Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Налаштування параметра мутації генетичного алгоритму

Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
к-т фіз.-мат. наук, доцент
Гулаєва Н.М.

_____
(Підпис)
"___" _____ 2020 року

Виконав студент КН-4
Кобєлєв М. Д.
"___" _____ 2021 року

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Викладач кафедри інформатики,

к-т фіз.-мат. наук, доцент

_____ Гулаєва Н.М.

„____” _____ 2020р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Кобєлєву Михайлу Дмитровичу

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: **Налаштування параметра мутації генетичного алгоритму**

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
Розділ 1. Дослідження впливу параметрів ГА на його збіжність
Розділ 2. Розробка програмного застосунку визначення Pmax
Розділ 3. Експериментальне налаштування Pmax
Висновки.
Список літератури
Додатки (за необхідністю)

Дата видачі „___” _____ 2020 р. Керівник_____

(підпис)

Завдання отримав _____

(підпис)

Тема: **Налаштування параметра мутації генетичного алгоритму**

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	1.10.2020	
2.	Огляд літератури за темою роботи.	1.11.2020	
3.	Проведення експериментів з різними способами оцінювання і ініціалізації.	1.12.2020	
3.	Аналіз результатів експериментів першого завдання.	20.01.2021	
4.	Написання програмного коду.	14.01.2020	
5.	Проведення декількох серій експериментів.	21.02.2020	
6.	Аналіз результатів експериментів другого завдання.	1.03.2020	
7.	Аналіз всіх експериментів та підбиття підсумків.	14.03.2020	
8.	Редагування та доповнення пояснювальної роботи.	31.03.2020	
9.	Оформлення слайдів для доповіді.	4.04.2020	
10.	Захист курсової роботи	12.04.2020	

Студент _____

Керівник _____

“_____”_____ 2021

Зміст

Анотація

Розроблено програмний застосунок для пошуку максимального значення ймовірності мутації Pmax, за якого спостерігається збіжність ГА. З використанням застосунку знайдено значення Pmax для низки тестових задач. Досліджено, який вплив на значення Pmax мають інші параметри ГА, та надано рекомендації щодо вибору значення ймовірності мутації при розв’язуванні практичних задач.

Ключові слова:

Генетичний алгоритм, збіжність генетичного алгоритму, параметр мутації, ймовірність мутації.

Вступ

Обрати правильні вхідні параметри генетичного алгоритму – це непроста та важлива задача. Від них суттєво залежить якість знайденого розв’язку та кількість ітерацій до збіжності популяції. Роботу було присвячено дослідженню саме параметру мутації. Якщо вибрати замале значення для цього параметру – пошук оптимального розв’язку може сильно сповільнитись, або унеможливитись, оскільки мутації привносять нові риси в популяцію. На противагу, якщо взяти завелике значення, то пошук просто перетворюється на випадковий, бо генетичний матеріал не передається до наступних поколінь. Через це, алгоритм може не збігтись і потрібно штучно зупиняти його роботу (після фіксованої кількості ітерацій), при цьому результату не буде отримано.

Однак, робота з генетичними алгоритмами показує, що параметр мутації залежить від інших параметрів алгоритму, таких як розмір особини, популяції, методи відбору та ініціалізації та ін. Існує припущення, що зрозумівши краще суть цих залежностей, ми зможемо передбачати значення параметру мутації, за яких алгоритм буде давати найкращі результати, виходячи з інших початкових параметрів.

Саме цьому і приділено увагу в цій роботі. Розробивши програмний застосунок для налаштування параметра мутації та провівши багато експериментів, ми намагаємось аналізувати ці дані, щоб краще зрозуміти суть залежності максимального значення мутації, за якого зберігається збіжність ГА, від інших вхідних параметрів алгоритму.

Актуальність теми:

Підбір найкращих параметрів для роботи генетичного алгоритму надалі залишається нетривіальною задачею, а залежності між значенням параметрів та результатами алгоритму комплексно не досліджені.

Отримані в ході роботи результати зможуть допомогти визначати значення ймовірності мутації, базуючись на інших параметрах генетичного алгоритму

Об'єкт дослідження:

Збіжність генетичного алгоритму.

Предмет дослідження:

Параметр «ймовірність мутації» та його вплив на збіжність генетичного алгоритму.

Методи дослідження:

Експериментальний метод: розроблено програму для підбору та тестування значень параметру Pmax та згенеровано набори значень, за яких відбувається або не відбувається збіжність ГА.

Структура роботи:

Робота має три розділи:

1) В першому розділі досліджується вплив різних факторів на значення ймовірності, за якого збіжність в ГА зберігається. За основу для досліджень були взяті як результати з інших робіт, так і поставлено нові експерименти.
2) В другому розділі описується процес розробки програмного застосунку для автоматичного визначення та тестування параметру мутації.
3) Третій розділ – проведення експериментів за допомогою програмного застосунку. Аналіз результатів цих експериментів та дослідження залежностей параметру мутації від інших вхідних параметрів.

Розділ 1. Дослідження впливу параметрів ГА на його збіжність

1.1 Постановка задачі

В цій роботі розглядається ГА з генераційним типом репродукції та досліджуються умови, за яких алгоритм збіжний.

Схема роботи алгоритму:

1. Ініціалізація
2. Моделювання еволюційного процесу (репродукція):
   2.1. Оцінювання. Якщо виконується умова зупинки, перехід на п.3
   2.2. Відбір
   2.3. Застосування оператора мутації
   2.4. Перехід на п.2.1
3. Завершення роботи

Алгоритм збігається, якщо протягом останніх NUM_ITER = 10 ітерацій, середнє здоров’я популяції змінюється не більше ніж на EPS = 0.0001.

Визначимо оптимальне значення параметру мутації Pmax як максимальне значення ймовірності мутації, за якого збіжність алгоритму відбувається у 100% прогонів. Тобто збільшивши це значення, на приклад, на 20%, ми втратимо збіжність у деяких прогонах.

Для отримання значення Pmax, ми спочатку підбираємо це значення за допомогою алгоритму підбору (додаток А), а потім тестуємо такі випадки:

- За Pmax * 80% - збіжність очікується у 100% прогонів;
- За Pmax – збіжність очікується у 100% прогонів;
- За Pmax * 120% - збіжність у <100% прогонів.

Таким чином, ми отримуємо значення Pmax з похибкою, що може досягати 20%.

1.2 Вплив способу кодування та розміру популяції

В [1] було проведено експериментальний підбір значень Pmax для різних наборів вхідних параметрів (розміру популяції, довжини особини, методу відбору), на прикладі функцій, що базуються на відстані Гемінга до ідеальної особини. Було проаналізовано залежності Pmax від цих параметрів та виведені рівняння, що описують ці залежності.

Зокрема, спостерігали зворотно пропорційну залежність Pmax від L * N, де L – довжина особини, а N – кількість особин в популяції:

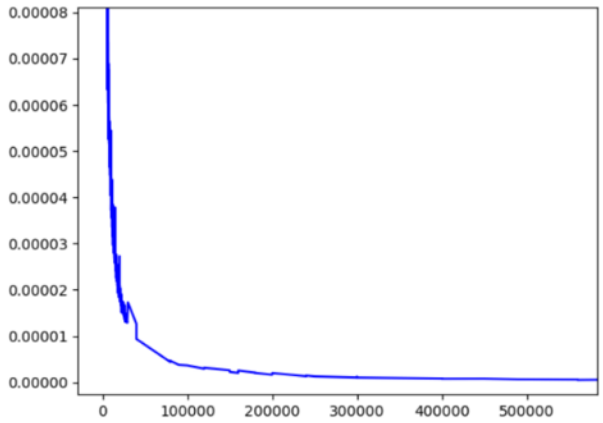


*Рисунок 1.1 Залежність Pmax від (L * N). Взято з [1]*

Якщо побудувати графік 1 / (L * N), дійсно отримуємо пряму:

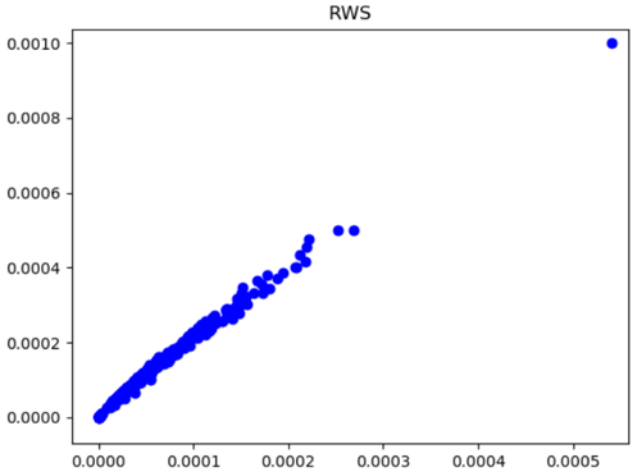


*Рисунок 1.2 Залежність Pmax від 1 / (L * N) . Взято з [1]*

В [1] також вдалося вивести рівняння для різних видів відбору:

- Турнір: $P_{max}(L,N) = \frac{0.69157678}{L*N}$

- Рулетка: $P_{max}(L, N) = \frac{0.39869718}{L*N}$

Надалі буде проведене дослідження залежності Pmax від інших параметрів ГА, зокрема від функції оцінювання (1.3) та способу ініціалізації популяції (1.4).

