Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

**Студента** Жука Андрія Віталійовича

**академічної групи** 124м–22–1

**спеціальності** 124 Системний аналіз

**на тему:** «Розробка статистичного оператора локального пошуку для евристичних та метаевристичних алгоритмів»

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| **кваліфікаційної роботи** | *к.т.н., доц. Желдак Т. А.* | | | |
| розділів: | | | | |
| Інформаційно-аналітичний розділ | *к.т.н., доц. Желдак Т. А.* | | | |
| Спеціальний розділ | *к.т.н., доц. Желдак Т. А.* | | | |
| Рецензент | *к. ф-м. н., доц. Лебідь О. Ю.* | | | |
| Нормоконтролер | *к.ф.-м.н., доц. Хом'як Т. В.* | | | |

Дніпро
2023

ЗАТВЕРДЖЕНО:
**завідувач кафедри**
*Системного аналізу та управління*
(повна назва)
_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище, ініціали)

«_____»_____ 20___року

**ЗАВДАННЯ**
**на кваліфікаційну роботу**
**ступеня** *магістра*

**студенту** *Жуку А.В.* **академічної групи** *124м–22–1*

**спеціальності:** *124 Системний аналіз*

**на тему:** *«*Розробка статистичного оператора локального пошуку для евристичних та метаевристичних алгоритмів*»*

затверджену наказом ректора НТУ «Дніпровська політехніка»
від *09.10.2023 р. №1227-с*

| Розділ | Зміст | Термін виконання |
|---|---|---|
| 1. Інформаційно-аналітичний | Дослідити проблематику математичної оптимізації у сфері вирішення комбінаторних задач, проаналізувати прикладні методи їх вирішення. | 04.09.2023 – 18.10.2023 |
| 2. Спеціальний | Обгрунтувати вибір алгоритму для вирішення комбінаторних задач з використанням локального пошуку. Підготувати набір тестових функцій, реалізувати обраний алгоритм та провести порівняльний аналіз ефективності роботи його операторів. | 18.10.2023 – 30.11.2023 |

Завдання видано _____ *доц. Желдак Т. А.*
(підпис) (прізвище, ініціали)

Дата видачі: 04.09.2023 р.

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ Жук А.В.
(підпис студента) (прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 80 с., 7 рис., 13 табл., 5 додатків, 86 джерел.

*Об'єктом дослідження* в роботі є використання евристичних та метаевристичних алгоритмів, для вирішення задач комбінаторної оптимізації.

*Предметом дослідження є* локальний пошук у задачах комбінаторної оптимізації.

*Метою* даної кваліфікаційної роботи є пошук та дослідження можливого впливу статистичних оцінок на вирішення комбінаторних задач у евристичних та метаевристичних алгоритмах.

*Методи дослідження*: метод наукового узагальнення і систематизації, метод формалізації, методи моделювання стохастичних пошукових евристик.

*В інформаційно-аналітичному розділі* були розглянуті основні поняття теорії оптимізації, описана проблематика комбінаторної оптимізації, були розглянуті практичні методи вирішення задач даного класу. Також проведений аналіз використання метаевристик пов’язаних з локальним пошуком.

*У спеціальному розділі* була запропонована нова метаевристика, пов’язана з застосуванням оператора локального пошуку при вирішені комбінаторних задач за допомогою стохастичних пошукових евристик, реалізовано алгоритм штучної імунної системи з різними варіантами операторів, підготовлено набір тестових функцій, виконано аналіз отриманих результатів тестування.

*Практична цінність* отриманих результатів полягає у тому, що запропонований новий підхід до використання локального пошуку, базованого на статистичних оцінках популяційного різноманіття, при вирішені задач комбінаторної оптимізації з використанням стохастичних пошукових евристик.

*Апробація результатів дослідження* проводилася на всеукраїнському конкурсі студентських наукових робіт зі штучного інтелекту 2023.

*Ключові слова*: КОМБІНАТОРНА ОПТИМІЗАЦІЯ, СТОХАСТИЧНІ ПОШУКОВІ ЕВРИСТИКИ, МЕТАЕВРИСТИКА, ЛОКАЛЬНИЙ ПОШУК, ШТУЧНА ІМУННА СИСТЕМА.

# ABSTRACT

Explanatory note: 80 p., 7 fig., 13 table, 5 appendix, 86 references.

*The object of research* in this paper is heuristic and metaheuristic algorithms used to solve combinatorial optimization problem.

*The subject of research* is local search in combinatorial optimization.

*The purpose of this qualification work* is to find and study the possible impact of statistical estimates on the search for solutions to combinatorial problems when using evolutionary algorithms.

*Research methods:* method of scientific generalization and systematization, method of formalization, methods of modelling stochastic search heuristics.

*In the information and analytical section*, the basic concepts of optimization theory were considered, the problems of combinatorial optimization were described, practical methods for solving problems of this class were considered, and the use of metaheuristics related to local search was analyzed.

*In a special section*, a new approach to the use of local search in solving combinatorial problems using stochastic search heuristics was proposed, an artificial immune system algorithm with different variants of operators was implemented, a set of test functions was prepared and a comparative analysis of the test results was carried out.

*The practical value of the results* obtained is that a new approach to the use of local search in solving combinatorial optimization problems using stochastic search heuristics based on statistical estimates of population diversity is proposed.

*The research results were tested* at the All-Ukrainian competition of student research papers on artificial intelligence 2023.

*Keywords:* COMBINATORIAL OPTIMISATION, STOCHASTIC SEARCH HEURISTICS, METAHEURISTICS, LOCAL SEARCH, ARTIFICIAL IMMUNE SYSTEM.

**ЗМІСТ**

# ВСТУП

Прогрес. На перший погляд, здається, що це просте слово з 7 літер, яке не містить нічого важливого та стає повсякденним у теперішній час. Майже кожного місяця у новинах можливо побачити згадку про прогресивну розробку, модель, думку, проект, закон тощо. За цим всім ми втрачаємо справжній зміст цього слова. Так, виходячи з тлумачення словника української мови: «Прогрес – це розвиток по висхідній лінії, удосконалення в цьому процесі, перехід від нижчого до вищого, від простого до більш складного…»[84].

Дійсно, вивчаючи історію людства, ми можемо побачити увесь його шлях розвитку по висхідній лінії, який привів нас до сьогодення. При цьому, одним з головних аспектів даного розвитку є питання того, як отримати максимальний результат за умови обмежених ресурсів.

Таким чином, ми приходимо до розуміння того, яким самим чином було винайдено поняття оптимізація, та відповідну їй наукову дисципліну – математичне програмування. При цьому, в залежності від того, про яку множину змінних йде мова, дискретну або глобальну, відповідну оптимізаційну задачу поділяють на комбінаторну [23, 31] або неперервну [26, 30].

Використання алгоритмів локального пошуку для задач комбінаторної оптимізації наразі є дуже поширеною практикою [17], та набуває нової популярності через зміну поглядів на їх побудову. Проте, ця задача не є такою простою, як здається на перший погляд. Проблема полягає у тому, що основна мета локального пошуку - оптимізація певної цільової функції, а не пошук оптимального рішення. У свою чергу, це випливає з того факту, що пошук відбувається на певному кроці роботи того чи іншого алгоритму, без прив’язки до минулих станів та їх результатів. Вибір алгоритму для вирішення конкретної задачі комбінаторної оптимізації є темою окремих досліджень, результати яких відрізняються, в залежності від типу та розмірності самої проблеми.

У даній роботі розглядається такі оптимізаційні проблеми, як:

- задача про наплічник (Knapsack problem);
- задача про покриття множини (Set covering problem);
- задача про мінімальне вершинне покриття (Minimal vertex cover);
- задача про призначення (Set partition problem).

Емпіричні та теоретичні дослідження показують, що успіх локального пошуку сильно залежить від типу задачі та внутрішніх операторів пошуку. У зв'язку з цим у літературі було представлено декілька внутрішніх операторів локального пошуку. Однак, рішення про те, який саме варіант слід використовувати і з якими операторами є дуже складним і вимагає тривалого процесу спроб і помилок. При цьому, при роботі з практичними задачами завжди виникає потреба у ручному налаштуванні оператора локального пошуку, щоб краще їй відповідати [64].

Хоча, встановлення правильної комбінації операторів має вирішальне значення, ручне налаштування передбачає, що при застосуванні до іншої задачі, ймовірно, оптимальна комбінація буде іншою. Дана думка цілком узгоджується з «Теоремою про відсутність безкоштовних сніданків Вульперта та Маккріді», яка доводить, що не існує єдиного алгоритму, який найкраще розв’язував би всі задачі[68].

Проте, Дросте, Джансес та Вегенер розглядаючи цю теорему прийшли до наступного висновку, назвавши цю теорему як «Безкоштовних сніданків (майже) не існує» [14].

Однією з причин для такого висновку є ідея об’єднання специфічних знань, різних підходів до вирішення певної задачі з метою отримання максимальної ефективності. Такий підхід отримав назву – Метаевристичність, і полягає у тому, що певну процедуру алгоритму вкладають у іншу стратегію. Такий тип алгоритмів працює ефективно та є результативним для задач функціональної оптимізації або проблем реального світу [53, 36].

Варто зазначити, що більшість метаевристичних алгоритмів є компромісними за своєю природою. У свою чергу, багато досліджень доводять, що використання оператора локального пошуку, при знаходженні глобального

оптимуму, підвищує продуктивність самого алгоритму [56]. За останні роки було багато спроб створити гібридний алгоритм за даними принципом, що призвело до отримання чудових результатів оптимізації функції однієї [61, 32] та кількох цілей [73,75].

Аналогічна ситуація і з вирішенням практичних задач, що пов’язані з комбінаторної оптимізацією. Поєднання оператора локального пошуку з евристичним алгоритмом дає змогу вирішити NP – складну задачу за відносно швидкий проміжок часу [41, 27].

*Мета дослідження*: пошук та дослідження можливого впливу статистичних оцінок на вирішення комбінаторних задач у евристичних та метаевристичних алгоритмах.

*Наукова ідея:* запропонувати такий оператор локального пошуку, який міг бути застосований до будь-яких задач комбінаторної оптимізації (за умови їх дискретності), зі збереженням своєї ефективності. При чому, ефективність роботи даного оператора забезпечується використанням статистичного апарату.

*Об'єкт дослідження:* використання евристичних та метаевристичних алгоритмів, для вирішення задач комбінаторної оптимізації.

*Предмет дослідження:* локальний пошук у задачах комбінаторної оптимізації.

*Методи дослідження*:

- метод наукового узагальнення і систематизації. Для проведення аналізу питань комбінаторної оптимізації та огляду прикладних методів пошуку рішень;
- метод формалізації. Для формального опису кожної з задач та опису запропонованого рішення локального пошуку
- методи моделювання стохастичних пошукових евристик. Для тестування запропонованого оператора та визначення його ефективності пошуку рішення.

*Наукова новизна результату дослідження*: запропонована нова метаевристична методика використання локального пошуку у популяційних алгоритмах.

*Практичне значення результатів роботи:* ефективність роботи даного оператора на базових задачах дає можливість покращення процес їх вирішення для прикладних задач.

*Отримані результати дослідження* були представлені на всеукраїнському конкурсі студентських наукових робіт зі штучного інтелекту 2023.

# РОЗДІЛ 1 ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ

## 1.1 Огляд типових проблем комбінаторної оптимізації

Дослідження будь – якого методу вирішення поставленої проблеми, починається з її математичного опису. Такий опис, передбачає визначення цільової, тестової функції максимізації або мінімізації, за умови використання асоційованого набору екземплярів.

Прикладом задачі комбінаторної оптимізації може бути пара $(S, f)$, де $S$ – це скінчена множина розв'язків, та $f: S \to R$ – це функція, яка ставить кожному $s \in S$ значення $f(s)$. Зазвичай $f(s)$ називають цільовою функцією. Мета самої задачі комбінаторної оптимізації полягає у знаходженні такого рішення $s \in S$, для якого значення цільової функції є мінімальним, тобто $f(s_{opt}) \leq f(s') \; \forall \; s' \in S$. У даному випадку, $s_{opt}$ називається глобальним, оптимальним, розв'язком задачі $(S, f)$, а $S_{opt}$ – множиною всіх глобальних рішень [60].

Варто зазначити, що оптимізаційна задача мінімізації дуже легко може бути перетворена на задачу максимізації своєї цільової функції. Проте, у випадку задачі мінімізації цільову функцію також називають функцією вартості, а саме отримуване значення – вартістю. При цьому, припускається, що функція витрат набуває лише невід'ємних значень, а саме $f\ (s) \geq 0 \; \forall \; s \in S$.

У свою чергу, задачі даного типу можуть бути розв'язані у трьох різних варіантах, а саме:

- варіант пошуку: за заданими даними $(S, f)$, знайти таке оптимальне рішення, щоб елемент $s_{opt} \in S$;
- версія оцінювання: за заданими даними $(S, f)$, знайти таке оптимальне значення цільової функції $f(s_{opt})$;

- версія пошуку рішення: за заданими даними $(S, f)$, та обмежень $L$ визначити, чи існує такий допустимий розв'язок, при якому $s \in S$, а $f(s) \leq L$.

Очевидним є те, що версія пошуку є найзагальнішою з трьох, оскільки знаючи оптимальний розв'язок, версія оцінювання та пошуку рішення вирішується тривіально. Простір $S$ будемо називати простором пошуку. Скінченність $S$ вказує на те, що будь-який приклад $(S, f)$, може бути вирішений шляхом перебору всієї множини розв'язків і вибрати такий, що має мінімальну вартість. Проте, такий підхід неможливо використовувати для багатьох задач, оскільки розмір простору пошуку, який позначається через $|S|$, зростає поліномінально з розміром прикладу [60].

Для перевірки та порівняння ефективності роботи алгоритмів оптимізації було обрано чотири комбінаторні задачі оптимізації, а саме:

- задача про наплічник (Knapsack problem);
- задача про покриття множини (Set covering problem);
- задача про мінімальне вершинне покриття (Minimal vertex cover);
- задача про призначення (Set partition problem).

### 1.1.1 Постановка задачі про наплічник

У загальному вигляді, задача про наплічник ( в літературі часто зустрічається і інша її назва, а саме задача про ранець) є однією з класичних у комбінаторній оптимізації. Основна ідея, полягає в тому, щоб вибрати підмножину предметів з певної, наперед заданої множини так, щоб сумі їх ваги не перевищувала певного обмеження ( в даному випадку ваги рюкзака), а сума їхньої цінності була максимальна.

Математично, це виглядає наступним чином. Нехай, ми маємо певну множину предметів $Q = \{q_1, q_2, \dots, q_n\}$, для кожного з яких відома цінність $c_i$, та визначений об'єм $a_i$. Необхідно упакувати наплічник $W$ таким чином, щоб

загальна цінність запакованих предметів була найбільша, без перевищення зазначеного об’єму.

Зазвичай, для вказання того, чи запаковано той чи інший предмет використовується двійковий вектор $X = (x_1, x_2, \dots, x_n)$, де компонент $x$ дорівнює нулю або одиниці в залежності від пакування певного предмету.

Таким чином, отримується математичне формулювання, задачі про наплічник (3.1.1), за умови виконання умов (3.1.2).

$$F = c_1x_1 + c_2x_2 + \dots + c_nx_n = \sum_{i=1}^{n} c_ix_i \to max, \quad (1.1)$$

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = \sum_{i=1}^{n} a_ix_i \leq W. \quad (1.2)$$

Варто зазначити, що якщо математична моделі задачі містить тільки одне обмеження вигляду (1.2.2), то задача про наплічник називається одновимірною, а в протилежному випадку – багатовимірною.

Незважаючи на свій загальний опис, даний тип задач має широке коло застосування в прикладних сферах людського життя, наприклад:

- логістика та управління запасами. При плануванні та управлінні запасами компанії часто необхідно вибирати товари для поставок або зберігання в залежності від їх ваги та вартості;
- фінанси. В інвестиційному портфелі можливо розглядати цінні папери, як предмети, а вартість імовірного прибутку або ризику – як цінність. Мета полягає у максимізації отриману цінність при обмеженому обсягу інвестицій;
- комбінаторна хімія. При пошуку оптимальних комбінацій реагентів для синтезу хімічних сполук, де кожен реагент має свою вагу та ефективність;
- маркетинг. При виборі набору дій (наприклад, рекламних заходів), максимізуючи їх ефективність при обмежених ресурсах( наприклад, бюджет).

### 1.1.2 Постановка задачі про покриття множини

Дана задача є надзвичайно, NP – складною, задачею комбінаторної оптимізації. Основна проблема полягає у тому, що необхідно знайти мінімальну кількість наборів, яка покриває усю множину.

Тобто, ми маємо два набори, а саме: $S$, як множину елементів, та $U$, як підмножину вищезгаданої множини. Кожна підмножина $U$ має свою вартість $C$, а об'єднання усіх підмножин повністю покриває $S$.

Аналогічно, до задачі про наплічник, використовується двійковий вектор $X = (x_1, x_2, \dots, x_n)$, де компонент $x$ дорівнює нулю або одиниці в залежності від вибору тієї чи іншої підмножини. Таким чином, отримується наступна цільова функція:

$$\sum_{i=1}^{n} c_i x_i \to min, \tag{1.3}$$

при виконанні наступних вимог

$$\sum_{i=1}^{n} x_i \geq 1, \forall i = 1, \dots, n \tag{1.4}$$

Задача покриття множини має важливі застосування в практиці, такі як:

- оптимізація маршрутів. Вибір мінімальної кількості рейсів авіакомпанії для покриття всіх міст або аеропортів, які необхідно обслуговувати;
- проектування тестових випробувань. Полягає у виборі мінімальної кількості тестів для покриття всіх можливих дефектів або властивостей системи;
- розміщення сенсорів. Розміщення мінімальної кількості сенсорів для виявлення подій або покриття певної області;
- розсилання рекламних листів. Вибір мінімальної кількості рекламних листів для охоплення всієї цільової аудиторії.

### 1.1.3 Постановка задачі про мінімальне вершинне покриття

Дана задача полягає у пошуку мінімального вершинного покриття у певному графі $G(V, E)$, де $V$ – множина вершин, а $E$ – множина ребер. Найпростіше формулювання полягає у тому, що існує множина вершин даного графа $C$, яка є вершинним покриттям, де для кожного ребра $(u, v) \in E$, принаймні одна з вершин $u$ або $v$ має відноситися до множини $C$ . При цьому, аналогічно до попередніх задач, для пошуку мінімального покриття, використовується двійковий вектор $X = (x_1, x_2, \dots, x_n)$, де компонент $x$ дорівнює нулю або одиниці в залежності від вибору тієї чи іншої вершини. Тоді, математичне формулювання виглядає наступним чином:

$$\sum_{v \in V} x_v \to min, \tag{1.5}$$

При дотриманні наступних умов

$$x_u + x_v \geq 1 \ \ \forall \{u, v\} \in E \tag{1.6}$$

Задача мінімального вершинного покриття має численні застосування в різних областях, наприклад:

- соціальні мережі. В графах соціальних мереж можливо використовувати мінімальне вершинне покриття для визначення найменшої кількості осіб, які представляють групу чи спільноту;
- визначення областей внутрішнього покриття. У сфері дизайну чи архітектури дана задача може бути використана для визначення мінімального набору об’єктів чи пунктів, які визначають область внутрішнього покриття;
- мережеве планування телекомунікацій. Можливе використання даної задачі для визначення оптимального розташування ретрансляційних веж або інфраструктури для покриття всієї території з мінімальними витратами.

### 1.1.4 Постановка задачі про призначення

Дана задача полягає у тому, щоб визначити як елементи однієї множини можуть бути розбиті на певну кількість малих підмножин. З практичної точки зору – це може бути задача найму.

Нехай, для виконання певної роботи необхідний набір навичок $U$. Також, є група людей, де кожна особа володіє певними навичками. Необхідно сформувати групу, де усі навички були присутніми та унікальними, а вартість послуг була мінімальна.

Нехай, $n-$ кількість претендентів, а $m$ – кількість навичок. Тобі ми будемо мати матрицю $A(n \times m)$ та вектор $C = (c_1, c_2, \dots, c_n)$, які містять перелік навичок та вартість послуг всіх претендентів. При цьому, аналогічно до попередніх задач, використовується двійковий вектор $X = (x_1, x_2, \dots, x_n)$, де компонент $x$ дорівнює нулю або одиниці в залежності від вибору того чи іншого претендента. Таким чином отримуємо наступну задачу:

$$F = \sum_{i=1}^{n} c_i x_i \to min, \tag{1.7}$$

При дотриманні наступних умов

$$\forall\, y_j = 1 : Y = A \times X, j = \overline{1, m} \tag{1.8}$$

Аналогічно задача про призначення може бути застосована і до інших сфер, а саме:

- розподіл ресурсів. У задачах, де ресурси (наприклад, час, бюджет) повинні бути розподілені між підпроектами чи завданнями;
- планування подій. У випадках, коли потрібно розподілити ресурси або простір для проведення подій чи конференцій.

## 1.2 Питання алгоритмічної складності задач комбінаторної оптимізації

Перед тим, як перейти до огляду підходів до вирішення комбінаторних задач необхідно звернути увагу на те, яка складність їм присутня. Теорія алгоритмів включає класифікацію задач за їхньою обчислювальною складністю. Два основних класи проблем, що виникають у цьому контексті, це P та NP

Клас P (Polynomial Time) включає у себе ті обчислювальні задачі, для яких існують алгоритми, що працюють у часовій складності, обмеженій поліноміальною функцією від розміру вхідних даних. Прикладом таких задач може виступати сортування масиву чисел або знаходження найменшого елемента, тощо.

Клас NP (Nondeterministic Polynimal Time) включає ті обчислювальні задачі, для яких відповіді можуть бути перевірені за поліноміальний час, хоча існуючі алгоритми працюють у часовій складності, обмеженій експоненціальною функцією від розміру даних. Комбінаторні задачі оптимізації, наведені у підрозділах 1.1.1 – 1.1.4 якраз відносяться до даного класу.

## 1.3 Огляд застосування точних методів оптимізації

Одним з перших підходів до вирішення задач комбінаторної оптимізації є використання методів, які засновані на використанні математичних моделях та гарантують отримання оптимального рішення шляхом дослідження всіх можливих варіантів. Проте, одним з головних недоліків даного класу алгоритмів є експоненціальне зростання часу пошуку при збільшенні розмірності вхідних даних.

### 1.3.1 Метод повного перебору

Даний метод є одним з найперших і полягає у переборі всіх можливих варіантів розв’язку. Не зважаючи на те, що цей підхід дуже просто реалізувати і результат завжди є гарантованим, вартість реалізації зростає у відповідності до кількості змінних. Саме тому використання даного алгоритму є доцільне тільки коли кількість змінних є малою.

### 1.3.2 Метод гілок і меж

Метод гілок та меж є алгоритмічним підходом для вирішення комбінаторних задач оптимізації. Він використовує стратегію розділення задачі на менші частини (гілки) та встановлення верхніх та нижніх меж для кожної гілки з метою ефективного обмеження простору пошуку. Основна ідея полягає в тому, щоб систематично розглядати всі можливі варіанти розв’язків та ефективно відсікати гілки, які не можуть призвести до оптимального рішення.

Процес роботи методу включає в себе гілкування, що означає розбиття простору станів на підпростори, оцінювання, де встановлюються верхні та нижні межі для обмеження гілок,  і вибір гілки для подальшого розгляду на основі встановлених меж та критеріїв вибору.

Даний підхід є ефективний для великих задач зі значним простором станів, де метод повного перебору стає неефективним. Він забезпечує гарантію оптимальності, проте успішність методу напряму залежить від стратегій гілкування, що напряму веде до ускладнення його реалізації.

### 1.3.3 Метод гілок і відтинань

Метод гілок та відтинань є розширенням методу гілок та меж, спрямованим на вирішення задач комбінаторної оптимізації та лінійного програмування. Основна ідея полягає в комбінації стратегій гілкування з лінійним програмуванням для оптимізації пошуку рішення в задачах з великим простором станів.

Аналогічно методу гілок та меж, він використовує техніки розбиття задачі на менші гілки та використовує лінійне програмування для швидкого та ефективного обчислення меж для кожної гілки. У свою чергу, обмеження (відтинання) генеруються з метою відсічення гілок, які не можуть призвести до оптимального рішення.

При цьому, варто відмітити, що для цього підходу важливим є те, щоб ціль дорівнювала верхній межі. У такому випадку знаходження оптимального розв'язку є гарантованим [46].

### 1.3.4 Метод послідовного аналізу варіантів

Метод послідовного аналізу варіантів також є одним з відомих підходів до вирішення задачі комбінаторної оптимізації. Загальна схема роботи алгоритму , за даним методом, розроблена Михалевичем В. С. на основі ідей теорії послідовних рішень та динамічного програмування [82].

Сама методика послідовного розвитку, аналізу та відсіву варіантів передбачає їх етапний розвиток та створення відповідних операторів аналізу, що дозволяє відкидати безнадійні частини перед повною їх конструкцією. Такий підхід дозволяє заощаджувати обчислювальні ресурси, оскільки при відсіві несприятливих початкових частин варіантів відсівається весь набір їх можливих продовжень.

### 1.3.5 Динамічне програмування

Динамічне програмування є одним з найбільш популярних точних методів вирішення задач комбінаторної оптимізації, починаючи з моменту його заснування Річардом Беллманом у 1950 – х роках. Схема застосування такого підходу передбачає можливість спрощення складної задачі шляхом розбиття її на малі під-задачі, рекурсивним чином. Хоча деякі проблеми пов'язані з прийняттям рішень не можуть бути розбиті таким чином, рішення, які охоплюють кілька моментів часу, часто розбивають рекурсивно.

Таким чином, важливо відокремити дві ключові характеристики, які повинна мати задача, щоб до неї можливо було застосувати динамічне програмування:

- оптимальна підструктура. Розв’язок задачі оптимізації може бути отриманий комбінацією оптимальних розв’язків її підзадач;
- перекриття підзадач. Тобто, простір підзадач повинен бути невеликий, для того, щоб рекурсивний алгоритм розв’язував одні й ті ж підзадачі знову і знову, а не генерував нові.

У випадку, якщо задача не володіє цими характеристиками, то стратегія пошуку рішення називається «розділяй і володарюй» [8].

### 1.3.6 Спеціальні алгоритми

Спеціальні алгоритми будуються на основі врахування специфіки конкретної задачі оптимізації, що розв’язується, тому мають вузький спектр застосування.

Приклад – метод Балаша (угорський метод) для розв’язання лінійної задачі про призначення. Один з небагатьох прикладів поліноміального алгоритму для розв’язання задачі комбінаторної оптимізації. Однак, при додаванні однієї чи кількох обмежувальних умов виникають подібні задачі, але зазначений алгоритм уже не можу бути застосованим чи його модифікація уже не має поліноміальної складності [83].

## 1.4 Огляд застосування наближених методів оптимізації

В наш час, широке застосування мають наближені методи оптимізації. На відміну від точних методів, які обчислюють оптимальне значення цільової функції, наближені обчислюють значення, яке знаходиться в межах деякого

коефіцієнта $\alpha$ ( константа або функція розміру задачі) від оптимального значення для всіх екземплярів задачі.

Серед основних переваг та причин популярності наближених методів виокремлюють наступні фактори:

- наближені методи мають меншу складність виконання порівняно з точними, в більшості лінійних або майже лінійних за розміром задачах. Точні підходи з поліноміальним часом виконання можуть бути занадто повільними для масивних графів, тому швидші наближені є набагато практичними. Крім того, на практиці за допомогою даних методів досягається розв'язок, який є майже оптимальним;
- концептуально, сам підхід є набагато простіший ніж точний. Відповідно, докази правильності роботи кожного з них є простішим;
- наближені практики легше реалізувати у порівнянні з точними, що є практичною причиною їх популярності;
- наближені алгоритми можуть мати в собі більше паралелізму ніж точні. Саме з точки зору реалізації на паралельних комп'ютерах простота реалізації є одним з основних аспектів вибору.

### 1.4.1 Жадібні алгоритми

Жадібним алгоритмом можливо назвати будь-який алгоритм, який слідує евристиці розв'язання задач, що полягає у виборі локально оптимального рішення на кожному етапі. У багатьох задачах використання такого підходу дає лише локально оптимальні рішення, які наближаються до глобального за допустимий проміжок часу.

Причина, чому сімейство даних алгоритмів має таку поведінку, полягає у відсутності можливості оперування усім набором даних. Це призводить до того, що алгоритм занадто рано приймає рішення, що

заважає надалі знайти глобальний оптимум. Все ж таки, вони є корисними під час розв'язування задач комбінаторної оптимізації, саме через швидке прийняття рішень та гарне наближення до глобального оптимуму.

### 1.4.2 Детермінований локальний пошук

Детермінований локальний пошук – це сімейство алгоритмів, які намагаються поліпшити поточне рішення, переміщуючись по сусіднім просторах станів та вибираючи кращі можливі варіанти.

Найбільш простий варіант локального пошуку - це ітеративне покращення, яке змінює лише поточне рішення і зупиняється, як тільки жодних кращих, сусідніх рішень не можливо більше знайти.

