просторі пошуку.

2. Розмноження: кожна насінина виростає в дорослу рослину та дає насіння, кількість яких пропорційна якості рослини. Кількість насіння, утвореного будь-яким бур'яном, змінюється лінійно від максимально можливої (для найкращого розв'язку) до мінімально можливої (для найгіршого розв'язку). Згідно прийнятності популяції, кожній особині популяції дозволено виробляти насіння в межах визначеного регіону з центром у його власній позиції.

3. Спектральний розкид: насіння, вироблене групою в нормальному розподілі із стандартним відхиленням. На кожному наступному кроці ітерації відхилення зменшується. Для того, щоб попередити передчасну збіжність до локальних оптимумів, рекомендовано використовувати стандартні оператори мутації.

4. Конкурентне виведення: якщо кількість бур'яну перевищує максимально-дозволену кількість у колонії, бур'яни з найгіршою придатністю видаляються з колонії, щоб у колонії залишалася постійна кількість особин.

5. Цей процес триває, доки не буде досягнуто максимальної кількості ітерацій, а потім популяція досліджується на придатність.

Автори вказують, що такий алгоритм за своїми властивостями може перевищувати більш розповсюджені алгоритми, наприклад, генетичні.

Алгоритм «великого вибуху – великого стиснення». Автори [13, 14] вказують, що класичний генетичний алгоритм або його різновиди страждають від надмірно повільної конвергенції, тобто вони повільно досягають глобального оптимуму. Однак у них є можливість «надійно» вирішити проблему, знайшовши найбільш перспективні місця у всьому просторі пошуку. Запропонований метод Великого вибуху – Великого стиснення знаходить точну глобальну оптимальну точку для простору в межах максимальної кількості дозволених ітерацій. Метод Великого вибуху – Великого стиснення складається з двох фаз: фази Великого вибуху та фази Великого стиснення. У фазі Великого вибуху варіанти розв'язків випадковим чином розподіляються у просторі пошуку параметрів. Подібно до інших еволюційних та генетичних алгоритмів, початкові розв'язки під час першого Великого вибуху розповсюджуються в просторі параметрів рівномірним чином.

Автори пов'язали випадковий характер Великого вибуху до розсіювання енергії або перетворення з впорядкованого стану до безладу чи до стану хаосу (новий набір кандидатів у розв'язки). Саме створення початкової випадкової популяції автори називають фазою Великого вибуху. За фазою Великого вибуху слідує фаза Великого стиснення. Під цією фазою автори мають на увазі фазу оператора конвергенції, який має багато входів, але лише один вихід, та називають його «центр маси». Під терміном «маса» мається на увазі значення, обернене до значення функції якості. Точку центру маси пропонується знаходити як відношення суми відношень векторів точок до значень їх функцій якості до суми зворотних значень функцій якості.

Після фази Великого стиснення алгоритм створює нові розв'язки, які будуть використані як Великий вибух наступного кроку ітерації. Для генерації нового покоління використовується попереднє знання про центр мас, навколо якого за допомогою нормального розподілу випадковим чином генеруються нові розв'язки, але у певних межах простору відносно центру мас. На кожній ітерації межі простору зменшуються.

Після другого вибуху центр мас перераховується. Ці послідовні етапи вибуху та стиснення виконуються неодноразово, доки не буде виконано критерій зупинки

алгоритму. Межі простору вибуху на кожній ітерації можуть бути фіксованими, але їх зменшення на кожній наступній ітерації дає кращі результати.

Підсумовуючи сказане вище, псевдокод алгоритму може бути наданий наступним чином:

1. Сформувати початкове покоління популяції випадковим чином.

2. Обчислити значення функції якості для всіх розв'язків.

3. Знайти центр мас. Найкращу за якістю особину з популяції можна використати у якості центра маси.

4. Обчислити та розташувати нових особин навколо центру мас.

5. Перейти до кроку 2, доки не буде виконано критерії зупинки.

Алгоритм імітації відпалу. Пропонується [15, 16] як хороший інструмент для задач пошуку та оптимізації. Це метод випадкового пошуку, який представляє абстракцію отримання кристалічної структури за допомогою фізичного процесу. Основними перевагами перед іншими методами пошуку є його гнучкість та здатність наближатися до глобальної оптимальності. З іншого боку, основними недоліками техніки є її дуже повільна конвергенція, а також компроміс між якістю розв'язків і часом, необхідним для обчислення. Робота з алгоритмом вимагає врахування різних класів обмежень та тонкого налаштування параметрів. Цей алгоритм працює таким чином, що поточний стан створює лише один наступний стан.

Сутність процесу складається з двох етапів. На першому етапі тверда речовина нагрівається шляхом підвищення температури до досягнення рідкої фази. За першим етапом йде другий, який включає зниження температури до тих пір, поки не буде досягнута кристалічна структура з мінімальною енергією (процес відпалу), і пошук мінімуму в більш загальній системі.

Моделювання відпалу починається з початкової популяції, потім генеруються сусідні до початкових розв'язки (або випадковим чином, або за допомогою певного попередньо визначеного правила). Моделювання відпалу засновано на критерії прийнятності Метрополіса, який моделює перехід термодинамічної системи від поточного розв'язку (стану) до потенційного, в якому вміст енергії мінімізується. Потенційний розв'язок приймається як поточний на основі ймовірності прийняття. Ймовірність прийняття є основним елементом механізму пошуку в симуляції відпалу. Якщо температура знижується досить повільно, то система може досягти рівноваги (стаціонарного стану) на кожній ітерації. Ця рівновага відповідає розподілу Больцмана.

Симуляція відпалу може бути описана у псевдокоді.

1. Випадковим чином генеруються початкові розв'язки.

2. Обрати лічильник зміни температури, температурний графік охолодження, початкову температуру.

3. Із поточного розв'язку створюється сусідній розв'язок шляхом зміщення поточного за кожним виміром на випадкову частку визначеного кроку.

4. Визначити значення цільової функції. Якщо поточний розв'язок має краще значення цільової функції, ніж старий, то поточний розв'язок приймається. В іншому випадку поточний розв'язок також можна прийняти, якщо значення задане розподілом Больцмана. Цей розподіл залежить від температури, тобто певного параметру керування рухом алгоритму.

5. Зменшення температури. Найпоширенішим графіком охолодження є геометричне правило зміни. Зниження температури призводить до зменшення розміру локального простору пошуку або розміру околу для кожного розв'язку, що досліджуються.

6. Перехід до 3.

7. Критерій припинення алгоритму.

Оптимізація на основі поведінки зграї сірих вовків. Цей алгоритм [17, 18] імітує соціальну ієрархію та полювання, яке притаманне зграї сірих вовків. Природно, сірі вовки живуть групою від 5 до 12 особин. Вони живуть у суворій соціальній ієрархії, лідери групи сірих вовків «альфа» – це самець і самка, які відповідають за прийняття рішень від імені інших вовків у групі. Після «альфи» наступним елементом у соціальній ієрархії сірих вовків є «бета», її відповідальність полягає в тому, щоб допомагати «альфі» в процесах прийняття рішень. «Бета» підсилює команди «альфи» в усій зграї та дає зворотний зв'язок до «альфи».

Найнижчий за рангом сірий вовк – «омега». «Омега» грає роль цапа-відбувайла. Омега-вовки завжди повинні підкорятися всім іншим домінуючим вовкам. Вони останні вовки, яким дозволено їсти. Якщо вовк не є «альфою», «бетою» чи «омегою», його називають підлеглим (або «дельтою»). «Дельта» підкоряється «альфі» та «беті», але домінують над «омегою». До цієї категорії відносяться розвідники, вартові, старійшини, сторожі.

Також автори вказують, що базовим є групове полювання, яке має наступні етапи:

1. Вистежування, переслідування та наближення до жертви.

2. Переслідування здобичі, доки вона не зупиниться, та її оточення.

3. Напад на здобич.

При математичному моделюванні найкращий розв'язок приймається як «альфа». Отже, другий і третій найкращі розв'язки приймаються «бета» і «дельта» відповідно. Решта розв'язків вважаються «омегами». В алгоритмі полювання (оптимізація) керується «альфою», «бетою» та «дельтою». За цими трьома вовками йдуть «омеги».

Стратегія полювання (оптимізації) будується наступним чином.

1. Проводиться ініціалізація початкової популяції вовків.

2. Проводиться оцінка якості кожної особини та виділення групи лідерів – «альфи», «бети» та «дельти».

3. «Альфа» досліджує простір навколо себе на певній відстані, обираючи кращий шлях, та здійснює переміщення в цю сторону.

4. «Бета» та «дельта» переміщуються у напрямку «альфи», але тримають до лідера певну відстань, що є випадковою у визначеному інтервалі.

5. «Омеги» займають випадковим чином простір між лідерами.

6. Перехід до етапу 3.

На кожній ітерації можуть бути змінені крок «альфи» та відстані між лідерами. Автори зазначають,

що такий підхід схильний до прив'язки до локальних розв'язань, тому рекомендують на початкових ітераціях не інтенсивно зменшувати крок та відстані. Також за можливості максимально використовувати фактор випадковості, параметри кроку «альфи» та відстані між лідерами не приймати фіксованими значеннями, а приймати випадковим значенням на заданому інтервалі.