1.3 Вплив функції пристосованості

Теоретично одне значення мутації може давати гарні результати на одній функції пристосованості, а на іншій – погані, ми хочемо це перевірити експериментально.

Ми зробили припущення, що функція пристосованості (тестова функція) має суттєвий вплив на Pmax. Щоб це перевірити, ми провели експерименти з функцією селективної переваги на біт. Тобто

$$f(x) = 1 * (l - k) + k * \sigma$$

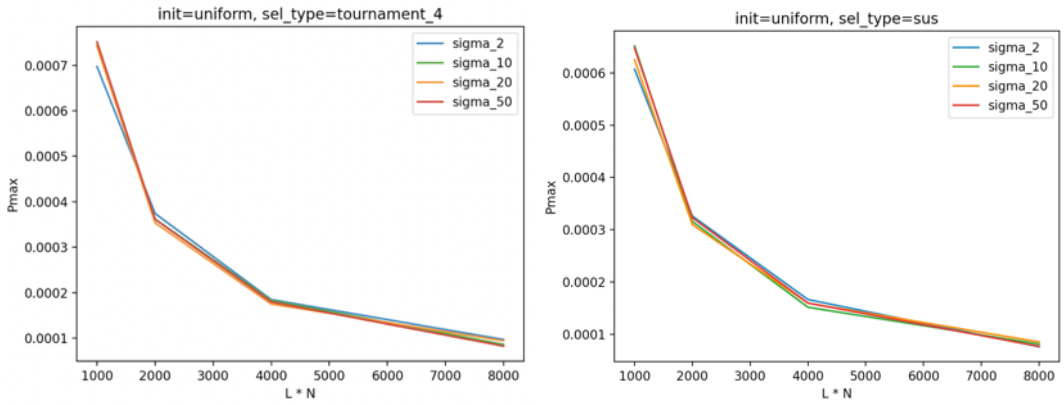
де k – кількість нулів в бінарному ланцюжку x, σ - параметр функції.

Експерименти проводились на таких параметрах: L = 10, 20; N = 100,200,400; σ = 2,10,20,50. Ініціалізація – всі 0, всі 1 та випадково за рівномірним розподілом, init = all_0, all_1, uniform. Повний набір параметрів та отримані значення Pmax наведено в додатку Б.

Зобразимо залежність Pmax від L * N, за допомогою запиту

```
SELECT N * L,  Pmax
FROM values
WHERE {condition}
```

`{condition}` – зазначено в заголовку кожного графіку, а `values` – вміст додатку Б. Отримаємо такі графіки:



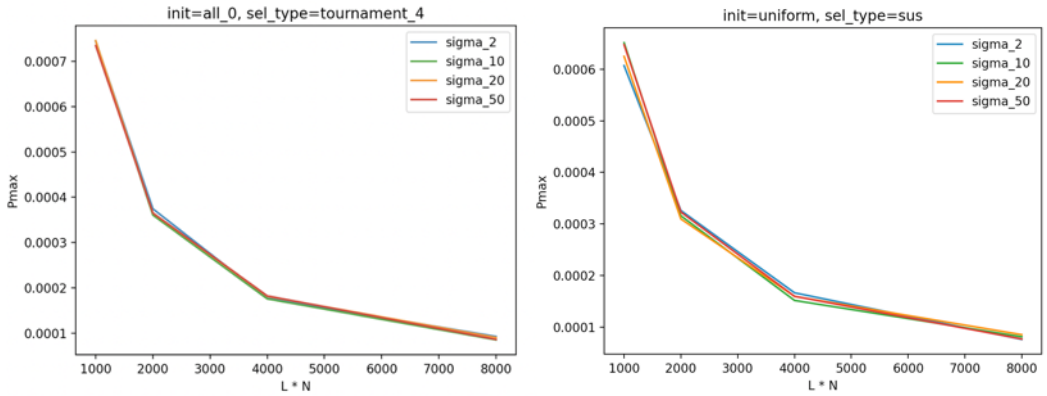


*Рисунок 1.3 Графіки залежностей Pmax від L * N для тестових функцій sigma*

Обрахувавши попарні різниці для кожного випадку init, sel_type, n, l за допомогою формули:

$$diff(Pmax_{sel1}, Pmax_{sel2}) = \frac{Pmax_{sel2} - Pmax_{sel1}}{Pmax_{sel2}} * 100\%$$

бачимо незначні відхилення:

- *sigma_2 – sigma_50*: avg_diff = **-1%**, std_diff = **6.6**
- *sigma_2 – sigma_10*: avg_diff = **0.8%**, std_diff = **6.7**
- *sigma_10 – sigma_20*: avg_diff = **-0.5%,** std_diff = **5.5**

Для інших пар ситуація така сама.

З наведених графіків та даних складно підтвердити або спростувати наше припущення, бо функції відрізняються лише одним параметром σ, що приймає значення 2, 10, 20, 50. Хоча незначні коливання є (особливо для типу відбору sus), вони все ще в межах похибки 20%.

Спробуємо змінити природу самої функції. До цього ми розглядали функції, що «нагороджували» особини за більшу кількість «0» або «1», проте в більшості задач, що розв'язують ГА, за допомогою «0» та «1» кодують можливі розв'язки задачі.

Візьмемо приклад кодування вузлами дискретизації. Особини довжини 10 та 20 – числа, записані в бінарній системі від 0 до 2^{10} - 1 та 2^{20} - 1 відповідно. Візьмемо також інші тестові функції: Еклі, сферичну, x^4,

Растригіна, Деба-2, Деба-4. Формули та графіки цих функцій можна знайти в додатку В.

Опис умов проведення цих експериментів, а також результати надано в Розділах 2 та 3.

Порівняємо залежність Pmax від L * N для цих функцій з попередніми функціями (sigma):