Локальний пошук не є новим методом вирішення NP – складних задач, так як перші згадки про нього були ще в пізні п'ятдесяті, на початку шістдесятих років минулого сторіччя. Проте, початковий інтерес у даного типу алгоритмів знижувався через брак нових концептуальних робіт та його успіх був заснований лише на практичній корисності. Також, при потребі забезпечення високої якості отриманого рішення, за умови розгляду великої проблеми, була необхідна велика обчислювальна потужність, яка не була наявна у ті часи [18, 60].

Також, до цього сімейства алгоритмів відноситься керований локальний пошук та Табу – пошук.

Керований локальний пошук заснований на використанні таких стратегій керування, що дозволяють уникнути потрапляння у локальний оптимум. До таких стратегій можливо віднести вибір самих локальний перетворень або ж оптимізація порядку їх виконання. У кінцевому результаті, основний акцент робиться на тих перетвореннях, що мають найбільший потенціал для поліпшення рішення через свою радикальність чи масовість.

При цьому, отримується адаптивність до змін у структурі простору рішень, що забезпечує здатність уникнення локальних мінімумів шляхом зміни стратегії пошуку. Проте, варто не забувати про потребу в узгодженості

стратегій, що використовуються та залежність від вибору самих параметрів та методів керування.

У свою чергу, Табу – пошук – це метод комбінаторної оптимізації, що поєднує в собі локальний пошук з забороненим списком. Заборонений список являє собою перелік локальних перетворень, що були виконані в останній час. Локальні перетворення, які є членами забороненого списку не можуть бути виконані в наступних ітераціях. Тобто, відбувається використання стратегії управління пам'яттю для уникнення повторних відвідувань підоптимальних рішень у просторі пошуку [20].

Таким чином, використання даної стратегії забезпечує можливість її застосування для різноманітних комбінаторних задач з уникненням циклічності та повторень в процесі пошуку. Негативною стороною даного підходу є залежність від правильного вибору стратегії управління самою пам’яттю та збільшена ймовірність потрапляння до локального мінімума чи максимума.

### 1.4.3 Стохастичний локальний пошук

Стохастичний локальний пошук представляє собою відокремлене сімейство алгоритмів вирішення задач комбінаторної оптимізації, яке передбачає обов’язкову наявність випадкових компонент.

Загалом, дане сімейство включає у себе цілий набір різних методів, починаючи від простих конструктивних та ітеративних процедур поліпшення до більш складних, такі як імітаційний відпал, ітеративний локальний пошук та навіть еволюційні алгоритми. Виходячи з терміну «стохастичний локальний пошук», випадковість може і часто саме відіграє важливу роль у цих методах. Випадковий вибір може бути використаний при генерації початкових розв’язків або при виборі кроку пошуку, який слід виконати далі. Наприклад, для розірвання зв’язків між еквівалентними альтернативами, а іноді евристично та імовірнісно вибрати з великих і різноманітних наборів можливих кандидатів.

При цьому, варто розуміти, що поняття «стохастичний локальний пошук» було визначено формально [25], і не тільки забезпечує об’єднуючу основу для

багатьох різних типів алгоритмів, включаючи раніше згадані конструктивні та ітераційні процедури покращення, але також надає широкий спектр більш складних методів пошуку, широко відомих як метаевристика [29].

Основними прикладами алгоритмів стохастичного локального пошуку є такі методи, як:

- алгоритм імітаційного відпалу;
- квантовий відпал;
- повторювальний локальний пошук.

Алгоритм імітаційного відпалу – це метод комбінаторної оптимізації, який імітує фізичний процес відпалу. У даному процесі тверде тіло нагрівається до високої температури, а потім поступово охолоджується. Під час охолодження, тверде тіло може приймати різні форми, проте воно прагне до такої, що має мінімальну потенційну енергію [40, 65].

Сам алгоритм працює аналогічним чином. Спочатку отримується певне, початкове рішення задачі. Потім алгоритм послідовно генерує нові рішення, які є близькими до початкового. При цьому, нові рішення вибираються за допомогою випадкового процесу. Якщо отримане рішення є краще за попереднє – воно обирається наступним. В іншому випадку воно приймається лише з заданою ймовірністю. Сама ж ймовірність цілком залежить від параметра температури, який є динамічним і змінюється у часі. Тому, чим менша стає значення параметру температури – тим менша ймовірність прийняття поганого рішення.

Перевагами застосування даного методу є здатність уникнення локальних мінімумів через прийняття гірших рішень, придатність для оптимізації великих просторів рішень, відсутність потреби у наявності знань про структуру задачі. Недоліками є залежність від правильного вибору параметрів та висока обчислювальна складність (у порівнянні з деякими евристичними методами), що зменшує швидкість отримання оптимального рішення.

Квантовий відпал являє собою метод комбінаторної оптимізації аналогічний імітаційному відпалу, проте головною обчислювальною одиницею є квантовий комп’ютер. Квантовий комп’ютер дозволяють генерувати нові рішення, які є більш близькими до глобального оптимуму, ніж за умови використання класичного комп’ютера. При цьому, для знаходження такого рішення знадобиться набагато менше часу.

Варто зазначити, що окрім звичайних недоліків, які наявні для усіх методів даного сімейства, квантові комп’ютери поки що є дорогими і складними в експлуатації.

Ітераційний локальний пошук генерує послідовність розв’язків шляхом почергового застосування механізму перестановки та допоміжного алгоритму локального пошуку. Таким чином, даний підхід можна вважати гібридом між пошуковими методами, що лягли в основу локального пошуку, та фазами перестановок.

Даний тип алгоритмів заснований на чотирьох головних компонентах. По-перше, -  це механізм, що використовується для генерації початкового рішення . По-друге, це процедура перестановки у локальному пошуку, що являє собою простий ітеративний алгоритм. По-третє, це процедура перестановки, що вносить мутації до обраних кандидатів. При цьому, важливо, щоб вплив перестановок, що внесли модифікацію у кандидата, не були легко повернуті до початкового стану за допомогою локального пошуку. Останнім елементом є критерій оцінки оптимальності отриманого рішення та вирішення долі його подальшого застосування.

Перевагою даного алгоритму є простота та ефективність у випадках, коли локальний пошук може ефективно поліпшити поточне рішення. При цьому він володіє усіма недоліками попередніх рішень.

### 1.4.4 Еволюційні алгоритми

Еволюційні алгоритми є потужним інструментом для розв’язання задач комбінаторної оптимізації у науковій та інженерній сфері. Це забезпечується

їхньою здатністю до глобального імовірнісного пошуку, що базується на біологічній еволюції, а саме селекції та мутації [55, 1]. Поєднання науки та інженерії є головною вимогою для еволюційних алгоритми, що використовуються для вирішення задач комбінаторної оптимізації [66].

Основним представником даного класу є генетичний алгоритм. Він починається з деякого набору рішень, які називаються популяцією. Потім, алгоритм послідовно виконує наступні кроки доти, поки не буде досягнуто граничного числа ітерацій або поки не буде знайдено рішення, яке не може бути покращено:

- селекція. Із популяції вибираються деякі рішення, які є найбільш придатними;
- схрещування. Вибрані рішення схрещуються між собою з метою отримання нових рішень;
- мутація. Нові рішення мутують для створення ще більшої різноманітності;
- стиснення. Отриманні рішення додаються до популяції, в той же час, старі рішення видаляються.

Головна ідея даного методу полягає в тому, що нові рішення, які є більш придатними – мають більшу ймовірність бути відібраними для схрещування та мутації. Таким чином, алгоритм прагне створити популяцію рішень, які є все більш придатними.

Основними перевагами є можливість ефективного пошуку рішень в просторах великої розмірності та здатність уникнути потрапляння в локальний оптимум через випадковість та множинність самого простору. Недоліками є залежність від великої кількості параметрів, та пов’язана з цим обчислювальна складність.

Ще одним прикладом виступають міметичні алгоритми. Цей тип методів поєднує в собі ідеї генетичних алгоритмів і локального пошуку. Відмінності полягають у тому, що замість схрещування та мутації популяція рішень

оптимізується за рахунок локального пошуку. Основна ідея полягає у тому, що генетичні операції дозволяють алгоритму генерувати нові рішення, які є далеко від локальних оптимумів, а локальний пошук допомагає їх уникнути.

Основним недоліком при роботі з цим методом є потреба у пошуку ефективних методик локального пошуку та селекції.

Популярним на сьогоднішній день є використання алгоритмів штучної імунної системи. Даний клас алгоритмів використовує ідеї біологічної імунної системи для пошуку глобального оптимуму.

Принцип роботи алгоритму має наступний вигляд:

- створення популяції антитіл. Створюється початкова популяція антитіл, які представляють можливі рішення задачі оптимізації. Кожне антитіло має свою характеристику, що відображає його ефективність;
- визначення афінності. Афінність антитіл визначається на основі їхньої ефективності. Це може включати в себе вимірювання відстані між антитілами та цільовою функцією;
- відбір імунітету. Антитіла з високою афінністю вибираються для участі в наступних етапах, імітуючи відбір в природному імунітеті;
- клонування та мутація. Відбувається клонування вибраних антитіл, а потім їхні копії піддаються мутації. Цей процес дозволяє виробляти різноманітні версії антитіл для розширення простору пошуку;
- оцінка антитіл. Відбувається оцінка антитіл за новими характеристиками;
- регулювання імунного відгуку. На основі оцінок регулюється імунний відгук. Кращі антитіла можуть залишитися в популяції, а менш ефективні можуть бути вилучені.

Основною перевагою використання алгоритмів даного типу є їх здатність регулювати інтенсивність пошуку на основі якості поточних рішень.

Також, при вирішенні задач комбінаторної оптимізації за допомогою еволюційних алгоритмів використовують такий клас методів, як диференціальна еволюція. Ці методи є подібні до генетичного алгоритму, і використовують лише оператор генерації нової популяції та селекції. Основна ідея полягає в тому, що нове рішення створюється за допомогою випадкового процесу. У свою чергу, це дозволяє алгоритму генерувати нові рішення, які можуть бути кращі за існуючі.

Проте, обмеження в експертизі та ресурсах звичайних користувачів перешкоджає ефективному вирішенню задач даного типу за допомогою цих алгоритмів. На практиці, більшість користувачів, що мають справу з комбінаторною оптимізацією, володіє низькими теоретичними знаннями, та недостатніми навичками програмування для реалізації еволюційного алгоритму. В першу чергу, це пов’язано з тим, що алгоритми засновані на біологічній еволюції вимагають великої кількості ітераційних операцій для пошуку наближеного оптимального рішення, що в свою чергу – споживає значні обчислювальні ресурси. З точки зору користувачів, маючи комбінаторну оптимізаційну задачу, вони не можуть її ефективно розв’язати через обмеженість обчислювальних ресурсів.

Одним з перспективних рішень є те, що хмарний сервер надає користувачам еволюційну обчислювальну послугу. У свою чергу, сам сервер обладнаний достатньою кількістю сховища та великою обчислювальною потужністю, та може запропонувати гнучкий обчислювальний сервіс, такий як навчання та виведення моделей машинного навчання, що отримало назву «Машинне навчання як послуга» (MLaaS) [13, 51, 52, 54, 77].

### 1.4.5 Ройові алгоритми

В природі групи з тисяч, мільйонів або трильйонів окремих елементарних сутностей можуть самоорганізовуватися у різноманітні форми, щоб відповідати функціональним цілям виключно через локальні та звичайні взаємодії. Формально, термін «ройовий інтелект» був запропонований Бені та Ванг, як

залаштунки вивчення клітинних роботичних систем [21]. Фактично, ройові алгоритми володіють властивостями самоорганізації, паралельної роботи, розподілених операцій, гнучкості та стійкості.

Найвідомішим прикладом ройових алгоритмів є мурашина колонія. На практиці цей алгоритм симулює поведінку мурах, стосовно знаходження шляху до їжі чи уникнення ризиків. Його основна ідея полягає в тому, щоб імітувати залежність мурах від феромонів і керувати діями кожної мурахи, використовуючи позитивний відгук.

Одна з головних переваг даного алгоритму є здатність виявляти оптимальні шляхи в графах, проте негативною стороною є можливість передчасного збільшення феромонів на коротших шляхах, що призводить до недооцінки довших шляхів.

Іншим поширеним варіантом є метод оптимізації потоком частинок. Він ґрунтується на поведінці зграй птахів або риб. У даному алгоритму зграя частинок зберігає рух в обмеженому просторі параметрів, взаємодіючі одна з одною, оновлюючи свою швидкість та положення на основі власної інформації та інформації своїх сусідів для знаходження глобального оптимуму [10, 71].

Початковий стан алгоритму являє собою групу випадкових частинок, що складають рій та представляють випадковий простір розв’язків. При цьому, після завершення пошуку глобального розв’язку його можливо вважати найкращим [47, 74]. Основним недоліком даного алгоритму вважається можливість швидкої збіжності до недостатньо дослідженого регіону простору рішень.

Також, окремої уваги заслуговують алгоритми, що засновані на поведінці бджолиної колонії. Це сімейство алгоритмів бере свій початок у 2005 році, коли Карабога запропонував його використання для вирішення багатовимірної задачі оптимізації функції, зважаючи на інтелектуальну поведінку бджолиного рою під час пошуку їжі [12]. Бджоли утворюють групи комах, і хоча поведінка окремої комахи надзвичайно проста, група окремих особин представляє дуже складну поведінкову модель. Реальні бджолині популяції можуть збирати

нектар з джерел їжі (квітів) з високою ефективністю в будь-якому середовищі [11].

В оригінальному алгоритмі бджоли мають розподіл на три типи особин: зайняті бджоли, бджоли – спостерігачі та бджоли – розвідники. Кожна бджола відповідає певному джерелу їжі (вектору розв’язків), а саме поле джерела їжі ітеративно перебирається [4].

Безперечними перевагами даного алгоритму є можливість адаптуватися до змінних умов та різноманітних просторів рішень. Проте вони також мають свої недоліки, що визначаються в залежності від правильного вибору параметрів та можливості передчасного перебування бджіл у локальних оптимумах.

### 1.4.6 Методи сканування

Окремий клас методів вирішення задач комбінаторної оптимізації становлять алгоритми сканування. Дані методи використовують послідовний або паралельний перебір можливих рішень задачі. Прикладом такого методу є розсіяний пошук.

Розсіяний пошук – це метаевристичний алгоритм, робота якого заснована на сусідському Табу-пошуку [19]. Основна ідея полягає в тому, що алгоритм генерує нові рішення на основі розсіяних точок, які розташовані в різних областях простору рішень. У свою чергу, це дозволяє знаходити такі рішення, які не є близькими до початкових.

Варто відзначити, що хоча застосування розсіяного пошуку є ефективним для великих та складних задач, воно безпосередньо пов’язане з великими обчислювальними витратами, що не гарантують отримання глобального оптимуму.

### 1.4.7 Спеціальні методи

Спеціальними методами, ми можемо називати алгоритми, практичне застосування яких відноситься лише до обмеженого числа задач ( частіше за все

до однієї). Прикладом таких алгоритмів може бути евристика Ліна – Кернігана [57], та заснований на ній метод для вирішення задачі комівояжера [16, 35, 37, 50,].

Задача комівояжера (Traveling salesman problem) являє собою класичну задачу оптимізації, де основна мета полягає в тому, щоб знайти найкоротший можливий маршрут, який містить заданий набір міст і повертається до місця старту.

Один з свіжих підходів є застосування модифікації, а саме алгоритму Ліна – Кернігана – Хельсгауна [37, 16], як одного з найсучасніших алгоритмів локального пошуку для задачі комівояжера. Даний алгоритм оптимізує розв'язки шляхом використання метода $k - opt$ [58], який досліджує простір розв'язків замінюючи $k$ ребра графа поточного туру. Щоб зменшити область пошуку та підвищити ефективність даної модифікації, вона визначає ребра – кандидати так само, як алгоритм Ліна – Кернігана, накладаючи обмеження на те, що ребра, які додаються під час процесу $k - opt$ повинні бути вибрані з ребер кандидатів. Алгоритм Ліна – Кернігана – Хельсгауна асоціює множину кандидатів для кожного міста, де записано декілька інших міст – кандидатів. При чому, ребра кандидати складаються з ребер між кожним містом та містами – кандидатами [35].

### 1.5 Огляд застосування метаевристик

Досвід застосування локального пошуку разом з метаевристикою, до задач комбінаторної оптимізації, не є новим. Пов'язано це з тим, що локальний пошук з ітеративним покращенням може зупинитися на дуже низькій якості локальних мінімумів. Найпростішим рішенням для покращення цієї ситуації є перезапуск локального пошуку з новим, випадково або жадібно згенерованим розв'язком, доки не буде досягнуто певного критерію зупинки. Таким чином, виходячи з того, що зі збільшенням розмірності простору пошуку – кількість

локальних мінімумів може зростати експоненціальна, то на допомогу приходить метаевристика [60].

Також варто відмітити, що в останні роки метаевристичний підхід до вирішення задач комбінаторної оптимізації розвивається дуже стрімко і успішно застосовується до великого різноманіття складних та практично прикладних задач [70, 33].

Переходячи до конкретики, а саме задач, що були визначені вище, можливо відмітити вдалі дослідження на тему застосування локального пошуку. Так, для задачі про наплічник досліджувалося використання різних стратегій локального пошуку поєднаних з популяційними алгоритмами. Метод сходження на вершину [28], комбінований жадібний [62], розсіяний [59] та ітераційний пошук [59] вважаються досить ефективними для даної задачі.

Задача про покриття множини представлена низкою досліджень, де використовується адаптивний алгоритм чорної діри [9], мурашиної колонії та бджолиного рою [42, 49], динамічних штрафів [2], зважених рядів [24], ітераційного локального та Табу – пошуку [49, 39]. Варто відзначити, що задача про розбиття множини також може бути віднесена до цього класу, проте окремі дослідження також наявні [43].

Задача про мінімальне вершинне покриття також представлена низкою досліджень, що пов’язують локальний пошук та метаевристику. Такими виступають адаптивний жадібний пошук [45], стратегія зваженої перевірки конфігурації та вершин [44, 6], локальний пошук заснований на низькорівневих евристиках та їх покращені [5, 67].

Таким чином, провівши аналіз наявних прикладних методів вирішення задач комбінаторної оптимізації, успішним досвідом використання метаевристичного підходу для пошуку глобального оптимуму у типових проблемах є алгоритм штучної імунної системи. Вибір пояснюється наявністю у даного методу широкого спектру еволюційних операторів та здатністю регулювати інтенсивність пошуку на основі якості поточних рішень.

## 1.6 Висновки до розділу

Отже, виходячи з вищезгаданого, задачі комбінаторної оптимізації мають широкий спектр застосувань у реальному житті. Серед найбільш популярних задач, для подальшого вивчення та дослідження було обрано чотири, а саме:

- задача про наплічник;
- задача про покриття множини;
- задача про мінімальне вершинне покриття;
- задача про призначення.

Аналізуючи питання алгоритмічної складності було встановлено, що усі наведені задачі є $NP-$складними, що приводить нас до висновку про те, що відповіді можуть бути перевірені за поліноміальний час, хоча існуючі алгоритми працюють у часовій складності, обмеженій експоненціальною функцією від розміру даних.

Був проведений аналіз точних та наближених методів вирішення задач комбінаторної оптимізації, де були наведені принципи роботи кожної сім'ї наведених алгоритмів з вказанням їх сильних та слабких сторін. За результатом аналізу було встановлено, що для дослідження чотирьох різних класів задач найкращим рішенням буде розробка метаевристичного алгоритму на базі штучних імунних систем. Основними аргументами для цього є те, що дане сімейство алгоритмів володіє широким набором різних еволюційних операторів, що забезпечує не тільки зменшення шансів знаходження локального оптимуму замість глобального, а також здатність регулювати інтенсивність пошуку на основі якості поточних рішень.

Досліджуючи, наскільки актуальним є ідея побудови метаевристичного алгоритму за стратегією локального пошуку, була відмічена актуальність даної ідеї. Існує багато досліджень, пов'язаних з даними класами оптимізаційних проблем, для яких успішно застосовувалися стратегії локального пошуку. Проте, звертаючи окрему увагу на ряд популяційних метаевристик, була

помічена відсутність запропонованих рішень, які використовували статистичне оцінювання якості рішень поточної популяції.

Таким чином, у роботі була порушена актуальна науково-дослідницька тема можливості впливу статистичних оцінок, поточних рішень у популяційних алгоритмах, на покращення процесу пошуку рішення.

Також визначено, що *об'єктом дослідження* в даній кваліфікаційній роботі є використання евристичних та метаевристичних алгоритмів, для вирішення задач комбінаторної оптимізації.

Відповідно до вищезгаданого, *Предметом дослідження є* розробка універсального оператора локального пошуку, який міг би застосовуватися у евристичних та метаевристичних алгоритмах.

Враховуючи проведений аналіз, актуальною вважається наступна *мета* даної кваліфікаційної роботи, а саме: пошук та дослідження можливого впливу статистичних оцінок на вирішення комбінаторних задач у евристичних та метаевристичних алгоритмах.

Для побудови метаевристичного алгоритму, що буде використовувати стратегію локального пошуку на базі статистичних оцінок отриманих рішень поточної популяції, використовується дискретний варіант гібридного імунного алгоритму $HINO - SF$.

Тож, в ході дослідження, для досягнення його мети, необхідно вирішити наступні задачі;

- вибір еволюційних операторів, для подальшого їх застосування у обраному алгоритмі;
- формальний опис та впровадження оператора локального пошуку, який би надавав статистичну оцінку поточній популяції рішень, для швидшого наближення до глобального оптимуму;
- реалізація дискретного варіанту імунного алгоритму $HINO - SF$;
- початкове тестування розробленої моделі;

- дослідження впливу різних варіантів оператора селекції та кросинговеру на отриманий результат роботи;
- дослідження ефективності запропонованого оператора локального пошуку для вирішення обраних задач комбінаторної оптимізації.

# РОЗДІЛ 2 СПЕЦІАЛЬНИЙ

## 2.1 Метод моделювання штучних імунних систем

Природна імунна система має унікальну здатність виробляти нові типи антитіл та відбирати на найбільш придатні із них, для взаємодії з антигенами, які потрапили в організм. Методом спроб та помилок імунна система виробляє велику кількість антитіл проти великої множини невідомих антигенів. Важливою характеристикою процесу адаптації системи імунітету при її взаємодії з вірусами та бактеріями, які надходять із зовнішньої середовища, - є забезпечення різноманіття типів антитіл. При цьому, з математичної точки зору можемо стверджувати, що підтримання різноманіття в імунній системі - це задача оптимізації мультимодальної функції, яка має кратне (не одне) рішення, що є дуже важливим, під час практичного застосування.

Таким чином, можемо стверджувати, що метод, який заснований на моделюванні штучних імунних систем – це яскравий приклад метаевристики, який є прямою альтернативою звичного генетичного алгоритму. При цьому, сам імунний алгоритм імітує властивості природної імунної системи і заснований на принципах соматичної теорії [45] і мережевої гіпотези.

Соматична теорія стверджує, що збільшення різноманітності антитіл відбувається за рахунок соматичної рекомбінації і мутації генів. В рамках мережевої гіпотези обґрунтовується припущення, згідно з яким контроль розмноження клонів здійснюється в результаті взаємного розпізнавання антитіл, що функціонують як єдина мережа [45, 72].

Пропонований нижче імунний алгоритм заснований на принципах ідіотипічної мережі, яка описує регуляцію імунної системи антиідіотипічними антитілам (антитіл, які вибирають інші антитіла). Саме наявність такої мережі слугує збереженням імунної відповіді упродовж досить довгого часу,

підтримки клітин пам’яті та прискоренню утворення антитіл при вторинній імунній відповіді [63].

В процесі еволюції імунна система вищих ссавців набула здатності знищувати антигени за допомогою антитіл (рис. 2.1.).

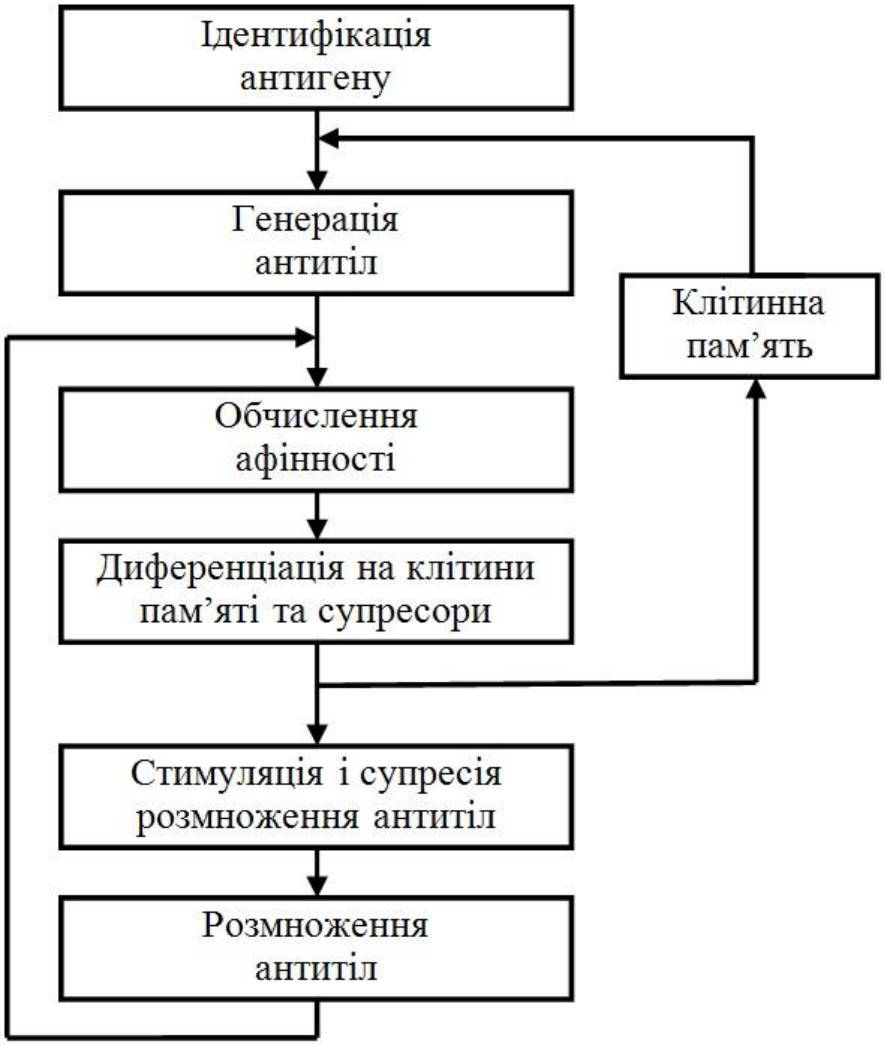


Рис. 2.1. Механізм формування антитіл в імунній системі

На рис. 2.1 – показана робота імунної системи з точки зору процесів обробки і управління інформацією. Користуючись цією схемою, інформаційні процеси в імунній системі можна представити у вигляді алгоритму, який вкладається у наступну послідовність формалізованих кроків:

Крок 1. Ідентифікація антигену. Потрапляння патогену до організму, та його перетворення на антиген – провокує імунну відповідь. У свою чергу цей процес відповідає встановленню виду задачі оптимізації.

Крок 2. Генерація антитіл. Клітини пам'яті активуються і виробляють антитіла для знищення даного антигену. Це відповідає створенню початкової популяції антитіл за допомогою випадкової генерації.

Крок 3. Обчислення афінності. На основі біохімічного розпізнавання поверхневої структури нового антигену відбувається відбір клітин, що продукують найбільш підходящі антитіла. На основі величини афінності відбираються такі антитіла, які найбільше відповідають антигену. Це відповідає пошуку оптимального рішення, як міри ступеню взаємозв’язку між даними (антигенами) і детекторами (антитілами).

Крок 4. Диференціація на клітини пам’яті та супресори. При великій концентрації деякі В-лімфоцити, антитіла яких відповідають антигену, стають клітинами пам'яті або супресорними клітинами відповідно до значення їх афінності. Це можна інтерпретувати як відбір потенційно кращих детекторів з метою пришвидшення знаходження деякого, близького до оптимального рішення.

Крок 5. Стимулювання і супресія розмноження клітин. Базуючись на результатах попереднього кроку, відбувається оцінка рівня та ступеню стимуляції клітини пам’яті. Тобто, якщо клітина розпізнала іншу клітину, але не сам антиген, то вона підлягає високому ступеню стимуляції. Таким чином, відбувається мутація клітини з метою розпізнавання інших клітин антигену. При цьому розмноження таких клітин стає мінімальне. Обернена ситуація відбувається, коли клітина розпізнала саме антиген. З практичної точки зору – це забезпечення різноманітності напрямків пошуку рішення.

Крок 6. Розмноження антитіл. Для відповіді на антигени, які раніше не зустрічалися і не схожі ні на одну з відомих рішень, в кістковому мозку виробляється утворення нових лімфоцитів замість антитіл, видалених через їх недостатню пристосованість.

У природі описаний процес є безперервним - антитіла виробляються організмом як до зустрічного антигену, так і до можливих його варіантів (мутацій). При моделюванні процесу на комп'ютері роботу алгоритму можна

зупиняти після певної кількості поколінь, через заданий час або після певної кількості поколінь, протягом яких пристосованість антитіл не підвищується.