Алгоритм оптимізації на основі поведінки ворон. Це схема розв'язання оптимізаційних задач ройовим інтелектом [19–20]. Автори відзначають основні характерні риси поведінки ворон, які стали базовими у формуванні підходу. Ворони стежать один за одним, спостерігають і відстежують місця зберігання їжі інших ворон та крадуть їжу, коли господар йде. Спираючись на поведінкові звички ворон, алгоритм має чотири основні принципи:

1) ворони живуть групами;

2) ворони запам'ятовують місце, де схована їжа;

3) ворони слідують одна за одною, щоб вкрасти їжу;

4) ворони мають здатність сприйняття. Коли вони відчувають, що за ними стежать, вони змінюють місце сховку їжі, щоб запобігти крадіжці.

Загалом, основні фази алгоритму можна надати наступним чином.

1. Ініціалізація зграї ворон, що випадковим чином розосереджена у багатовимірному просторі.

2. Кожна ворона на своїй позиції оцінюється за якістю, значення якості ставиться як початкове значення пам'яті.

3. Ворона вибирає випадковим чином іншу ворону та генерує випадкове число від 0 до 1, яке характеризує ймовірність обізнаності наявності схованки їжі у іншої ворони.

4. Ворона оновлює свою позицію в залежності від ймовірності обізнаності. Якщо вона вища заданого значення, то ворона рухається на випадкову відстань у бік обраної ворони, якщо нижче, то ворона переміщується у випадкову позицію.

5. Оновлення пам'яті. Якщо нова позиція ворони за якістю краще попередньої, то вона залишається на ній, якщо гірше – ворона повертається на попередню позицію.

6. Аналіз якості популяції.

7. Перехід до 3.

Автори вказують на чудові характеристики наданого алгоритму. Існують поліпшені та гібридні варіанти. Алгоритм потребує небагато вхідних параметрів, а його структура проста та легка для реалізації. Він був успішно застосований у багатьох галузях і на практиці показав хороший ефект застосування. Хоча вказується, що алгоритм має такі недоліки, як низька швидкість та передчасне наближення до локального оптимуму, що потребує додаткових досліджень урізноманітнення популяції.

Алгоритм пошуку гармонії. Це відносно нова метаевристична техніка пошуку розв'язків [21, 22], яка імітує процес музичної імпровізації для досягнення ідеального стану гармоній, що відтворюються одночасно кількома музичними інструментами. Це означає, що висота звуку кожного інструменту поєднуються разом, щоб отримати найбільш гармонійний ритм, що оцінюється на основі естетичного сприйняття. Алгоритм пошуку гармонії базується на розв'язанні, де імпровізація висоти звуку відбувається ітеративно за допомогою набору розв'язків у пам'яті гармонії. Гармонія в музиці відповідає вектору розв'язків, кожен музичний інструмент відповідає кожному параметру проєктування, діапазон висоти музичного інструменту відповідає діапазону можливих значень параметрів проєктування, естетика аудиторії відповідає цільовій функції, а музичні імпровізації відповідають локальному та глобальному пошуку в оптимізації.

Щоб виконати алгоритм, існує п'ять основних кроків:

1. Ініціалізація проблеми та параметрів алгоритму.

2. Ініціалізація пам'яті гармонії. На початку процесу пошуку гармонії початкова сукупність векторів гармонії генерується випадковим чином.

3. Імпровізація нової гармонії. Щоб імпровізувати одну висоту для кожного інструменту, музикант випадково обирає з трьох варіанти: 1) зіграти будь-яку висоту з пам'яті гармонії, 2) зіграти висоту, яка трохи відрізняється від висоти з пам'яті гармонії на певне значення, або 3) зіграти повністю випадкову висоту із допустимих діапазонів кроку.

4. Оновлення пам'яті гармонії за виключенням найгіршої та найкращої гармонії. Пам'ять гармонії оновлюється шляхом оцінки між розв'язками нової гармонії та існуючими. Якщо існуюча гармонія гірша за нову, її буде виключено та замінено новою у пам'яті.

5. Перевірка критеріїв зупинки; якщо вони не задовольняються, перехід до кроку 3. Процес продовжується та завершується, коли кількість імпровізацій досягнуто.

Алгоритм оптимізації на основі поведінки горобців. Наданий авторами алгоритм [23, 24] базується на поведінці горобців. Як зазначається, у популяції горобці поділені відповідно до поведінки на два типи: виробники та крадії. Виробники ведуть активний пошук джерел їжі, в той час як крадії відбирають їжу у виробників. Крім того, вказується, що птахи зазвичай можуть динамічно змінювати поведінкові стратегії і перемикатися між виробником та крадієм. Поведінка стратегії пошуку їжі переважно залежить від власних енергетичних запасів особин, горобці з низьким запасом енергії більше крадуть. Також на поведінку та переміщення впливає територіальне положення особини відносно загальної зграї.

Вказані вище та інші положення поведінки горобців були ідеалізовані та сформовані відповідні правила для наступного алгоритму.

1. Виробники мають високий рівень енергетичних запасів та вказують на зони пошуку їжі або напрямки для всіх крадіїв. Вони відповідають за визначення районів, де можна знайти багаті джерела їжі. Рівень енергетичних запасів залежить від показника значення функції якості індивіда.

2. Як тільки горобець виявляє крадія, особини починають цвірінькати на знак тривожного сигналу. Коли значення тривоги перевищує певний безпечний поріг, виробники повинні відігнати всіх крадіїв у безпечну зону.

3. Кожен горобець може стати виробником, доки він шукає кращі джерела їжі, але частка виробників та

крадіїв незмінна в усій популяції.

4. Горобці, чия енергія стає вищою, починають діяти як виробники. Голодуючі крадії прагнуть змінити становище та летять в інші місця за їжею, щоб отримати більше енергії.

5. Крадії слідують за виробниками, які можуть забезпечити найкращу їжу. Тим часом деякі крадії можуть постійно контролювати виробників та конкурувати за їжу, щоб збільшити рівень крадіжок.

6. Горобці на краю групи швидко рухаються до безпечної зони, щоб зайняти кращу позицію, тоді як горобці в середині групи рухаються навмання, щоб бути ближче до інших.

У симуляційному експерименті використовують віртуальних горобців. Виробники несуть відповідальність за пошук їжі та керування рухом всієї популяції. Виробники можуть шукати їжу у ширшому діапазоні місць, ніж крадії.

Основні етапи алгоритму можна надати наступним чином.

1. Створити та ініціалізувати розв'язок. На цьому етапі визначаються розмір популяції, максимальна кількість ітерацій, співвідношення горобців виробників та крадіїв. Початкова позиція кожного горобця в популяції вибирається випадковим чином.

2. Розрахувати функцію якості для кожної особини.

3. Розподіл популяції в залежності від показників якості на виробників та крадіїв (10–20%).

4. Переміщення особин популяції. Виробники та крадії випадковим чином переміщуються на певний крок. Якщо поряд є крадій, виробник рухається від нього. Якщо крадія поряд немає, виробник обирає випадковий напрям.

5. Переміщення крадіїв на певний крок у напрямку найближчого якісного виробника.

6. Якщо нова позиція крадія краща за попередню, вона закріплюється.

7. Перехід до 3.

Автори вказують, що запропонований алгоритм може забезпечити високу конкурентоспроможність порівняно з іншими сучасними алгоритмами з точки зору точності пошуку, швидкості збіжності та стабільності. Крім того, результати практичних досліджень показують, що наданий алгоритм має високу продуктивність у різноманітних просторах пошуку та хорошу здатність до дослідження потенційної області глобального оптимуму.

Алгоритм гравітаційного пошуку. Є новим алгоритмом стохастичного пошуку [25, 26], у якому агенти розглядаються як об'єкти, а їх активність вимірюється їх масою. Алгоритм гравітаційного пошуку можна розглядати як ізольовану систему мас, як маленький штучний світ мас, які підкоряються ньютонівським законам тяжіння та руху. З часом маси будуть притягуватися найважчою масою, яка є оптимальним розв'язанням у просторі пошуку. Автори вказують, що для гравітаційного пошуку підтверджено високу продуктивність у розв'язанні різних задач оптимізації.

Маса кожного агента розраховується в залежності від його та кожного з інших агентів значень цільових функцій. Загальну силу, яка діє на кожного агента у визначеному вимірі у певний момент часу, пропонується розраховувати відповідно до теорії гравітації Ньютона. Відповідно до маси та сили може бути знайдене прискорення агента. Також послідовно можуть бути знайдені швидкість та положення агента.

Параметри, які контролюють продуктивність алгоритму, пропонуються наступні: максимальна кількість ітерацій, розмір популяції, початкове значення гравітаційної сталої. Слід зазначити, що гравітаційна стала ініціалізується на початку та з часом зменшується для впливу на точність пошуку.

Псевдокод алгоритму може бути наданий наступним чином.

1. Визначення простору пошуку. Ініціалізація параметрів керування алгоритму.

2. Ініціалізація початкової популяції агентів випадковим чином у межах простору параметрів.

3. Обчислити значення цільової функції.

4. Обчислити значення мас агентів та гравітаційної сталої.

5. Розрахувати загальну силу для різних вимірів простору.

6. Розрахувати прискорення кожного агента.

7. Розрахувати швидкості кожного агента.

8. Оновити позицію кожного агента.

9. Повтор кроків 3–8, доки не буде досягнуто критерію зупинки, тобто заздалегідь визначеної кількості ітерацій.

10. Визначити найкращий розв'язок.