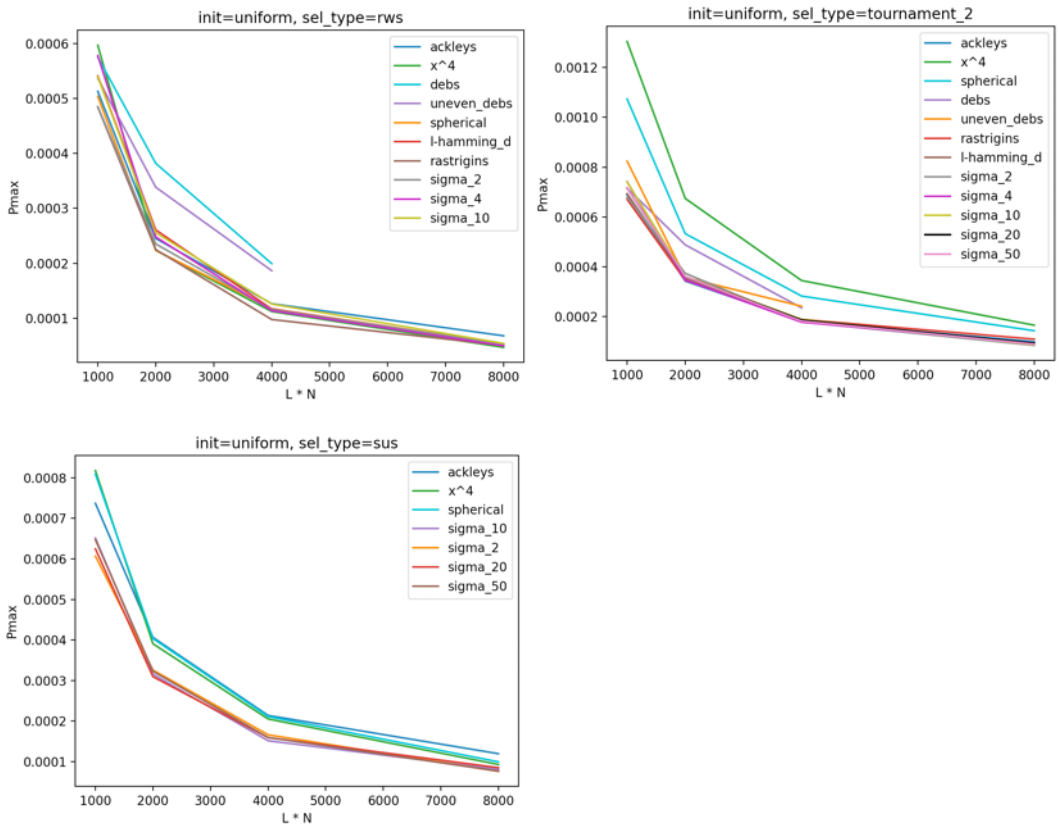


*Рисунок 1.4 Графіки залежностей Pmax від L * N для різних тестових функцій.*

Обрахуємо значення попарних різниць за тією ж формулою, що і для функцій sigma:

- *sigma_10 - сферична*: avg_diff = **-23.8%**, std_diff = **19.5**
- *Деба 2 - sigma_50*: avg_diff = **-25.5%**, std_diff = **16.2**
- x^4 – *sigma_10*: avg_diff = **-57.4%**, std_diff = **44.5**
- x^4 – *сферична*: avg_diff = **-9.7%**, std_diff = **18.6**

Аналогічні ситуації спостерігаємо для інших пар функцій.

Бачимо велику розбіжність в значеннях Pmax для різних тестових функцій. Особливо помітна різниця на графіках для маленьких L * N.

Отже, аналізуючи графіки та дані, ми не можемо бути впевнені, що однакові значення Pmax будуть мати хороші результати на різних за природою функціях. А з цього випливає, що для кожної функції (для кожної задачі, що розв’язується) потрібно підбирати своє значення Pmax

1.4 Вплив методу ініціалізації

З цих експериментів ми можемо дослідити вплив методу ініціалізації на Pmax. Порівняємо три методи:

1) Всі 0 – тобто всі «оптимальні» особини **all_0**;
2) Всі 1 – найгірший випадок **all_1;**
3) Випадковий за рівномірним розподілом **uniform.**

Залежності Pmax від L, N окремо:

```
SELECT L,  Pmax
FROM values
WHERE {condition}
```

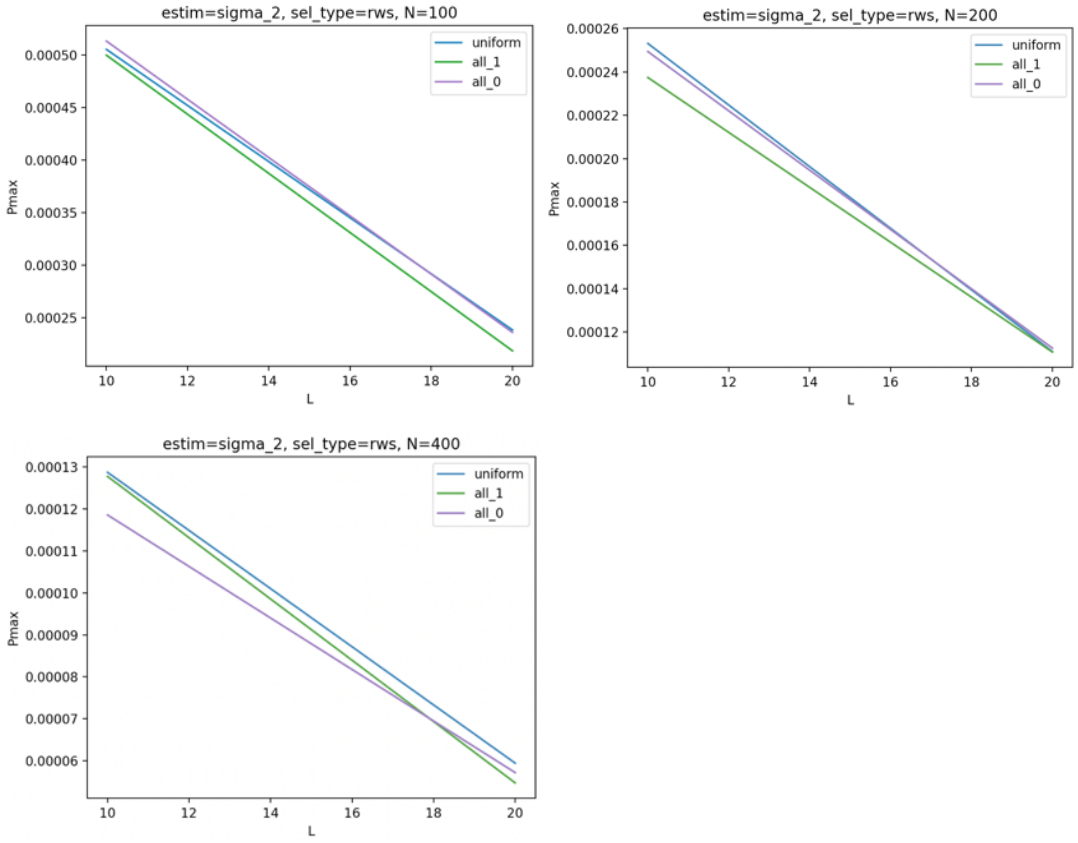


Рисунок 1.5 Графіки залежностей Pmax від L для різних способів ініціалізації Для N = 100, 200, 400.

```
SELECT N,  Pmax
FROM values
WHERE {condition}
```

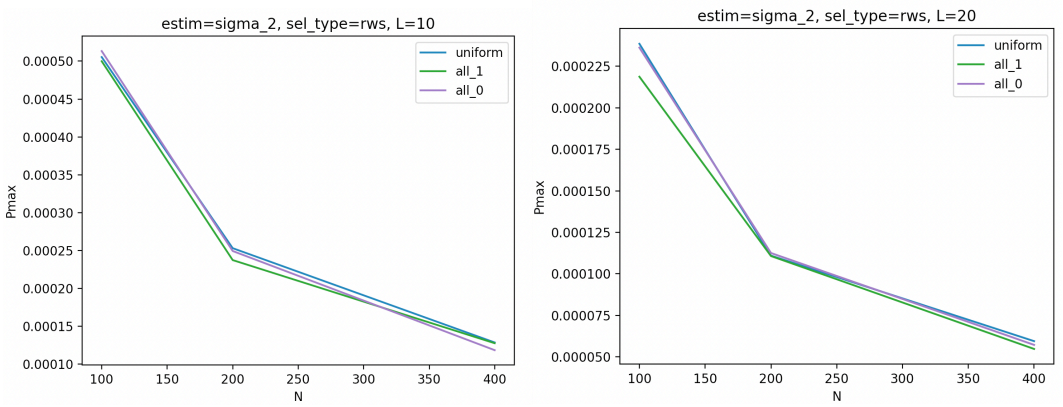
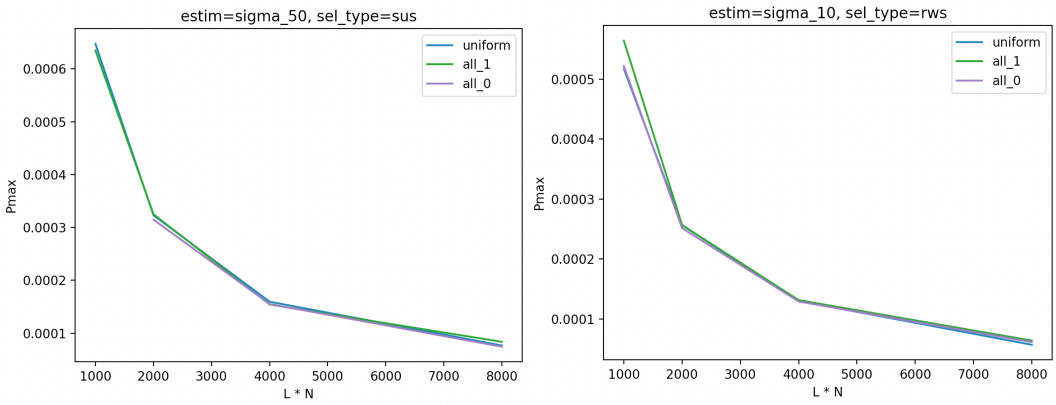