Результатом роботи алгоритму буде сімейство рішень, що відповідають антитілам в клітинах пам'яті, максимально пристосованих до антигену. У випадку пошуку екстремумів мультимодальних функцій це сімейство відповідатиме масиву з векторів-рішень, афінність яких максимально наближена до глобального оптимуму, тоді як самі рішення попарно відмінні не менш як на задану граничну величину.

Показниками ефективності роботи алгоритму можуть бути час виконання та кількість звернень до цільової функції при відшуканні глобального оптимуму, а також надійність (повторюваність) отриманих рішень при повторних запусках. Останнє є доволі важливим, адже евристичний метод, що розглядається, відноситься до групи наближених з класу метаевристичних методів[79].

Історично першою реалізацією ідеї моделювання штучних імунних систем була ідея селекції та витіснення незалежних пошукових агентів, запропонована у алгоритмі CLONALG[3]. У наш час, виходячи з популярності застосування цього алгоритму, стали широко відомими його різні модифікації. Так одна з існуючих модифікацій, алгоритм AIRS, використовується для вирішення задачі класифікації [15]. Важливим є те, що алгоритми клональної селекції є дуже схожими на еволюційні алгоритми. Це зумовлено наявністю операторів клонування, селекції, мутації та стиснення.

Оператор стиснення популяції вилучає надлишковість популяції, виключаючи гірше з близьких рішень, що задовольняють умові

$$\|X_a - X_b\| < r, \quad a \neq b, \quad a, b \in \left[1 : N_p\right], \tag{2.1}$$

де $r$ – певний рівень близькості; $\|*\|$ – символ норми, яка розраховується в залежності від метрики простору; $N_p$ – розмір популяції.

До недоліків алгоритму CLONALG слід віднести відсутність адаптації, обмежену точність, обумовлену двійковим кодуванням та невикористання як такого механізму пам’яті.

Інший алгоритм моделювання імунних систем – алгоритм BCA [38] – фактично є версією еволюційної стратегії з двійковим кодуванням і оператором стиснення. Він використовує одразу два оператори мутації: у 3/4 випадків бітову, а у 1/4 випадків – суміжну (кілька бітів, що стоять поруч, інвертуються одночасно). Метод показує високу ефективність пошуку при невеликих розмірах популяції $N_p = 3 \div 5$, однак, як і CLONALG, на багатовимірних тестових задачах за якістю рішень програє відомим еволюційним стратегіям.

Певним кроком вперед є алгоритм opt-AiNet [72], головною особливістю якого є робота безпосередньо у дійсному просторі та збереження всіх знайдених локальних і глобальних оптимумів цільової функції у пам’яті, розмір якої динамічно змінюється. Алгоритм використовує також динамічний розмір популяції, керований механізмом стискання. Здійснюється мутація кожного клону пропорційне корисності його батьківської клітини за формулою

$$X_j^{C_i} = X_j^{C_i} + \alpha_j \cdot N(0;1), \quad \alpha_j = \rho e^{-\hat{\varphi}_j}, \quad j = \left[1{:}\, N_p\right], \tag{2.2}$$

де $X_j^{C_i}$ – $i$ −тий клон $j$ −тої клітини попередньої популяції; $\alpha_j$ – радіус мутації для клонів $j$ −тої клітини; $\hat{\varphi}_j$ – відносна афінність $j$ −тої клітини у поколінні $\hat{\varphi}_j = \frac{\varphi_j - \varphi_{max}}{\varphi_{min} - \varphi_{max}}$; $\rho$ – вільний позитивний параметр методу; $N(0;1)$ – нормально розподілена випадкова величина з математичним очікуванням 0 і середньоквадратичним відхиленням, що дорівнює 1.

Практична реалізація даного алгоритму на ряді функцій показала його значну перевагу над алгоритмом CLONALG у швидкості отримання порівняного за якістю рішення [7].

Також при вирішенні багатьох прикладних задач з використанням методу моделювання імунних систем застосовується алгоритм HIA [7, 45]. Його можна розглядати як модифікацію методу opt-AiNet. На відміну від останнього, розмір

популяції в алгоритмі HIA залишається постійним в процесі всього ітераційного процесу й підтримується оператором стискання (2.1). Крім того, кожна з клітин має додатковий атрибут $t$, званий віком клітини. За його допомогою визначаються так звані «елітні» клітини, що прожили $t \geq t_{max}$. Це – рішення, які тривалий час не були покращенні мутацією, тому вважаються локальними оптимумами. Вони заносяться в пам'ять й використовуються для генерації клонів у кожному новому поколінні.

Мутацію клонів здійснюють двома способами. Перший, відповідальний за інтенсифікацію пошуку в околі знайденого рішення, полягає у покоординатній зміні клона з використанням адаптивного нормального розподілу рівня мутації

$$X_j^{C_i} = X_j^{C_i} + N(0; \alpha_j), \tag{2.3}$$

де $\alpha_j(t)$ – СКВ нормального розподілу, що для кожної ітерації $t$ визначається залежністю

$$\alpha_j(t) = \begin{cases} 2\left(X_j(t-1) - X_j^{C_i}(t-1)\right), & if\ f\left(X_j^{C_i}(t-1)\right) < f\left(X_j(t-1)\right); \\ \alpha_j(t-1) \quad overwise \end{cases}. \tag{2.4}$$

Другий спосіб має на меті диверсифікувати напрямок пошуку за рахунок обстеження ще не розглянутих ділянок і полягає у зміні однієї координати $i$ випадковим числом з усієї області пошуку

$$x_{j,i} = U(x_{min,i}; x_{max,i}), \quad i = [1:n]. \tag{2.5}$$

Тут $n$ – розмірність простору, а $U(x_{min,i}; x_{max,i})$ – рівномірно розподілена випадкова величина в межах $D = \{x_{i,min} \dots x_{i,max}\}$.

Алгоритм HIA, у порівнянні з попередніми, демонструє надійність пошуку глобального оптимум багатоекстремальних функцій за доволі обмежений час.

Для побудови метаевристичного алгоритму, що працює за стратегією локального пошуку, для вирішення комбінаторних задач, алгоритм HINO-SF [79]. При цьому його адаптація буде називатися HINCO-SL (Hybrid Immune Network Combinatorial Optimization algorithm with Saaty selection and local search).

Даний, гібридний, метаевристичний алгоритм складається з наступних кроків:

1) Випадково генерується популяція антитіл. Лічильник поколінь встановлюється в $t = 0$;
2) Якщо досягнуто наперед заданий максимум кількості поколінь, здійснюється перехід до кроку 11; в іншому випадку – перехід до кроку 3;
3) За оцінкою пристосованості клітин поточного покоління призначається кількість клонів для кожної з клітин поточного покоління за допомогою оператору селекції;
4) Клонування клітин у кількості, що визначена оператором селекції;
5) Рекомбінація клонованих особин за допомогою модифікованого оператору ймовірнісного кросинговеру;
6) Розрахунок динамічної ймовірності мутації для клонів та виконання адаптивного оператора мутацій;
7) Визначення статистичних оцінок якості рішень поточної популяції та їх різноманіття. Якщо отримані значення задовольняють встановлені обмежені – відбувається процес локального пошуку. Інакше одразу йде перехід до наступного кроку.
8) Стиснення спільної популяції батьків та клонів за рахунок відповідного оператору до заданого рівня $N_p$;
9) Знищення з поточного покоління клітин, вік яких досяг заданого значення $t_{max}$. Їх місце в основній популяції займають клітини, згенеровані за випадково-рівномірним принципом в області пошуку;
10) Лічильник поколінь $t = t + 1$. Перехід на крок 2;
11) Виведення поточного покоління, як множини субоптимальних рішень;
12) Зупинка алгоритму.

Важливим кроком, до розуміння принципів роботи алгоритму є розгляд його головних еволюційних операторів, а саме:

- оператор селекції;
- оператор кросинговеру;
- оператор мутації;
- оператор локального пошуку
- оператор стиснення.

### 2.1.1. Оператор селекції

Першим розглядається оператор селекції. Для визначення кількості клонів, що припадає на кожну з особин поточного покоління, пропонується використовувати оператор, що застосовує метод аналізу ієрархій Сааті з відтинанням, вперше застосований у операторах клональної селекції еволюційних оптимізаційних стратегій [85]. Автор називає свій оператор $h$ −зрізом множини клітин поточного покоління, що описується рівнянням:

$$M_h = \left\{X_j \in D | \mu\left(f\left(X_j\right)\right) > h_{min}\{\}\left[1{:}\,N_p\right]\right\}, \tag{2.6}$$

де $h_{min} = \frac{1}{N_p}$ - критична пристосованість клітини; $D$ - область пошуку; $\mu(f(X_j))$ - функція належності клітини множині якісних рішень.

Функція належності клітини визначається наступним чином:

– рішення нормуються за значенням афінності на відрізок $[0; 1]$;

– розраховується рейтинг кожного з них $a_j = 1 + \lfloor \varphi\left(X_j\right) * k \rfloor$, де $k -$ розмірність шкали (за умовчанням $k = 9$);

– будується матриця попарних порівнянь, за допомогою наступного виразу:

$$b_{ij} = \begin{cases} \frac{a_1}{a_i}, & if\ i = 1; \\ \frac{b_{1j}}{b_{1i}}, & if\ i > 1; \end{cases} \tag{2.7}$$

– функції належності розраховуються як:

$$\mu\left(f\left(X_j\right)\right) = \frac{1}{\sum_j b_{ji}}; \tag{2.8}$$

– клітини, з $\mu\left(f\left(X_j\right)\right) > h_{min}$, клонуються у кількості, пропорційній функції належності

$$n_c = N_c * \mu(f(X_j)); \tag{2.9}$$

а, решта батьківського покоління відкидається.

Для ефективної роботи подальших операторів кросинговеру та мутації, бажано, аби особини з високою афінністю продукували достатньо велику кількість клонів.

Проте доцільним є внесення змін до запропонованого оператору селекції з метою покращення роботи алгоритму. Зміни полягають у іншому підході для побудови матриці попарних порівнянь та розрахунку функції належності.

Матриця попарних порівнянь будується за наступним принципом:

$$b_{ij} = \begin{cases} \left|a_i - a_j\right| + 1, & if\ j \geq i; \\ \dfrac{1}{b_{ji}}, & if\ j \leq i. \end{cases} \tag{2.10}$$

Розрахунок функції належності відбувається наступним чином:

$$\mu\left(f(X_i)\right) = \sqrt[n]{\prod_{j=1}^{n} b_{ji}}\,. \tag{2.11}$$

Варто зазначити, що окрім запропонованих варіантів існує також стандартні оператори селекції, що широко використовуються у різних популяційних алгоритмах, а саме:

- турнірний. Обирає найкраще рішення з певної групи. Він є універсальним оператором селекції, проте його якість роботи страждає у випадку наявності рішень з близькими оцінками;
- ранговий. В залежності від оцінки певного рішення – йому видається ранг. Надалі відбувається сортування за значенням рангу, і рішення з найбільшим рангом обирається. На відміну від турнірного оператора

є більш кращим при роботі за умови наявності близьких оцінок рішень;

– пропорційний. Аналогічно ранговому оператору, вибір рішення залежить від його оцінки. Проте, на базі оцінки визначається не ранг, а ймовірність, з якою те чи інше рішення може бути обране. Має такі самі недоліки як і турнірний оператор.

### 2.1.2. Оператор кросинговеру

Наступним за важливістю йде оператор кросинговеру. Іншою ключовою відмінністю запропонованого алгоритму є застосування до клонів адаптивного оператору кросинговеру, що передбачає обмін генетичною інформацією. Оскільки кодування інформації є двійковим, то для рекомбінації генів пропонується застосування модифікованого оператору кросинговеру [80].

Модифікований оператор кросинговеру передбачає лише один параметр, що задається користувачем:

$P_{cross}$ – ймовірність, що для клону буде виконано оператор кросинговеру.

Якщо випадкове число, рівномірно розподілене між 0 та 1 (тут і надалі позначаємо як $\mathrm{U}(0;1)$) менше ймовірності $P_{cross}$ для двох випадкових клонів з номерами $j$ та $k$, то поточна координата $i$ клонів піддається схрещуванню наступним чином:

- Спочатку визначаються два випадкових числа, що рівномірно розподілене між 2 та n-1 $Y(2, n-1)$ та $Z(2, n-1)$. Ці числа позначають собою так звані першу та другу точку поділу, які використовуються при кросинговері.
- Перше дочірнє антитіло формується таким чином, що її набір клітин пам’яті частково відображає набір другого батьківського антитіла. Важлива відмінність модифікованого оператора кросинговеру від звичайного двоточкового полягає у процесі відображення набору клітин

пам’яті від Y до Z (або навпаки). Тобто, якщо Y менше за Z, то від 1 до Y, набір клітин пам’яті відповідає набору першого батьківського антитіла, а від Z до останнього, набору другого. У той же час від Y до Z набір клітин відповідає результату порівняння, який був отриманий за допомогою логічного оператору **OR**, для цього набору клітин пам’яті батьківських особин. Варто зазначити, що якщо Y більше за Z, то порівняння відбувається за допомогою логічного оператору **AND**. Друге дочірнє антитіло формується аналогічним чином, проте вже з частковим відображенням першого антитіла. Візуалізація процесу кросинговеру наведена на рис. 2.2 – 2.3.

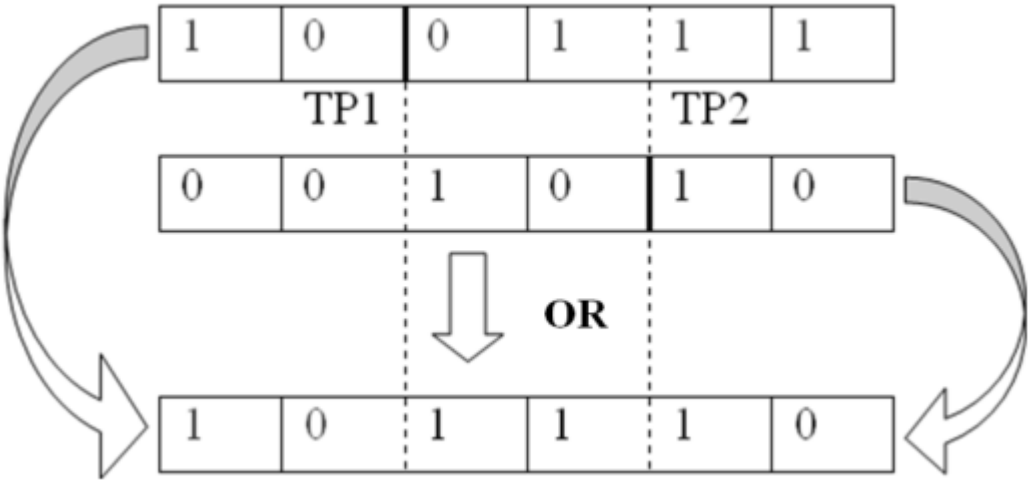


Рис. 2.2. Логічний оператор кросинговеру «OR»

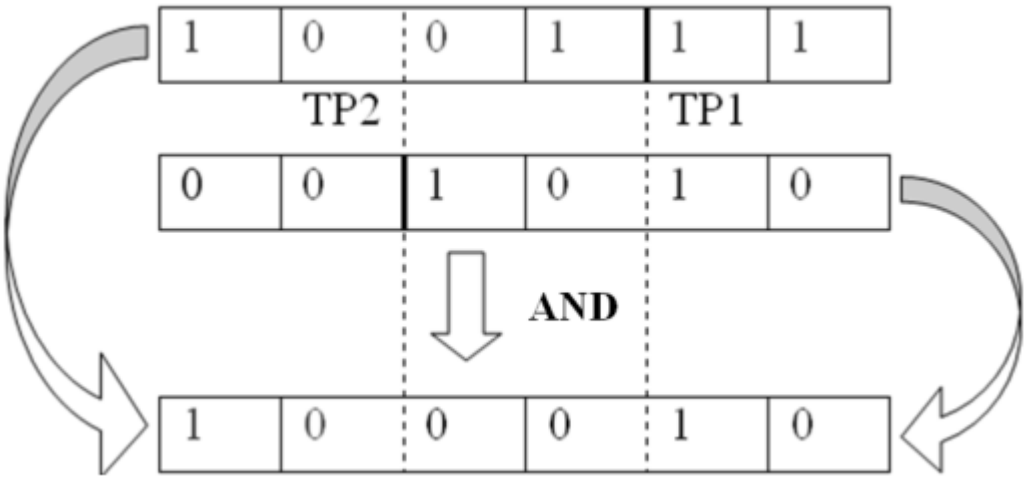


Рис. 2.3. Логічний оператор кросинговеру «AND»

Варто зазначити, що окрім запропонованих варіантів існує також стандартні оператори кросинговеру, що широко використовуються у різних популяційних алгоритмах, а саме:

- одноточковий. Передбачає обмін антитілами між двома батьківськими особами в одній точці розриву;
- двоточковий. Передбачає обмін антитілами між двома батьківськими особами в двох точках розриву;
- багатоточковий. Передбачає обмін антитілами між двома батьківськими особами в багатьох точках розриву;
- однорідний. Обмін антитілами між батьківськими особами відбувається випадковим чином;
- перетасувальний. Передбачає перестановку антитіл в батьківських особинах і їх подальший обмін;
- перетасувальний зі зменшеною кількістю перестановок. Основна відмінність від попереднього полягає у обмеженнях кількості перестановок в батьківських хромосомах.

### 2.1.3. Оператор мутації

Головним оператором, що відповідає за пошук рішення, в запропонованому алгоритмі, як і в більшості реалізацій методу штучних імунних систем, залишається оператор мутації. Пропонується адаптивний оператор, який випадковим чином застосовує інверсійну та сальтаційну мутацію.

Отримуючи на вході традиційні параметри $P_{mut}^{min}$ – мінімальний рівень мутації, $mut\_rate$ – відносне значення рівня мутації (частини генів, які попадуть під зміну оператором), $mutpoz$ – випадкове значення точки мутації антитіла (рівномірно розташовано між 1 та n, та визначається окремо для кожного покоління), та $T_{max}$ – максимальну припустиму кількість поколінь.

На кожній ітерації алгоритм розраховує ступінь мутації генів:

$$mut\_level = \begin{cases} P_{mut}^{min}\left(1 + mut\_rate - \dfrac{mut\_rate * 2t}{T_{max}}\right), \text{якщо } t < \dfrac{T_{max}}{2}; \\ P_{mut}^{min}, \text{якщо } t \geq \dfrac{T_{max}}{2}, \end{cases} \quad (2.12)$$

де $t$ – номер поточного покоління (ітерації).

Якщо при переборі координат випадкове число менше $mut\_level$, координата піддається інверсійній мутації, інакше – сальтаційній.

Під інверсійною мутацією розуміється, що від поточної клітини пам’яті до клітини яка обрана як мутаційна, відбувається перестановка цих самих клітин у зворотному порядку. Якщо мутаційна клітина, та поточна має одну й ту саму координату відбувається інверсія. Тобто замість 1 отримується 0, та навпаки.

Сальтаційна мутація заснована на ефекті сальтації, - стрибкоподібному переміщенні зерен матеріалу під дією несучого середовища. У даному випадку розуміємо цей процес, як стрибкоподібне переміщення клітин пам’яті у межах одного антитіла. Для цього спочатку випадково визначається певне число таких переміщень – saltation_num(1,(n-mutpoz+1)), яке рівномірно розподілене від 1 до різниці між розмірністю антитіл та позиції клітини мутації.

Після отриманого saltation_num відбувається переміщення за наступним принципом:

- Якщо поточна позиція клітини пам’яті відповідає мутаційній, відбувається інверсія.
- Якщо поточна позиція клітини пам’яті не відповідає мутаційній, то відбувається переміщення за наступним принципом:

$$\begin{aligned} & x_{ji}^{c} = x_{ji}, \\ & x_{ji} = x_{j\ mutpoz+k-1}, \\ & x_{j\ mutpoz+k-1} = x_{ji}^{c}, \\ & k = \overline{1, saltation\_num}. \end{aligned} \quad (2.13)$$

Вибір – яку саме мутацію виконувати: широку за (2.13) чи стислу у вигляді інверсії – здійснюється на основі синтетичної оцінки поточної клітини в

масштабі оцінок корисності на поточній ітерації по всій популяції. Остання визначається для кожної клітині індивідуально за формулою:

$$mut\_select\ _j = \frac{1}{2}\left(1 - \gamma\left(\frac{t}{T_{max}}\right)\right)\frac{\varphi_{max} - \varphi_j}{\varphi_{max} - \varphi_{min}}, j = [1{:}\,N_c], \quad (2.14)$$

де $\gamma$ – максимальна припустима доля сальтаційних мутацій в загальній кількості.

Якщо при зверненні до клону випадкове рівномірне число виявиться меншим за значення $mut\_select$, то для генів клона виконуватиметься лише мутація інверсійна, інакше – сальтаційна.

### 2.1.4. Оператор локального пошуку

Пропонується використання оператора локального пошуку для дискретних, метаевристичних, алгоритмів, що заснований на принципах статистики. Справа в тому, що зазвичай мається справа з порівняно маленькою популяцією, яка має велику кількість клонів. Це у свою чергу дає змогу отримати вибірку даних, які підлягають статистичному аналізу.

У даному випадку локальний пошук використовується для популяції клонів та складається з 8 кроків:

1. Для тих антитіл, наявність або відсутність яких була зафіксована раніше, відбувається повторна «фіксація».
2. Визначаються допустимі рамки статистичного відхилення від середнього значення кількості клонів.
3. Обчислюється генетичне різноманіття клонів. Виходячи з того, що маємо справу з дискретними величинами, які представлені як 0 та 1, визначається наявність кожного з генів.
4. Якщо кількість клонів з наявним геном відповідає допустимим рамкам статистичного відхилення – номер гену запам’ятовується.
    a. Якщо, на кроці 4 не було знайдені ні одного гену, що задовольняють умови – відбувається перехід до кроку 6.

b. Якщо, на кроці 4 було знайдено гени, то для них визначається середнє значення цільової функції, у випадку наявності та відсутності даного гену. За отриманими значеннями визначається та запам’ятовується, що саме наближає до отримання оптимального значення (мінімального чи максимального).

5. Визначений ген додається або забирається у цілого покоління за одним з наступних принципів:
   a. З усіх претендентів визначається той, що має найбільш оптимальне значення цільової функції. Саме його ознака і підлягає фіксації.
   b. Претенденти сортуються за абсолютним оптимальним значенням статистичного відхилення. Ознака першого підлягає фіксації.
   c. Усе відбувається аналогічно 5.b, з тією різницею, що фіксації підлягають не тільки перший претендент, а усі наступні, якщо значення його відхилення дорівнює 0.
6. Локальний пошук завершується.

Для більш детального розуміння принципу роботи даних варіантів локального пошуку, пропонується розглянути його роботу на прикладі популяції з 10 особин, кожна з яких, має 10 антитіл. Причому, будемо вважати, що ці антитіла (рішення) пройшли вище наведені оператори.

Таблиця 2.1

**Представлення популяції антитіл, що подані на вхід оператору локального пошуку**

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |
| №6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №8 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |

У таблиці 2.1 – перші 10 колонок відображають наявність чи відсутність певного антитіла, а остання містить значення цільової функції (антигену) для конкретної особи.

Перший крок роботи оператора локального пошуку передбачає фіксацію попередньо визначених антитіл, що дають оптимальний результат. При першій ітерації даний крок ігнорується, проте при усіх наступних є обов'язковим. Полягає він у тому, що успішні результати роботи на минулих ітераціях беруться до уваги у поточній.

Математично це можливо сформулювати наступним чином. Нехай, ми маємо множину U, яка являє собою набір номерів антитіл, які підлягають фіксації, та множину V, яка містить значення для фіксації цих антитіл. При чому, маємо популяцію рішень A, а її розмір дорівнює m. Тоді, для переходу на наступний крок має виконуватися умова:

$$A_{ij} = v_j \; \forall \, j = U, i = \overline{1,m} \qquad (2.15)$$

Другий крок, передбачає визначення меж статистичного відхилення. Для того, щоб проводити локальний пошук серед певної популяції, треба визначитися з параметрами, які будуть для нас важливими. Виходячи з того, що запропонований локальний пошук заснований на генетичному різноманітті, важливими параметром є кількісна міра наявності певного антитіла у всієї популяції.

Таким чином, для того, щоб провести локальний пошук, необхідно визначитися з верхнім та нижнім порогом наявності антитіла у цілої популяції. Це необхідно, так як антитіла, які знаходяться в даних межах підлягають подальшому дослідженню.

Перед тим, як кількісно визначати ці пороги, необхідно визначитися з рівнем допустимої, статистичної похибки $P$. Рекомендується брати значення в

10%, так як при визначені меж завжди отримується ціле число, яке не підлягає округленню, що впливає на якість локального пошуку.

Верхній ($U_b$) та нижній ($L_b$) поріг генетичного різноманіття визначається наступним чином:

$$U_b = \frac{m}{2} + mP \quad (2.16)$$

$$L_b = \frac{m}{2} - mP \quad (2.17)$$

Для прикладу популяції, що наведена у таблиці 2.1 дані межі будуть мати наступний вигляд:

$$U_b = \frac{10}{2} + 10 * 0{,}1 = 6$$

$$L_b = \frac{10}{2} - 10 * 0{,}1 = 4.$$

Третій крок, полягає у визначені генетичного різноманіття. Визначившись, з верхнім та нижнім порогом генетичного різноманіття, можливо переходити до наступного кроку. Генетичне різноманіття популяції полягає у статистичній оцінці наявності певного антитіла у всієї популяції. Тобто, дана оцінка може бути сформульована наступним чином:

$$\sum_{i=1}^{m} x_{jm} \ \forall j = U \quad (2.18)$$

У випадку, коли антитіло має оцінку, що задовольняє визначені межі – воно береться до подальшого розгляду. Для прикладу популяції, що наведена у таблиці 2.1 дані оцінки, разом з позначення обраних антитіл наведені нижче:

Таблиця 2.2

**Обчислення генетичного різноманіття**

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -10 | -11- | -12- |
| №1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |

Продовження таблиці 2.2

| -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -10 | -11- | -12- |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |
| №8 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |
| Значення генетичного різноманіття | 8 | 7 | 4 | 6 | 4 | 2 | 7 | 3 | 4 | 2 | - |

За результатом обчислень, для подальшого дослідження підходять лише 4 антитіла, так як їхнє генетичне різноманіття задовольняє визначені межі. Це антитіла №3, №4, №5 та №9.

У випадку, коли генетичне різноманіття жодного з антитіл не задовольняє встановлені межі – локальний пошук припиняється.

На четвертому кроці відбувається аналіз відібраних антитіл. А саме досліджується який вплив несе обране антитіло на цільову функцію (антиген). Для цього, визначається середнє значення цільової функції популяції (С), за умови наявності або відсутності даного антитіла.

$$One = \frac{\sum_{i=1}^{m} c_{jm}}{\sum_{i=1}^{m} x_{jm}} \; \forall \, x_j = 1 \qquad (2.19)$$

$$Zero = \frac{\sum_{i=1}^{m} c_{jm}}{m - \sum_{i=1}^{m} x_{jm}} \; \forall \, x_j = 0 \qquad (2.20)$$

Для прикладу популяції, що наведена у таблиці 2.2 дане дослідження виглядає наступним чином:

- Антитіло №3:
  - $One = \frac{2298+5680+3185+740}{4} = 2975.75;$
  - $Zero = \frac{687+8544+449+1802+5162+3274}{6} = 3319.67.$
- Антитіло №4:
  - $One = \frac{687+449+2298+3185+740+3274}{6} = 1772.17;$
  - $Zero = \frac{8544+5680+1802+5162}{4} = 5297.$

- Антитіло №5:
  - $One = \frac{687+1802+740+3274}{4} = 1625.75;$
  - $Zero = \frac{8544+449+2298+5680+3185+5162}{6} = 4219.67.$
- Антитіло №9:
  - $One = \frac{687+449+5680+740}{4} = 1889;$
  - $Zero = \frac{8544+2298+1802+3185+5162+3274}{6} = 4044.17.$

Виходячи з того, що розглядаються задачі мінімізації значення цільової функції, то для подальшого кроку обираються характеристики з найменшою оцінкою. Тобто, для усіх визначених антитіл у популяції, їх наявність є краще.

На п’ятому кроці відбувається повторна фіксація антитіла, в залежності від того, яка методика була обрана.

Даний крок є заключним, і він пов’язаний з фіксацією (додаванням або відніманням визначеного антитіла у поточній популяції. Надалі, для вже нової популяції визначаються актуальні значення цільової функції та відбувається її стиснення.

Важливо розуміти, що шляхи фіксації можуть бути різні, і є темою окремого дослідження, проте, наразі пропонуються три різні варіанти. Також, після кожної фіксації, номер антитіла та його оптимальний стан запам’ятовується.

Для більш кращого розуміння процесу фіксації, усі варіанти будуть описані, на прикладі популяції, що представлена у таблиці 2.1. Проміжні дані, за допомогою яких приймається рішення про фіксацію, наведені у таблиці 2.3.