Алгоритм дерев, що ростуть, або посіву та вирощування дерев [27, 28] був натхненний природними явищами – ростом дерев. Цей алгоритм має дві фази: фазу посіву та фазу вирощування. Автори пропонують розглянути простір розв'язків як сад саджанців, де саджанці розподілені рівномірно. Щоб розв'язати задачу, довільні розв'язки, які мають бути згенеровані спочатку, повинні бути рівномірно розподілені у можливому просторі пошуку. Спочатку кожен саджанець не має гілок. Початкові саджанці в саду (початковий розв'язок в просторі пошуку) можна створити як однорідний сад. Наступним етапом вирощування саджанців є паруванння. Метою оператора парування є створення нового саджанця з існуючих шляхом обміну існуючою інформацією. Для кожної пари саджанців буде свій фактор парувування, оскільки відстань між парою є найважливішим фактором, який впливає на вірогідність парувування.

Наступний оператор, що може бути використаний, –це формування гілок на саджанцях, тобто зміна одного із параметрів саджанця. Наслідуючи природу, автори пропонують при формуванні гілок використовувати випадковість, обираючи параметр, де формується гілка. Далі вказується на можливість використати оператор щеплення. Процес щеплення може відбуватися між двома різними саджанцями у випадку їх подібності. Успішність щеплення пропорційна подібності саджанців.

Представлені оператори автори пропонують використовувати послідовно, а отримані саджанці розглядаються як тимчасові розв'язання. Після використання операторів спарювання, розгалуження та щеплення оцінюються отримані тимчасові розв'язки. Наступне покоління буде отримано шляхом вибору певної кількості найкращих саджанців із поточної генера-

ції та тимчасових розв'язків.

Псевдокод алгоритму виглядає наступним чином:

1. Висаджування саджанців у початковий сад.

2. Обчислення цільової функції для кожного саджанця.

3. Парування.

4. Формування гілок.

5. Щеплення .

6. Обчислення цільової функції для нових форм саджанців.

7. Вибір кращих форм саджанців.

8. Перехід до 1.

Алгоритм перемішаних стрибаючих жаб.

Алгоритм перемішаних стрибаючих жаб [29, 30] є меметичним метаевристичним алгоритмом, запропонованим для розв'язання задач комбінаторної оптимізації. Як і інші метаевристичні алгоритми, цей алгоритм є ітераційним алгоритмом на основі дослідження популяції. Він моделюється з урахуванням соціальної поведінки жаб. Мета жаб у популяції – отримати максимум їжі, використовуючи мінімальну кількість кроків. Усі жаби складають популяцію алгоритму, а кожна жаба представляє розв'язок. Кожна жаба у популяції має значення пристосованості, яке представляє ступінь близькості до їжі. На початку алгоритму генерується випадкова сукупність особин. Потім пристосованість кожної жаби обчислюється за допомогою цільової функції, специфічної для задачі оптимізації, і популяція сортується за значеннями пристосованості. Популяція, відсортована за пристосованістю, розділяється на мемеплекси (групи). Для кожного мемеплексу існує своя петля меметичного еволюційного процесу. Меметичний еволюційний процес у мемеплексах є частиною локального пошуку алгоритму. З кожним кроком еволюції якість жаб'ячих мемів стає кращою та йде до найкращого розв'язку. Алгоритм використовує кращих жаб ефективніше, ніж гірших жаб на етапі меметичної еволюції. Після меметичного еволюційного процесу для кожного мемеплекса алгоритм збирає останні та перемішує їх. Наступними кроками стає сортування популяції та нове відокремлення мемеплексів. Це частина глобального пошуку алгоритму.

Етапи алгоритму можна сформувати у наступний псевдокод.

1. Ініціалізація параметрів алгоритму: кількість мемеплексів, кількість жаб у кожному мемеплексі.

2. Створення випадкової популяції та обчислення пристосованості для кожної жаби.

3. Сортування сукупності у порядку зменшення значення пристосованості.

4. Розділити жаб на мемеплекси.

5. Процес меметичної еволюції в мемеплексах. Меметична еволюція виконується для кожного мемеплексу.

5.1. Ініціалізація початкових значень та параметрів.

5.2. Вибрати підмемеплекс із поточного мемеплексу. Стратегія вибору підмемеплексу залежить від пристосованості жаб у поточному мемеплексі. Жаби, які мають кращу пристосованість, мають більше шансів бути відібраними, ніж гірші жаби. Визначають найкращу жабу та найгіршу жабу в підмемеплексі.

5.3. Оновити позицію найгіршої жаби шляхом випадкового кроку у бік найкращої жаби.

5.4. Обчисліть пристосованість найгіршої жаби в новій позиції. Якщо нове значення пристосованості краще за старе, тоді перейдіть до кроку 5.4, інакше генеруйте нову позицію для найгіршої жаби випадковим чином, доки позиція не стане кращою.

5.5. Оновити мемеплекс, з якого вибрано підмемеплекс. Відсортуйте поточний мемеплекс за значенням пристосованості жаб.

5.6. Перейдіть до кроку 5.2.

6. Перетасувати мемеплекси. Після попередньо визначеної кількості ітерацій для меметичної еволюції в кожному мемеплексі жаби перемішуються. Відсортуйте масив за значенням пристосованості жаб та оновіть позицію найкращої жаби.

7. Критерії завершення керування для алгоритму. Якщо номер ітерації завершено, алгоритм зупиняється. В іншому випадку перейдіть до кроку 4.

Зозулиний пошук. Побудований на природній поведінці зозуль [31, 32] та імітує їх специфічні особливості розповсюдження. Як і інші подібні алгоритми, він починається з початкової популяції зозуль. Ці початкові зозулі відкладають кілька яєць в гнізда птахів-господарів. Деякі з цих яєць, які більше схожі на яйця птахів-господарів, мають можливість вирости та стати дорослою зозулею. Інші яйця виявляються птахами-господарями та гинуть. Яйця, що вижили, вказують на придатність гніздування у цій місцевості. Чим більше яєць виживає у певній місцевості, тим більше вона приваблива. Зозулі шукають найбільш підходящу ділянку для відкладання яєць, щоб максимізувати відсоток виживання останніх. Після того, як залишені яйця вилуплюються та перетворюються на зрілу зозулю, вони утворюють кілька суспільств. Кожне суспільство має свій регіон проживання. Найкраще середовище існування з усіх суспільств буде місцем призначення зозуль з інших суспільств. Потім вони емігрують до цього найкращого середовища існування. Вони будуть жити десь поблизу найкращого середовища існування. Враховуючи кількість яєць, які має кожна зозуля, а також відстань зозулі до точки цілі (найкращого середовища існування), для неї призначається певний радіус відкладання яєць. Потім зозуля починає відкладати яйця в кілька випадкових гнізд всередині радіуса кладки яєць. Цей процес триває до тих пір, поки не буде отримано найкращу позицію із максимальним значенням якості, і більшість популяції зозулі не збереться навколо тієї самої позиції.

У природі кожна зозуля відкладає від 5 до 20 яєць. Ці значення використовуються як верхня та нижня межі кількості яєць для кожної зозулі на різних ітераціях. Ще одна звичка зозуль полягає в тому, що вони відкладають яйця на максимальній відстані від місця проживання. Тож цей максимальний діапазон називатиметься «радіус відкладання яєць», який пропорційний загальній кількості яєць зозулі.

Зважаючи на вказані вище особливості поведінкової особливості зозуль та прийняті положення алгоритму, псевдокод алгоритму можна надати у наступному вигляді.

1. Ініціалізація випадковим чином початкової популяції зозуль. Місце життя зозулі визначає розташування пробної точки у просторі параметрів.

2. Призначити випадковим чином кожній зозулі кількість яєць у межах вказаного діапазону.

3. Визначити для кожної зозулі радіус відкладання яєць.

4. Розподілити випадковим чином яйця у межах визначеного радіусу.

5. Вбити ті яйця, які розпізнані птахами-господарями. Тобто ліквідувати певну кількість яєць з найнижчим показником якості.

6. Визначити якість місця життя зозуль, що виросли із залишених яєць.

7. Обмежити максимальну кількість зозуль та вбити певну кількість особин із найнижчими показниками якості.

8. Провести кластерізацію популяції.

9. Визначити середню якість кожної популяції.

10. Визначити кращу популяцію за показником середньої якості особин.

11. Перемістити зозуль на певний випадковий крок у напрямку кращої популяції.

12. Зупинка алгоритму, або перейти до 2.

Алгоритм штучних імунних систем. Відносно нова [33, 34] метаевристична техніка має назву алгоритму штучної імунної системи. Це адаптивна система, яка була натхненна спостереженнями за імунними функціями та теоретичною імунологією, моделями та принципами, які застосовуються до складних проблемних областей. Алгоритм імунної системи первинно був розроблений та змодельований на основі імунної мережі. Також зазначається, що алгоритм здатний виконувати паралельну обробку.

Основні базові природні принципи, що лягли в основу штучних імунних систем, можна сформувати наступним чином:

1. Складна система з декількома механізмами захисту від чужорідних організмів.

2. Можливість розпізнати всі клітини господаря та класифікувати їх, щоб викликати відповідну імунну відповідь.

3. Вчиться розрізняти свої та не свої клітини.

4. Має багато бажаних характеристик з точки зору обчислень, таких як: унікальність, розпізнавання образів, різноманітність, стійкість до помилки, навчання та пам'ять, самоорганізація, надійність, співпраця між різними рівнями.