Рисунок 1.6 Графіки залежностей Pmax від N для різних способів ініціалізації. Для L = 10, 20

Залежність Pmax від L * N:

```
SELECT L*N,  Pmax
FROM values
WHERE {condition}
```



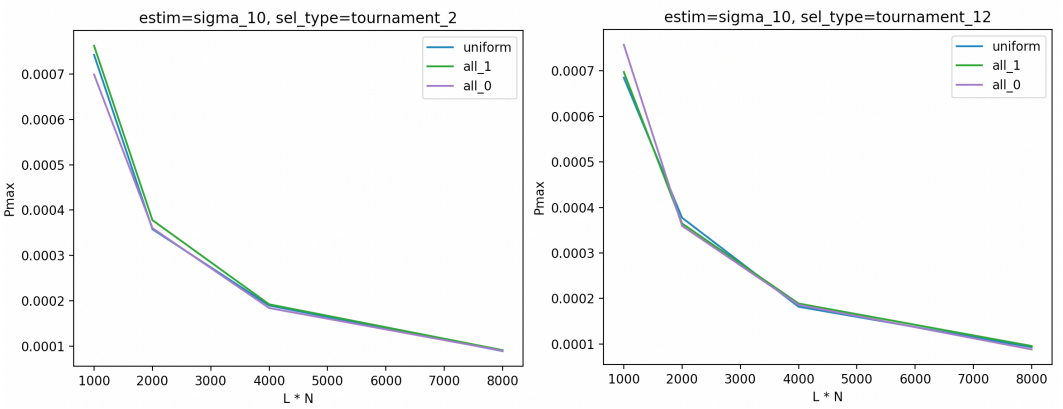


*Рисунок 1.7 Графіки залежностей Pmax від L*N для різних способів ініціалізації.*

Оцінимо попарні різниці для способів ініціалізації для кожного випадку estim, sel_type, L, N:

$$diff(Pmax_{init1}, Pmax_{init2}) = \frac{Pmax_{init2} - Pmax_{init1}}{Pmax_{init2}} * 100\%$$

- *all_0 – all_1*: avg_diff = **0.35%**, std_diff = **5.4**
- *all_0 – uniform*: avg_diff = **-0.6%**, std_diff = **5.6**
- *all_1 – uniform*: avg_diff = **-1.1%**, std_diff = **5.9**

З цих даних ми бачимо, що спосіб ініціалізації популяцій якщо і має вплив, то незначний і в межах похибки 20%.

Також оцінимо точність алгоритму за різних способів ініціалізації. Оскільки в методі **all_0** вся популяція складається з оптимальних особин, то припускаємо, що за такого методу, частка прогонів з правильною збіжністю (тобто коли найкраще/середнє здоров'я популяції знаходиться дуже близько до оптимуму) більша.

Оскільки функції оцінювання (sigma) набувають лише натуральних значень, то вважатимемо, що популяція збіглась правильно, якщо середнє здоров'я на фінальній ітерації було в межах $(f_{opt} - 1, f_{opt} + 1)$, де f_{opt} – значення оптимуму.

Порахуємо для кожного типу ініціалізації, частку значень Pmax, за яких відбулася правильна збіжність алгоритму в усіх прогонах, а також середню та максимальну абсолютну відстань від знайденого розв'язку до оптимуму:

init	Правильна збіжність, %	Відстань, $max(\|f_{found} - f_{opt}\|)$	Відстань, $avg(\|f_{found} - f_{opt}\|)$
all_0	90.65	5.5	0.19
uniform	79.63	8.1	0.50
all_1	68.52	68.6	1.92

Подивимось на таку саму інформацію окремо для кожного типу відбору:

selection	init	Правильна збіжність, %	Відстань, $max(\|f_{found} - f_{opt}\|)$	Відстань, $avg(\|f_{found} - f_{opt}\|)$
rws	*all_0*	91.67	0.6	0.05
rws	*uniform*	33.33	8.1	1.68
rws	*all_1*	8.33	40.5	5.92
sus	*all_0*	60.87	5.5	0.84
sus	*uniform*	62.50	6	0.95
sus	*all_1*	58.33	5.1	0.86
tournament_2	*all_0*	100.00	0.02	0.00
tournament_2	*uniform*	100.00	0.05	0.01
tournament_2	*all_1*	83.33	4.9	0.49

Спостерігаємо особливо велику різницю у значеннях для відбору за методом рулетки, хоча і для всіх методів загалом картина залишається такою, що правильна збіжність за **all_0** більша, ніж за **uniform** чи **all_1**.

Підсумовуючи графіки та дані, спостерігаємо, що спосіб відбору не має сильного впливу на значення Pmax. Проте, також бачимо, що спосіб відбору може вплинути на точність алгоритму (збіжність до глобального максимуму). Оскільки розв’язуючи задачу за допомогою ГА, ми зазвичай не знаємо, де лежить екстремум, то підхід випадкової ініціалізації популяції може давати в середньому кращі результати, ніж ініціалізація в одній точці (особливо, якщо ця точка виявиться найгіршим випадком).

Розділ 2. Розробка програмного застосунку автоматичного визначення Pmax

2.1 Алгоритм налаштування Pmax

Тепер коли ми маємо уявлення про залежності Pmax від вхідних параметрів, ми хочемо протестувати ці значення на функціях з тестових наборів, які використовуються для порівняльного аналізу стохастичних алгоритмів [2].

Загальний алгоритм дій:

1. По можливості, ми беремо значення Pmax вже знайдене за минулих досліджень або вираховуємо його з рівнянь лінійних залежностей, наведених в 1.2.
2. Тестуємо це значення (збіжність має відбуватись при 100% прогонів, але при збільшенні Pmax на 20%, збіжність має бути не в усіх прогонах). Якщо тестування вдале – переходимо на крок 4, якщо ні – на крок 3.
3. Беремо те значення, що тестували за початкове, проводимо дії описані в алгоритмі підбору Pmax (Додаток А). Результат алгоритму передаємо на крок 2.
4. Кінець.

Як бачимо, цей підхід є ітеративним та повністю автоматизованим.

При роботі алгоритму нам важливо зберігати низку метрик популяції, як фінальної (розв’язку), так і по ходу ітерацій, більш детально про це в пункті 2.2.

2.2 Структура бази даних

Оскільки для аналізу Pmax необхідно зберігати інформацію з експериментів у структурованому вигляді, було вирішено використовувати реляційну базу даних.

Окрім інформації про метрики популяцій, також було необхідно мати параметри експериментів в базі для зручного доступу до них при аналізі.

Отже, було розроблено ER-модель БД та виокремлено такі сутності:

- **Function** – тестова функція (сферична, Деба, і тд);
- **FuncParam** – параметри функції (інтервал, на якому проводимо експерименти, точність);
- **FuncCase** – конкретний випадок функції (напр., сферична на інтервалі [-1, 1] з точністю 3 знаки після коми);
- **ParamSet** – набір параметрів генетичного алгоритму;
- **ExperimentSuite** – експеримент з певним набором параметрів;
- **TestSuite** – тест значення Pmax (окремо для 80% від Pmax, Pmax та 120% від Pmax);
- **RunSet** – і-ий крок алгоритму підбору Pmax (всього 15 на один експеримент);
- **Run** – 1 прогін у тестах (відношення до TestSuite) або в алгоритмі пошуку Pmax (відношення до RunSet).