Таблиця 2.3

**Проміжні результати роботи оператора локального пошуку**

| Номер антитіла | Оптимальний стан | Середнє значення цільової функції в оптимальному стані | Відхилення генетичного різноманіття від середнього значення |
|---|---|---|---|
| -1- | -2- | -3- | -4- |
| 3 | 1 | 2975.75 | 1 |
| 4 | 1 | 1772.17 | 1 |

Продовження таблиці 2.3

| -1- | -2- | -3- | -4- |
|---|---|---|---|
| 5 | 1 | 1625.75 | 1 |
| 9 | 1 | 1889 | 1 |

– Фіксація за генетичним різноманіттям:

При даній варіації фіксації, кандидати сортуються за значенням статистичного відхилення генетичного різноманіття. Перший з кандидатів обирається на фіксацію. Виходячи з того, що у прикладі, що розглядається, відхилення у всіх однакове, фіксації буде підлягати антитіло №3.

Таблиця 2.4

**Популяція до локального пошуку та після фіксації за генетичним різноманіттям**

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |
| №6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |
| №8 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |
| №6 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |
| №8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |

– Фіксація за значенням цільової функції:

При даній варіації фіксації антитіл, кандидати сортуються за середнім значенням цільової функції в оптимальному стані. Перший з кандидатів обирається на фіксацію. У прикладі, що розглядається, найменше середнє значення має антитіло №5. Таким чином фіксації буде підлягати антитіло №5.

Таблиця 2.5

**Популяція до локального пошуку та після фіксації за значенням цільової функції**

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |
| №6 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |
| №8 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |

| Антитіло / Особа | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 | Антиген |
|---|---|---|---|---|---|---|---|---|---|---|---|
| №1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 687 |
| №2 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 8544 |
| №3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 449 |
| №4 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2298 |
| №5 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 5680 |
| №6 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1802 |
| №7 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3185 |
| №8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 5162 |
| №9 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 740 |
| №10 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3274 |

– Фіксація за значенням цільової функції

При даній варіації фіксації антитіл, усі відбувається аналогічно попередньому варіанту. Різниця полягає у тому, фіксації підлягає не тільки перший кандидат, а й усі наступні, у випадку, коли статистичне відхилення від середнього значення для цих антитіл дорівнює 0. Виходячи з того, що у прикладі, який розглядається, ніхто з кандидатів не має нульового значення статистичного відхилення генетичного різноманіття – результат роботи локального пошуку ідентичний тому, що зображений у таблиці 2.5.

### 2.1.5. Оператор стиснення

Вагому роль, у пошуку рішення близького до оптимального грає оператор стиснення. По закінченні усіх пошукових операторів клітини нового покоління оцінюються за якістю. Разом з попереднім поколінням вони утворюють конфліктну множину, з якої, використовуючи принцип [45], відбирають $N_p$ клітин з найкращою афінністю, що лежать не ближче, ніж у радіусі $r$ одна від одної. Для цього виконуються наступні операції:

– сортування клонів за зменшенням афінності;

– циклічно від першої клітини до спустошення множини ще не розглянутих:

– вилучення усіх клітин, близьких до поточної і гірших за неї.

У даному випадку оператор стиснення використовує наступну відстань між точками:

$$dim^{threshold}, \tag{2.21}$$

де dim – розмірність простору пошуку, а $threshold$ – коефіцієнт стиснення, який приймає значення від 0 до 1.

## 2.2 Програмна реалізація і тестування гібридного, метаевристичного алгоритму HINCO-SL

Реалізація алгоритму гібридного, метаевристичного алгоритму HINCO-SL відбувалася у середовищі Visual Studio Code, за допомогою мови програмування Python. Варто відмітити, що з метою забезпечення найкращої швидкодії були використані такі модулі, як Pandas, Numpy та Numba.

У якості вихідних даних для кожної з задач комбінаторної оптимізації використовувався свій набір даних, а саме:

– задача про наплічник – knapPI_3_1000_1000_1 (розмірність простору 1000);
– задача про покриття множини – scpcyc07(розмірність простору 672);
– задача про мінімальне вершинне покриття – graph100-01 (розмірність простору 100);
– задача про призначення – sppnw41 ( розмірність простору 197).

Варто відмітити, що для тестування обиралися набори даних з найбільшою розмірністю простору. Даний вибір зумовлений бажанням збільшити вплив ймовірнісних складових на пошук рішення. Відповідні набори вихідних даних до кожної задачі, лістинг усього програмного коду, реалізованого алгоритму, тестових функцій та операторів наведені у додатках В і Г відповідно.

Саме тестування відбувається у автоматичному режимі за допомогою функції. Усі результати являють собою математичне очікування 10,

послідовних запусків алгоритму. Також варто зазначити, що оптимальні налаштування алгоритму знайдені на кожному з кроків – також використовуються і у наступних.

За допомогою тестування досліджується вплив, на пошук близького до оптимального, рішення наступних параметрів алгоритму:

– розмір популяції (5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20);
– кількість клонів (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200);
– ймовірність кросинговеру (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
– ймовірність мутації (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
– ймовірність координатного обміну (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
– доля мутаційних координат (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
– параметр вибору мутації (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
– коефіцієнт стиснення (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1).

Обладнання на якому відбувалося тестування:

– OS: Microsoft Windows 11 Pro 23H2 (22631.2792);
– CPU: AMD Ryzen 5 4600G with Radeon Graphics 3.70 GHz (6 ядер, 12 потоків);
– RAM: 8Gb DDR4 2133MHz;
– SSD: Goodram PX500 Gen.2 256GB M.2 (Система);
– HDD: TOSHIBA DT01ACA100 1TB (Інше).

Зупинка роботи роботи алгоритму відбувається за умови отримання рішення, яке дорівнює 0, якщо за 20 останніх ітерацій покращення поточного рішення не відбулося, кількість поколінь більша за 300, кількість викликів цільової функції більша за розмірність простору у 100 разів, або час становить більше 150 секунд.

Першим відбулося тестування роботи алгоритму для задачі про наплічник. Отримані результати зображені на рисунку 2.4 та таблиці 2.6. Варто зазначити, що виходячи з різного порядку цифри ключових оцінок роботи алгоритму, тут і

надалі усі результати представляють собою нормовану гістограму з накопиченням.

Таблиця 2.6

**Отримані результати тестування алгоритму HINCO-SL на задачі про наплічник**

| | К-сть ітерацій | Значення функції | Час | К-сть викликів функціїї |
|---|---|---|---|---|
| Без локального пошуку | 22 | 0,6 | 0,218 | 4420 |
| Локальний пошук 1 | 19 | 0 | 0,450 | 11619 |
| Локальний пошук 2 | 25,8 | 0,1 | 0,449 | 15836 |
| Локальний пошук 3 | 15,7 | 0,2 | 0,194 | 9088 |

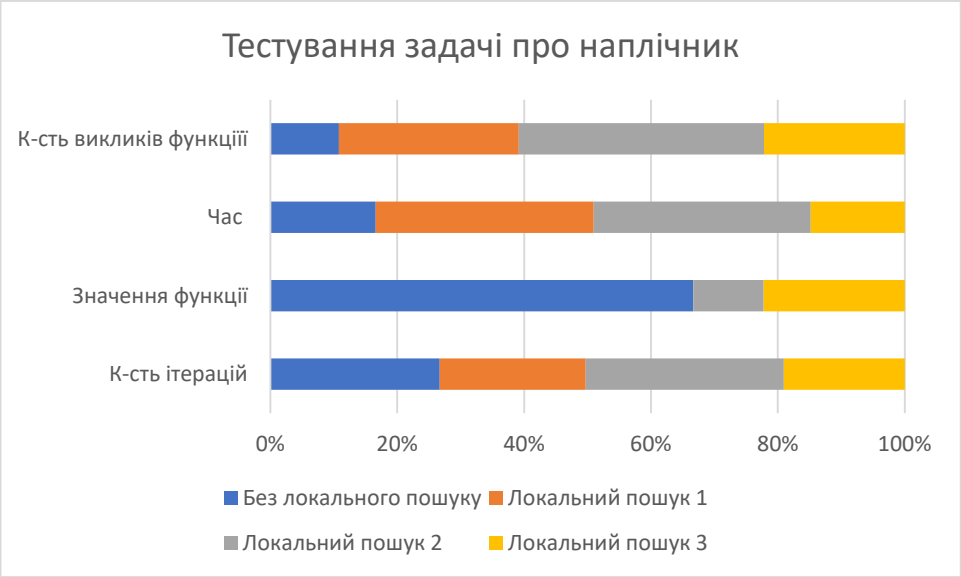


Рис. 2.4. Порівняльний аналіз результатів тестування алгоритму HINCO-SL

Аналізуючи отримані результати можна казати про наочний результат ефективності застосування стратегії статистичного локального пошуку для даного типу задач.

Локальний пошук за першим типом фіксації виконав оптимізацію чисто. Тобто, за 10 запусків оптимальне рішення було знайдене завжди. Проте негативною стороною є те, що час пошуку рішення є найдовшим.

При цьому, порівнюючи за кількістю ітерацій та витраченим часом – цікаві результати показує локальний пошук за третім варіантом фіксації. Йому

вдалося покращити середнє рішення без локального пошуку у 3 рази, швидше на 12 %, хоча кількість викликів цільової функції виросла у 2 рази.

У свою чергу, локальний пошук за другим варіантом показує гарні результати за значенням цільової функції, проте за іншими програє усім.

Якщо аналізувати результати з точки зору визначення основних операторів, що мали найбільший вплив, то маємо наступну ситуацію:

– без локального пошуку – мутація покращує значення цільової функції в середньому у 24% ітерацій;
– з локальним пошуком 1 – кросовер покращує рішення у 0,5% ітерацій, а мутація та локальний пошук у 26 та 12 відсотках відповідно;
– з локальним пошуком 2 – кросовер покращує рішення у 1,5% ітерацій, а мутація та локальний пошук у 26 та 14 відсотках відповідно;
– з локальним пошуком 3 – мутація та локальний пошук покращують рішення у 34 та 68 відсотках відповідно.

Оптимальні налаштування роботи алгоритму наведені у таблиці 2.7.

Таблиця 2.7

**Оптимальні налаштування роботи алгоритму HINCO-SL для вирішення задачі про наплічник**

| | Без локального пошуку | Локальний пошук 1 | Локальний пошук 2 | Локальний пошук 3 |
|---|---|---|---|---|
| -1- | -2- | -3- | -4- | -5- |
| Розмір популяції | 20 | 19 | 16 | 10 |
| Кількість клонів | 100 | 200 | 200 | 170 |

Продовження таблиці 2.7

| -1- | -2- | -3- | -4- | -5- |
|---|---|---|---|---|
| Ймовірність кросинговеру | 0,3 | 0,5 | 0,7 | 0,1 |
| Ймовірність мутації | 0,6 | 0,6 | 0,9 | 0,8 |
| Доля мутаційних координат | 0,8 | 1 | 0,7 | 0,2 |
| Параметр вибору мутації | 0,2 | 1 | 0,5 | 0,2 |

| | | | | |
|---|---|---|---|---|
| Коефіцієнт стиснення | 0,2 | 0,9 | 0,8 | 0,7 |

Наступним кроком відбувалося тестування запропонованого алгоритму для вирішення задачі про покриття множини. Отримані результати представлені на рисунку 2.5 та таблиці 2.8. У свою чергу, оптимальні налаштування наведені у таблиці 2.9

Таблиця 2.8

**Отримані результати тестування алгоритму HINCO-SL на задачі про покриття множини**

| | К-сть ітерацій | Значення функції | Час | К-сть викликів функціїї |
|---|---|---|---|---|
| Без локального пошуку | 36 | 335 | 17,861 | 7210 |
| Локальний пошук 1 | 24 | 323 | 8,416 | 9510 |
| Локальний пошук 2 | 26 | 335 | 17,247 | 10310 |
| Локальний пошук 3 | 25 | 330 | 15,977 | 9810 |

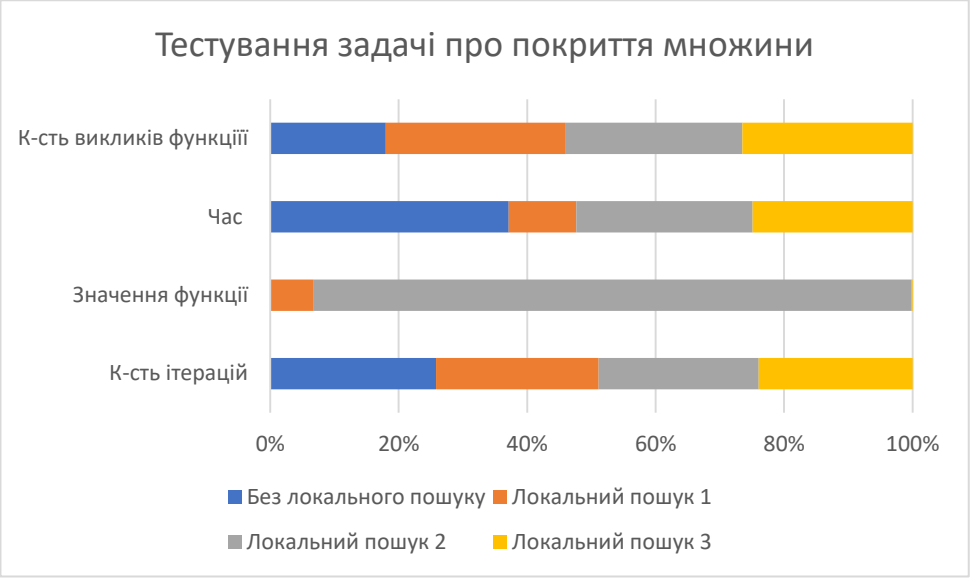


Рис. 2.5. Порівняльний аналіз результатів тестування алгоритму HINCO-SL

Отримані результати також підтверджують ефективність застосування даного алгоритму до даного класу проблем.

Локальний пошук за першим типом фіксації показав найкращі результати в розрізі таких оцінок, як загальна кількість ітерацій, значення цільової функції

та час. Серед усіх запропонованих варіантів локального пошуку даний варіант показує найнижчу кількість викликів цільової функції. При цьому, відносно алгоритму без застосування даного оператора кількість викликів збільшилася на 32% при зменшенні кількості ітерацій та часу у 1,5 – 2 рази.

Локальний пошук за другим варіантом фіксації хоч і показує покращення, але з огляду на кількість ітерації має дуже схожі результати, як у випадку відсутності. При цьому кількість викликів цільової функції зросла на 43%.

У свою чергу, локальний пошук за третім варіантом показує проміжні результати за усіма оцінками між першим та другим.

Якщо аналізувати результати з точки зору визначення основних операторів, що мали найбільший вплив, то маємо наступну ситуацію:

- без локального пошуку – відсутнє покращення рішень за рахунок оператора мутації та кросинговера ;
- з локальним пошуком 1 – покращення рішення відбувається за рахунок мутації та локального пошуку у 8 та 52 відсотках відповідно;
- з локальним пошуком 2 – покращення рішення відбувається за рахунок мутації та локального пошуку у 12 та 50 відсотках відповідно;
- з локальним пошуком 3 – мутація та локальний пошук покращують рішення у 32 та 50 відсотках відповідно.

Таблиця 2.9

**Оптимальні налаштування роботи алгоритму HINCO-SL для вирішення задачі про покриття множини**

| | Без локального пошуку | Локальний пошук 1 | Локальний пошук 2 | Локальний пошук 3 |
|---|---|---|---|---|
| Розмір популяції | 16 | 12 | 19 | 5 |
| Кількість клонів | 130 | 90 | 100 | 20 |
| Ймовірність кросинговеру | 0,1 | 0,5 | 0,2 | 0,4 |
| Ймовірність мутації | 0,6 | 0,6 | 0,3 | 0,7 |
| Доля мутаційних координат | 0,1 | 0,1 | 0,1 | 0,2 |
| Параметр вибору мутації | 0,9 | 0,3 | 0,5 | 0,3 |
| Коефіцієнт стиснення | 0,3 | 0,2 | 0,3 | 1 |

Далі було проведено тестування запропонованого алгоритму для вирішення задачі про мінімальне вершинне покриття . Отримані результати представлені на рисунку 2.6 та таблиці 2.10. У свою чергу, оптимальні налаштування наведені у таблиці 2.11

Таблиця 2.10

**Отримані результати тестування алгоритму HINCO-SL на задачі про мінімальне вершинне покриття**

| | К-сть ітерацій | Значення функції | Час | К-сть викликів функціїї |
|---|---|---|---|---|
| Без локального пошуку | 35 | 5499 | 15,26 | 7010 |
| Локальний пошук 1 | 38,5 | 5675 | 18,423 | 14610 |
| Локальний пошук 2 | 34,5 | 5773 | 19,83 | 13160 |
| Локальний пошук 3 | 33,5 | 5585 | 19,094 | 12260 |

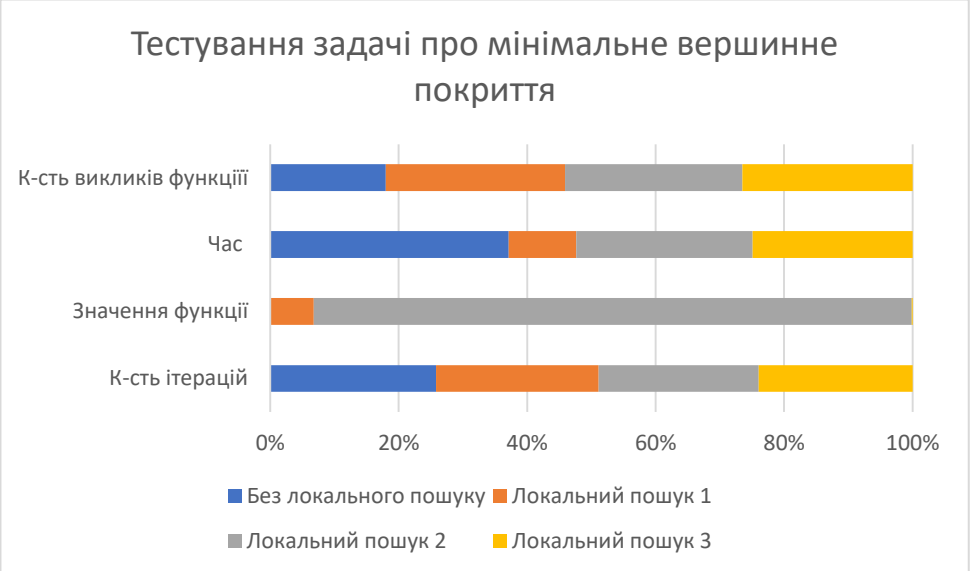

Рис. 2.6. Порівняльний аналіз результатів тестування алгоритму HINCO-SL

Аналіз результатів тестування на задачі про мінімальне вершинне покриття показав відсутність ефективності роботи запропонованих операторів локальних пошуків. Усі результати є близькими та гіршими за значення цільової функції до алгоритму, що не використовує локальний пошук.

Якщо аналізувати результати з точки зору визначення основних операторів, що мали найбільший вплив, то маємо наступну ситуацію:

– без локального пошуку – мутація покращила рішення у 14% ітерацій;
– з локальним пошуком 1 – покращення рішення відбувається за рахунок мутації та локального пошуку у 10 та 3 відсотках відповідно;
– з локальним пошуком 2 – покращення рішення відбувається за рахунок мутації та локального пошуку у 10 та 5 відсотках відповідно;
– з локальним пошуком 3 – мутація та локальний пошук покращують рішення у 13 та 8 відсотках відповідно.

Таблиця 2.11

**Оптимальні налаштування роботи алгоритму HINCO-SL для вирішення задачі про мінімальне вершинне покриття**

| | Без локального пошуку | Локальний пошук 1 | Локальний пошук 2 | Локальний пошук 3 |
|---|---|---|---|---|
| Розмір популяції | 14 | 16 | 11 | 9 |
| Кількість клонів | 120 | 110 | 20 | 200 |
| Ймовірність кросинговеру | 0,2 | 0,3 | 0,5 | 0,2 |
| Ймовірність мутації | 0,6 | 0,3 | 0,3 | 0,6 |
| Доля мутаційних координат | 0,1 | 0,5 | 0,5 | 0,1 |

| | | | | |
|---|---|---|---|---|
| Параметр вибору мутації | 0,3 | 0,2 | 0,6 | 0,8 |
| Коефіцієнт стиснення | 0,1 | 0,1 | 0,9 | 0,1 |

Останнім проведено тестування запропонованого алгоритму для вирішення задачі про призначення. Отримані результати представлені на рисунку 2.7 та таблиці 2.12. У свою чергу, оптимальні налаштування наведені у таблиці 2.13.

Таблиця 2.12

**Отримані результати тестування алгоритму HINCO-SL на задачі про призначення**

| | К-сть ітерацій | Значення функції | Час | К-сть викликів функціїї |
|---|---|---|---|---|
| Без локального пошуку | 28 | 38601 | 6,838 | 5610 |
| Локальний пошук 1 | 27,5 | 2268185 | 1,945 | 8710 |
| Локальний пошук 2 | 27 | 31639725 | 5,057 | 8610 |
| Локальний пошук 3 | 26 | 38601 | 4,589 | 8260 |

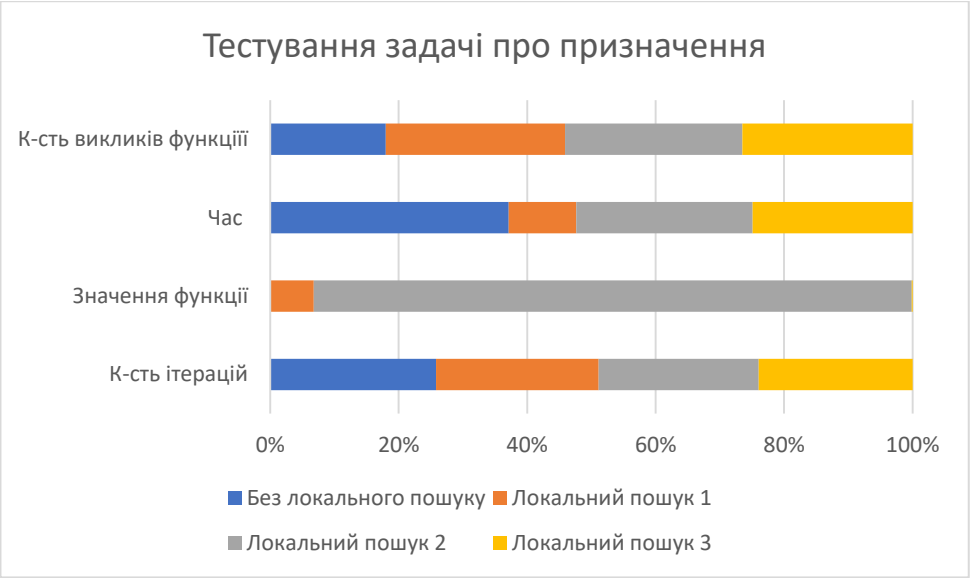


Рис. 2.7. Порівняльний аналіз результатів тестування алгоритму HINCO-SL

Аналіз результатів тестування на задачі про призначення показав відсутність ефективності роботи операторів локальних пошуку №1 та №2. Визначається це шляхом того, що отримане значення функції є великим за порядком до інших, що свідчить про роботу штрафної функції.

У той же час, локальний пошук при фіксації за 3 типом показує кращі результати в розрізі оцінки часу та кількості ітерацій. Загальне покращення становить 8 та 32 відсотка відповідно.

Якщо аналізувати результати з точки зору визначення основних операторів, що мали найбільший вплив, то маємо наступну ситуацію:

– без локального пошуку – мутація покращила рішення у 12,5% ітерацій;

– з локальним пошуком 1 – покращення рішення відбувається за рахунок мутації та локального пошуку у 43 та 4 відсотках відповідно;

– з локальним пошуком 2 – покращення рішення відбувається за рахунок мутації та локального пошуку у 41 та 4 відсотках відповідно;

– з локальним пошуком 3 – мутація та локальний пошук покращують рішення у 15 та 7 відсотках відповідно.

Таблиця 2.13

**Оптимальні налаштування роботи алгоритму HINCO-SL для вирішення задачі про призначення**

| | Без локального пошуку | Локальний пошук 1 | Локальний пошук 2 | Локальний пошук 3 |
|---|---|---|---|---|
| -1- | -2- | -3- | -4- | -5- |
| Розмір популяції | 18 | 19 | 13 | 14 |
| Кількість клонів | 190 | 120 | 150 | 190 |
| Ймовірність кросинговеру | 0,2 | 0,3 | 0,1 | 0,2 |
| Ймовірність мутації | 0,7 | 1 | 1 | 0,7 |
| Доля мутаційних координат | 0,7 | 1 | 1 | 1 |
| Параметр вибору мутації | 0,6 | 0 | 0,6 | 0,1 |
| Коефіцієнт стиснення | 0,9 | 0 | 0,8 | 0,4 |

Таким чином, були проведенні тестування роботи алгоритму HINCO-SL на усіх чотирьох задачах комбінаторної оптимізації. Запропоновані оператори локальних пошуків показали свою ефективність в усіх поставлених задачах, окрім задачі про мінімальне вершинне покриття

## 2.3 Висновки до розділу

Отримані у спеціальному розділі, результати дають змогу пересвідчитися, що ідея використання статистичних оцінок для вирішення задач комбінаторної оптимізації була обрана вірно, а вибір штучних імунних систем для побудови метаевристичного алгоритму зі стратегією локального пошуку не був помилковий.

Так, розглядаючи принцип роботи популяційних алгоритмів була помічена особливість, яка полягала у відсутності оцінок простору отриманих рішень, під час кожної ітерації. Так, зазвичай дослідження велося лише у напрямку вибору кращих, або гірших рішень. Прикладом є такі еволюційні оператори, як селекції та стиснення популяції. Проте, незважаючи на сукупну різноманітність

множини отриманих рішень на проміжних етапах пошуку (оператори кросинговеру, мутації) було відсутнє дослідження впливу конкретних ознак на остаточне рішення.

Таким чином, зважаючи на достатню статистичну вибірку під час проміжних етапів роботи популяційних алгоритмів був запропонований загальний вигляд оператора локального пошуку. Так, як він був заснований на отриманні статистичних оцінок якості отриманого рішення, було запропоновано три різні підходи до їх використання.

З практичної точки зору, було прийнято рішення про побудову гібридного, метаевристичного алгоритму HINCO-SL, що використовує стратегію локального пошуку, на базі штучних імунних систем. Цей вибір пов’язаний з намаганням зробити його універсальним, для вирішення різних задач комбінаторної оптимізації. Універсальність полягає у тому, що запропонований варіант алгоритму використовує велику низку еволюційних операторів та потребує мінімальну кількість евристичної інформації.

У ході дослідження було показано, що:

- запропоновані оператори локального пошуку показують різну ступінь своєї ефективності на 3 з 4 поставлених задач. Для задачі про мінімальне вершинне покриття ефективність застосованих операторів не була доведена;
- для вирішення задачі про наплічник найкраще підходить використання оператора локального пошуку за 1 типом фіксації. На другому місці за отриманими результатами є локальний пошук за 3 типом фіксації. При цьому, 2 тип фіксації показує гарну тенденцію сходження до глобального оптимуму, проте програє за усіма іншими показниками;
- локальний пошук за 1 типом також найкраще підходить для вирішення задачі про покриття множини. Варто зазначити, що для даної задачі усі варіанти оператора локального пошуку показали кращі результати, ніж алгоритм HINCO-SL, за їх відсутності. Головними критеріями є зменшення кількості ітерацій та часу пошуку рішення;

- запропонований оператор локального пошуку не є ефективним при вирішенні задачі про мінімальне вершинне покриття. Основна причина може полягати у тому, що на відміну від інших, дана задача відноситься до класу оптимізаційних проблем на графі, а також володіє найменшою розмірністю простору;
- при вирішенні задачі про призначення, локальний пошук за 3 типом фіксації показав свою ефективність в розрізі кількості ітерацій та часу на 8 та 32 відсотки відповідно. При цьому, інші варіанти не змогли досягти оптимального значення цільової функції, а тому їх неможливо порівнювати;
- спираючись на аналіз оптимальних параметрів алгоритму для кожної задачі, можемо стверджувати, що основне покращення отриманих результатів відбувається за рахунок мутації. Також, варто зазначити, що використання будь-якого оператора локального пошуку, майже завжди призводить до зменшення розміру популяції або кількості клонів разом зі збільшенням ймовірності мутації, долі мутаційних координати та коефіцієнту стиснення.

# ВИСНОВКИ

У даній роботі було дано вступ до проблематики використання локального пошуку, при вирішенні задач комбінаторної оптимізації. Були розглянути чотири типові проблеми комбінаторної оптимізації, а саме:

- задача про наплічник;
- задача про покриття множини;
- задача про мінімальне вершинне покриття;
- задача про призначення.

Після наданої постановки кожної, з вищезгаданих задач було розглянуто питання їх алгоритмічної складності та основні методи, що застосовуються для точної та наближеної оптимізації. Окремо увагу було приділено використання метаевристичних підходів з використанням стратегії локального пошуку.