Виділяють дві найпопулярніших теорії: теорія клонального відбору та теорія імунної мережі. Принцип клонального відбору – це стратегія, що описує основні властивості адаптивної імунної відповіді на антигенний стимул. Сутність клонального відбору полягає в тому, що адаптивна імунна система реагує лише тоді, коли до неї вторгається зовнішній стимул (тобто антиген). Таким чином, адаптивна реакція або відповідь імунної системи визначається виробленням антитіл. Отже, якщо антиген присутній, антитіла з найвищим рівнем розпізнавання відбираються для проліферації, виробляючи величезні обсяги антитіл, які називаються клонами. Розмноження відбувається безстатевим шляхом. Під час цієї стадії розмноження клони піддаються соматичній гіпермутації та сильному тиску селекції. Теорія імунної мережі полягає в тому, що антитіла мають деякі частини, які називаються ідіотопами; вони можуть бути розпізнані іншими вільними антитілами або молекулами рецепторів. Ця здатність розпізнавати та бути розпізнаною іншими елементами імунної системи надає системі властиву динамічну поведінку. Натепер існують як безперервні, так і дискретні моделі імунної мережі.

Псевдокод алгоритму простої штучної клональної імунної системи може бути поданий наступним чином.

1. Випадково генеруйте певну кількість антитіл.

2. Повторіть заздалегідь визначену кількість разів:

2.1. Визначити якість кожного антитіла.

2.2. Сортувати антитіла.

2.3. Клонувати всі антитіла. Антитіла клонуються пропорційно до їхньої якості.

2.4. Провести мутацію всіх клонів.

2.5. Визначте якість кожного клону.

2.6. Сортувати клони.

2.7. Виберіть певну кількість найкращих осіб із популяції антитіл та популяції клонів.

2.8. Замініть антитіла з найнижчою якістю новими створеними випадково особинами.

3. Кінець.

Алгоритм поведінки кажанів є відносно новим та потужним алгоритмом [35, 36], який належить до парадигми алгоритмів, що були натхненні природою. Цей надійний та ефективний алгоритм оптимізації заснований на популяції кажанів, які літають у просторі розв'язків. Кожний розв'язок всього багатовимірного простору параметрів представлено у вигляді кажана. На вході в алгоритм початкова позиція та швидкість кажанів ініціалізуються випадковим чином. Алгоритм побудовано на ітераційній схемі, яка визначає значення придатності для всієї сукупності особин. Положення кожного кажана оновлюється за допомогою нового вектору швидкості. Найкращий розв'язок, досягнутий на кожній ітерації, використовується як еталон для наступної ітерації, доки не буде досягнуто критеріїв завершення. Деякі основні положення, які були використані для розробки алгоритму.

1. Кажани використовують свою дивовижну здатність до ехолокації, щоб знаходити їжу й уникати перешкод у темряві.

2. Кажани летять випадково із поточної позиції, зі швидкістю та фіксованою частотою для пошуку здобичі чи їжі. Довжину хвилі та швидкість випромінювання імпульсів можуть автоматично регулювати відповідно до близькості до цілі.

3. Гучність імпульсів може змінюватися від мінімального значення до максимального.

Базуючись на наведених вище положеннях, основні кроки алгоритму можна звести до наступного псевдокоду.

1. Ініціалізація параметрів алгоритму (кількість ітерацій, розмір простору пошуку, кількість кажанів, мінімальна та максимальна частота тощо).

2. Ініціалізація популяції кажанів.

3. Ініціалізація частоти, частоти пульсації та гучності ехолокації, які є випадково-залежними та визначають щільність та глибину дослідження простору параметрів навколо особини.

4. Оцінка початкової популяції та визначення початкового найкращого розв'язку.

5. Налаштування частоти ехолокації для створення нового розв'язку.

6. Оновити швидкість та положення кажанів.

7. Створіть локальні розв'язки навколо поточного найкращого.

8. Обчисліть значення придатності для нових розв'язків.

9. Якщо новий розв'язок кращий попереднього, то прийняти його.

10. Оновити частоти, частоти пульсації та гучності ехолокації

11. Перехід до 5.

12. Постпроцесінг та збереження результатів.

Алгоритм оптимізації на основі поведінки китів. Це новий метаевристичний алгоритм оптимізації [37, 38], натхненний природою, який імітує соціальну поведінку горбатих китів. Найцікавіше у горбатих китах – це їх особливий спосіб полювання. Така поведінка пошуку їжі має назву метод живлення за допомогою бульбашкової сітки. Горбаті кити вважають за краще полювати на зграю криля або дрібну рибу поблизу поверхні. Було помічено, що цей пошук їжі здійснюється шляхом створення характерних бульбашок уздовж кола або шляху у формі «9». Відомо два маневри, пов'язаних з бульбашками, які мають назву «висхідні спіралі» та «подвійні петлі». Зазначається, що годування бульбашковою сіткою є унікальною поведінкою, яку можна спостерігати лише у горбатих китів. Автори надихнулися вказаною поведінкою китів та створили алгоритм, який математично моделює цей маневр для проведення оптимізації.

Алгоритм передбачає, що поточний найкращий розв'язок-кандидат є цільовою здобиччю, тобто близьким до оптимального. Після визначення найкращого пошукового агента інші пошукові агенти намагатимуться оновити свої позиції відносно нього. Це відбувається шляхом здійснення випадкового кроку у напрямку поточного найкращого агента. Наступним етапом є оточення по спіралі. Спочатку обчислюється відстань між китом та жертвою. Потім створюється спіральне рівняння між положенням кита та жертви, щоб імітувати рух горбатих китів у формі спіралі. Зверніть увагу, що горбаті кити одночасно плавають навколо здобичі в межах кола, що звужується, та по спіралеподібній траєкторії. Щоб змоделювати таку одночасну поведінку, ми припускаємо, що існує ймовірність 50% вибору між механізмом скорочувального оточування або спіральною моделлю для оновлення положення китів під час оптимізації. На додаток до методу бульбашкової мережі горбаті кити шукають здобич випадковим чином. Псевдокод алгоритму може бути наданий наступним чином.

1. Ініціалізація популяції китів випадковим чином.

2. Обчислення функції якості кожного пошукового агента.

3. Знайти найкращого пошукового агента.

4. Для кожного пошукового агента розрахувати параметри напряму руху в залежності від фактору випадковості.

5. Оновити позицію поточного пошукового агента за рівнянням. В залежності від фактору випадковості кожен з агентів може рухатись або в лінійному напрямку до кращого агента, або по спіральній траєкторії навколо кращого агента, або випадковим чином змінювати своє положення у просторі пошуку.

6. Перейти до 2.

7. Кінець.

Автори випробовували вказаний алгоритм на задачах параметричного та структурного проєктування. Результати оптимізації доводять, що алгоритм є конкурентоспроможнім порівняно з сучасними метаевристичними алгоритмами, а також порівняно зі звичайними методами.

Пошук на основі поведінки косяка штучних риб. Природа виробила багато складних систем, здатних справлятися із важкими труднощами. Рибні зграї, наприклад, отримують велику користь від великої кількості особин для підвищення спільної живучості. Автори пропонують цікавий підхід [39, 40] до пошуку у просторах великої розмірності, який базується на поведінці зграї риб. Похідний алгоритм складається з трьох операторів: годування, плавання та розмноження. Разом ці оператори забезпечують відповідні якості: широкі можливості пошуку, можливість автоматичного перемикання між розвідкою та експлуатацією та глобальне керівництво для процесу пошуку.

Базові принципи, що сформували основу алгоритму, отримані при спостереженні за зграєю реальних риб.

1. Годування: є природним інстинктом особин (риб) шукати їжу, щоб вирости сильнішими та мати можливість розмножуватися. Їжа є метафорою для оцінки варіантів розв'язків у процесі пошуку.

2. Плавання. Має на меті імітувати скоординований та єдиний очевидний колективний рух, створений усіма рибами у зграї. Плавання обумовлюється годуванням і, зрештою, керуватиме процесом пошуку.

3. Розмноження. Відповідає механізмам природного відбору, який наділяє успішних істот здатністю до продовження роду, тоді як інші, тобто слабкі особини, мають більшу ймовірність загинути.

Алгоритм починається з випадкового генерування зграї риб відповідно до параметрів, які контролюють розмір риби та її початкове положення. Що стосується динаміки, центральна ідея алгоритму полягає в тому, що всі оператори працюють незалежно один від одного. Процес пошуку укладено в цикл, де відбуваються виклики раніше представлених операторів, доки не буде виконано принаймні одну умову зупинки.

Псевдокод даного алгоритму надаємо наступним чином.

1. Ініціалізація позицій всіх риб випадковим чином.

2. Ініціалізація умовної відносної ваги всіх риб. На початку для всіх риб цей параметр приймається 1.

3. Для кожної риби обрати нову сусідню випадкову позицію для здійснення індивідуального руху.

4. Оцінити функцію якості нової позиції.

5. Якщо нова позиція краща за попередню, перемістити рибу у нову сусідню випадкову позицію.

6. Знайти прирощення маси риби як відношення прирощення її власної якості до максимального прирощення у зграї.

7. Розрахувати вектор інстинктивного руху як середньозважене векторів переміщень риб, що покращили свою якість.

8. Виконати інстинктивний рух.

9. Обчислити барицентр як умовний центр ваги зграї.

10. Для кожної риби виконати рух у вигляді випадкового кроку в напрямку барицентра.

11. Перейти до 3.

12. Кінець.