Також маємо окрему допоміжну таблицю **InitPopulation**, яка служить як seed для ініціалізації популяції. Тобто ми хочемо для одного і того самого методу відбору, значень L, N, способу ініціалізації та номера прогону мати одну і ту саму початкову популяцію.

Ці сутності та взаємозв’язки зображено в ER-діаграмі:

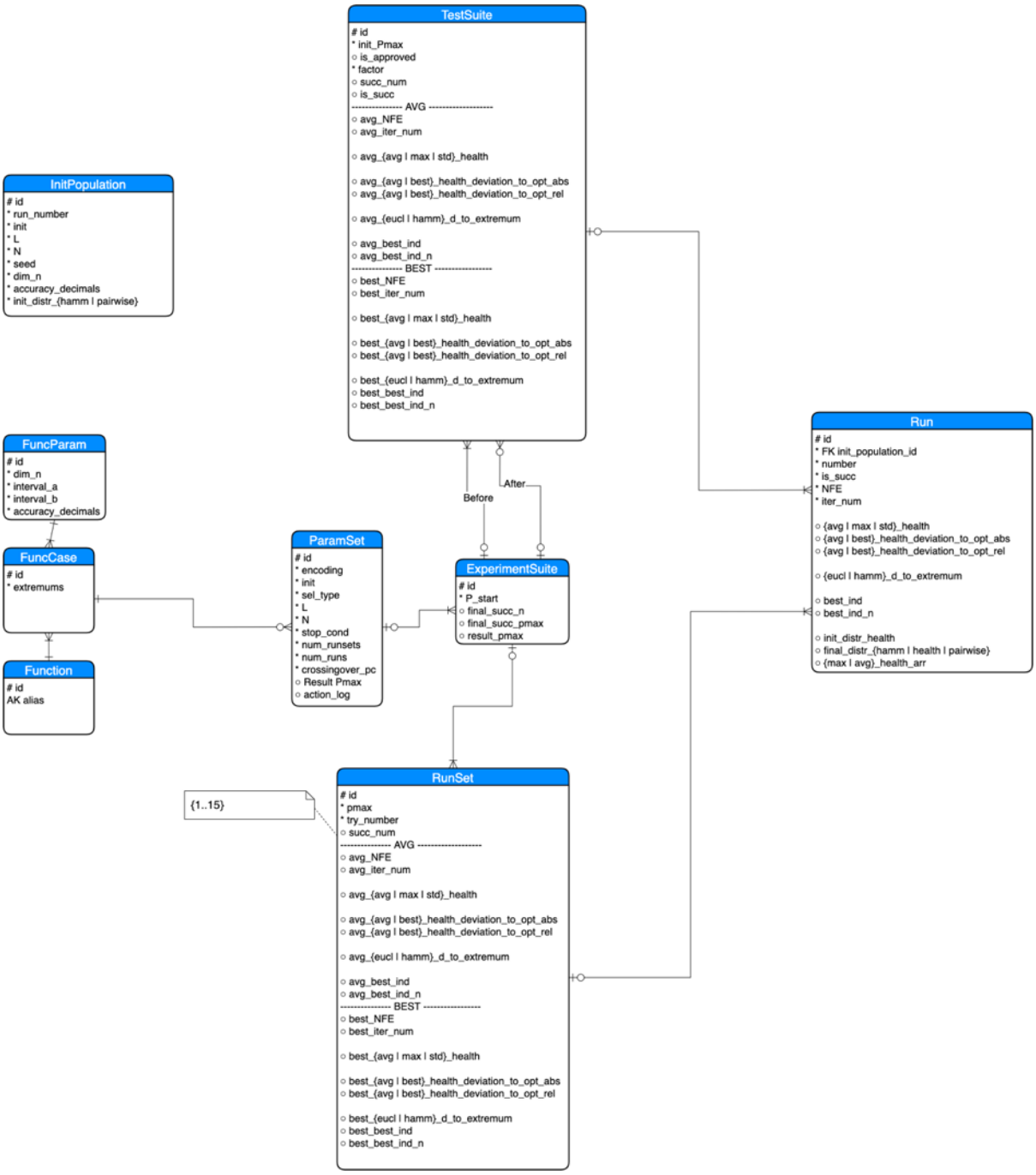


Рисунок 2.1 ER-діаграма предметної області

Позначка виду `field_{a | b}` – використана для позначення полів, назви яких мають спільну частину `field_` та відрізняються частиною в дужках `{}`.

Всі метрики в сутності Run, за винятком `max_health_arr`, `avg_health_arr` обраховуються для фінальної популяції, тобто коли

алгоритм збігся. Якщо популяція не збіглась за MAX_ITER (20000) ітерацій, залишаємо ці поля `null`. Відповідні метрики в сутностях TestSuite та RunSet обраховуються як середнє (з префіксом `avg_`) або як максимальне (з префіксом `best_`) зі значень у відповідних їм Run.

Повну R-модель бази даних з усіма типами даних та зв’язками можна знайти в додатку Г.

2.3 Алгоритм роботи програми

Підготовка:

- Вносимо в базу даних вручну інформацію про функції та експерименти (Function, FuncParam, FuncCase).
- Генеруємо початкові популяції (InitPopulation).
- Генеруємо набори параметрів (ParamSet).

Робота програми (для кожного окремого ParamSet):

1. Беремо необхідний ParamSet, створюємо ExperimentsSuite, Вираховуємо Pmax за рівнянням, що наведене в 1.2 (заповнюємо Pstart в ExpSuite).
2. Проводимо тести (TestSuite) з отриманим значенням.
3. Якщо тести успішні TestSuite.is_succ -> крок 5 (заповнюємо результат у ExpSuite.final_succ_pmax), інакше -> крок 4.
4. Проводимо алгоритм підбору описаний в додатку А. З отриманим значенням переходимо на крок 2.
5. Кінець.

Паралельно обраховуємо середні та найкращі показники та заносимо їх у базу.

Описану схему роботи можна зобразити у UML-flow діаграмі:

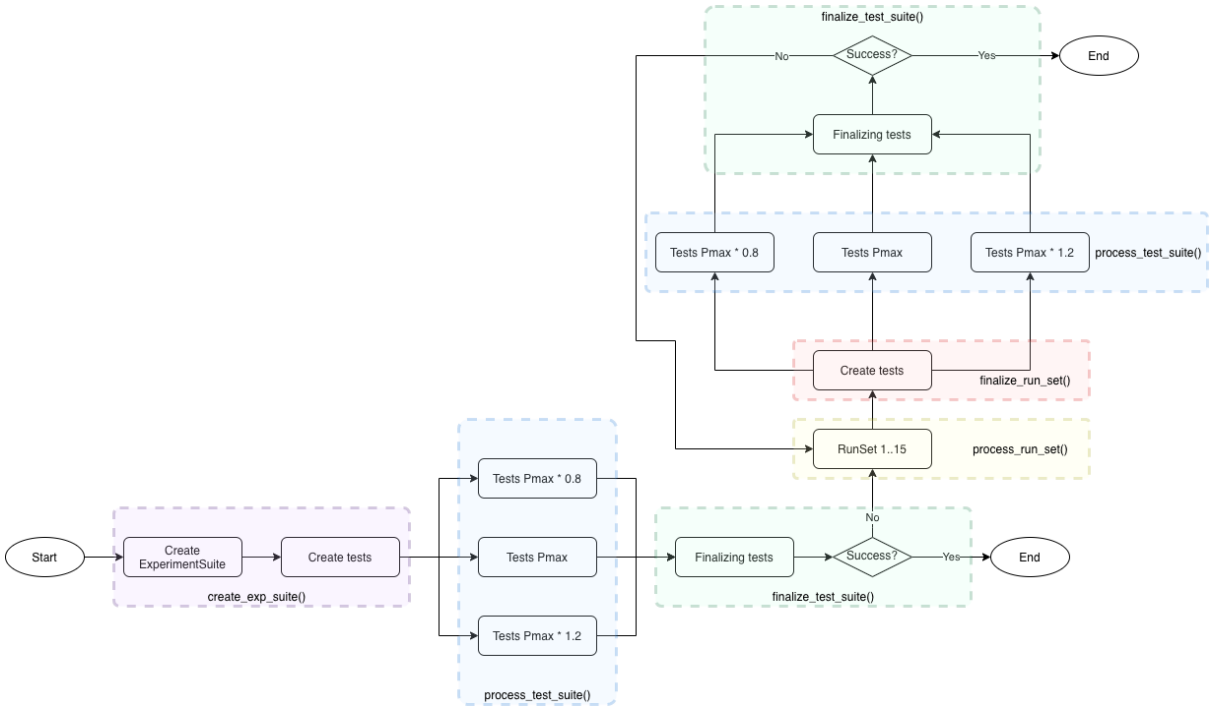


Рисунок 2.2 UML-flow діаграма роботи застосунку

Різними кольорами зображено підзадачі, що відповідають крокам 1 – 4 описаним вище.