Таким чином була сформульована наукова ідея стосовно запропонування такого оператора локального пошуку, який міг бути застосований до будь-яких задач комбінаторної оптимізації (за умови їх дискретності), зі збереженням своєї ефективності. При чому, ефективність роботи даного оператора забезпечується використанням статистичного апарату.

У спеціальному розділі було обґрунтовано вибір розробки метаевристичного алгоритму HINCO-SL на базі штучних імунних систем. Там же, під час опису основних операторів, була запропонована основна ідея використання статистичного локального пошуку у популяційних алгоритмах, та наведені три варіанти їх реалізації. Проведена надалі програмна реалізація та тестування гібридного, метаевристичного алгоритму HINCO-SL показала ефективність використання запропонованих операторів локального пошуку на трьох з чотирьох обраних задач комбінаторної оптимізації.

Таким чином, була отримана наукова новизна результатів дослідження, що полягала у новій, метаевристичній методиці використання оператора локального пошуку у популяційних алгоритмах.

Практичне значення результатів роботи полягає у тому, що наведена ефективність даного оператора на базових задачах дає можливість покращення процесу їх вирішення для прикладних задач.

Подальший розвиток об’єкту дослідження можливий у розширенні переліку базових функцій та проведення порівняльного аналізу його використання у метаевристичній стратегії в поєднанні з іншими алгоритмами, що були зазначені у розділі 1.4 .

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Radhakrishnan and G. Jeyakumar, “Evolutionary algorithm for solving combinatorial optimization—a review,” Innovations in Computer Science and Engineering, pp. 539–545, 2021.

2. Bilal, N., Galinier, P. and Guibault, F. (2013). A New Formulation of the Set Covering Problem for Metaheuristic Approaches. ISRN Operations Research, 2013, pp.1–10. doi:https://doi.org/10.1155/2013/203032.

3. Brownlee J., Clonal Selection Algorithms [Текст] / Technical Report 070209A, Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, 2007.

4. J. C. Bansal, H. Sharma, and S. S. Jadon, “ Artificial bee colony algorithm: A survey,” Int. J. Adv. Intell. Paradig., vol.5, no. 1–2, pp.123–159, 2013.

5. Cai, S. (2015). Balance between complexity and quality: local search for minimum vertex cover in massive graphs. International Conference on Artificial Intelligence, pp.747–753.

6. Cai, S., Su, K. and Sattar, A. (2011). Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence, 175(9-10), pp.1672–1696. doi:https://doi.org/10.1016/j.artint.2011.03.003.

7. Chen J., Q. Lin, Zh. Ji European Journal of Operational Research. 2010. Vol. 204. pp. 294-302.

8. Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001), Introduction to Algorithms (2nd ed.), MIT Press & McGraw–Hill, ISBN 0-262-03293-7 . pp. 344.

9. Sofo, R., Crawford, B., Olivares, R., Taramasco, C., Figueroa, I., Gómez, Á., Castro, C. and Paredes, F. (2018). Adaptive Black Hole Algorithm for Solving the Set Covering Problem. Mathematical Problems in Engineering, 2018, pp.1–23. doi:https://doi.org/10.1155/2018/2183214.

10. D. B. Fogel and H. G. Beyer, “A note on the empirical evaluation of intermediate recombination,” Evol. Comput., vol.3, no. 4, pp.491–495, Dec. 1995.

11. D. Karaboga and B. Akay, “ A comparative study of artificial bee colony algorithm,” Appl. Math. Comput., vol. 214, no. 1, pp. 108–132, Aug. 2009.

12. D. Karaboga, “ An idea based on honey bee swarm for numerical optimization,” Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.

13. D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, “CrypTFlow2: Practical 2-party secure inference,” in Proceedings of ACM Conference on Computer and Communications Security, 2020, pp. 325–342

14. Droste, S., Jansen, T. and Wegener, I. (2002). Optimization with randomized search heuristics—the (A)NFL theorem, realistic scenarios, and difficult functions. 287(1), pp.131–144. doi:https://doi.org/10.1016/s0304-3975(02)00094-4.

15. Dudek G. Artificial Immune System With Local Feature Selection for ShortTerm Load Forecasting, IEEE Trans. Evol. Computat., т. 21, вид. 1, сс. 116–130, лют. 2017, doi: 10.1109/TEVC.2016.2586049.

16. E. D. Taillard and K. Helsgaun, “POPMUSIC for the travelling ´ salesman problem,” European Journal of Operational Research, vol. 272, no. 2, pp. 420–429, 2019.

17. E.H.L. Aarts and J.K. Lenstra, editors. Local Search in Combinatorial Optimization. John Wiley & Sons, 1997.

18. E.H.L. Aarts. Local Search Algorithms. Invited Lecture at the Combinatorial Optimization 1998 conference, Brussels, Belgium, 1998.

19. F. Glover, “Tabu search—part I,” ORSA Journal on Computing, vol. 1, no. 3, pp. 190–206, 1989.

20. F. Glover, M. Laguna: Tabu Search (Kluwer, Boston 1997)

21. G. Beni and J. Wang, “Swarm intelligence in cellular robotic systems,” in Robots and Biological Systems: Towards a New Bionics?, P. Dario, G. Sandini, and P. Aebischer, Eds. Berlin, Heidelberg, Germany: Springer, 1993, pp. 703−712.

22. G.A. Croes. A Method for Solving Traveling Salesman Problems. Operations Research, 6:791–812, 1958.

23. G.L. Nemhauser and L.A. Wolsey. Integer and Combinatorial Optimization. John Wiley & Sons, New York, 1988.

24. Gao, C., Yao, X., Weise, T. and Li, J. (2015). An efficient local search heuristic with row weighting for the unicost set covering problem. European Journal of Operational Research, 246(3), pp.750–761. doi:https://doi.org/10.1016/j.ejor.2015.05.038.

25. H.H. Hoos, T. Stützle: Stochastic Local Search— Foundations and Applications (Morgan Kaufmann, San Francisco 2004)

26. Handbook of global optimization / ed. R. Horst, ed. P. Pardalos. — Springer US, 1995.

27. Haneen Algethami (2023). Local Search-Based Metaheuristic Methods for the Solid Waste Collection Problem. Applied Computational Intelligence and Soft Computing, 2023, pp.1–11. doi:https://doi.org/10.1155/2023/5398400.

28. Holte, R.C. (2001). Combinatorial Auctions, Knapsack Problems, and Hill-Climbing Search. In: Stroulia, E., Matwin, S. (eds) Advances in Artificial Intelligence. Canadian AI 2001. Lecture Notes in Computer Science(), vol 2056. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45153-6_6

29. Hoos, Holger & Stützle, Thomas. (2015). Stochastic Local Search Algorithms: An Overview. 1085-1105. 10.1007/978-3-662-43505-2_54.

30. Horst R. Introduction to global optimization / R. Horst, P. Pardalos, N. V. Thoai. — Springer US, 2000.

31. I.H. Osman and J.P. Kelly (Eds.). Meta-Heuristics: Theory & Applications. Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

32. J. Brest, M. S. Maučec, and B. Bošković, “Single objective real-parameter optimization: algorithm jSO,” in Proceedings of the 2017 IEEE congress on evolutionary computation (CEC), Donostia-San Sebastián, Spain, June 2017.

33. J. E. Bell and P. R. McMullen, “Ant colony optimization techniques for the vehicle routing problem,” Advanced Engineering Informatics, vol. 18, no. 1, pp. 41–48, 2004.

34. Tang, G. Liu and Q. Pan, "A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends,"

in IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 10, pp. 1627-1643, October 2021, doi: 10.1109/JAS.2021.1004129.

35. Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, Chu-Min Li. Reinforced Lin–Kernighan–Helsgaun algorithms for the traveling salesman problems. Knowledge-Based Systems, 2023, 260, pp.110144. ⟨10.1016/j.knosys.2022.110144⟩. ⟨hal-04124984⟩

36. K. Biswas, P. M. Vasant, J. A. G. Vintaned, and J. Watada, "A review of metaheuristic algorithms for optimizing 3D well-path designs," Archives of Computational Methods in Engineering, vol. 28, no. 3, pp. 1775–1793, 2021.

37. K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," European Journal of Operational Research, vol. 126, no. 1, pp. 106–130, 2000.

38. Kelsey J., Timmis J. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. 2003. URL: http://www.cs.york.ac.uk/rts/docs/GECCO_2003/papers/2723/27230207.pdf

39. Kinney, G.W., Barnes, J.W. and Colletti, B.W. (2007). A Reactive Tabu Search algorithm with variable clustering for the Unicost Set Covering Problem. International Journal of Operational Research, 2(2), p.156. doi:https://doi.org/10.1504/ijor.2007.012458.

40. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi: Optimization by simulated annealing, Science 220, 671–680 (1983)

41. Lagos, C., Guerrero, G., Cabrera, E., Niklander, S., Johnson, F., Paredes, F. and Vega, J. (2016). A Matheuristic Approach Combining Local Search and Mathematical Programming. Scientific Programming, 2016, pp.1–7. doi:https://doi.org/10.1155/2016/1506084.

42. Lanza-Gutierrez, J.M., Caballe, N.C., Crawford, B., Soto, R., Gomez-Pulido, J.A. and Paredes, F. (2020). Exploring Further Advantages in an Alternative Formulation for the Set Covering Problem. Mathematical Problems in Engineering, 2020, pp.1–24. doi:https://doi.org/10.1155/2020/5473501.

43. Lewis, M., Kochenberger, G.A. and Bahram Alidaee (2008). A new modeling and solution approach for the set-partitioning problem. Computers & Operations Research, 35(3), pp.807–813. doi:https://doi.org/10.1016/j.cor.2006.04.002.

44. Li, R., Hu, S., Zhang, H. and Yin, M. (2016). An efficient local search framework for the minimum weighted vertex cover problem. 372, pp.428–445. doi:https://doi.org/10.1016/j.ins.2016.08.053.

45. Lucinska M. Hybrid Immune Algorithm for Multimodal Function Optimization [Текст] / Lucinska M., Wierzchon S.T. // Recent Advances in Intelligent Information Systems, 2009, pp. 301-313.

46. Luedtke, Jim. “The Branch-and-Cut Algorithm for Solving Mixed-Integer Optimization Problems.” Institute for Mathematicians and Its Applications, 10 Aug. 2016, https://www.ima.umn.edu/materials/2015-2016/ND8.1-12.16/25397/Luedtke-mip-bnc-forms.pdf.

47. M. Clerc, Particle Swarm Optimization. Hoboken, USA: John Wiley & Sons, 2010.

48. Minakshi Kalra, Shobhit Tyagi, Vijay Kumar, Manjit Kaur, Wali Khan Mashwani, Habib Shah, Kamal Shah, "A Comprehensive Review on Scatter Search: Techniques, Applications, and Challenges", Mathematical Problems in Engineering, vol. 2021, Article ID 5588486, 21 pages, 2021. https://doi.org/10.1155/2021/5588486

49. Musliu, N. (2006). Local Search Algorithm for Unicost Set Covering Problem. In: Ali, M., Dapoigny, R. (eds) Advances in Applied Artificial Intelligence. IEA/AIE 2006. Lecture Notes in Computer Science(), vol 4031. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11779568_34

50. O. C. Martin, S. W. Otto, and E. W. Felten, “Large-step markov chains for the traveling salesman problem,” Complex Systems, vol. 5, no. 3, 1991.

51. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, “Delphi: A cryptographic inference service for neural networks,” in Proceedings of USENIX Security Symposium, 2020, pp. 2505–2522.

52. Mohassel and P. Rindal, “ABY3: A mixed protocol framework for machine learning,” in Proceedings of ACM SIGSAC conference on computer and communications security, 2018, pp. 35–52.

53. R. Elshaer and H. Awad, “A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants,” Computers & Industrial Engineering, vol. 140, p. 106242, 2020.

54. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in Proceedings of International Conference on Machine Learning. PMLR, 2016, pp. 201–210.

55. Wang and Z. Zhang, “Set theory-based operator design in evolutionary algorithms for solving knapsack problems,” IEEE Transactions on Evolutionary Computation, vol. 25, no. 6, pp. 1133–1147, 2021

56. S. Gao, C. Vairappan, Y. Wang, Q. Cao, and Z. Tang, “Gravitational search algorithm combined with chaos for unconstrained numerical optimization,” Applied Mathematics and Computation, vol. 231, pp. 48–62, 2014.

57. S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” Operations Research, vol. 21, no. 2, pp. 498–516, 1973.

58. S. Lin, “Computer solutions of the traveling salesman problem,” Bell System Technical Journal, vol. 44, no. 10, pp. 2245–2269, 1965.

59. Sapra, D., Sharma, R. and Agarwal, A. (2017). Comparative study of metaheuristic algorithms using Knapsack Problem. doi:https://doi.org/10.1109/confluence.2017.7943137.

60. Stützle, T.G. (1999). Local Search Algorithms for Combinatorial Problems.

61. T. Nguyen, “A high performance social spider optimization algorithm for optimal power flow solution with single objective optimization,” Energy, vol. 171, pp. 218–240, 2019.

62. Tian, X., Ouyang, D., Wang, Y., Zhou, H., Jiang, L. and Zhang, L. (2023). Combinatorial optimization and local search: A case study of the discount knapsack

problem. Computers & Electrical Engineering, 105, pp.108551–108551. doi:https://doi.org/10.1016/j.compeleceng.2022.108551.

63. Timmis Jon; Neal Mark; Hunt John. An artificial immune system for data analysis. Biosystems, 2000, 55.1-3: 143-150

64. Turky, A., Sabar, N.R., Dunstall, S. and Song, A. (2020). Hyper-heuristic local search for combinatorial optimisation problems. Knowledge-Based Systems, 205, p.106264. doi:https://doi.org/10.1016/j.knosys.2020.106264.

65. Cerný: A thermodynamical approach to the traveling salesman problem, J. Optim. Theory Appl. 45(1), 41–51 (1985)

66. T. Paschos, Applications of combinatorial optimization (2nd Edition). John Wiley & Sons, 2017.

67. Wagner, M., Friedrich, T. and Lindauer, M. (2017). Improving local search in a minimum vertex cover solver for classes of networks. doi:https://doi.org/10.1109/cec.2017.7969507.

68. Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1):67–82.

69. Luo, Q. Lv and P. Qian, "An Improved Iterated Local Search Algorithm for Invest Strongly Correlated 0/1 Knapsack Problem," 2009 International Forum on Computer Science-Technology and Applications, Chongqing, China, 2009, pp. 167-171, doi: 10.1109/IFCSTA.2009.162.

70. X.-S. Yang, “Optimization and metaheuristic algorithms in engineering,” Metaheuristics in Water, Geotechnical and Transport Engineering, pp. 1–23, 2013.

71. H. Shi and R. C. Eberhart, “Parameter selection in particle swarm optimization,” in Proc. 7th Int. Conf. Evolutionary Programming, San Diego, California, USA, 1998, pp. 591−600.

72. Yildiz A.R. A novel hybrid immune algorithm for global optimization in design and manufacturing [Текст] // Robotics and Computer-Integrated Manufacturing. – 2009. – vol. 25. – pp. 261-270.

73. Z. Cui, J. Zhang, Y. Wang et al., “A pigeon-inspired optimization algorithm for many-objective optimization problems,” Science China Information Sciences, vol. 62, no. 7, pp. 70212–70221, 2019.

74. Z. G. Ren, T. H. Chen, and Z. Z. Wu, “Optimal matching control of a low energy charged particle beam in particle accelerators,” IEEE/CAA J. Autom. Sinica, vol. 6, no.2, pp.460–470, Mar. 2019

75. Z.-M. Gu and G.-G. Wang, “Improving NSGA-III algorithms with information feedback models for large-scale many-objective optimization,” Future Generation Computer Systems, vol. 107, pp. 49–69, 2020.

76. Zhang, Y., Wu, J., Zhang, L., Zhao, P., Zhou, J. and Yin, M. (2018). An Efficient Heuristic Algorithm for Solving Connected Vertex Cover Problem. Mathematical Problems in Engineering, 2018, pp.1–10. doi:https://doi.org/10.1155/2018/3935804.

77. Zhao, Bowen, et al. "Evolution as a service: A privacy-preserving genetic algorithm for combinatorial optimization." arXiv preprint arXiv:2205.13948 (2022).

78. Zheldak T.Efficiency Improvement of the Algorithm Based on an Artificial Immune System Modeling Applied to Continuous and Combinatorial Problems / Zheldak, T., Ziborov, I., Lyman, V., Zhuk, A. / / CEUR Workshop Proceedings, (2021) – 3106, pp. 82–95.

79. Алгоритм моделювання штучної імунної системи з селективним оператором Сааті та одновимірним локальним пошуком / Т.А. Желдак, В.В. Слєсарєв // Искусственный интеллект. — 2013. — № 4. — С. 101–112. — Бібліогр.: 12 назв. — укр.

80. Дубровін В. І. Дослідження та розроблення генетичних алгоритмів та операторів схрещування / В. І. Дубровін, Є. М. Федорченко // Вісник Національного університету "Львівська політехніка". – 2010. – № 673 : Інформаційні системи та мережі. – С. 97-104.

81. Литвиненко, В. І. Побудова штучних імунних систем. Наукові праці [Чорноморського державного університету імені Петра Могили]. Сер.: Комп’ютерні технології, 2010, 134, Вип. 121: 166-178.

82. Михалевич В.С. Последовательные алгоритмы оптимизации и их применение/ В.С. Михалевич // Кибернетика.— 1965.— №1.— С. 45—55; №2.— С. 85—88.

83. Прикладні методи комбінаторної оптимізації : навч. посіб. / Л. Ф. Гуляницький, О. Ю. Мулеса. – К. : Видавничо-поліграфічний центр "Київський університет", 2016. – 133 с.

84. ПРОГРЕС – Академічний тлумачний словник української мови [Електронний ресурс] // Академічний тлумачний словник української мови. – Режим доступу: http://sum.in.ua/s/proghres (дата звернення: 10.12.2023). – Назва з екрана.

85. Снитюк В.Є. Спрямована оптимізація і особливості еволюційної генерації потенційних розв’язків [Текст] / В.Є. Снитюк // VI міжнародна школасемінар - "Теорія прийняття рішень", Ужгород, 1-6 жовтня 2012 р. – Ужгород: "Інвізор", 2012 – с. 182-183.

86. Тимофієва Н. К.; Гриценко В. І. Розв’язання задачі планування з теорії розкладів методом структурно-алфавітного пошуку та гібридним алгоритмом. Управляющие системы и машины, 2011, 3: 21-36.

# ДОДАТКИ

## Додаток А. Відомість матеріалів кваліфікаційної роботи

| № з/п | Позначення | Назва | Кількість | Примітки |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | Документація | | |
| 3 | | | | |
| 4 | САУ.КР.23.7.ПЗ | Пояснювальна записка | 80 | Формат А4 |
| 5 | | | | |
| 6 | | Демонстраційні матеріали | 20 | Презентація на CD-R |
| 7 | | | | |
| 8 | | Копія роботи | 1 | Диск CD-R |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

| | | | | | САіУ.КР.23.7.ДА.ПЗ. | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Змін. | Аркуш | № докум. | Підпис | Дата | | | | |
| Розроб. | | Жук А. В. | | | **Матеріали кваліфікаційної роботи** | Літ. | Аркуш | Аркушів |
| К. розд. | | Желдак Т.А. | | | | | | |
| Керівн. | | Желдак Т.А. | | | | НТУ «ДП» 12; 124м-22-1 | | |
| Н.контр. | | Хом'як Т.В. | | | | | | |
| Зав. каф. | | Желдак Т.А. | | | | | | |

**Додаток Б.**

Відгук на кваліфікаційну роботу магістра
студента групи 124м–22–1 спеціальності 124 Системний аналіз
Жука Андрія Віталійовича

Тема кваліфікаційної роботи: «Розробка статистичного оператора локального пошуку для евристичних та метаевристичних алгоритмів»

Обсяг кваліфікаційної роботи 80 стор.

Мета кваліфікаційної роботи: пошук та дослідження можливого впливу статистичних оцінок на вирішення комбінаторних задач у евристичних та метаевристичних алгоритмах.

Тема роботи є актуальною, оскільки дозволяє вирішити важливу науково-практичну проблему підвищення ефективності евристичних пошукових алгоритмів при розв’язанні алгоритмічно складних комбінаторних оптимізаційних задач. Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 124 Системний аналіз.

Виконані в кваліфікаційній роботі завдання відповідають вимогам ступеня магістра. Оригінальність наукових рішень полягає в розробці власного оператора локального пошуку, а також дослідження ефективності його застосування до ряду задач комбінаторної оптимізації.

Практичне значення результатів кваліфікаційної роботи полягає в тому, що запропонований оператор локального пошуку, застосований в тих чи інших пошукових евристичних алгоритмах, підвищує швидкість пошуку рішення (збіжність цих алгоритмів), що в свою чергу призводить до зменшення часу на прийняття рішень. Висновки підтверджують можливість використання результатів роботи при розв’язанні широкого кола прикладних задач.

Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами. Роботу виконано самостійно, відповідно до завдання та у повному обсязі. Суттєвих недоліків не зауважую.

Кваліфікаційна робота в цілому заслуговує оцінки: «відмінно» ( 100 балів).

З урахуванням викладеного автор заслуговує присвоєння освітньої кваліфікації «магістр з системного аналізу».

Керівник кваліфікаційної роботи,
К.т.н., доц., зав.кафедрою САУ _____ / Т.А. Желдак

# Додаток В. Набори даних

knapPI_3_1000_1000_1.csv

1000,4990
585,485
194,94
426,326
606,506
348,248
516,416
521,421
1092,992
422,322
749,649
895,795
337,237
143,43
557,457
945,845
915,815
1055,955
546,446
352,252
522,422
109,9
891,791
1001,901
459,359
222,122
767,667
194,94
698,598
838,738
107,7
674,574
644,544
815,715
434,334
982,882
866,766
467,367
1094,994
1084,984
993,893
399,299
733,633
533,433
231,131
782,682
528,428
172,72
800,700
974,874
717,617
238,138
974,874
956,856
820,720
245,145
519,419
1095,995
894,794
629,529
296,196
299,199
1097,997
377,277
216,116
197,97
1008,908
819,719
639,539
342,242
807,707
207,107
669,569
222,122
637,537
170,70
1031,931
198,98
826,726
700,600
587,487
745,645
872,772
367,267
613,513
1072,972
181,81
995,895
1043,943
313,213
158,58
848,748
403,303
587,487
864,764
1023,923
636,536
129,29
824,724
774,674
889,789
640,540
579,479
654,554
242,142
567,467
439,339
146,46
741,641
810,710
296,196
653,553
594,494
291,191
166,66
824,724
924,824
830,730
308,208
1088,988
811,711
190,90
900,800
440,340
414,314
649,549
389,289
296,196
501,401
965,865
566,466
778,678
789,689
670,570
933,833
1036,936
325,225
822,722
344,244
751,651
949,849
223,123
213,113
531,431
479,379
608,508
461,361
685,585
165,65
953,853
586,486
742,642
786,686
1092,992
386,286
825,725
989,889
386,286
124,24
912,812
591,491
959,859
991,891
763,663
190,90
188,88
281,181
279,179
314,214
287,187
117,17
719,619
572,472
361,261
518,418
946,846
519,419
292,192
456,356
361,261
782,682
614,514
406,306
986,886
301,201
630,530
485,385
949,849
1052,952
394,294
600,500
899,799
294,194
491,391
837,737
430,330
424,324
398,298
1092,992
890,790
324,224
375,275
360,260
926,826
197,97
172,72
310,210
966,866
749,649
1051,951
1019,919
848,748
163,63
785,685
1058,958
1056,956
904,804
664,564
618,518
283,183
528,428
500,400
637,537
821,721
446,346
307,207
253,153
423,323
1071,971
711,611
762,662
216,116
297,197
209,109
191,91
895,795
629,529
443,343
226,126
962,862
847,747
785,685
569,469

110,10
870,770
981,881
1034,934
1084,984
823,723
503,403
995,895
460,360
668,568
549,449
272,172
641,541
1058,958
372,272
483,383
977,877
408,308
459,359
1070,970
807,707
683,583
408,308
148,48
870,770
1030,930
130,30
669,569
308,208
103,3
411,311
120,20
200,100
709,609
1039,939
987,887
522,422
925,825
885,785
1030,930
470,370
1004,904
1089,989
341,241
1069,969
479,379
243,143
476,376
1072,972
1062,962
128,28
989,889
161,61
543,443
738,638
316,216
448,348
438,338
447,347
260,160
166,66
506,406
491,391
259,159
738,638
131,31
395,295
304,204
926,826
520,420
296,196
253,153
549,449
525,425
955,855
431,331
243,143
665,565
587,487
938,838
240,140
109,9
664,564
1018,918
715,615
633,533
235,135
332,232
664,564
1057,957
460,360
691,591
893,793
676,576
263,163
846,746
959,859
477,377
860,760
958,858
811,711
186,86
762,662
534,434
259,159
658,558
760,660
379,279
368,268
940,840
1048,948
835,735
415,315
674,574
776,676
226,126
441,341
1012,912
789,689
839,739
994,894
921,821
806,706
725,625
590,490
1017,917
578,478
301,201
771,671
1093,993
1032,932
249,149
999,899
152,52
337,237
859,759
287,187
367,267
572,472
356,256
872,772
883,783
198,98
217,117
1006,906
616,516
1011,911
280,180
735,635
125,25
325,225
480,380
923,823
812,712
264,164
366,266
443,343
316,216
832,732
548,448
602,502
641,541
840,740
764,664
676,576
1054,954
712,612
826,726
1002,902
872,772
554,454
631,531
511,411
1043,943
1073,973
850,750
803,703
427,327
950,850
1017,917
177,77
105,5
320,220
213,113
902,802
1013,913
503,403
891,791
281,181
1098,998
110,10
959,859
625,525
445,345
1019,919
531,431
768,668
775,675
627,527
933,833
562,462
538,438
391,291
623,523
705,605
1016,916
557,457
520,420
505,405
215,115
517,417
760,660
379,279
361,261
785,685
872,772
696,596
488,388
407,307
864,764
324,224
943,843
422,322
306,206
940,840
507,407
1075,975
739,639
501,401
952,852
191,91
642,542
427,327
160,60
430,330
857,757
282,182
182,82
703,603
737,637
893,793
193,93
715,615
714,614
833,733
236,136
964,864
287,187

116,16
202,102
963,863
1072,972
1087,987
263,163
406,306
601,501
134,34
577,477
940,840
592,492
800,700
221,121
806,706
180,80
887,787
238,138
205,105
760,660
934,834
329,229
898,798
927,827
410,310
548,448
709,609
330,230
790,690
932,832
661,561
589,489
679,579
686,586
160,60
391,291
488,388
765,665
409,309
760,660
507,407
802,702
300,200
253,153
413,313
706,606
1070,970
223,123
133,33
353,253
373,273
809,709
377,277
932,832
1097,997
208,108
140,40
988,888
327,227
581,481
960,860
383,283
1040,940
958,858
708,608
384,284
1090,990
811,711
690,590
950,850
906,806
339,239
152,52
1043,943
901,801
295,195
864,764
195,95
810,710
510,410
486,386
764,664
693,593
1000,900
150,50
212,112
594,494
1063,963
256,156
854,754
1036,936
558,458
1065,965
119,19
186,86
325,225
823,723
669,569
284,184
197,97
968,868
294,194
440,340
854,754
512,412
702,602
587,487
161,61
309,209
795,695
446,346
447,347
960,860
819,719
407,307
200,100
195,95
197,97
921,821
943,843
1041,941
845,745
921,821
386,286
687,587
592,492
301,201
216,116
765,665
631,531
627,527
524,424
912,812
353,253
524,424
526,426
559,459
513,413
443,343
1080,980
936,836
391,291
606,506
933,833
279,179
586,486
1016,916
109,9
530,430
675,575
696,596
1022,922
908,808
245,145
369,269
997,897
612,512
504,404
911,811
842,742
785,685
828,728
667,567
369,269
876,776
203,103
955,855
845,745
818,718
135,35
953,853
879,779
197,97
171,71
246,146
180,80
846,746
942,842
115,15
228,128
1065,965
1041,941
612,512
327,227
580,480
130,30
1058,958
442,342
765,665
705,605
291,191
631,531
1064,964
464,364
615,515
459,359
706,606
967,867
922,822
920,820
636,536
413,313
793,693
307,207
119,19
1011,911
435,335
408,308
201,101
530,430
1022,922
785,685
394,294
741,641
1010,910
213,113
510,410
241,141
350,250
790,690
646,546
800,700
829,729
659,559
709,609
581,481
820,720
603,503
1076,976
866,766
859,759
107,7
898,798
982,882
174,74
959,859
748,648
282,182
399,299
525,425
885,785
642,542
946,846
783,683
490,390
953,853
997,897
1038,938