Оптимізація на основі поведінки котячої зграї. У запропонованому авторами алгоритмі оптимізації [41, 42] дві основні моделі поведінки спочатку моделюються у дві підмоделі, а саме, режим пошуку та режим стеження. Завдяки поєднанню цих двох режимів із визначеною користувачем пропорцією алгоритм може продемонструвати кращу продуктивність. Автори використовують котів та модель їхньої поведінки для розв'язання задач оптимізації, тобто використовують котів для зображення наборів розв'язків. Кожен кіт має власну позицію у багатовимірному просторі, швидкості для кожного виміру, значення пристосованості, яке визначається функцією якості, та прапорця, який вказує, перебуває кіт у режимі пошуку чи відстеження. Остаточним розв'язком буде найкраща позиція одного з котів.

Режим пошуку використовується для моделювання кота, який відпочиває, озирається навколо та шукає наступне положення, куди можна перейти. У режимі пошуку визначаються чотири важливі фактори: пошуковий пул пам'яті, пошуковий діапазон вибраного виміру, кількість вимірів, які потрібно змінити, та врахування власного положення. Режим відстеження – це підмодель для моделювання дій кота під час відстеження деяких цілей.

Псевдокод алгоритму може бути наданий наступним чином.

1. Вкажіть верхню та нижню межі для параметрів проєктування.

2. Випадково згенеруйте певну кількість котів та випадково розповсюдьте їх у багатовимірному просторі, в якому кожен кіт має випадкове значення швидкості, не більше попередньо визначеного максимального значення.

3. Випадково класифікуйте кішок на шукачів та відстежувачів.

4. Оцініть значення функції якості всіх кішок.

5. Виберіть кращу кішку.

6. Перемістіть котів відповідно до їхніх ролей. Якщо кіт перебуває в режимі пошуку, застосовуйте до нього процес режиму пошуку, інакше застосовуйте до нього процес режиму пересування.

6.1. Режим пошуку:

6.1.1. Зробити деяку кількість копій поточної позиції кішки.

6.1.2. Для кожної копії випадковим чином змінити позиції.

6.1.3. Обчислити значення якості для всіх особин.

6.1.4. Випадково виберіть точку, до якої потрібно перейти, та замініть позицію.

6.2. Режим відстеження:

6.2.1. Оновіть швидкості для кожного виміру в залежності від поточного положення кішки та положення кращої особини.

6.2.2. Перевірте, чи швидкості знаходяться в діапазоні максимальної швидкості. Якщо нова швидкість виходить за межі діапазону, встановіть її рівною межі.

6.2.3. Оновіть положення кішки відповідно до визначеної швидкості.

7. Після того, як коти пройдуть режим пошуку або відстеження, для наступної ітерації випадковим чином перерозподіліть котів у режими пошуку або відстеження.

8. Перевірте умову завершення; якщо задоволені – завершити програму; інакше повторіть кроки 4–7.

Алгоритм інтелектуальних крапель води. Одним із нещодавно запропонованих алгоритмів у сфері рійового інтелекту є алгоритм інтелектуальних крапель води [43, 44]. Алгоритм базується на динаміці річкових систем, діях та реакціях, які відбуваються серед крапель води в річках. Авторів надихнули спостереження за річками в природі: безліч поворотів на їхніх шляхах та принципи їх створеня, та чи є за ними якась логіка. Авторами пропонується використовувати механізми, які діють у річках, а також сформовано алгоритм, який є кроком у цьому напрямку.

Алгоритм створено з урахуванням двох основних властивостей: швидкість та ґрунт.

Обидві властивості можуть змінюватися протягом терміну роботи алгоритму, поки він «тече» від джерела до місця призначення. Штучний потік починає свою подорож із початковою швидкістю та нульовим ґрунтом. Під час своєї подорожі він подорожує в середовищі, з якого видаляє трохи ґрунту, та може набрати деяку швидкість. Штучний потік має проходити окремими кроками. Від поточного розташування до наступного швидкість штучного потоку збільшується на величину, нелінійно зворотньо пропорційну ґрунту між двома місцями. Таким чином, доріжка з меншим вмістом ґрунту дозволяє алгоритму працювати швидше, ніж доріжка з більшою кількістю ґрунту. Штучний потік збирає ґрунт під час своєї подорожі в навколишньому середовищі та видаляє його зі шляху, що з'єднує дві локації.

Кількість ґрунту, доданого до штучного потоку, є нелінійно пропорційною до часу, необхідного для проходження потоку від поточного місця розташування до наступного. Цей проміжок часу розраховується за простими законами фізики прямолінійного руху. Таким чином, витрачений час пропорційний швидкості штучного потоку та обернено пропорційний відстані між двома місцями. Крім того, ті частини навколишнього середовища, які використовуються з більшою кількістю потоку, матимуть менше ґрунту. Можна сказати, що ґрунт є джерелом інформації, тому що як середовище, так і краплі води мають пам'ять про ґрунт.

Штучний потік потребує механізму вибору шляху до наступного місця або кроку. У цьому механізмі потік віддає перевагу доріжкам із низьким ґрунтом перед доріжками з високим ґрунтом. Така поведінка вибору шляхів реалізується шляхом накладення рівномірного випадкового розподілу на ґрунти доступних шляхів. Тоді ймовірність вибору наступного шляху обернено пропорційна ґрунтам доступних шляхів, а шляхи з меншим рівнем ґрунту мають вищі шанси бути обраними потоком.

Псевдокод алгоритму можна надати наступним чином.

1. Ініціалізація статичних параметрів: кількість крапель води, кількість ітерацій.

2. Ініціалізація динамічних параметрів – швидкості крапель води.

3. Розподілити штучні краплі води у просторі випадковим чином.

4. Оновити список вузлів, що будуть відвідуватися кожною штучною краплею води.

5. Для кожної штучної краплі води повторити:

5.1. Для краплі, що знаходиться у вузлі, виберіть наступний вузол, який не входить до списку вже відвіданих вузлів.

5.2. Для кожної краплі, що рухається від вузла до вузла, оновіть швидкість залежно від положення та функції якості.

5.3. Для кожної краплі, що рухається від вузла до вузла, обчисліть кількість ґрунту, що є еквівалентом якості, який крапля захоплює зі шляху.

5.4. Оновіть загальну кількість ґрунту, що захоплено цією краплею.

6. Знайдіть найкращий розв'язок для ітерації.

7. Оновіть ґрунти на шляхах, які формують поточний ітераційний найкращий розв'язок.

8. Оновіть загальний найкращий розв'язок.

9. Збільшити номер ітерації.

10. Кінець

Алгоритм наслідування поведінки мавп. Це алгоритм на основі агента, що імітує поведінку мавпи [45, 46]. Існує декілька ідеологічних варіацій алгоритму.

Перший алгоритм розрахований на пошук серед бінарних систем. Він головним чином зосереджений на діяльності мавпи, яка піднімається на дерево у пошуках їжі. У алгоритмі пошуку передбачається, що мавпа досліджує дерева та дізнається точні шляхи, що призводять до кращого харчування. Вважається, що дерево, по якому лазить мавпа, має розв'язок задачі оптимізації, а його гілки представляють збурення, які можуть перетворювати один розв'язок у інший. За допомогою випадкового механізму на кожному кроці мавпа вибирає гілки. Цей випадковий механізм базується на значеннях цільової функції нових розв'язків. Мавпа віддає перевагу кращим розв'язкам. Кожен раз, коли мавпа знаходить кращий розв'язок, вона повертається до кореня та зберігає його як поточний найкращий розв'язок у пам'яті. Мавпа знову піднімається на дерево, націлюючись на гілки з кращими значеннями. Для цього мавпячого пошуку використовується структура даних бінарного дерева. Передбачається, що дерева, на які піднімається мавпа, мають дві гілки, тобто ліву та праву. Структура даних двійкового дерева складається з:

1. Набір вузлів, з'єднаних шляхами, перший вузол є коренем.

2. Вважається, що корінь та вузли мають джерела харчування, які відповідають ймовірним розв'язкам задачі, що оптимізується.

3. Кожна гілка пов'язана зі обуренням поточних розв'язків. Гілки дають змогу мавпам підніматися по дереву від одного вузла до іншого, вказуючи на перехід від одного розв'язку до іншого.

4. Адаптивна пам'ять підтримується для зберігання інформації, отриманої під час процесу сканування простору розв'язків, та на неї посилаються у процесі пошуку.

Другий алгоритм, використовується для розв'язання оптимізації багатовимірних систем та будується на метафорі особливостей поведінки підйому мавпи на гору. Передбачається, що на полі є кілька гір, які позначають можливий простір пошуку задачі. Цей алгоритм був розроблений з трьома процесами, такими як процес підйому, процес стрибок-спостереження та процес сальто.

Основні етапи алгоритму можна подати наступним чином.

1. Ініціалізація популяції.

2. Індивідуальний процес лазіння мавп. Ця поведінка підйому моделюється як процес пошуку оптимального розв'язку у просторі локального пошуку відповідно до інформації про псевдоградієнт цільової функції.

3. Процес стрибку-спостереження. Після підйому 2 кожна мавпа досягає своєї вершини. Потім вона спостерігає та намагається визначити, чи є інші точки навколо неї, які є вищими за поточну. Якщо так, то мавпа перейде туди з поточної позиції. Цей процес реалізовується певною кількістю випадкових у визначеному діапазоні кроків у випадковому напрямку.