2.4 Середовище роботи програми

Алгоритм та застосунок реалізовані мовою Python за допомогою бібліотек **numpy** (бібліотека для швидких математичних обчислень над векторами та матрицями [3]), **peewee** (ORM [4]), **psycopg2** (з'єднання з базою даних). В якості БД було використано PostgreSQL.

Проведення експериментів локально було практично неможливе через їх велику кількість. Проведення експериментів на платформі Google Colab також є незручним через обмеження в часі активності сесії. Натомість було прийняте рішення спробувати проводити експерименти на віртуальних машинах cloud-провайдера.

В якості cloud-провайдера було обрано Digital Ocean, оскільки він надає студентам 50$, які можуть бути спожиті на хмарні сервіси без обмеження у часі через програму Github Student Pack. Доступ до програми можна отримати підключивши студентський корпоративний email до GitHub акаунту.

Проте використання віртуальних машин несе свої недоліки. Зокрема, для оптимізації витрат, була необхідність часто створювати/знищувати машини. Для полегшення запуску коду на віртуальних машинах було використано Docker.

Для використання Docker було створено Dockerfile:

```
FROM python:3.6.8-stretch

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the
container at /app
COPY requirements.txt /app/requirements.txt

# Install the dependencies
RUN pip install -r requirements.txt

# Copy source code
ADD . /app
```

Після побудови образу (`docker build -t <tag> .`) потрібно його завантажити на репозиторій (`docker push -t <tag>`). Платформа Docker Hub пропонує хостинг необмеженої кількості відкритих образів [5]:

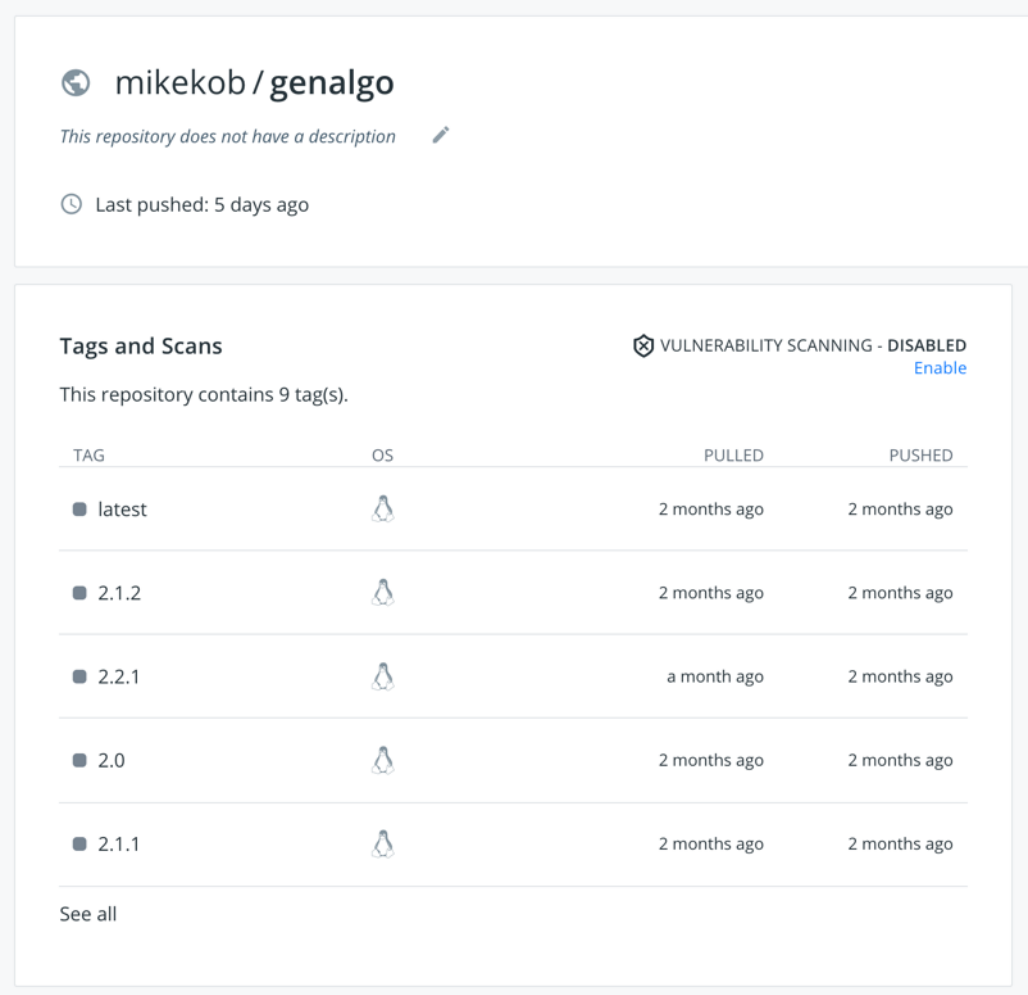


Рисунок 2.3 Образи збережені на DockerHub

Далі на будь якій машині можна запускати контейнери – оброблювачі задач за допомогою команди `docker run <tag>`, вказавши необхідні env vars. На приклад, щоб запустити контейнер для оброки задач, достатньо запустити команду:

```
docker run --env DB_USER=… --env DB_NAME=… --env DB_PASSWORD=… --env DB_HOST=… -d mikekob/genalgo:3.3 python run.py process_name
```

Розділ 3. Експериментальне налаштування Pmax

3.1 Умови проведення експериментів

Для тестування значень Pmax ми обрали такі функції:

- Сферична функція (тестова функція Де Йонга)
- Функція x^4
- Функція спадаючих максимумів (тестова функція Деба 2)
- Функція нерівномірних спадаючих максимумів (тестова функція Деба 4)
- Функція шляху Еклі
- Функція Растригіна

Формули та графіки функцій можна знайти в додатку В.

Всі функції мають векторизовану програмну реалізацію з використанням бібліотеки **numpy**. Тобто на вхід подається масив чисел (аргументів), будь-якої розмірності, на вихід – одновимірний масив чисел – значень функції. Переглянути реалізації можна в додатку Д.

Для проведення цієї серії експериментів ми зафіксували такі параметри:

- Максимальна кількість ітерацій MAX_ITER = 20000
- Кількість ітерацій з незмінним здоров’ям, необхідна для збіжності STEPS_BACK = 10
- Максимальна різниця здоров’я поточної ітерації з минулою допустима для збіжності EPS = 0.0001
- Кількість прогонів при тестуванні NUM_PROGONS = 10
- Кількість кроків алгоритму пошуку Pmax NUM_RUNSETS = 15
- Спосіб ініціалізації INIT = uniform (випадковий за рівномірним розподілом)

Інші параметри могли приймати такі значення:

- Розв’язки кодувались вузлами дискретизації з точністю q = 2, 3 (знаків після коми). Використовували бінарне кодування та коди Грея.
- Окрім одновимірного випадку, експерименти проведено з n = 2, 3, 4, 5 для окремих функцій.
- Розмір популяції складав N = 100, 200, 400.
- Розмір особини вираховувався з параметрів кодування (з інтервалу, точності та розмірності) і складав L = 10, 20, 30, 40, 50, 60, 80, 100 для різних випадків.
- Тип відбору – рулетка, турнір (з параметрами 2,4,12) та універсальний стохастичний відбір, sel_type = rws, tournament, sus.