516,416
189,89
937,837
724,624
347,247
281,181
393,293
978,878
244,144
1033,933
1038,938
680,580
567,467
352,252
580,480
307,207
1055,955
758,658
765,665
120,20
126,26
1054,954
664,564
447,347
549,449
859,759
126,26
546,446
621,521
114,14
401,301
757,657
421,321
810,710
713,613
1080,980
488,388
288,188
979,879
736,636
1090,990
303,203
288,188
532,432
1063,963
229,129
920,820
730,630
400,300
609,509
922,822
337,237
355,255
473,373
377,277
938,838
253,153
495,395
500,400
792,692
327,227
203,103
618,518
318,218
547,447
484,384
415,315
1031,931
913,813
496,396
424,324
489,389
602,502
740,640
399,299
387,287
880,780
397,297
476,376
1077,977
616,516
612,512
312,212
319,219
429,329
655,555
910,810
498,398
586,486
389,289
585,485
898,798
516,416
698,598
1052,952
149,49
994,894
672,572
458,358
797,697
121,21
870,770
747,647
825,725
144,44
642,542
783,683
314,214
461,361
987,887
914,814
145,45
331,231
194,94
108,8
382,282
265,165
182,82
596,496
419,319
799,699
726,626
952,852
246,146

559,459
641,541
635,535
974,874
414,314
825,725
1044,944
877,777
406,306
654,554
559,459
308,208
737,637
887,787
547,447
1076,976
508,408
1012,912
207,107
544,444
1062,962
103,3
665,565
401,301
507,407
264,164
957,857
537,437
417,317
752,652
198,98
864,764
918,818
141,41
700,600
837,737
388,288
276,176
1002,902
236,136
667,567
228,128
290,190
494,394
849,749
335,235
1008,908
142,42
376,276
924,824
848,748
784,684
692,592
1079,979
493,393
668,568
463,363
467,367
263,163
152,52
1083,983
742,642
279,179
435,335
116,16
226,126
578,478
230,130
359,259
683,583
968,868
229,129
395,295
852,752
696,596
267,167
171,71
956,856
722,622
740,640
964,864
870,770
744,644
531,431
105,5
295,195
465,365
548,448
529,429
801,701
164,64
1052,952
954,854
980,880
870,770
268,168
188,88
179,79
257,157
1061,961
468,368
572,472
521,421
317,217
819,719
809,709
371,271
247,147
1075,975
396,296
916,816
1009,909
1062,962
864,764
635,535
351,251
538,438
978,878
796,696
853,753
981,881
670,570
418,318
353,253

145,45
766,666
152,52
1031,931
596,496
533,433
413,313
910,810
238,138
778,678
434,334
223,123
1088,988
312,212

scpcyc07.csv
672,448,0,0,0
2,4,3,1
5,7,6,1
5,9,8,2
13,15,14,1
13,17,16,2
13,22,21,5
33,35,34,1
33,37,36,2
33,42,41,5
33,54,53,13
81,83,82,1
81,85,84,2
81,90,89,5
81,102,101,13
81,130,129,33
193,195,194,1
193,197,196,2
193,202,201,5
193,214,213,13
193,242,241,33
193,306,305,81
6,11,10,3
14,19,18,3
14,24,23,6
34,39,38,3
34,44,43,6
34,56,55,14
82,87,86,3
82,92,91,6
82,104,103,14
82,132,131,34
194,199,198,3
194,204,203,6
194,216,215,14
194,244,243,34
194,308,307,82
8,12,10,16,4
20,18,16,4
27,26,8,4
36,40,38,4
36,47,46,8
36,59,58,16
84,88,86,4
84,95,94,8
84,107,106,16
84,135,134,36
196,200,198,4
196,207,206,8
196,219,218,16
196,247,246,36
196,311,310,84
18,30,29,10
38,50,49,10
38,62,61,18
86,98,97,10
86,110,109,18
86,138,137,38
198,210,209,10
198,222,221,18
198,250,249,38
198,314,313,86
9,12,11,7
21,25,23,7
21,28,26,9
41,45,43,7
41,48,46,9
41,66,65,21
89,93,91,7
89,96,94,9
89,114,113,21
89,142,141,41
201,205,203,7
201,208,206,9
201,226,225,21
201,254,253,41
201,318,317,89
23,31,29,11
43,51,49,11
43,69,68,23
91,99,97,11
91,117,116,23
91,145,144,43
203,211,209,11
203,229,228,23
203,257,256,43
203,321,320,91
26,32,29,12
46,52,49,12
46,73,72,26
94,100,97,12
94,121,120,26
94,149,148,46
206,212,209,12
206,233,232,26
206,261,260,46
206,325,324,94
49,77,76,29
97,125,124,29
97,153,152,49
209,237,236,29
209,265,264,49
209,329,328,97
17,20,19,15
22,25,24,15
22,28,27,17
53,57,55,15
53,60,58,17
53,67,65,22
101,105,103,15
101,108,106,17
101,115,113,22
101,158,157,53
213,217,215,15
213,220,218,17
213,227,225,22
213,270,269,53
213,334,333,101
24,31,30,19
55,63,61,19
55,70,68,24
103,111,109,19
103,118,116,24
103,161,160,55
215,223,221,19
215,230,228,24
215,273,272,55
215,337,336,103
27,32,30,20
58,64,61,20
58,74,72,27
106,112,109,20
106,122,120,27
106,165,164,58
218,224,221,20
218,234,232,27
218,277,276,58
218,341,340,106
61,78,76,30
109,126,124,30
109,169,168,61
221,238,236,30
221,281,280,61
221,345,344,109
28,32,31,25
65,71,68,25
65,75,72,28
113,119,116,25
113,123,120,28
113,174,173,65
225,231,228,25
225,235,232,28
225,286,285,65
225,350,349,113
68,79,76,31
116,127,124,31
116,178,177,68
228,239,236,31
228,290,289,68
228,354,353,116
72,80,76,32
120,128,124,32
120,183,182,72
232,240,236,32
232,295,294,72
232,359,358,120
124,188,187,76
236,300,299,76
236,364,363,124
37,40,39,35
42,45,44,35
42,48,47,37
54,57,56,35
54,60,59,37
54,67,66,42
129,133,131,35
129,136,134,37
129,143,141,42
129,159,157,54
241,245,243,35
241,248,246,37
241,255,253,42
241,271,269,54
241,370,369,129
44,51,50,39
56,63,62,39
56,70,69,44
131,139,137,39
131,146,144,44
131,162,160,56
243,251,249,39
243,258,256,44
243,274,272,56
243,373,372,131
47,52,50,40
59,64,62,40
59,74,73,47
134,140,137,40
134,150,148,47
134,166,164,59
246,252,249,40
246,262,260,47
246,278,276,59
246,377,376,134
62,78,77,50
137,154,152,50
137,170,168,62
249,266,264,50
249,282,280,62
249,381,380,137
48,52,51,45
66,71,69,45
66,75,73,48
141,147,144,45
141,151,148,48
141,175,173,66
253,259,256,45
253,263,260,48
253,287,285,66
253,386,385,141
69,79,77,51
144,155,152,51
144,179,177,69
256,267,264,51
256,291,289,69
256,390,389,144
73,80,77,52
148,156,152,52
148,184,182,73
260,268,264,52
260,296,294,73
260,395,394,148

152,189,187,77
264,301,299,77
264,400,399,152
60,64,63,57
67,71,70,57
67,75,74,60
157,163,160,57
157,167,164,60
157,176,173,67
269,275,272,57
269,279,276,60
269,288,285,67
269,406,405,157
70,79,78,63
160,171,168,63
160,180,177,70
272,283,280,63
272,292,289,70
272,410,409,160
74,80,78,64
164,172,168,64
164,185,182,74
276,284,280,64
276,297,294,74
276,415,414,164
168,190,187,78
280,302,299,78
280,420,419,168
75,80,79,71
173,181,177,71
173,186,182,75
285,293,289,71
285,298,294,75
285,426,425,173
177,191,187,79
289,303,299,79
289,431,430,177
182,192,187,80
294,304,299,80
294,437,436,182
299,443,442,187
85,88,87,83
90,93,92,83
90,96,95,85
102,105,104,83
102,108,107,85
102,115,114,90
130,133,132,83
130,136,135,85
130,143,142,90
130,159,158,102
305,309,307,83
305,312,310,85
305,319,317,90
305,335,333,102
305,371,369,130
92,99,98,87
104,111,110,87
104,118,117,92
132,139,138,87
132,146,145,92
132,162,161,104
307,315,313,87
307,322,320,92
307,338,336,104
307,374,372,132
95,100,98,88
107,112,110,88
107,122,121,95
135,140,138,88
135,150,149,95
135,166,165,107
310,316,313,88
310,326,324,95
310,342,340,107
310,378,376,135
110,126,125,98
138,154,153,98
138,170,169,110
313,330,328,98
313,346,344,110
313,382,380,138
96,100,99,93
114,119,117,93
114,123,121,96
142,147,145,93
142,151,149,96
142,175,174,114
317,323,320,93
317,327,324,96
317,351,349,114
317,387,385,142
117,127,125,99
145,155,153,99
145,179,178,117
320,331,328,99
320,355,353,117
320,391,389,145
121,128,125,100
149,156,153,100
149,184,183,121
324,332,328,100
324,360,358,121
324,396,394,149
153,189,188,125
328,365,363,125
328,401,399,153
108,112,111,105
115,119,118,105
115,123,122,108
158,163,161,105
158,167,165,108
158,176,174,115
333,339,336,105
333,343,340,108
333,352,349,115
333,407,405,158
118,127,126,111
161,171,169,111
161,180,178,118
336,347,344,111
336,356,353,118
336,411,409,161
122,128,126,112
165,172,169,112
165,185,183,122
340,348,344,112
340,361,358,122
340,416,414,165
169,190,188,126
344,366,363,126
344,421,419,169
123,128,127,119
174,181,178,119
174,186,183,123
349,357,353,119
349,362,358,123
349,427,425,174
178,191,188,127
353,367,363,127
353,432,430,178
183,192,188,128
358,368,363,128
358,438,436,183
363,444,442,188
136,140,139,133
143,147,146,133
143,151,150,136
159,163,162,133
159,167,166,136
159,176,175,143
369,375,372,133
369,379,376,136
369,388,385,143
369,408,405,159
146,155,154,139
162,171,170,139
162,180,179,146
372,383,380,139
372,392,389,146
372,412,409,162
150,156,154,140
166,172,170,140
166,185,184,150
376,384,380,140
376,397,394,150
376,417,414,166
170,190,189,154
380,402,399,154
380,422,419,170
151,156,155,147
175,181,179,147
175,186,184,151
385,393,389,147
385,398,394,151
385,428,425,175
179,191,189,155
389,403,399,155
389,433,430,179
184,192,189,156
394,404,399,156
394,439,436,184
399,445,442,189
167,172,171,163
176,181,180,163
176,186,185,167
405,413,409,163
405,418,414,167
405,429,425,176
180,191,190,171
409,423,419,171
409,434,430,180
185,192,190,172
414,424,419,172
414,440,436,185
419,446,442,190
186,192,191,181
425,435,430,181
425,441,436,186
430,447,442,191
436,448,442,192
197,200,199,195
202,205,204,195
202,208,207,197
214,217,216,195
214,220,219,197
214,227,226,202
242,245,244,195
242,248,247,197
242,255,254,202
242,271,270,214
306,309,308,195
306,312,311,197
306,319,318,202
306,335,334,214
306,371,370,242
204,211,210,199
216,223,222,199
216,230,229,204
244,251,250,199
244,258,257,204
244,274,273,216
308,315,314,199
308,322,321,204
308,338,337,216
308,374,373,244
207,212,210,200
219,224,222,200
219,234,233,207
247,252,250,200
247,262,261,207
247,278,277,219
311,316,314,200
311,326,325,207
311,342,341,219
311,378,377,247
222,238,237,210
250,266,265,210
250,282,281,222
314,330,329,210
314,346,345,222
314,382,381,250
208,212,211,205
226,231,229,205
226,235,233,208
254,259,257,205
254,263,261,208
254,287,286,226

318,323,321,205
318,327,325,208
318,351,350,226
318,387,386,254
229,239,237,211
257,267,265,211
257,291,290,229
321,331,329,211
321,355,354,229
321,391,390,257
233,240,237,212
261,268,265,212
261,296,295,233
325,332,329,212
325,360,359,233
325,396,395,261
265,301,300,237
329,365,364,237
329,401,400,265
220,224,223,217
227,231,230,217
227,235,234,220
270,275,273,217
270,279,277,220
270,288,286,227
334,339,337,217
334,343,341,220
334,352,350,227
334,407,406,270
230,239,238,223
273,283,281,223
273,292,290,230
337,347,345,223
337,356,354,230
337,411,410,273
234,240,238,224
277,284,281,224
277,297,295,234
341,348,345,224
341,361,359,234
341,416,415,277
281,302,300,238
345,366,364,238
345,421,420,281
235,240,239,231
286,293,290,231
286,298,295,235
350,357,354,231
350,362,359,235
350,427,426,286
290,303,300,239
354,367,364,239
354,432,431,290
295,304,300,240
359,368,364,240
359,438,437,295
364,444,443,300
248,252,251,245
255,259,258,245
255,263,262,248
271,275,274,245
271,279,278,248
271,288,287,255
370,375,373,245
370,379,377,248
370,388,386,255
370,408,406,271
258,267,266,251
274,283,282,251
274,292,291,258
373,383,381,251
373,392,390,258
373,412,410,274
262,268,266,252
278,284,282,252
278,297,296,262
377,384,381,252
377,397,395,262
377,417,415,278
282,302,301,266
381,402,400,266
381,422,420,282
263,268,267,259
287,293,291,259
287,298,296,263
386,393,390,259
386,398,395,263
386,428,426,287
291,303,301,267
390,403,400,267
390,433,431,291
296,304,301,268
395,404,400,268
395,439,437,296
400,445,443,301
279,284,283,275
288,293,292,275
288,298,297,279
406,413,410,275
406,418,415,279
406,429,426,288
292,303,302,283
410,423,420,283
410,434,431,292
297,304,302,284
415,424,420,284
415,440,437,297
420,446,443,302
298,304,303,293
426,435,431,293
426,441,437,298
431,447,443,303
437,448,443,304
312,316,315,309
319,323,322,309
319,327,326,312
335,339,338,309
335,343,342,312
335,352,351,319
371,375,374,309
371,379,378,312
371,388,387,319
371,408,407,335
322,331,330,315
338,347,346,315
338,356,355,322
374,383,382,315
374,392,391,322
374,412,411,338
326,332,330,316
342,348,346,316
342,361,360,326
378,384,382,316
378,397,396,326
378,417,416,342
346,366,365,330
382,402,401,330
382,422,421,346
327,332,331,323
351,357,355,323
351,362,360,327
387,393,391,323
387,398,396,327
387,428,427,351
355,367,365,331
391,403,401,331
391,433,432,355
360,368,365,332
396,404,401,332
396,439,438,360
401,445,444,365
343,348,347,339
352,357,356,339
352,362,361,343
407,413,411,339
407,418,416,343
407,429,427,352
356,367,366,347
411,423,421,347
411,434,432,356
361,368,366,348
416,424,421,348
416,440,438,361
421,446,444,366
362,368,367,357
427,435,432,357
427,441,438,362
432,447,444,367
438,448,444,368
379,384,383,375
388,393,392,375
388,398,397,379
408,413,412,375
408,418,417,379
408,429,428,388
392,403,402,383
412,423,422,383
412,434,433,392
397,404,402,384
417,424,422,384
417,440,439,397
422,446,445,402
398,404,403,393
428,435,433,393
428,441,439,398
433,447,445,403
439,448,445,404
418,424,423,413
429,435,434,413
429,441,440,418
434,447,446,423
440,448,446,424
441,448,447,435

graph100-01.csv
100,3465
1,2
1,3
1,4
1,5
1,6
1,7
1,8
1,9
1,10
1,11
1,12
1,13
1,14
1,15
1,16
1,17
1,18
1,19
1,20
1,21
1,22
1,23
1,24
1,25
1,26
1,27
1,28
1,29
1,30
1,32
1,33
1,34
1,35
1,36
1,37
1,38
1,39
1,40
1,41
1,42
1,43
1,44
1,45
1,46
1,47
1,48
1,49
1,50
1,51
1,52
1,53
1,54

1,55
1,56
1,57
1,59
1,60
1,63
1,64
1,65
1,66
1,67
1,69
1,71
1,72
1,73
1,74
1,75
1,76
1,77
1,78
1,79
1,81
1,82
1,83
1,84
1,85
1,86
1,87
1,88
1,89
1,90
1,91
1,92
1,93
1,95
1,96
1,97
1,98
1,99
1,100
2,3
2,4
2,5
2,7
2,8
2,9
2,10
2,12
2,16
2,17
2,18
2,19
2,20
2,21
2,22
2,23
2,24
2,25
2,26
2,27
2,28
2,29
2,30
2,32
2,33
2,34
2,35
2,36
2,37
2,38
2,39
2,40
2,41
2,42
2,43
2,44
2,45
2,46
2,47
2,49
2,50
2,51
2,52
2,53
2,54
2,57
2,58
2,59
2,60
2,61
2,62
2,63
2,64
2,65
2,66
2,67
2,68
2,69
2,70
2,74
2,75
2,76
2,77
2,78
2,79
2,81
2,82
2,83
2,84
2,85
2,86
2,88
2,89
2,90
2,91
2,93
2,94
2,95
2,96
2,98
2,99
2,100
3,4
3,7
3,9
3,10
3,12
3,14
3,17
3,19
3,21
3,22
3,24
3,25
3,27
3,28
3,30
3,31
3,33
3,35
3,37
3,39
3,40
3,42
3,44
3,48
3,50
3,52
3,53
3,59
3,60
3,62
3,63
3,65
3,66
3,69
3,71
3,72
3,74
3,76
3,77
3,78
3,80
3,82
3,83
3,85
3,86
3,88
3,90
3,92
3,94
3,96
3,98
3,99
4,5
4,7
4,8
4,10
4,11
4,13
4,14
4,15
4,16
4,17
4,19
4,20
4,22
4,24
4,26
4,27
4,28
4,29
4,30
4,32
4,33
4,34
4,35
4,36
4,37
4,38
4,39
4,40
4,41
4,43
4,44
4,45
4,46
4,47
4,48
4,50
4,51
4,53
4,54
4,56
4,58
4,59
4,60
4,61
4,62
4,63
4,64
4,66
4,67
4,68
4,69
4,70
4,71
4,72
4,73
4,74
4,76
4,77
4,78
4,79
4,80
4,82
4,84
4,85
4,86
4,88
4,89
4,91
4,92
4,94
4,95
4,96
4,97
4,98
4,99

4,100
5,6
5,9
5,10
5,12
5,14
5,15
5,17
5,21
5,22
5,24
5,25
5,27
5,28
5,31
5,33
5,35
5,37
5,39
5,42
5,44
5,46
5,48
5,50
5,52
5,53
5,55
5,57
5,59
5,60
5,62
5,63
5,65
5,66
5,68
5,69
5,71
5,72
5,74
5,76
5,77
5,80
5,82
5,83
5,85
5,86
5,90
5,92
5,99
6,7
6,8
6,9
6,10
6,11
6,12
6,13
6,14
6,15
6,16
6,17
6,18
6,19
6,20
6,21
6,22
6,23
6,24
6,25
6,26
6,27
6,28
6,29
6,30
6,31
6,34
6,36
6,37
6,38
6,39
6,40
6,41
6,42
6,43
6,45
6,46
6,47
6,48
6,49
6,50
6,51
6,52
6,53
6,54
6,55
6,56
6,57
6,58
6,59
6,62
6,64
6,65
6,66
6,67
6,68
6,69
6,71
6,72
6,73
6,74
6,75
6,76
6,77
6,78
6,79
6,80
6,83
6,84
6,85
6,88
6,89
6,90
6,93
6,95
6,97
6,98
6,99
7,8
7,9
7,10
7,11
7,12
7,13
7,14
7,15
7,16
7,17
7,19
7,20
7,21
7,22
7,23
7,24
7,25
7,26
7,27
7,28
7,29
7,30
7,31
7,33
7,34
7,35
7,36
7,37
7,38
7,39
7,40
7,41
7,43
7,44
7,47
7,48
7,50
7,52
7,53
7,55
7,56
7,57
7,58
7,59
7,60
7,61
7,62
7,63
7,64
7,65
7,66
7,67
7,69
7,70
7,71
7,72
7,73
7,74
7,75
7,76
7,77
7,78
7,79
7,80
7,81
7,82
7,83
7,84
7,85
7,86
7,87
7,88
7,90
7,91
7,92
7,93
7,95
7,97
7,98
7,100
8,9
8,10
8,12
8,14
8,15
8,17
8,19
8,21
8,22
8,24
8,25
8,27
8,30
8,31
8,33
8,35
8,37
8,40
8,42
8,44
8,46
8,48
8,52
8,53
8,55
8,57
8,59
8,60
8,62
8,63
8,65
8,66
8,68
8,69
8,72
8,76
8,77
8,78
8,80
8,82
8,83
8,85

8,86
8,88
8,90
8,92
8,96
8,99
9,10
9,11
9,12
9,13
9,15
9,17
9,18
9,19
9,20
9,22
9,23
9,24
9,25
9,26
9,28
9,29
9,30
9,31
9,33
9,34
9,35
9,36
9,37
9,38
9,39
9,40
9,42
9,43
9,44
9,45
9,46
9,47
9,48
9,49
9,50
9,51
9,52
9,54
9,56
9,57
9,58
9,60
9,61
9,62
9,63
9,64
9,65
9,66
9,67
9,68
9,69
9,70
9,73
9,74
9,75
9,76
9,77
9,79
9,80
9,81
9,82
9,83
9,86
9,87
9,89
9,90
9,91
9,92
9,93
9,94
9,95
9,97
9,98
9,99
9,100
10,11
10,12
10,13
10,14
10,18
10,19
10,20
10,21
10,22
10,23
10,24
10,25
10,26
10,27
10,28
10,29
10,30
10,31
10,32
10,33
10,34
10,35
10,36
10,37
10,38
10,39
10,40
10,41
10,42
10,43
10,45
10,46
10,47
10,48
10,51
10,52
10,53
10,54
10,55
10,56
10,57
10,58
10,59
10,61
10,62
10,63
10,64
10,65
10,66
10,67
10,69
10,70
10,71
10,72
10,73
10,74
10,75
10,76
10,79
10,80
10,81
10,82
10,83
10,85
10,86
10,87
10,89
10,90
10,91
10,92
10,96
10,97
10,99
10,100
11,12
11,14
11,15
11,17
11,19
11,21
11,22
11,24
11,25
11,27
11,28
11,30
11,31
11,33
11,35
11,37
11,39
11,40
11,42
11,44
11,46
11,48
11,50
11,52
11,53
11,55
11,57
11,59
11,60
11,62
11,63
11,65
11,66
11,68
11,69
11,71
11,72
11,74
11,76
11,77
11,78
11,80
11,82
11,83
11,85
11,86
11,88
11,90
11,94
11,98
11,99
12,13
12,14
12,15
12,16
12,17
12,18
12,19
12,20
12,21
12,22
12,24
12,25
12,26
12,27
12,28
12,29
12,30
12,31
12,32
12,34
12,35
12,36
12,37
12,38
12,40
12,41
12,42
12,43
12,45
12,46
12,48
12,50
12,51
12,52
12,53
12,54
12,55
12,56
12,57
12,58
12,59
12,60

12,61
12,62
12,63
12,64
12,65
12,66
12,67
12,69
12,70
12,71
12,72
12,73
12,74
12,75
12,76
12,77
12,78
12,79
12,82
12,83
12,84
12,85
12,87
12,88
12,89
12,90
12,91
12,92
12,94
12,95
12,97
12,98
12,99
12,100
13,14
13,15
13,19
13,21
13,22
13,24
13,25
13,27
13,30
13,31
13,33
13,35
13,39
13,40
13,44
13,46
13,48
13,50
13,52
13,53
13,55
13,57
13,59
13,60
13,62
13,63
13,65
13,66
13,68
13,71
13,72
13,74
13,76
13,77
13,78
13,80
13,83
13,85
13,86
13,88
13,92
13,94
13,96
13,98
13,99
14,15
14,16
14,17
14,18
14,19
14,20
14,21
14,22
14,23
14,24
14,25
14,26
14,27
14,29
14,30
14,31
14,33
14,34
14,35
14,36
14,37
14,38
14,39
14,40
14,41
14,42
14,44
14,46
14,47
14,49
14,50
14,51
14,52
14,54
14,56
14,58
14,59
14,60
14,61
14,63
14,64
14,65
14,67
14,68
14,69
14,70
14,74
14,75
14,76
14,77
14,78
14,79
14,80
14,83
14,84
14,85
14,86
14,87
14,88
14,89
14,90
14,91
14,93
14,94
14,95
14,96
14,98
14,99
14,100
15,16
15,17
15,18
15,19
15,20
15,23
15,24
15,25
15,26
15,27
15,28
15,29
15,30
15,31
15,32
15,34
15,35
15,36
15,37
15,38
15,39
15,40
15,42
15,43
15,45
15,46
15,47
15,49
15,52
15,53
15,54
15,55
15,57
15,59
15,61
15,62
15,63
15,65
15,66
15,68
15,70
15,71
15,72
15,73
15,74
15,75
15,76
15,78
15,79
15,80
15,81
15,83
15,85
15,86
15,87
15,88
15,89
15,91
15,92
15,93
15,94
15,95
15,96
15,97
15,98
15,99
15,100
16,17
16,19
16,21
16,22
16,24
16,25
16,27
16,28
16,31
16,33
16,35
16,37
16,39
16,40
16,42
16,44
16,48
16,52
16,55
16,59
16,60
16,62
16,63
16,65
16,66
16,68
16,69
16,71
16,72
16,74
16,76
16,77
16,78

16,80
16,83
16,85
16,86
16,88
16,90
16,94
16,96
16,99
17,18
17,21
17,22
17,24
17,25
17,26
17,27
17,28
17,31
17,32
17,33
17,34
17,35
17,37
17,38
17,39
17,40
17,41
17,42
17,43
17,45
17,46
17,48
17,49
17,50
17,51
17,53
17,54
17,55
17,56
17,57
17,58
17,59
17,60
17,62
17,63
17,64
17,65
17,66
17,67
17,68
17,69
17,70
17,71
17,73
17,75
17,76
17,77
17,78
17,81
17,82
17,83
17,84
17,85
17,86
17,87
17,88
17,89
17,90
17,91
17,92
17,93
17,94
17,95
17,97
17,98
17,99
18,19
18,21
18,22
18,24
18,25
18,27
18,28
18,31
18,33
18,35
18,37
18,39
18,40
18,42
18,44
18,48
18,50
18,52
18,53
18,55
18,57
18,59
18,62
18,63
18,65
18,66
18,68
18,69
18,71
18,72
18,74
18,76
18,77
18,78
18,82
18,83
18,85
18,86
18,88
18,94
18,96
18,98
18,99
19,20
19,21
19,22
19,25
19,26
19,27
19,28
19,29
19,30
19,31
19,32
19,33
19,35
19,36
19,37
19,39
19,40
19,41
19,42
19,43
19,44
19,45
19,46
19,47
19,48
19,49
19,50
19,51
19,52
19,53
19,54
19,55
19,56
19,57
19,58
19,59
19,61
19,62
19,63
19,64
19,65
19,66
19,67
19,68
19,69
19,70
19,71
19,72
19,73
19,74
19,75
19,76
19,77
19,78
19,80
19,81
19,82
19,83
19,84
19,85
19,86
19,88
19,89
19,90
19,91
19,94
19,95
19,97
19,99
19,100
20,24
20,25
20,28
20,30
20,31
20,33
20,35
20,37
20,39
20,40
20,42
20,44
20,46
20,48
20,50
20,52
20,53
20,55
20,57
20,60
20,62
20,63
20,65
20,66
20,68
20,69
20,71
20,72
20,74
20,77
20,78
20,82
20,83
20,85
20,86
20,88
20,90
20,92
20,94
20,96
20,98
20,99
21,22
21,23
21,24
21,25
21,26
21,27
21,28
21,29
21,30
21,31
21,32
21,33
21,34
21,36
21,37
21,38
21,39

21,40
21,41
21,42
21,44
21,45
21,47
21,48
21,49
21,50
21,52
21,54
21,55
21,57
21,59
21,60
21,61
21,62
21,63
21,64
21,65
21,66
21,68
21,69
21,70
21,71
21,72
21,73
21,74
21,75
21,76
21,77
21,78
21,79
21,80
21,81
21,82
21,84
21,85
21,86
21,87
21,88
21,89
21,90
21,91
21,92
21,93
21,95
21,96
21,97
21,98
21,99
21,100
22,23
22,24
22,25
22,26
22,28
22,29
22,30
22,31
22,32
22,33
22,34
22,35
22,36
22,37
22,38
22,40
22,42
22,43
22,44
22,45
22,46
22,49
22,50
22,51
22,52
22,53
22,54
22,55
22,56
22,57
22,58
22,59
22,60
22,61
22,62
22,63
22,64
22,65
22,66
22,67
22,68
22,69
22,70
22,72
22,73
22,74
22,76
22,77
22,78
22,79
22,80
22,81
22,83
22,84
22,85
22,86
22,87
22,88
22,89
22,90
22,91
22,92
22,93
22,94
22,95
22,96
22,97
22,98
22,99
23,24
23,27
23,28
23,30
23,31
23,33
23,35
23,37
23,39
23,40
23,44
23,48
23,52
23,53
23,57
23,59
23,60
23,62
23,65
23,66
23,68
23,69
23,71
23,72
23,74
23,76
23,77
23,78
23,82
23,83
23,88
23,90
23,92
23,94
23,96
23,99
24,25
24,26
24,27
24,29
24,30
24,31
24,33
24,34
24,36
24,37
24,38
24,39
24,40
24,41
24,42
24,43
24,44
24,45
24,47
24,48
24,49
24,50
24,51
24,52
24,54
24,55
24,56
24,57
24,58
24,59
24,61
24,63
24,64
24,65
24,66
24,67
24,68
24,69
24,70
24,72
24,73
24,74
24,75
24,76
24,77
24,78
24,79
24,81
24,82
24,83
24,86
24,87
24,88
24,89
24,90
24,91
24,93
24,95
24,96
24,98
24,99
24,100
25,26
25,28
25,29
25,30
25,31
25,32
25,33
25,35
25,36
25,38
25,39
25,40
25,42
25,43
25,44
25,45
25,46
25,47
25,48
25,49
25,50
25,51
25,52
25,53
25,54
25,55
25,56
25,57
25,59