4. Процес сальто, що переносить на дослідження нових гірських вершин. У цьому процесі вибирається барицентр із поточних позицій усіх мавп, який називається опорою. Мавпи переносяться в нову позицію вперед або назад відносно точки опори. Максимальна відстань, на яку може переноситися мавпа, має назву інтервал сальто, її можна використовувати для керування процесом, який забезпечує наявність нових розв'язків у глобальному просторі пошуку.

5. Перевірте критерії завершення за допомогою попередньо визначеного номера циклу.

Якщо умова завершення не виконується, перейдіть до кроку 2 та повторіть ще раз. В іншому випадку виведіть оптимальний розв'язок та об'єктивні значення.

Електромагнітний пошук. Ідея, запропонована авторами [47, 48], проводить аналогію з механізмом тяжіння-відштовхування теорії електромагнетизму. Подібно до елементарного електромагнетизму, пропонується розглядати кожну точку як заряджену частинку, яка вивільняється у простір. Заряд кожної точки відноситься до значення цільової функції, яку намагаються оптимізувати. Цей заряд також визначає величину тяжіння або відштовхування точки над сукупністю вибірки – чим краще значення цільової функції, тим вище величина тяжіння. Після обчислення цих зарядів вони використовуються, щоб знайти напрямок руху кожної точки в наступних ітераціях. Цей напрямок вибирається через оцінку комбінованої сили, що діє на точку через інші точки. Також у алгоритмі застосовують процедуру локального пошуку для покращення деяких значень цільової функції, що спостерігаються у популяції.

Евристика пошуку складається з чотирьох фаз. Це ініціалізація алгоритму, обчислення сумарної сили, що діє на кожну частинку, рух уздовж напрямку сили та застосування пошуку сусідів для використання локальних мінімумів.

Загальну схему алгоритму можна подати так.

1. Ініціалізація. Точки випадковим чином розподіляються у допустимій області, яка є багатовимірним гіперкубом.

2. Обчислити значення цільової функції для точки.

3. Сортування. Визначити найкращу точку.

4. Локальний пошук. Процедура виконується наступним чином: спочатку обчислюється максимально можлива довжина кроку. По-друге, у межах цього кроку випадковим чином аналізують певну кількість точок. Якщо одна з цих точок краще початкової, то початкову точку замінюють на нову.

5. Розрахуйте вектор повної сили. Принцип суперпозиції теорії електромагнетизму стверджує, що сила, яка діє на точку через інші точки, обернено пропорційна відстані між точками та прямо пропорційна добутку їхніх зарядів. Відповідно до метафори, обчислюються заряди точок як значення цільової функції. Заряд кожної точки не є постійним і змінюється від ітерації до ітерації. Потім визначається напрямок сили між двома точками після порівняння значень їх цільових функцій.

6. Переміщення. Після оцінки повного вектору сили точка переміщується в напрямку сили на випадкову довжину кроку.

7. Перейти до 2.

8. Кінець. Критерієм завершення може бути максимальна кількість ітерацій або кількість ітерацій, проведених без зміни поточної найкращої точки.

Алгоритм наслідування поведінки мурашиної колонії. Це клас алгоритмів, основна ідея яких навіяна поведінкою справжніх мурах [49, 50] та полягає у паралельному пошуку кількох конструктивних обчислювальних потоків. Цей процес проходить на основі даних локальної проблеми та динамічної структури пам'яті, що містить інформацію про якість попередньо отриманого результату. Колективна поведінка, що виникає в результаті взаємодії різних потоків пошуку, виявилася ефективною у розв'язанні задач комбінаторної оптимізації.

Функціонування алгоритму можна представити наступним чином. Набір обчислювальних паралельних та асинхронних агентів (колонія мурах) переміщується через стани проблеми, що відповідають частинним розв'язкам задачі. Вони рухаються, застосовуючи стохастичну локальну політику прийняття рішень, засновану на двох параметрах, які називаються слідами та привабливістю. Рухаючись, кожна мураха поступово будує розв'язок задачі. Коли мураха завершує розв'язок або під час фази побудови, вона оцінює розв'язок та змінює значення сліду для компонентів, які використовуються у розв'язанні. Ця інформація про феромони керуватиме пошуками майбутніх мурах. Крім того, алгоритм включає ще два механізми: випаровування сліду та, за бажанням, фонові процеси. Випаровування слідів зменшує всі значення слідів з часом, щоб уникнути необмеженого накопичення слідів над деяким компонентом. Фонові процеси можна використовувати для реалізації централізованих дій, які не можуть виконуватися окремими мурахами. Наприклад, виклик локальної процедури оптимізації або оновлення глобальної інформації, яка буде використовуватися для вирішення питання про зміщення процесу пошуку з нелокальної точки зору.

Тобто мураха – це простий обчислювальний агент, який ітеративно створює розв'язок. В основі алгоритму лежить цикл, у якому на кожній ітерації кожна мураха переходить (виконує крок) із стану в інший стан, що відповідає більш повному частинному розв'язку. Тобто на кожному кроці кожна мураха обчислює набір можливих розширень до свого поточного стану та з певною ймовірністю переходить до одного з них.

Алгоритм наслідування поведінки мурашиної колонії для випадку пошукової оптимізації у просторі параметрів наступний.

1. Встановити параметри, ініціалізувати сліди феромонів.

2. Конструювати розв'язок за допомогою мурах. Тобто ініціювати комбіновану випадково-феромонну подорож мурах у просторі параметрів з аналізом якості поточного місця та з поміткою феромонами більш якісних позицій.

3. Застосувати локальний пошук.

4. Оновити феромони.

5. Перейти до 2.

6. Кінець.

Алгоритм світляків. Був створений авторами [51, 52] при натхненні природною поведінкою світлячків. Світляки використовують свою можливість світитися, щоб наближатися один до одного в темряві. Автори запропонували три припущення. По-перше, всі світлячки можуть світитися незалежно від статі, тобто кожен світлячок може притягуватися до інших світлячків. По-друге, привабливість пов'язана з інтенсивністю, яка є функцією відстані між відповідним світлячком та іншими світлячками. Зі збільшенням відстані привабливість зменшується. Нарешті, яскравість або інтенсивність світла світлячка визначається значенням функції якості поставленої задачі.

Етапи алгоритму можна представити наступним чином.

1. Ініціалізація параметрів (розмір популяції, кількість ітерацій, константи).

2. Визначити інтенсивність світла як залежність від функції якості.

3. Визначити відстані між усіма світлячками.

4. Попарно порівняти інтенсивність світла та відстані між усіма світлячками.

5. Для кожної пари перемістіть менш привабливого світлячка до більш привабливого на відстань, що є функцією інтенсивності світла та відстані між ними.

6. Перейти до 2, доки не настав критерій зупинки.

7. Ранжувати розв'язки та знайти найкраще.

Алгоритм пошуку на основі поведінки фламінго. Автор [53] представляє цікавий алгоритм оптимізації ройового інтелекту: алгоритм пошуку на основі поведінки фламінго, який створено за метафорою їхньої міграційної поведінки та їх методів пошуку їжі. Математична модель поведінки побудована так, що алгоритм має можливості локального та глобального дослідження простору. Основні метафоричні ідеї, що лягли в основу алгоритму викладені далі.

1. Фламінго комунікують один з одним, щоб повідомити своє місцезнаходження, а також наявність їжі в даному місці.

2. Популяція фламінго не знає, де найбільше їжі у поточній зоні пошуку. Натомість вони лише оновлюють місцезнаходження кожного фламінго (на яке впливає поведінка пошуку їжі та міграційна поведінка), щоб

знайти місце з більшою кількості їжі. Тобто пошукові агенти – це фламінго, які досліджують простір пошуку та розвиваються шляхом обміну інформацією між собою, також вони мають фіксовані правила переміщення локації, що в кінцевому підсумку призводить до оптимального розв'язку.

3. Правила зміни позиції засновані на поведінці фламінго: пошук їжі та міграційна поведінка. Поведінку пошуку їжі можна розділити на два процеси: сканування дзьобом та рух ніг.

Основні етапи алгоритму можуть бути представлені наступним чином.

1. Ініціалізація популяції.

2. Отримуються значення функції якості кожного окремого фламінго, та сукупність сортується відповідно до якості. Фламінго з низькими показниками якості стають мігруючими, а з високими – тими, що живляться. Особина, що живиться, робить за ітерацію декілька кроків визначеної довжини у бік особини з найкращим показником якості, аналізуючи при цьому якість кожного положення. Мігруючі особини змінюють положення випадковим чином у межах простору пошуку.

3. Змініть положення мігруючих особин та тих, хто живиться.

4. Перевірте наявність фламінго, які знаходяться поза межами поля пошуку.

5. Якщо досягнуто максимальної кількості ітерацій, перейдіть до кроку 6; інакше перейдіть до кроку 2.

6. Виведіть оптимальний розв'язок.

Алгоритм штучної бджолиної колонії. Був запропонований відносно нещодавно [54, 55], він є одним з підходів, що використовуються для пошуку оптимального розв'язку різноманітних задач. Цей алгоритм був побудований, спираючись на поведінку медоносних бджіл під час пошуку якісного джерела їжі. Схема алгоритму є відносно простою, швидкою та базується на дослідженні популяції технікою стохастичного пошуку.