Детальніше набір параметрів цієї серії експериментів, а також підібрані значення Pmax, можна знайти в додатку Е.

3.2 Результати експериментів

3.2.1 Вплив розміру популяції та розмірності задачі

Розглянемо графіки залежностей Pmax від довжини особини та розміру популяції за умов зазначених в 3.1.

Зобразимо залежність Pmax від L за допомогою запиту:

```
SELECT L,  Pmax
FROM values
WHERE {condition}
```

де `{condition}` – наведене в заголовку кожного графіку, а `values` – вміст додатку Е.

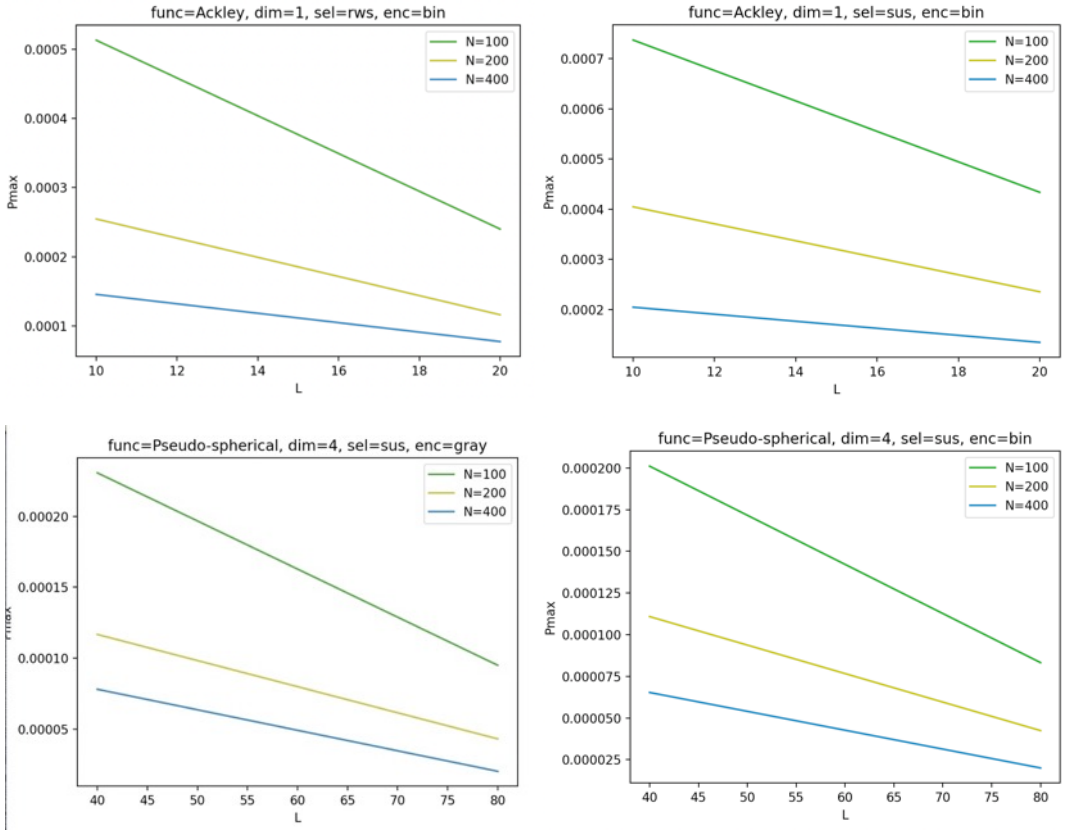


Рисунок 3.1 Графіки залежностей Pmax від L для різних значень N.

Залежність Pmax від N:

```
SELECT N,  Pmax
FROM values
WHERE {condition}
```

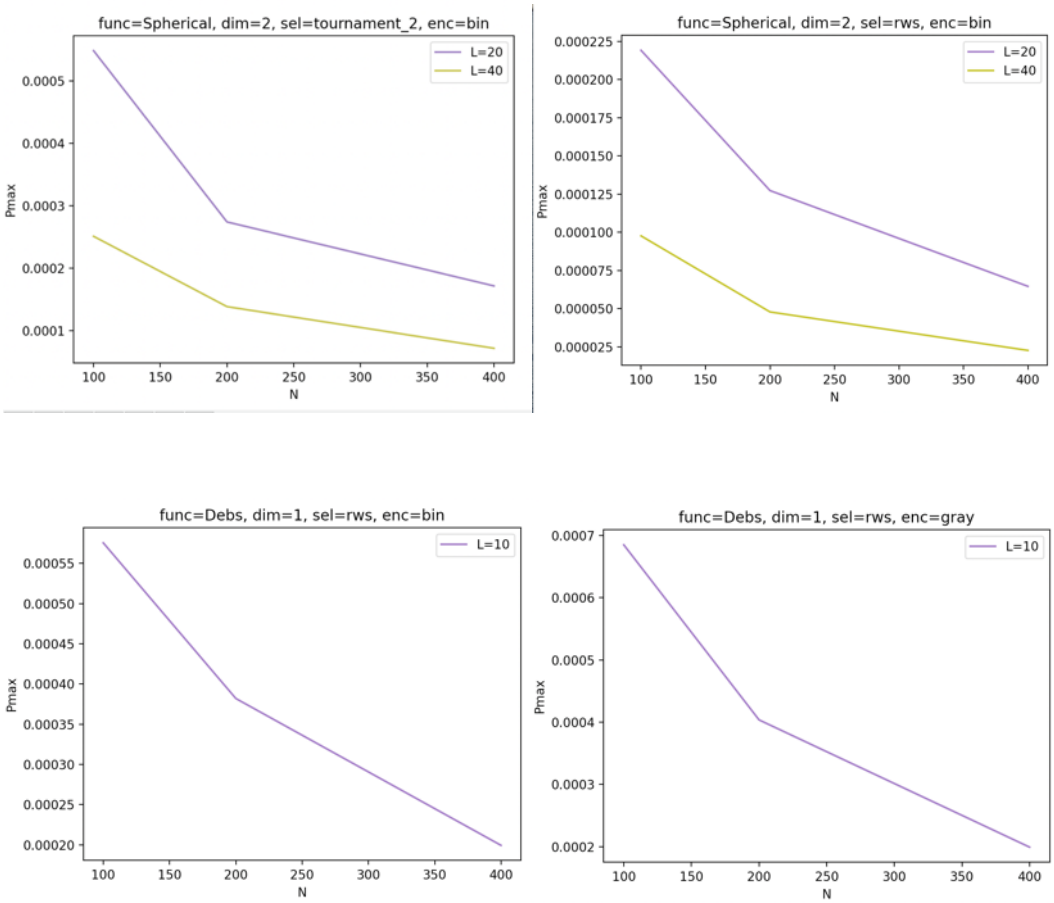


Рисунок 3.2 Графіки залежностей Pmax від N для різних значень L.

Бачимо з графіків, що залежності зберігаються за умов зазначених в 3.1. Так само зберігається залежність Pmax від L * N, що має зворотно пропорційну природу:

```
SELECT L*N,  Pmax
FROM values
WHERE {condition}
```