25,60
25,62
25,63
25,64
25,65
25,67
25,68
25,69
25,70
25,72
25,74
25,75
25,76
25,78
25,80
25,81
25,82
25,83
25,84
25,85
25,86
25,87
25,88
25,89
25,90
25,91
25,93
25,94
25,95
25,96
25,97
25,98
25,99
25,100
26,28
26,33
26,35
26,37
26,39
26,40
26,42
26,44
26,46
26,48
26,50
26,53
26,57
26,62
26,63
26,66
26,68
26,69
26,71
26,72
26,74
26,76
26,78
26,80
26,83
26,85
26,86
26,88
26,90
26,92
26,96
26,98
26,99
27,28
27,29
27,31
27,33
27,34
27,35
27,36
27,39
27,40
27,42
27,45
27,46
27,47
27,48
27,49
27,51
27,52
27,53
27,54
27,55
27,56
27,57
27,58
27,59
27,61
27,62
27,63
27,65
27,66
27,68
27,69
27,70
27,71
27,73
27,74
27,75
27,76
27,77
27,79
27,80
27,81
27,82
27,83
27,84
27,86
27,87
27,88
27,90
27,91
27,92
27,93
27,94
27,95
27,96
27,97
27,99
27,100

28,29
28,30
28,31
28,32
28,33
28,36
28,37
28,38
28,39
28,41
28,42
28,43
28,44
28,45
28,47
28,48
28,49
28,50
28,51
28,52
28,53
28,54
28,55
28,56
28,57
28,58
28,59
28,60
28,61
28,62
28,63
28,65
28,66
28,67
28,69
28,70
28,71
28,74
28,76
28,78
28,79
28,80
28,82
28,83
28,84
28,86
28,88
28,91
28,92
28,93
28,94
28,96
28,97
28,98
28,99
28,100
29,31
29,33
29,35
29,39
29,42
29,44
29,46
29,50
29,53
29,57
29,60
29,62
29,63
29,65
29,66
29,68
29,69
29,71
29,74
29,78
29,80
29,82
29,83
29,85
29,86
29,88
29,94
29,96
29,98
29,99
30,31
30,32
30,33
30,34
30,35
30,36
30,37
30,38
30,39
30,40
30,42
30,43
30,44
30,45
30,46
30,47
30,49
30,51
30,53
30,54
30,55
30,57
30,58
30,59
30,60
30,61
30,62
30,63
30,64
30,66
30,67
30,68
30,69
30,70
30,72
30,73
30,74
30,75

30,77
30,80
30,81
30,82
30,83
30,84
30,85
30,86
30,87
30,89
30,91
30,92
30,93
30,94
30,95
30,96
30,97
30,98
30,100
31,32
31,33
31,34
31,35
31,36
31,37
31,39
31,40
31,41
31,42
31,43
31,45
31,46
31,47
31,48
31,49
31,51
31,52
31,53
31,54
31,55
31,57
31,58
31,59
31,60
31,64
31,65
31,66
31,67
31,69
31,70
31,71
31,72
31,73
31,74
31,75
31,76
31,77
31,78
31,79
31,80
31,82
31,83
31,84
31,85
31,86
31,87
31,91
31,92
31,93
31,94
31,95
31,96
31,97
31,98
31,99
31,100
32,33
32,35
32,37
32,39
32,40
32,42
32,44
32,46
32,48
32,50
32,52
32,53
32,59
32,60
32,63
32,65
32,68
32,69
32,71
32,72
32,76
32,77
32,78
32,82
32,83
32,85
32,86
32,88
32,90
32,94
32,96
32,98
32,99
33,34
33,35
33,36
33,38
33,39
33,41
33,42
33,43
33,44
33,46
33,47
33,49
33,50
33,51
33,52
33,53
33,54
33,57
33,64
33,65
33,67
33,68
33,69
33,70
33,71
33,72
33,74
33,75
33,76
33,77
33,79
33,80
33,81
33,82
33,83
33,85
33,86
33,87
33,89
33,90
33,91
33,92
33,93
33,94
33,96
33,97
33,98
33,99
33,100
34,35
34,37
34,39
34,40
34,42
34,44
34,50
34,52
34,53
34,55
34,57
34,59
34,60
34,62
34,63
34,65
34,66
34,68
34,69
34,72
34,74
34,76
34,77
34,80
34,82
34,83
34,85
34,86
34,88
34,90
34,92
34,94
34,96
34,98
34,99
35,36
35,37
35,38
35,40
35,41
35,42
35,43
35,44
35,45
35,47
35,48
35,49
35,50
35,51
35,52
35,53
35,54
35,55
35,56
35,57
35,58
35,60
35,62
35,63
35,64
35,65
35,66
35,67
35,68
35,69
35,70
35,71
35,72
35,73
35,74
35,75
35,76
35,77
35,78
35,79
35,80
35,81
35,82
35,83
35,84
35,85
35,86
35,87
35,88
35,89
35,90
35,91
35,92
35,93
35,94

35,95
35,96
35,97
35,98
35,100
36,37
36,39
36,40
36,42
36,44
36,46
36,50
36,52
36,53
36,55
36,57
36,59
36,60
36,62
36,63
36,65
36,66
36,68
36,69
36,71
36,72
36,74
36,76
36,77
36,80
36,82
36,83
36,86
36,88
36,90
36,92
36,94
36,96
36,98
36,99
37,38
37,39
37,40
37,41
37,42
37,43
37,44
37,45
37,46
37,47
37,48
37,49
37,50
37,51
37,52
37,53
37,54
37,55
37,56
37,57
37,58
37,59
37,60
37,61
37,62
37,64
37,65
37,66
37,67
37,68
37,69
37,70
37,71
37,73
37,74
37,75
37,76
37,78
37,79
37,80
37,81
37,82
37,83
37,84
37,85
37,86
37,87
37,88
37,89
37,90
37,91
37,92
37,93
37,94
37,97
37,99
37,100
38,39
38,40
38,42
38,44
38,46
38,48
38,52
38,53
38,55
38,57
38,59
38,60
38,62
38,63
38,65
38,66
38,68
38,71
38,72
38,74
38,77
38,82
38,83
38,86
38,88
38,90
38,92
38,94
38,96
38,98
38,99
39,40
39,41
39,42
39,44
39,45
39,46
39,47
39,49
39,50
39,52
39,53
39,54
39,55
39,56
39,57
39,59
39,60
39,61
39,62
39,63
39,65
39,66
39,67
39,68
39,69
39,70
39,71
39,72
39,74
39,76
39,77
39,78
39,79
39,81
39,82
39,83
39,84
39,86
39,88
39,89
39,91
39,92
39,93
39,94
39,95
39,96
39,97
39,98
39,99
39,100
40,41
40,42
40,44
40,46
40,47
40,49
40,50
40,51
40,52
40,54
40,55
40,56
40,59
40,61
40,62
40,63
40,64
40,65
40,66
40,67
40,68
40,69
40,70
40,71
40,72
40,73
40,74
40,75
40,77
40,78
40,79
40,80
40,81
40,83
40,84
40,85
40,86
40,87
40,88
40,89
40,91
40,92
40,94
40,95
40,96
40,97
40,98
40,99
40,100
41,42
41,46
41,48
41,50
41,52
41,53
41,55
41,59
41,60
41,62
41,63
41,65
41,66
41,68
41,69
41,71
41,76
41,77
41,78
41,80
41,82

41,83
41,85
41,86
41,88
41,90
41,94
41,96
41,98
41,99
42,43
42,46
42,47
42,48
42,50
42,51
42,52
42,54
42,55
42,56
42,57
42,58
42,59
42,61
42,62
42,63
42,64
42,66
42,67
42,69
42,73
42,75
42,76
42,77
42,78
42,79
42,80
42,81
42,82
42,83
42,84
42,86
42,88
42,89
42,91
42,92
42,93
42,94
42,95
42,96
42,97
42,99
42,100
43,44
43,46
43,48
43,50
43,52
43,53
43,59
43,62
43,63
43,65
43,66
43,68
43,69
43,72
43,74
43,76
43,77
43,78
43,80
43,82
43,83
43,85
43,86
43,90
43,92
43,94
43,96
43,98
43,99
44,45
44,46
44,47
44,48
44,49
44,50
44,52
44,53
44,54
44,55
44,56
44,57
44,58
44,59
44,60
44,61
44,63
44,64
44,65
44,66
44,67
44,68
44,69
44,70
44,71
44,73
44,74
44,75
44,76
44,77
44,79
44,80
44,82
44,83
44,84
44,85
44,87
44,88
44,90
44,91
44,92
44,93
44,94
44,95
44,96
44,97
44,98
44,99
45,46
45,48
45,50
45,52
45,53
45,57
45,59
45,60
45,62
45,63
45,65
45,66
45,69
45,71
45,72
45,74
45,77
45,78
45,80
45,82
45,83
45,85
45,86
45,88
45,90
45,92
45,94
45,98
45,99
46,47
46,48
46,49
46,50
46,51
46,52
46,53
46,54
46,55
46,56
46,58
46,59
46,60
46,61
46,62
46,63
46,64
46,66
46,67
46,68
46,70
46,72
46,73
46,74
46,75
46,76
46,78
46,79
46,82
46,83
46,84
46,85
46,86
46,87
46,88
46,89
46,90
46,91
46,92
46,93
46,94
46,95
46,96
46,98
46,99
46,100
47,48
47,50
47,53
47,55
47,57
47,59
47,60
47,62
47,63
47,65
47,66
47,68
47,69
47,71
47,74
47,76
47,77
47,78
47,80
47,82
47,85
47,86
47,88
47,92
47,96
47,98
47,99
48,49
48,50
48,52
48,53
48,54
48,55
48,56
48,58
48,64
48,66
48,67
48,68
48,69
48,70
48,71
48,72
48,74

48,75
48,76
48,77
48,78
48,79
48,82
48,83
48,84
48,85
48,86
48,87
48,88
48,89
48,90
48,91
48,92
48,94
48,95
48,96
48,97
48,98
48,99
48,100
49,50
49,53
49,55
49,59
49,60
49,62
49,65
49,66
49,68
49,69
49,71
49,72
49,76
49,78
49,80
49,82
49,85
49,86
49,88
49,90
49,92
49,94
49,99
50,51
50,52
50,54
50,55
50,57
50,58
50,59
50,60
50,61
50,62
50,63
50,65
50,66
50,67
50,68
50,69
50,70
50,71
50,72
50,74
50,75
50,76
50,77
50,78
50,79
50,80
50,81
50,82
50,83
50,84
50,85
50,86
50,87
50,88
50,89
50,90
50,91
50,92
50,93
50,94
50,95
50,96
50,98
50,99
50,100
51,52
51,53
51,55
51,57
51,60
51,63
51,65
51,66
51,69
51,72
51,74
51,76
51,77
51,78
51,80
51,82
51,83
51,85
51,86
51,90
51,94
51,96
51,98
52,53
52,54
52,55
52,56
52,57
52,58
52,59
52,60
52,61
52,63
52,64
52,65
52,66
52,67
52,69
52,70
52,71
52,72
52,74
52,75
52,77
52,78
52,80
52,83
52,84
52,85
52,88
52,90
52,91
52,92
52,94
52,95
52,96
52,97
52,98
52,99
52,100
53,54
53,55
53,56
53,59
53,61
53,63
53,64
53,65
53,66
53,67
53,69
53,70
53,71
53,72
53,73
53,74
53,76
53,77
53,78
53,79
53,80
53,82
53,83
53,84
53,86
53,88
53,89
53,90
53,91
53,92
53,93
53,94
53,95
53,97
53,98
53,99
53,100
54,59
54,60
54,62
54,63
54,66
54,69
54,71
54,72
54,74
54,76
54,78
54,80
54,83
54,85
54,86
54,88
54,90
54,92
54,94
54,96
54,99
55,56
55,58
55,59
55,60
55,61
55,62
55,63
55,65
55,66
55,68
55,69
55,71
55,72
55,73
55,74
55,76
55,77
55,78
55,79
55,81
55,83
55,84
55,85
55,86
55,89
55,90
55,92
55,93
55,95
55,96
55,97
55,98
55,99
55,100
56,57
56,59
56,60
56,62
56,65

56,66
56,68
56,69
56,72
56,76
56,77
56,78
56,80
56,82
56,83
56,85
56,86
56,88
56,90
56,92
56,94
56,96
56,98
56,99
57,58
57,60
57,61
57,63
57,64
57,66
57,67
57,68
57,69
57,71
57,72
57,73
57,74
57,76
57,78
57,79
57,80
57,81
57,82
57,83
57,84
57,85
57,86
57,88
57,89
57,90
57,91
57,92
57,94
57,95
57,96
57,98
57,99
58,59
58,62
58,63
58,65
58,68
58,69
58,71
58,74
58,76
58,77
58,78
58,80
58,82
58,83
58,85
58,86
58,88
58,90
58,96
58,99
59,60
59,61
59,62
59,64
59,65
59,66
59,67
59,68
59,69
59,70
59,71
59,72
59,73
59,74
59,75
59,76
59,77
59,78
59,79
59,80
59,81
59,82
59,83
59,84
59,85
59,87
59,88
59,89
59,90
59,91
59,92
59,93
59,95
59,96
59,99
60,61
60,63
60,65
60,66
60,67
60,68
60,70
60,71
60,72
60,73
60,74
60,76
60,77
60,78
60,79
60,80
60,81
60,82
60,83
60,84
60,86
60,87
60,88
60,89
60,90
60,91
60,92
60,93
60,94
60,95
60,96
60,97
60,98
60,100
61,62
61,63
61,66
61,68
61,69
61,71
61,72
61,74
61,76
61,77
61,78
61,80
61,83
61,86
61,88
61,90
61,92
61,94
61,96
61,98
61,99
62,63
62,64
62,65
62,66
62,67
62,68
62,69
62,70
62,71
62,72
62,76
62,77
62,80
62,81
62,82
62,83
62,84
62,85
62,86
62,87
62,88
62,90
62,92
62,93
62,94
62,95
62,97
62,98
62,100
63,64
63,65
63,67
63,68
63,70
63,71
63,73
63,74
63,75
63,76
63,77
63,78
63,79
63,80
63,82
63,83
63,84
63,86
63,88
63,90
63,91
63,93
63,96
63,97
63,98
63,99
63,100
64,65
64,66
64,68
64,69
64,71
64,72
64,74
64,76
64,78
64,82
64,83
64,85
64,86
64,90
64,92
64,96
64,98
64,99
65,66
65,67
65,68
65,69
65,70
65,71
65,73
65,74
65,75
65,76
65,77
65,78

65,79
65,80
65,81
65,82
65,83
65,85
65,86
65,87
65,88
65,89
65,91
65,92
65,93
65,94
65,95
65,96
65,98
65,99
66,67
66,69
66,70
66,71
66,72
66,73
66,74
66,75
66,76
66,77
66,78
66,79
66,80
66,81
66,82
66,83
66,84
66,85
66,86
66,87
66,88
66,89
66,91
66,92
66,93
66,94
66,95
66,96
66,97
66,98
66,99
66,100
67,68
67,69
67,71
67,72
67,74
67,77
67,78
67,80
67,82
67,83
67,85
67,86
67,88
67,92
67,94
67,96
67,98
67,99
68,69
68,70
68,71
68,72
68,73
68,74
68,76
68,77
68,78
68,80
68,84
68,85
68,87
68,88
68,89
68,90
68,91
68,92
68,93
68,94
68,96
68,97
68,98
68,99
68,100
69,70
69,71
69,72
69,74
69,75
69,76
69,77
69,78
69,79
69,81
69,82
69,83
69,84
69,85
69,86
69,87
69,88
69,89
69,91
69,92
69,93
69,94
69,95
69,96
69,97
69,98
69,99
69,100
70,71
70,72
70,74
70,77
70,82
70,83
70,85
70,86
70,88
70,90
70,94
70,96
70,98
71,72
71,73
71,74
71,77
71,79
71,82
71,86
71,87
71,88
71,90
71,91
71,92
71,93
71,94
71,96
71,98
71,99
71,100
72,73
72,74
72,75
72,76
72,77
72,78
72,80
72,82
72,83
72,84
72,85
72,88
72,89
72,90
72,91
72,92
72,93
72,94
72,95
72,96
72,97
72,98
73,74
73,76
73,77
73,78
73,80
73,82
73,83
73,85
73,86
73,88
73,92
73,94
73,98
73,99
74,75
74,76
74,77
74,78
74,79
74,80
74,81
74,82
74,83
74,84
74,85
74,86
74,87
74,89
74,90
74,91
74,92
74,93
74,94
74,95
74,96
74,97
74,99
74,100
75,76
75,82
75,83
75,86
75,88
75,90
75,92
75,96
75,98
75,99
76,77
76,78
76,79
76,80
76,81
76,82
76,83
76,85
76,86
76,88
76,89
76,90
76,91
76,92
76,93
76,95
76,96
76,97
76,98
76,99
76,100
77,78
77,79
77,80
77,81
77,82

77,83
77,84
77,86
77,88
77,89
77,90
77,91
77,92
77,93
77,94
77,95
77,96
77,97
77,98
77,99
78,79
78,82
78,83
78,84
78,85
78,86
78,87
78,88
78,89
78,90
78,91
78,92
78,93
78,94
78,95
78,96
78,97
78,98
78,99
78,100
79,80
79,82
79,83
79,85
79,86
79,88
79,92
79,94
79,98
79,99
80,81
80,83
80,84
80,85
80,86
80,87
80,88
80,89
80,92
80,93
80,95
80,96
80,97
80,98
80,99
80,100
81,82
81,83
81,85
81,86
81,88
81,90
81,92
81,98
81,99
82,83
82,84
82,85
82,88
82,89
82,91
82,92
82,93
82,94
82,95
82,96
82,97
82,99
82,100
83,84
83,85
83,86
83,88
83,89
83,90
83,92
83,93
83,94
83,95
83,96
83,97
83,99
83,100
84,85
84,86
84,88
84,90
84,94
84,98
84,99
85,86
85,87
85,88
85,89
85,92
85,93
85,94
85,95
85,96
85,97
85,98
85,99
85,100
86,87
86,88
86,89
86,90
86,91
86,92
86,93
86,94
86,95
86,96
86,97
86,98
86,99
87,88
87,90
87,92
87,94
87,99
88,89
88,90
88,91
88,92
88,94
88,95
88,96
88,97
88,98
88,99
88,100
89,90
89,94
89,96
89,98
89,99
90,91
90,92
90,93
90,94
90,96
90,97
90,98
90,99
91,92
91,96
91,99
92,93
92,94
92,97
92,98
92,99
92,100
93,96
93,98
93,99
94,95
94,96
94,97
94,99
94,100
95,96
95,98
95,99
96,97
96,98
96,99
96,100
97,98
97,99
98,99
98,100
99,100

sppnw41.csv
17,197,0,0,0,0,0,0,0
2259,5,1,3,4,8,10
3309,4,1,3,4,11
4497,3,1,3,4
4965,4,1,4,9,11
5961,3,1,4,9
768,2,1,4
3915,2,1,4
3417,2,1,8
4752,3,2,5,11
5454,5,2,6,11,13,17
2112,3,2,7,11
3675,2,2,7
5943,5,2,11,13,14,17
5247,4,2,11,13,17
5298,4,2,11,14,15
4542,3,2,11,14
5484,3,2,11,15
3810,2,2,11
5385,1,2
4140,4,3,5,8,10
5199,3,3,5,11
4809,6,3,6,8,10,13,17
5868,5,3,6,11,13,17
5796,4,3,7,8,10
5307,6,3,8,10,13,14,17
6252,6,3,8,10,13,15,17
4611,5,3,8,10,13,17
4662,5,3,8,10,14,15
3897,4,3,8,10,14
2862,3,3,8,10
6357,5,3,11,13,14,17
5661,4,3,11,13,17
5712,4,3,11,14,15
4956,3,3,11,14
3912,2,3,11
5100,1,3
4767,5,4,5,10,12,16
3822,5,4,5,10,16,17
4047,3,4,5,10
5661,3,4,9,10
5562,5,4,10,12,13,14
4548,5,4,10,12,14,16
6261,5,4,10,12,15,16
4767,5,4,10,13,14,17
6507,5,4,10,13,15,17
4608,4,4,10,14,15
3603,5,4,10,14,16,17
3357,3,4,10,14
5316,5,4,10,15,16,17
4746,3,4,10,15
2820,2,4,10
6345,4,5,7,12,16
6090,4,5,7,16,17
4239,5,5,8,10,12,16
5739,5,5,8,10,16,17
3294,5,5,8,10,16,17

3507,3,5,8,10
6525,5,5,12,13,16,17
1833,3,5,12,16
5268,3,5,12,16
1158,3,5,16,17
1158,3,5,16,17
1272,1,5
6423,5,6,7,9,13,17
6141,6,6,7,12,13,16,17
6375,5,6,9,11,12,16
5958,5,6,9,11,13,17
5421,5,6,9,11,16,17
5667,3,6,9,11
5079,5,6,9,13,15,17
3477,4,6,9,13,17
6426,4,6,9,16,17
6495,6,6,11,12,13,16,17
3918,7,6,12,13,14,15,16,17
4077,6,6,12,13,15,16,17
2700,5,6,12,13,16,17
2445,3,6,12,13
5313,3,6,12,16
2727,5,6,13,14,15,17
2661,4,6,13,15,17
2301,3,6,13,17
4368,3,6,16,17
1569,1,6
4563,5,7,9,11,12,16
4599,4,7,9,11,15
3606,5,7,9,11,16,17
3831,3,7,9,11
5559,4,7,9,12,16
6225,4,7,9,13,17
5508,3,7,9,14
5595,3,7,9,15
4614,4,7,9,16,17
4836,2,7,9
5901,5,7,12,13,14,15
6477,6,7,12,13,15,16,17
6078,4,7,12,13,15
4887,5,7,12,14,15,16
6123,4,7,12,14,16
5040,4,7,12,15,16
3486,3,7,12,16
5100,5,7,13,14,15,17
5286,4,7,13,15,17
3933,5,7,14,15,16,17
3699,3,7,14,15
5868,4,7,14,16,17
4095,4,7,15,16,17
3522,2,7,15
2541,3,7,16,17
204,1,7
4995,4,8,9,10,14
5133,3,8,9,10
6585,6,8,10,12,13,14,15
5418,7,8,10,12,13,14,16,17
5031,5,8,10,12,13,14
5571,6,8,10,12,14,15,16
4017,5,8,10,12,14,16
5730,5,8,10,12,15,16
5478,6,8,10,13,14,15,17
5841,6,8,10,13,14,15,17
4239,5,8,10,13,14,17
5979,5,8,10,13,15,17
4626,6,8,10,14,15,16,17
4080,4,8,10,14,15
3075,5,8,10,14,16,17
2829,3,8,10,14
4788,5,8,10,15,16,17
4215,3,8,10,15
5544,4,8,10,16,17
165,2,8,10
5901,4,9,11,12,16
5760,4,9,11,13,17
5046,3,9,11,14
5988,3,9,11,15
4956,4,9,11,16,17
1608,2,9,11
5214,2,9,11
3975,4,9,13,14,17
4884,4,9,13,15,17
3282,3,9,13,17
3333,3,9,14,15
3333,3,9,14,15
2565,2,9,14
3519,2,9,15
5373,3,9,16,17
2913,1,9
5637,5,11,12,13,16,17
5250,3,11,12,13
5661,5,11,12,14,15,16
6456,4,11,12,14,16
5814,4,11,12,15,16
4212,3,11,12,16
5874,5,11,13,14,15,17
5697,4,11,13,15,17
4458,3,11,13,17
4707,5,11,14,15,16,17
4473,3,11,14,15
6201,4,11,14,16,17
4869,4,11,15,16,17
4296,2,11,15
3267,3,11,16,17
156,1,11
3720,6,12,13,14,15,16,17
3360,4,12,13,14,15
6303,5,12,13,14,16,17
2988,5,12,13,14,16,17
2601,3,12,13,14
3870,5,12,13,15,16,17
3546,3,12,13,15
2724,4,12,13,16,17
2466,2,12,13
3174,4,12,14,15,16
2646,4,12,14,15,16
1818,3,12,14,16
5877,3,12,14,16
3375,3,12,15,16
4893,2,12,16
2664,2,12,16
2529,4,13,14,15,17
3048,4,13,14,15,17
2280,3,13,14,17
2685,3,13,15,17
2325,2,13,17
6291,4,14,15,16,17
5595,4,14,15,16,17
5787,4,14,15,16,17
2091,4,14,15,16,17
5913,4,14,15,16,17
1776,2,14,15
4767,3,14,16,17
1263,3,14,16,17
948,1,14
2820,3,15,16,17
5412,3,15,16,17
2451,1,15
4044,2,16,17
2109,2,16,17
4797,2,16,17