В алгоритмі колонія штучних бджіл складається з трьох груп бджіл: робочі бджоли, спостерігачі та розвідники. Одна частина колонії складається з робочих штучних бджіл, а друга – зі спостерігачів. Для кожного джерела їжі є лише одна бджола. В алгоритмі положення джерела їжі представляє можливий розв'язок задачі оптимізації, а кількість нектару джерела їжі відповідає якості (придатності) відповідного розв'язку. Кількість робочих бджіл та бджіл-спостерігачів дорівнює кількості розв'язків у популяції. На першому кроці алгоритм випадково генерує розподілену початкову сукупність розв'язків (позицій джерел їжі). Кожний розв'язок є багатовимірним вектором. Після ініціалізації популяції початкових позицій (розв'язків), вона піддається повторним циклам процесу пошуку робочими бджолами, бджолами-спостерігачами та бджолами-розвідниками. Робоча бджола робить модифікацію позиції (розв'язку) у своїй пам'яті залежно від місцевої інформації (візуальної інформації) та перевіряє кількість нектару (значення придатності) нового джерела (нового розв'язку). За умови, що кількість нектару нового джерела більше, ніж попереднього, бджола запам'ятовує нове джерело та забуває старе. В іншому випадку вона зберігає попередню позицію в пам'яті. Після того, як всі робочі бджоли завершують пошук, вони обмінюються інформацією про нектар джерел їжі та інформацією їх розташування із бджолами-спостерігачами. Бджола-спостерігач оцінює інформацію про нектар, отриману від усіх робочих бджіл, та вибирає джерело їжі з імовірністю, яка пов'язана з кількістю нектару джерела. Як і у випадку з робочими бджолами, вона вносить зміни до позиції у своїй пам'яті та перевіряє кількість нектару потенційного джерела. За умови, що у ньому нектару більше, ніж у попередньому, бджола запам'ятовує нове розташування та забуває старе. Автори пропонують використовувати випадковість для пошуку нової позиції джерела. Ця послідовність операцій зациклюється зі зменшеннями поля пошуку та кроку пошуку нового джерела. Вказується, що в алгоритмі є кілька контрольних параметрів: кількість джерел їжі, яка дорівнює кількості робочих та бджіл-спостерігачів, значення обмежень, максимальна кількість циклів.

Псевдокод алгоритму пропонується представити наступним чином.

1. Ініціалізація популяції розв'язань.

2. Оцінка популяції.

3. Створення нових розв'язків робочими бджолами та їх оцінка.

4. Застосування процесу відбору.

5. Обчислення значень імовірностей для розв'язків.

6. Створення нових розв'язків спостерігачами із початкових, що обрані залежно від їх оцінки вірогідності.

7. Застосування процесу відбору.

8. Визначити залишений розвідником розв'язок, якщо він існує, то замінити його на новий випадковий розв'язок.

9. Запам'ятати найкращий розв'язок, досягнутий на поточний момент.

10. Перейти до 2.

11. Кінець, якщо досягнута максимальна кількість циклів.

Автори вказують на відповідний рівень продуктивності алгоритму порівняно з іншими відомими сучасними алгоритмами, такими як генетичний алгоритм, диференціальна еволюція, оптимізація роєм частинок.

Також слід зазначити, що існує низка алгоритмів, які є поєднанням базових метаевристичних алгоритмів, їх модифікаціями та гібридами.

Наданий огляд цікавих та різноманітних метаевристичних алгоритмів з оригінальними метафорами, на думку авторів цієї роботи, наведений не тільки з ознайомчою метою. Він повинен дати поштовх до активізації думки розробника оптимізаційних алгоритмів та стратегій у двох напрямках.

Перший напрямок – допомога підвищення розуміння та інтенсифікація донесення ідеї до цільової аудиторії вже розробленого алгоритму чи стратегії пошуку оптимальних параметрів. Як зазначалося вище, сама метафора дає можливість відійти від «сухої» наукової мови та використати зрозумілі образи та дії для опису алгоритму чи стратегії.

Другий напрямок – розробка нових алгоритмів чи стратегій пошуку оптимальних параметрів. Подібно до деяких технічних наук, наприклад, біоніки, яка черпає принципи, структури та конструкції з природи, спосте-

реження за об'єктами живої та неживої природи можуть надихнути розробників на створення нових метаевристичних алгоритмів. До речі, значна частина розглянутих у цій роботі алгоритмів була розроблена саме за цим напрямком, на що прямо вказують самі автори алгоритмів у своїх роботах.

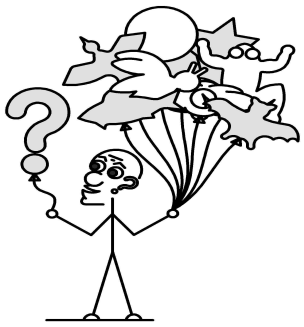

Рисунок 2 – Ілюстрація до формування нового знання у системі «ідея-алгоритм-метафора»

Формування та оформлення нового знання завжди є складним та творчим процесом (рис. 2), тож саме метафору можна використовувати для його спрощення. У такому випадку у системі «ідея-алгоритм» виникає додатковий допоміжний елемент – «метафора». А далі шляхи розвитку в цій системі можуть розвиватися досить різноманітно:

«ідея > метафора > алгоритм»,
«метафора > ідея > алгоритм»,
«ідея > алгоритм > метафора».

Висновки:

1. Доведено актуальність сучасних метаевристичних алгоритмів, освітлено ряд термінів та взаємозв'язків між ними, необхідність проведення класифікації, а також метафор, що використовуються для опису алгоритмів. Це дає змогу зрозуміти необхідність висвітлення вказаної теми та проведення досліджень літературних джерел стосовно питання.

2. Розглянуто ряд термінів і категорій, а також взаємозв'язки між ними, що дало змогу запропонувати класифікацію метаевристичних алгоритмів.

Запропоновано підхід до класифікації метаевристичних алгоритмів, що базується на термінах та поділі категорій, взятих із природничих наук. Відповідно до назв класів відбувається їх наповнення. Це дає змогу об'єднати певний сегмент знань у кластер з єдиною термінологією.

3. Розглянуто категорію «метафора» та її функції при формуванні метаевристичних алгоритмів, це дало змогу глибше зрозуміти можливості використання метафор у науковій діяльності та сформувати перелік вимог до них при описі алгоритмів.

4. Проведено огляд цікавих, оригінальних та різноманітних метаевристичних алгоритмів, що дало змогу зрозуміти сучасні тенденції стосовно цього питання, визначити переваги та недоліки цих алгоритмів, а також зрозуміти та сформувати роль метафори при їх формуванні чи описі. Також огляд дає можливість виділити два інтелектуальних напрямки використання метафор: допомога підвищення розуміння та інтенсифікація донесення ідеї до цільової аудиторії вже розробленого алгоритму чи стратегії та розробка нових алгоритмів або стратегій пошуку оптимальних параметрів.

5. Розглянуто систему формування та оформлення нового знання та ролі метафори у ній. Сформовано системний трикутник «ідея-алгоритм-метафора», а також можливі шляхи розвитку в цій системі, що дає можливість визначення чи вибору розробником шляху та наступних його етапів.

Список літератури / References (transliterated)

1. Mirjalili S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, no. 89, pp. 228–249.
2. Topal A. O., Altun O. (2016). A novel meta-heuristic algorithm: dynamic virtual bats algorithm. *Information Sciences*, no. 354, pp. 222–235.
3. Hashim F. A., Houssein E. H., Mabrouk M. S., Al-Atabany W., Mirjalili S. (2019). Henry gas solubility based algorithm. *Future Generation Computer Systems*, no. 101, pp. 646–667.
4. Abdelazim G. Hussien, Mohamed Amin, Mingjing Wang, Guoxi Liang, Ahmed Alsanad, Abdu Gumaei, Huiling Chen. (2016). Crow Search Algorithm: Theory, Recent Advances, and Applications. *IEEE Access*, vol. 4, no. 3024108, pp. 1–22.
5. Xinjie Yu, Mitsuo Gen. *Introduction to Evolutionary Algorithms*. Springer, 2010. 433 p.
6. Bondarenko O., Ustynenko O., Serykov V. (2020). Solving the problem of rational design for a two-stage reducer by using a modified evolutionary algorithm. *Proceedings of 7th International BAPT Conference Dedicated to the 90th Anniversary of Prof. Kiril Arnaudov «Power Transmissions 2020" June 10–13, 2020, Borovets, Bulgaria*. Sofia, Bulgaria: Scientific-technical union of mechanical engineering «Industry-4.0», pp. 115–122.
7. Passino K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, no. 22(3), pp. 52–67.
8. Passino, Kevin M. (2010). Bacterial foraging optimization. *Swarm. Intell. Res*., no. 1(1), pp pp.. 1–16.
9. Mohammadi-Balani A., Nayeri M. D., Azar A., Taghizadeh-Yazdi M. (2021). Golden eagle optimizer: A nature-inspired metaheuristic algorithm. *Comput. Ind. Eng*., vol. 152, no. 107050, pp. 1–12.
10. Trinh Dong-Nguyen, Vinh-Tiep Nguyen. (2022). An optimizing pulse coupled neural network based on golden eagle optimizer for automatic image segmentation. *Journal of Information Hiding and Multimedia Signal Processing,* vol. 13, no. 3, pp. 155–164.
11. Mehrabian A.R., Lucas C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, no. 1, pp. 355–366.
12. Jolaia F., Tavakkoli-Moghaddama R., Rabieeb M., Gheisarihac E. (2014). An enhanced invasive weed optimization for makespan minimization in a exible owshop scheduling problem. *Scientia Iranica, Industrial Engineering,* no. 21, pp. 1007–1020.
13. Osman K. Erol, Ibrahim Eksin. (2006). A new optimization method: Big Bang–Big Crunch. *Advances in Engineering Software,* no. 37, pp. 106–111.
14. Kaveh A, Talatahari S. (2010). A discrete Big Bang–Big Crunch algorithm for optimal design of skeletal structure. *Asian J Civil Eng*, no. 11(1), pp. 103–122.
15. Kirkpatrick S., Gelatt C. D. Optimization by Simulated Annealing. *Vecchi Science, New Series*, 1983. Vol. 220, No. 4598. P. 671–680.
16. Esin Onbas, Og Lu, Linet Özdamar. (2001). Parallel Simulated Annealing Algorithms in Global Optimization. *Journal of Global Optimization*, no. 19, pp. 27–50.
17. Emmanuel Gbenga Dada1, Stephen Bassi Joseph, David Opeoluwa Oyewola, Alaba Ayotunde Fadele, Haruna Chiroma, Shafi'i Muhammad Abdulhamid. (2022). Application of grey wolf optimization algorithm: recent trends, issues, and possible horizons. *Journal of Science*, no. 35(2), pp. 485–504.
18. Al-Tashi Q., Rais H. M., Abdulkadir S. J., Mirjalili S., Alhussian H. (2020). A Review of grey wolf optimizer-based feature selection methods for classification. *Evolutionary Machine Learning Techniques*, no. 1, pp. 273–286.
19. Askarzadeh A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, no. 169, pp. 1–12.
20. Fan Y., Zhang D. M., Chen Z. Y., Wang Y. R, Xu H, Wang Q. Q. (2020). Spectrum allocation scheme for vehicular networks based on

improved crow algorithm. *Computer science*, no. 47, pp. 273–278.
21. Geem Z. W. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, vol. 76, no. 2, pp. 60–68.
22. Geem Z. W., Lee K. S., Park Y. (2005). Application of harmony search to vehicle routing. *J. Appl. Sci*, vol. 2, no. 12, pp. 1552–1557.
23. Xue J., Shen B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Syst Sci Control Eng.,* no. 8(1), pp. 22–34.
24. Qi Xiong, Xinman Zhang, Shaobo He, Jun Shen. (2021). A fractional-order chaotic sparrow search algorithm for enhancement of long distance iris image. *Mathematics*, no. 9(2790), pp. 1–17.
25. Rashedi E., Nezamabadi-pour H., Saryazdi S. (2009). GSA: a gravitational search algorithm. *Information Science*, no. 179(13), pp. 2232–2248.
26. Rashedi E., Nezamabadi-pour N., Saryazdi S. (2011). Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 117–122.
27. Karci A., Alatas B., Akin E. (2006). Sapling growing up algorithm. *Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyomu*, no. 1, pp. 57–61.
28. Cengiz Hark, Taner Ucxkan, Ali Karcı. (2022). A new multi-document summarisation approach using saplings growing-up optimisation algorithms: Simultaneously optimised coverage and diversity. *Journal of Information Science*, no. 1, pp. 1–16.
29. Eusuff M, Lansey K, Pasha F. (2006). Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Eng Optim*, no. 38, pp. 129–154.
30. Eusuff M. M., Lansey K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Water Resour Plan Manag*, no. 129, pp. 210–225.
31. Xin-She Yang, Suash Deb. (2009). Cuckoo search via levy ights. *Proceedings of the World Congress on Nature & Biologically Inspired Computing*, *IEEE Publications*, pp. 210–214.
32. Venkata Vijaya Geeta Pentapalli, Ravi Kiran Varma. (2016). Cuckoo search optimization and its applications: a review. *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, iss, 11, pp. 556–562.
33. Johnny Kelsey, Jon Timmis. (2003). Immune inspired somatic contiguous hypermutation for function optimisation. *GECCO,* vol. 2723, pp. 207–218.
34. Jerzy Balicki. (2004). Multi-criterion evolutionary algorithm with model of the immune system to handle constraints for task assignments. *Artificial Intelligence and Soft Computing*, vol. 3070, pp. 394–399.
35. Yang X.-S. (2010). A new metaheuristic bat-inspired algorithm. *Natureinspired cooperative strategies for optimization*, pp. 65–74.
36. Amar Yahya Zebari, Saman M. Almufti, Chyavan Mohammed Abdulrahman. Bat algorithm (BA): review, applications and modifications. *International Journal of Scientific World*, 2020. No. 8(1). P. 1 –7.
37. Mirjalili S., Lewis A. The whale optimization algorithm. *Advances in engineering software,* 2016. Vol. 95. P. 51–67.
38. Kavita Jain, Akash Saxena, Ahmad M. Alshamrani, Adel Fahad Alrasheedi, Khalid Abdulaziz Alnowibet, Ali Wagdy Mohamed (2022). An amended whale optimization algorithm for optimal bidding in day ahead electricity market. *Axioms*, vol. 11, no. 456, pp. 1–33.
39. Fernando B. de Lima Neto, Anthony J. C. C. Lins, Antônio I. S. Nascimento, Marília P. Lim, Carmelo J. A. Bastos Filho. (2008). A novel search algorithm based on fish school behavior. *IEEE International Conference on Systems, Man and Cybernetics,* pp. 2646–2651.
40. Ananthi J., Ranganathan V., Sowmya B. (2016). Structure optimization using bee and fish school algorithm for mobility prediction. *Middle-East J. Sci. Res*., no. 24, pp. 229–235.
41. Chu S. C., Tsai P. W., Pan J. S. (2006). Cat swarm optimization. *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, Guilin, China, pp. 854–858.
42. Chu S. C., Tsai P. W. (2007). Computational intelligence based on the behavior of cats. *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 1, pp. 163–173.
43. Hamed Shah-Hosseini. (2009). The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. *Bio-Inspired Computation*, vol. 1, no. ½, pp. 71–79.
44. Broderick Crawford, Ricardo Soto, Gino Astorga, Carlos Castro, Fernando Paredes, Sanjay Misra, José-Miguel Rubio. (2018). Solving the software project scheduling problem using intelligent water drops. *Technical Gazette*, vol. 25, no. 2, pp. 350–357.
45. Mucherino A., Seref O. (2007). Monkey search: A novel metaheuristic search for global optimization. *AIP Conf. Proc*., vol. 953, no. 1, pp. 162–173.
46. Zhao R. (2008). Monkey algorithm for global numerical optimization. *J. Uncertain Syst*., vol. 2, no. 3, pp. 165–176.
47. Ilker Birbil S., Shu-Cherng Fang. (2003). An Electromagnetism-like Mechanism for Global Optimization. *Journal of Global Optimization*, no. 25, pp. 263–282.
48. Hosein Abedinpourshotorban, Siti Mariyam Shamsuddin, Dayang N. A. Jawawi, Zahra Beheshti. (2016). Electromagnetic field optimization:Aphysics-inspiredmetaheuristic optimization algorithm. *Swarmand Evolutionary Computation,* no. 6, pp. 8–22.
49. Dorigo M., Di Caro G. (1999). The ant colony optimization metaheuristic. *New Ideas in Optimization,* pp. 11–32.
50. Dorigo M., Di Caro G., Gambardella L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, vol. 5, no. 2, pp. 137–172.
51 Yang X.-S. (2009). Firefly algorithms for multimodal optimization. *Proceedings of the Stochastic Algorithms: Foundations and Applications; 5th International Symposium, SAGA,* pp. 169–178.
52. Smail Bazi, Redha Benzid, Yakoub Bazi, Mohamd Mahmoud Al Rahhal. (2021). A fast firefly algorithm for function optimization: application to the control of BLDC motor. *Sensors,* no. 21. Id. 5267.
53. Wang Zhiheng, Liu JIanhua. (2021). Flamingo search algorithm: a new swarm intelligence optimization algorithm. *ACCESS*, vol. 9. Id. 3090512.
54. Karaboga D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report-TR06*. Erciyes University, Engineering Faculty, Computer Engineering Department, p. 10.
55. Basturk B., Karaboga D. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, vol. 39, pp. 459–471.

Надійшла (received) 30.01.2023

Відомості про авторів / About the Authors

Бондаренко Олексій Вікторович / Bondarenko Oleksiy – кандидат технічних наук (PhD in Eng. S.), Національний технічний університет «Харківський політехнічний інститут», доцент кафедри теорії і систем автоматизованого проєктування механізмів і машин; м. Харків, Україна; тел.: (067) 189-97-00; ORCID: https://orcid.org/0000-0002-2693-5301; e-mail: avbondko@gmail.com

Устиненко Олександр Віталійович / Ustynenko Oleksandr – кандидат технічних наук (PhD in Eng. S.), доцент, старший науковий співробітник, Національний технічний університет «Харківський політехнічний інститут», професор кафедри теорії і систем автоматизованого проєктування механізмів і машин; м. Харків, Україна; тел.: (057) 707-64-78; ORCID: https://orcid.org/0000-0002-6714-6122; e-mail: ustin1964@tmm-sapr.org

Сєриков Володимир Іванович / Sierykov Volodymyr – кандидат технічних наук (PhD in Eng. S.), доцент, старший науковий співробітник, Національний технічний університет «Харківський політехнічний інститут», докторант кафедри теорії і систем автоматизованого проєктування механізмів і машин; м. Харків, Україна; тел.: (057) 707-64-78; ORCID: https://orcid.org/0000-0002-5295-3925; e-mail: SerikovVI@tmm-sapr.org