# Додаток Г. Лістинг програмного коду

```
'''FILE HINCO-SL '''
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numba import jit
import time
from FitnessFunction import Fitness

#Оцінка пристосованості
@jit(nopython=True)
def get_fitness(pop, dataset):
    #pop = pop.astype(np.float64)
    for p in pop:
        p[-1] = Fitness(p[:-1],dataset)
    return len(pop)

#Селекція
@jit(nopython=True)
def selection(pop, dataset, options):
    fit_fun_norm = np.ones(int(options[4]))
    max_fit_fun = np.max(pop[:,-1])
    min_fit_fun = np.min(pop[:,-1])
    if min_fit_fun == max_fit_fun:
        fit_fun_norm = (1/options[4]) * fit_fun_norm
    else:
        mean_fit_fun = max_fit_fun-min_fit_fun
        for p in range(int(options[4])):
            fit_fun_norm[p] = (max_fit_fun - pop[p,-1])/mean_fit_fun
    rating_fit_fun = np.round(fit_fun_norm*9)
    pairwise = np.zeros((int(options[4]),int(options[4])))
    mu_x = np.zeros(int(options[4]))
    for p in range(int(options[4])):
        for q in range(int(options[4])):
            if (q >= p):
                pairwise[p,q] = np.abs(rating_fit_fun[p]-rating_fit_fun[q])+1
            else:
                pairwise[p,q] = 1/pairwise[q,p]
        mu_x[p] = np.power(np.prod(pairwise[p,:]),(1/options[4]))
        if mu_x[p] < options[5]:
            mu_x[p] = 0
    sum_mu = np.sum(mu_x)
    mu_x = mu_x/sum_mu
    return mu_x

# Клонування
@jit(nopython=True)
def clonning(pop, selection_rate, options):
    clone_pool = np.ones((int(options[6]),int(options[0])))
    clone_count = 0
    n_clones = np.round(options[6]*selection_rate)
    while(np.sum(n_clones) < options[6]):
        n_clones[0] +=1
    while(np.sum(n_clones) > options[6]):
        n_clones[0] -=1
    for p in range(int(options[4])):
        for clone in range(int(n_clones[p])):
            clone_count+=1
            if(clone_count >= int(options[6])):
                break
            clone_pool[clone_count,:] = pop[p,:]
    return  clone_pool

#Кросинговер
@jit(nopython=True)
def crossover(pop, clone, options):
    for p in range(int(options[6]/2)):
        if np.random.rand() >= options[7]:
            parent1 = clone[p,:-1].astype(np.int64)
```

```
            parent2 = clone[int(options[6])-p-1,:-1].astype(np.int64)
            child1 = np.empty_like(parent1)
            child2 = np.empty_like(parent2)
            cross_point_1 = np.random.randint(int(options[0])-3)
            cross_point_2 = np.random.randint(int(options[0])-3)
            while cross_point_1 == 0 or cross_point_1 == cross_point_2 or cross_point_2
== 0:
                cross_point_1 = np.random.randint(int(options[0])-3)
                cross_point_2 = np.random.randint(int(options[0])-3)
            if cross_point_1 < cross_point_2:
                child1[0:cross_point_1] = parent1[0:cross_point_1]
                child1[cross_point_1:cross_point_2] =
(parent1[cross_point_1:cross_point_2] | parent2[cross_point_1:cross_point_2])
                child1[cross_point_2:] = parent2[cross_point_2:]
                child2[0:cross_point_1] = parent2[0:cross_point_1]
                child2[cross_point_1:cross_point_2] =
(parent1[cross_point_1:cross_point_2] | parent2[cross_point_1:cross_point_2])
                child2[cross_point_2:] = parent1[cross_point_2:]
            else:
                child1[0:cross_point_2] = parent1[0:cross_point_2]
                child1[cross_point_2:cross_point_1] =
(parent1[cross_point_2:cross_point_1] & parent2[cross_point_2:cross_point_1])
                child1[cross_point_1:] = parent2[cross_point_1:]
                child2[0:cross_point_2] = parent2[0:cross_point_2]
                child2[cross_point_2:cross_point_1] =
(parent1[cross_point_2:cross_point_1] & parent2[cross_point_2:cross_point_1])
                child2[cross_point_1:] = parent1[cross_point_1:]
            clone[p,:-1] = child1
            clone[int(options[6])-(p+1),:-1] = child2
    return clone

#Мутація
@jit(nopython=True)
def mutation(clone, generation, options):
    mut_select = np.zeros(int(options[6]))
    max_fi = clone[np.argmax(clone[:,-1]), -1]
    min_fi = clone[np.argmin(clone[:,-1]), -1]
    for p in range(int(options[6])):
        mut_select[p] =(1/2)*(1-options[10]*(generation/options[1]))*((max_fi-clone[p,-
1])/(max_fi-min_fi))
    if generation >= options[1]/2:
        mut_level = options[8]
    else:
        mut_level = options[8]*(1+options[9]*(1-2*generation/options[1]))
    for p in range(int(options[6])):
        if np.random.rand() < mut_level:
            #Точкова мутація
            if np.random.rand() < mut_select[p]:
                for gene in range (int(options[0]-1)):
                    if gene >= ((options[0]-1)*options[9]):
                        break
                    mutpoz = np.random.randint(int(options[0]-1))
                    if mutpoz == gene:
                        if clone[p,gene] == 0:
                            clone[p,gene] = 1
                        else:
                            clone[p,gene] = 0
                    else:
                        gene_temp = clone[p,mutpoz]
                        clone[p,mutpoz] = clone[p,gene]
                        clone[p,gene] = gene_temp
            #Сальтаційна мутація
            else:
                for gene in range (int(options[0]-1)):
                    if gene >= ((options[0]-1)*options[9]):
                        break
                    mutpoz = np.random.randint(int(options[0]-1))
                    saltation_num = np.random.randint(int(options[0])-(mutpoz+1))
                    for k in range(saltation_num):
                        gene_temp = clone[p,gene]
```

```
                        clone[p,gene] = clone[p,mutpoz+k]
                        clone[p,mutpoz+k] = gene_temp
    return clone

@jit(nopython=True)
def local_fitness(gene, clone):
    ones_fit = np.array([0], dtype = np.int64)
    zeros_fit = np.array([0], dtype = np.int64)
    for p in range(len(clone)):
        if clone[p, gene[0]] == 0:
            zeros_fit[0] = zeros_fit[0] + clone[p, -1]
        else:
            ones_fit[0] = ones_fit[0] + clone[p, -1]
    zeros_fit[0] = np.array(np.round(zeros_fit[0]/(len(clone) - gene[2])), dtype =
np.int64)
    ones_fit[0] = np.array(np.round(ones_fit[0]/gene[2]), dtype = np.int64)
    zeros = np.array([gene[0],zeros_fit[0], 0, gene[1]], dtype = np.int64)
    ones = np.array([gene[0],ones_fit[0], 1, gene[1]], dtype = np.int64)
    if zeros_fit < ones_fit:
        return  zeros
    else:
        return  ones

@jit(nopython=True)
#Фіксація 1
def local_search(clone, fixed_poz, options):
    for value in fixed_poz:
        clone[:,int(value[0])] = value[1]
    if np.random.rand() < options[12]:
        accept_error_low = int(options[6])/2 - int(options[6]) *0.1
        accept_error_high =  int(options[6])/2 + int(options[6]) *0.1
        accept_gene = np.empty((0,3), np.int64)
        for p in range(int(options[0]-1)):
            ones_num = np.int64(np.sum(clone[:,p]))
            if ones_num < accept_error_low or ones_num > accept_error_high:
                continue
            else:
                gene_error = np.int64(np.abs((options[6]/2) - ones_num))
                accept_gene = np.vstack((accept_gene, np.array([[p, gene_error,
ones_num]])))
        if len(accept_gene) > 0:
            point_num = np.shape(accept_gene)
            gene_fit = np.empty((0,4), np.int64)
            accept_gene = accept_gene[accept_gene[:,1].argsort()]
            for value in accept_gene:
                local_fit = local_fitness(value, clone)
                gene_fit = np.concatenate((gene_fit, local_fit.reshape(1, -1)), axis=0)
            gene_fit = gene_fit[gene_fit[:, 1].argsort()].astype(np.int64)
            fixed_poz = np.vstack((fixed_poz, np.array([[gene_fit[0,0],
gene_fit[0,2]]])))
            clone[:, gene_fit[0,0]] = gene_fit[0,2]
    return clone, fixed_poz

@jit(nopython=True)
#Фіксація 2
def local_search(clone, fixed_poz, options):
    for value in fixed_poz:
        clone[:,int(value[0])] = value[1]
    if np.random.rand() < options[12]:
        accept_error_low = int(options[6])/2 - int(options[6]) *0.1
        accept_error_high =  int(options[6])/2 + int(options[6]) *0.1
        accept_gene = np.empty((0,3), np.int64)
        for p in range(int(options[0]-1)):
            ones_num = np.int64(np.sum(clone[:,p]))
            if ones_num < accept_error_low or ones_num > accept_error_high:
                continue
            else:
                gene_error = np.int64(np.abs((options[6]/2) - ones_num))
                accept_gene = np.vstack((accept_gene, np.array([[p, gene_error,
ones_num]])))
```

```
        if len(accept_gene) > 0:
            point_num = np.shape(accept_gene)
            gene_fit = np.empty((0,4), np.int64)
            accept_gene = accept_gene[accept_gene[:,1].argsort()]
            for value in accept_gene:
                local_fit = local_fitness(value, clone)
                gene_fit = np.concatenate((gene_fit, local_fit.reshape(1, -1)), axis=0)
            gene_fit = gene_fit[gene_fit[:, 3].argsort()].astype(np.int64)
            fixed_poz = np.vstack((fixed_poz, np.array([[gene_fit[0,0],
gene_fit[0,2]]])))
            clone[:, gene_fit[0,0]] = gene_fit[0,2]
    return clone, fixed_poz

@jit(nopython=True)
#Фіксація 3
def local_search(clone, fixed_poz, options):
    for value in fixed_poz:
        clone[:,int(value[0])] = value[1]
    if np.random.rand() < options[12]:
        accept_error_low = int(options[6])/2 - int(options[6]) *0.1
        accept_error_high =  int(options[6])/2 + int(options[6]) *0.1
        accept_gene = np.empty((0,3), np.int64)
        for p in range(int(options[0]-1)):
            ones_num = np.int64(np.sum(clone[:,p]))
            if ones_num < accept_error_low or ones_num > accept_error_high:
                continue
            else:
                gene_error = np.int64(np.abs((options[6]/2) - ones_num))
                accept_gene = np.vstack((accept_gene, np.array([[p, gene_error,
ones_num]])))
        if len(accept_gene) > 0:
            point_num = np.shape(accept_gene)
            gene_fit = np.empty((0,4), np.int64)
            accept_gene = accept_gene[accept_gene[:,1].argsort()]
            for value in accept_gene:
                local_fit = local_fitness(value, clone)
                gene_fit = np.concatenate((gene_fit, local_fit.reshape(1, -1)), axis=0)
            gene_fit = gene_fit[gene_fit[:, 3].argsort()].astype(np.int64)
            fixed_poz = np.vstack((fixed_poz, np.array([[gene_fit[0,0],
gene_fit[0,2]]])))
            clone[:, gene_fit[0,0]] = gene_fit[0,2]
            for value in gene_fit[1:,:]:
                if value[3] == 0:
                    fixed_poz = np.vstack((fixed_poz, np.array([[value[0], value[2]]])))
                    clone[:, value[0]] = value[2]

    return clone, fixed_poz

@jit(nopython=True)
def compression(pop, clone, options):
    pop = np.vstack((pop, clone))
    pop = pop[pop[:, -1].argsort()]
    new_pop = np.zeros((int(options[4]), int(options[0])))
    new_pop_size = 0
    new_pop[new_pop_size,:] = pop[0,:]
    for p in range (int(options[4] + options[6]-1)):
        less_threshold = 0
        for m in range(new_pop_size):
            if np.sqrt(np.sum(np.power(pop[int(p+1),:-1] - new_pop[m,:-1],2))) <
options[11]:
                less_threshold = 1
                break
        if less_threshold == 0:
            new_pop_size += 1
            new_pop[new_pop_size,:] = pop[p,:]
            if new_pop_size == (options[4]-1):
                break
    for idx, value in enumerate(new_pop):
        if (not np.any(new_pop[idx,:])) == 1:
            new_pop[idx,:] = pop[int(idx),:]
```

```
    return new_pop

@jit(nopython=True)
def check_result(pop, options, stop, generation, n_eval_fun, history):
    if (stop >= options[2]) or (generation >= options[1]) or (n_eval_fun >= options[3])
or pop[0,-1] == 0:
        return 0
    else:
        if len(history) > 20:
            if history[-20] == history[-1] :
                return 0
        return 1

def HINCOSF(pop, dataset, n_eval_fun, options):
    start = time.time()
    generation = 0
    local_search_num = 0
    crossover_num = 0
    mutation_num = 0
    fixed_poz = np.empty((0,2), np.int64)
    history = np.array([])
    key = 1
    while key == 1:
        generation += 1
        selection_rate = selection(pop, dataset, options)
        clone = clonning(pop, selection_rate, options)
        current_fit = np.min(clone[:,-1])
        if options[7] > 0:
            clone = crossover(pop, clone, options)
            n_eval_fun += get_fitness(clone, dataset)
            new_fit = np.min(clone[:,-1])
            if new_fit < current_fit:
                crossover_num +=1
                current_fit = new_fit
            else:
                current_fit = np.min(clone[:,-1])
        if options[8] > 0:
            clone = mutation(clone, generation, options)
            n_eval_fun += get_fitness(clone, dataset)
            new_fit = np.min(clone[:,-1])
            if new_fit < current_fit:
                mutation_num +=1
                current_fit = new_fit
            else:
                current_fit = np.min(clone[:,-1])
        if (options[12] == 1 ):
            current_len = len(fixed_poz)
            if current_len > 0:
                for value in fixed_poz:
                   clone[:,int(value[0])] = value[1]
                n_eval_fun += get_fitness(clone, dataset)
            clone, fixed_poz = local_search(clone, fixed_poz, options)
            next_len = len(fixed_poz)
            if (next_len - current_len > 0):
                local_search_num +=1
                n_eval_fun += get_fitness(clone, dataset)
                new_fit = np.min(clone[:,-1])
                if new_fit < current_fit:
                    local_search_num +=1
        pop = compression(pop, clone, options)
        pop = pop[pop[:, -1].argsort()]
        stop = time.time() - start
        history = np.append(history, pop[0,-1])
        key = check_result(pop, options, stop, generation, n_eval_fun, history)
        statistics = [generation,pop[0,-1], stop, n_eval_fun, crossover_num,
mutation_num, local_search_num]
        #print('Generation: ', generation)
        #print('Best: ', pop[0,-1])
        #print('Time: ', stop)
        #print('Function calls: ',n_eval_fun)
```

```
    return(statistics)



def
main(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_prob
,threshold):
#def main ():
    #filename = "knapPI_3_1000_1000_1.csv"
    #filename = "scpcyc07.csv"
    #filename = "sppnw41.csv"
    filename = "graph100-01.csv"
    dataset = np.array(pd.read_csv(filename, header = None).fillna(0))
    #Параметри цільової функції
    dim = len(dataset) # розмірність простору
    #sppnw part
    '''
    matrix = np.zeros((int(dataset[0,1]),int(dataset[0,0]+1)))
    for index in range(int(dataset[0,1])):
        row = int(index + 1)
        for column in range(len(dataset[row,:])):
            if column == 0:
                matrix[index,-1] = dataset[row,column]
            else:
                if dataset[row,column] == 0:
                    break
                else:
                    matrix[index,int(dataset[row,column]-1)] = 1
    '''
    #end
    #mvc_graph part
    #'''
    matrix = np.zeros((int(dataset[0,0]),dim-1))
    for index in range(int(dataset[0,1])):
        row = int(index + 1)
        matrix[int(dataset[row,0]-1),index] = 1
        matrix[int(dataset[row,1]-1),index] = 1
    #'''
    #end
    #sppnw and mvc_graph part
    dataset = matrix
    #end
    #Критерії зупинки алгоритму
    max_gen = 300 # Максимальна кількість поколінь
    max_time = 150 # Максимальний час виконання алгоритму
    max_eval_fun = 100 * (dim-1); # Максимальна кількість викликів цільової функції
    # Параметри алгоритму
    #pop_size = 13 # Розмір популяції
    min_profit_value = 1 / pop_size # Мінімальне значення пристосованості
    #clone_pop_size = 100  # Кількість клонів
    #cross_prob_val = 0.9  # Ймовірність кросинговеру
    #mut_prob_val = 0.5  # Ймовірність мутації
    #mut_rate = 0.3 # Доля мутаційних координат
    #gamma = 0.5 # Параметр вибору мутації
    #threshold = 0.5 # Коефіцієнт стиснення
    compression_threshold = (dim-1)**(threshold)  # Параметр стиснення популяції
    #local_search_prob = 0 #Ймовірність локального пошуку
    pop = np.random.randint(2, size = (pop_size,dim)).astype(np.int64) # Генерація
початкової популяції
    options = np.array([dim,
                        max_gen,
                        max_time,
                        max_eval_fun,
                        pop_size,
                        min_profit_value,
                        clone_pop_size,
                        cross_prob_val,
                        mut_prob_val,
                        mut_rate,
```

```
                        gamma,
                        compression_threshold,
                        local_search_prob])
    n_eval_fun = 0  #Кількість обчислень цільової функції
    # Визначення пристосованості
    n_eval_fun += get_fitness(pop, dataset)
    fitness_value = HINCOSF(pop, dataset, n_eval_fun, options)
    #plt.plot(fitness_value)
    #plt.ylabel('Best fitnes')
    #plt.xlabel('Iteration')
    #plt.title('Fitness dynamic')
    #plt.show()
    return fitness_value



if __name__ == "__main__":
    np.set_printoptions(suppress=True)
    main()

'''FILE FitnessFunction '''
#from kp_func import function
#from scp_func import function
from spp_func import function
#from mvc_func import function
from numba import jit

@jit(nopython=True)
def Fitness(antigen, dataset):
    return function(antigen, dataset)

'''FILE kp_func '''
import numpy as np
from numba import jit

@jit(nopython=True)
def function (antigen, dataset):
    antigen = antigen.astype(np.float64)
    checkweight = np.dot(antigen,dataset[1:,1:].astype(np.float64))
    if checkweight[0] >= dataset[0,1]:
        return checkweight[0] - dataset[0,1]
    elif not(np.any(checkweight[0])):
        return 10**60
    else:
        return (checkweight[0] + dataset[0,1] )**2

'''FILE mvc_func '''
import numpy as np
from numba import jit

@jit(nopython=True)
def function (antigen, dataset):
    include_point = np.dot((dataset[:,:].astype(np.float64)),
(antigen.T.astype(np.float64)))
    extra_point = 0
    non_include_point = 0
    #print(include_point)
    for item in include_point:
        if item == 0:
            non_include_point +=1
        else:
            extra_point += np.abs(item - 1)
    #print(extra_point)
    if extra_point == 0 and non_include_point == 0:
        return np.sum(include_point)
    elif not(np.any(include_point)):
        return 10**60
    else:
        return
(extra_point+non_include_point)*(extra_point+non_include_point)+(len(antigen))
```

```
'''FILE scp_func '''
import numpy as np
from numba import jit

@jit(nopython=True)
def function (antigen, dataset):
    skills = np.empty((int(np.sum(antigen)),5), np.int64)
    skills_num = 0
    for idx, value in enumerate(antigen):
        if value == 1:
            skills[skills_num] = dataset[idx+1,:]
            skills_num += 1
    non_exist_skills = 0
    for point in range(int(dataset[0,1])):
        if np.any(skills == (point+1) ):
            continue
        else:
            non_exist_skills = non_exist_skills + 1
    if non_exist_skills  == 0:
        return np.sum(antigen)
    elif not(np.any(skills)):
        return 10**60
    else:
        return np.sum(antigen) + (non_exist_skills+1)**2

'''FILE spp_func '''
import numpy as np
from numba import jit

@jit(nopython=True)
def function (antigen, dataset):
    cost = np.dot((antigen.astype(np.float64)), (dataset[:,-1].astype(np.float64)))
    skills = np.dot((dataset[:,:-1].T.astype(np.float64)), (antigen.astype(np.float64)))
    extra_skills = 0
    for item in skills:
        extra_skills += np.abs(item - 1)
    if extra_skills == 0:
        return cost
    elif not(np.any(skills)):
        return 10**60
    else:
        return ((cost+extra_skills)**2)

'''FILE test '''
import numpy as np
import xlsxwriter
from numba import jit
from HINCOSF import main as main_1
from HINCOSF_2 import main as main_2
from HINCOSF_3 import main as main_3
pop_size_list = np.int64(np.linspace(5, 20, 16))
clone_pop_size_list = np.int64(np.linspace(10, 200, 20))
cross_prob_val_list = np.linspace(0, 1, 11)
mut_prob_val_list = np.linspace(0, 1, 11)
mut_rate_list = np.linspace(0, 1, 11)
gamma_list = np.linspace(0, 1, 11)
threshold_list = np.linspace(0, 1, 11)
pop_mean_stat = np.empty((len(pop_size_list),8))
clone_mean_stat = np.empty((len(clone_pop_size_list),8))
cross_mean_stat = np.empty((len(cross_prob_val_list),8))
mut_prob_mean_stat = np.empty((len(mut_prob_val_list),8))
mut_rate_mean_stat = np.empty((len(mut_rate_list),8))
gamma_mean_stat = np.empty((len(gamma_list),8))
threshold_mean_stat = np.empty((len(gamma_list),8))
stat = np.empty((10,7))
local_search_prob = 0
for local_search_module in range(4):
    if local_search_module == 1:
        local_search_prob = 1
```

```
        workbook = xlsxwriter.Workbook('LOCAL SEARCH 1.xlsx')
        print("LOCAL SEARCH 1")
    elif local_search_module == 2:
        workbook = xlsxwriter.Workbook('LOCAL SEARCH 2.xlsx')
        print("LOCAL SEARCH 2")
    elif local_search_module == 3:
        workbook = xlsxwriter.Workbook('LOCAL SEARCH 3.xlsx')
        print("LOCAL SEARCH 3")
    else:
        workbook = xlsxwriter.Workbook('LOCAL SEARCH OFF.xlsx')
        print("LOCAL SEARCH OFF")
    clone_pop_size = 100
    cross_prob_val = 0.5
    mut_prob_val = 0.5
    mut_rate = 0.5
    gamma = 0.5
    threshold = 0.5
    worksheet = workbook.add_worksheet("POP_SIZE_TEST")
    print("POP_SIZE_TEST")
    for idx,value in enumerate(pop_size_list):
        pop_size = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        worksheet.write(idx, 0, value)
        pop_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 1, pop_mean_stat[idx,0])
        worksheet.write(idx, 2, pop_mean_stat[idx,1])
        worksheet.write(idx, 3, pop_mean_stat[idx,2])
        worksheet.write(idx, 4, pop_mean_stat[idx,3])
        worksheet.write(idx, 5, pop_mean_stat[idx,4])
        worksheet.write(idx, 6, pop_mean_stat[idx,5])
        worksheet.write(idx, 7, pop_mean_stat[idx,6])
    pop_mean_stat = pop_mean_stat[pop_mean_stat[:, 2].argsort()]
    pop_size = int(pop_mean_stat[np.argmin(pop_mean_stat[:,1]),7])
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, pop_size)
    worksheet = workbook.add_worksheet("CLONE_POP_SIZE_TEST")
    print("CLONE_POP_SIZE_TEST")
    for idx, value in enumerate(clone_pop_size_list):
        clone_pop_size = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
```

```
        clone_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, clone_mean_stat[idx,0])
        worksheet.write(idx, 2, clone_mean_stat[idx,1])
        worksheet.write(idx, 3, clone_mean_stat[idx,2])
        worksheet.write(idx, 4, clone_mean_stat[idx,3])
        worksheet.write(idx, 5, clone_mean_stat[idx,4])
        worksheet.write(idx, 6, clone_mean_stat[idx,5])
        worksheet.write(idx, 7, clone_mean_stat[idx,6])
    clone_mean_stat = clone_mean_stat[clone_mean_stat[:, 2].argsort()]
    clone_pop_size = int(clone_mean_stat[np.argmin(clone_mean_stat[:,1]),7])
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, clone_pop_size)
    worksheet = workbook.add_worksheet("CROSS_PROB_VAL_TEST")
    print("CROSS_PROB_VAL_TEST")
    for idx, value in enumerate(cross_prob_val_list):
        cross_prob_val = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        cross_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, cross_mean_stat[idx,0])
        worksheet.write(idx, 2, cross_mean_stat[idx,1])
        worksheet.write(idx, 3, cross_mean_stat[idx,2])
        worksheet.write(idx, 4, cross_mean_stat[idx,3])
        worksheet.write(idx, 5, cross_mean_stat[idx,4])
        worksheet.write(idx, 6, cross_mean_stat[idx,5])
        worksheet.write(idx, 7, cross_mean_stat[idx,6])
    cross_mean_stat = cross_mean_stat[cross_mean_stat[:, 2].argsort()]
    cross_prob_val = cross_mean_stat[np.argmin(cross_mean_stat[:,1]),7]
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, cross_prob_val)
    worksheet = workbook.add_worksheet("MUT_PROB_VAL_TEST")
    print("MUT_PROB_VAL_TEST")
    for idx,value in enumerate(mut_prob_val_list):
        mut_prob_val = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
```

```
        mut_prob_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, mut_prob_mean_stat[idx,0])
        worksheet.write(idx, 2, mut_prob_mean_stat[idx,1])
        worksheet.write(idx, 3, mut_prob_mean_stat[idx,2])
        worksheet.write(idx, 4, mut_prob_mean_stat[idx,3])
        worksheet.write(idx, 5, mut_prob_mean_stat[idx,4])
        worksheet.write(idx, 6, mut_prob_mean_stat[idx,5])
        worksheet.write(idx, 7, mut_prob_mean_stat[idx,6])
    mut_prob_mean_stat = mut_prob_mean_stat[mut_prob_mean_stat[:, 2].argsort()]
    mut_prob_val = mut_prob_mean_stat[np.argmin(mut_prob_mean_stat[:,1]),7]
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, mut_prob_val)
    worksheet = workbook.add_worksheet("MUT_RATE_TEST")
    print("MUT_RATE_TEST")
    for idx, value in enumerate(mut_rate_list):
        mut_rate = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        mut_rate_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, mut_rate_mean_stat[idx,0])
        worksheet.write(idx, 2, mut_rate_mean_stat[idx,1])
        worksheet.write(idx, 3, mut_rate_mean_stat[idx,2])
        worksheet.write(idx, 4, mut_rate_mean_stat[idx,3])
        worksheet.write(idx, 5, mut_rate_mean_stat[idx,4])
        worksheet.write(idx, 6, mut_rate_mean_stat[idx,5])
        worksheet.write(idx, 7, mut_rate_mean_stat[idx,6])
    mut_rate_mean_stat = mut_rate_mean_stat[mut_rate_mean_stat[:, 2].argsort()]
    mut_rate = mut_rate_mean_stat[np.argmin(mut_rate_mean_stat[:,1]),7]
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, mut_rate)
    worksheet = workbook.add_worksheet("GAMMA_TEST")
    print("GAMMA_TEST")
    for idx, value in enumerate(gamma_list):
        gamma = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
```

```
        gamma_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, gamma_mean_stat[idx,0])
        worksheet.write(idx, 2, gamma_mean_stat[idx,1])
        worksheet.write(idx, 3, gamma_mean_stat[idx,2])
        worksheet.write(idx, 4, gamma_mean_stat[idx,3])
        worksheet.write(idx, 5, gamma_mean_stat[idx,4])
        worksheet.write(idx, 6, gamma_mean_stat[idx,5])
        worksheet.write(idx, 7, gamma_mean_stat[idx,6])
    gamma_mean_stat = gamma_mean_stat[gamma_mean_stat[:, 2].argsort()]
    gamma = gamma_mean_stat[np.argmin(gamma_mean_stat[:,1]),7]
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, gamma)
    worksheet = workbook.add_worksheet("THRESHOLD_TEST")
    print("THRESHOLD_TEST")
    for idx, value in enumerate(threshold_list):
        threshold = value
        if local_search_module == 1 or local_search_module == 0:
            for p in range(10):
                stat[p,:] =
main_1(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        elif local_search_module == 2:
            for p in range(10):
                stat[p,:] =
main_2(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        else:
            for p in range(10):
                stat[p,:] =
main_3(pop_size,clone_pop_size,cross_prob_val,mut_prob_val,mut_rate,gamma,local_search_pr
ob,threshold)
        threshold_mean_stat[idx,:] = [np.mean(stat[:,0]), np.mean(stat[:,1]),
np.mean(stat[:,2]), np.mean(stat[:,3]), np.mean(stat[:,4]), np.mean(stat[:,5]),
np.mean(stat[:,6]), value]
        worksheet.write(idx, 0, value)
        worksheet.write(idx, 1, threshold_mean_stat[idx,0])
        worksheet.write(idx, 2, threshold_mean_stat[idx,1])
        worksheet.write(idx, 3, threshold_mean_stat[idx,2])
        worksheet.write(idx, 4, threshold_mean_stat[idx,3])
        worksheet.write(idx, 5, threshold_mean_stat[idx,4])
        worksheet.write(idx, 6, threshold_mean_stat[idx,5])
        worksheet.write(idx, 7, threshold_mean_stat[idx,6])
    threshold_mean_stat = threshold_mean_stat[threshold_mean_stat[:, 2].argsort()]
    threshold = threshold_mean_stat[np.argmin(threshold_mean_stat[:,1]),7]
    worksheet.write(idx+1, 0, "Best: ")
    worksheet.write(idx+1, 1, threshold)
    workbook.close()
```

# Додаток Д. Рецензія всеукраїнського конкурсу студентських наукових робіт зі штучного інтелекту 2023

**РЕЦЕНЗІЯ**

Всеукраїнський конкурс студентських робіт зі штучного інтелекту 2023

Перший етап

на наукову роботу

назва Статистичний локальний пошук у евристичних та метаевристичних алгоритмах

шифр Trident Beach

представлену на Всеукраїнський конкурс студентських робіт зі штучного інтелекту

| № з/п | Характеристики та критерії оцінки рукопису наукової роботи | Рейтингова оцінка. Максимальна кількість балів (за 120-бальною шкалою) | Бали |
|---|---|---|---|
| 1 | Актуальність проблеми | 10 | 8.0 |
| 2 | Новизна та оригінальність ідей | 15 | 11.3 |
| 3 | Використані методи дослідження | 15 | 8.3 |
| 4 | Теоретичні наукові результати | 10 | 8.0 |
| 5 | Практична направленість результатів | 20 | 10.7 |
| 6 | Рівень використання наукової літератури та інших джерел інформації | 5 | 5.0 |
| 7 | Ступінь самостійності роботи | 10 | 9.0 |
| 8 | Якість оформлення | 5 | 4.7 |
| 9 | Наукові публікації | 10 | 0.0 |
| 10 | Недоліки роботи (пояснення зниження максимальних балів у пунктах 1-9):<br>Покращення алгоритму пошуку має місце, але висновок не повністю відображає особливості задачі. Алгоритм підвищив ефективність пошуку, але не суттєво.<br>Не показано приклади практичного застосування, де вирішення цієї задачі має застосування.<br>1. Не підтверджена можливість використання наявних засобів, і не подано жодних прикладів.<br>2. Не підтверджено новизну та оригінальність ідей, і відсутні кількісні статистичні дані.<br>3. Низький рівень обгрунтування використаних теоретичних засобів.<br>4. Обгрунтування вибору запропонованих моделей є недостатнім.<br>5. Обгрунтування практичного застосування запропонованої автоматизації недостатнє. Домінантним показником адекватності моделі є людський фактор. | | |
| 10.1 | Наявність розробленого програмного забезпечення, у т.ч. веб-сервісу, мобільного застосунку | 10 | 3.7 |

| № з/п | Характеристики та критерії оцінки рукопису наукової роботи | Рейтингова оцінка. Максимальна кількість балів (за 120-бальною шкалою) | Бали |
|---|---|---|---|
| 10.2 | Рівень отриманих практичних результатів наукової роботи | 10 | 8.3 |
| Сума балів | | | 77,0 |

Загальний висновок

**не рекомендується для захисту на науково-практичній конференції**

(рекомендується, не рекомендується для захисту на науково-практичній конференції)

Рецензенти члени Конкурсної комісії напряму «Обчислювальний інтелект»

(підпис) (Ім'я ПРІЗВИЩЕ)

(місце роботи, посада, науковий ступінь)

«08» листопада 2023 року