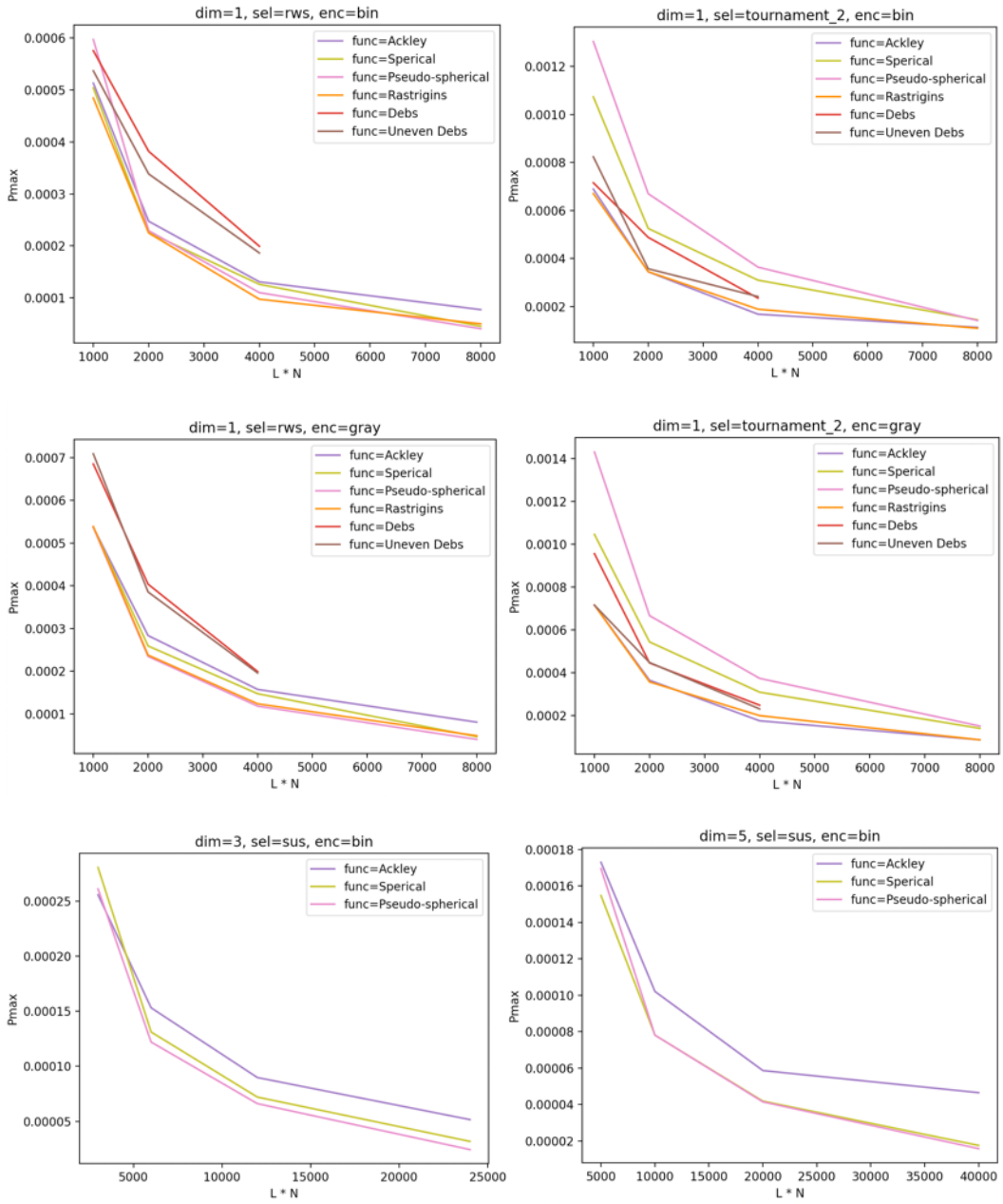


*Рисунок 3.3 Графіки залежностей Pmax від L*N для різних значень тестових функцій.*

Якщо зобразимо пари `(1/(L*N), Pmax)` точками, то отримуємо графік прямої:

```
SELECT 1/ (L*N),  Pmax
FROM values
WHERE {condition}
```

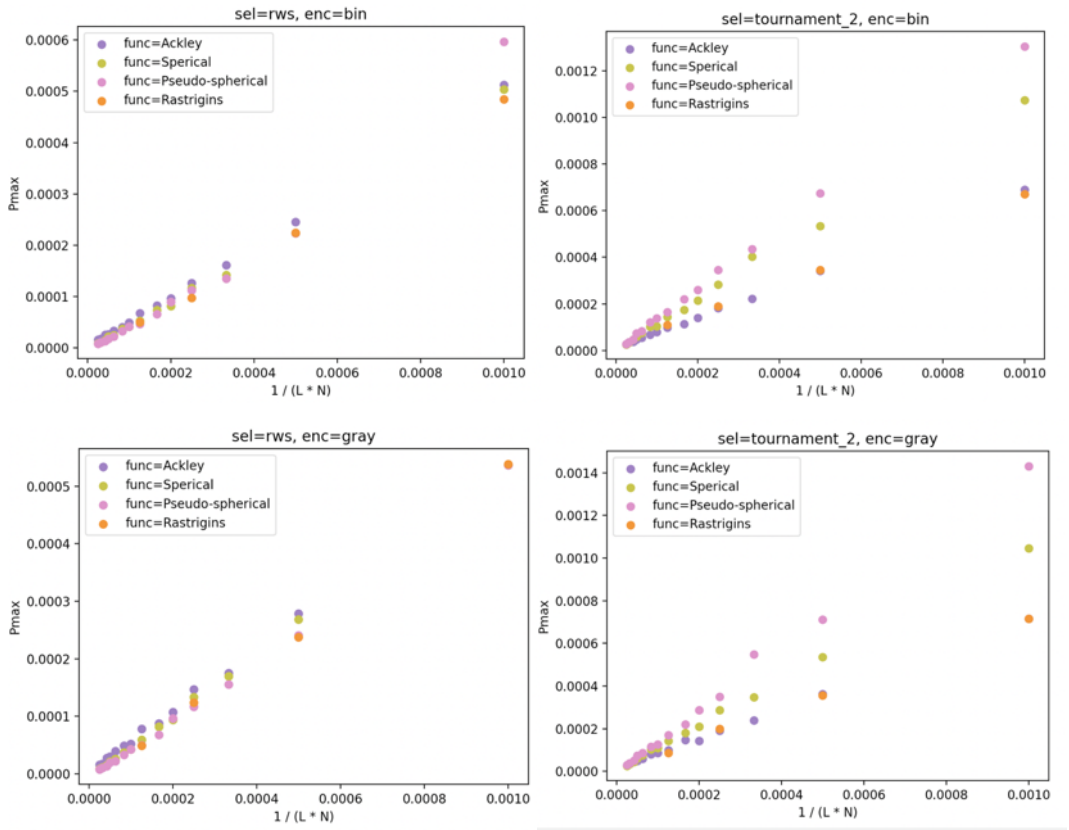


*Рисунок 3.3 Графіки залежностей Pmax від 1/(L*N) для різних тестових функцій*

Також бачимо, що нахил прямої сильно залежить також від тестової функції.

З цього можна зробити висновок, що природа залежності значення Pmax від значень L * N зберігається, проте як зазначено у пункті 1.3 параметри залежності для різних функцій різні, що підтверджується експериментами.

Зі 160 значень Pmax, що були підібрані в [1], лише 47 (29%) пройшли тестування для нових функцій. В інших 113 (71%) експериментах, потрібно було підбирати нове значення Pmax для поточної функції. Нове підібране значення відрізнялось від початкового в середньому на 41%.

Проте оскільки цього разу умови дещо змінились (зокрема експерименти з N=400 в [1] не проводились), то не для всіх випадків можна було взяти вже підібране значення Pmax. Для таких випадків дані

виводились із лінійних рівнянь залежностей знайдених в [1] та зазначених в 1.2. Якщо брати до уваги і ці значення також, то отримаємо таку статистику: 75% (181 з 242) не пройшли тестування, в середньому значення відрізняються на 62%.

3.2.2 Вплив розмірності задачі

В експериментах ми тестували функції, що мають як одновимірний, так і багатовимірний випадки. Щоб закодувати багатовимірний розв'язок $(x_1, x_2, \dots, x_n)$, ми кодували традиційним методом (вузлами дискретизації) кожен x_i і потім «склеювали» їх разом в одну особину, тобто довжина такої особини складала $L' = L * n$.

Виходячи із сильної зворотно пропорційної залежності значення Pmax від довжини особини, можна зробити припущення, що підібравши значення Pmax для розмірності задачі n = 1, ми зможемо вивести $P'_{\max}$ для n > 1. Таким чином, маючи

$$\mathrm{P}_{\max} = \frac{a}{L * N}, \qquad a \in \mathbb{R}$$

та

$$P'_{\max} = \frac{a}{L' * N} = \frac{a}{L * n * N}, \qquad a \in \mathbb{R}$$

Виходить:

$$P'_{\max} = \frac{P_{max}}{n}$$

Зауважимо, що базуючись на результатах експериментів, ми припускаємо, що коефіцієнт a однаковий для однакових умов (тестової функції, кодування, способу відбору та ініціалізації).

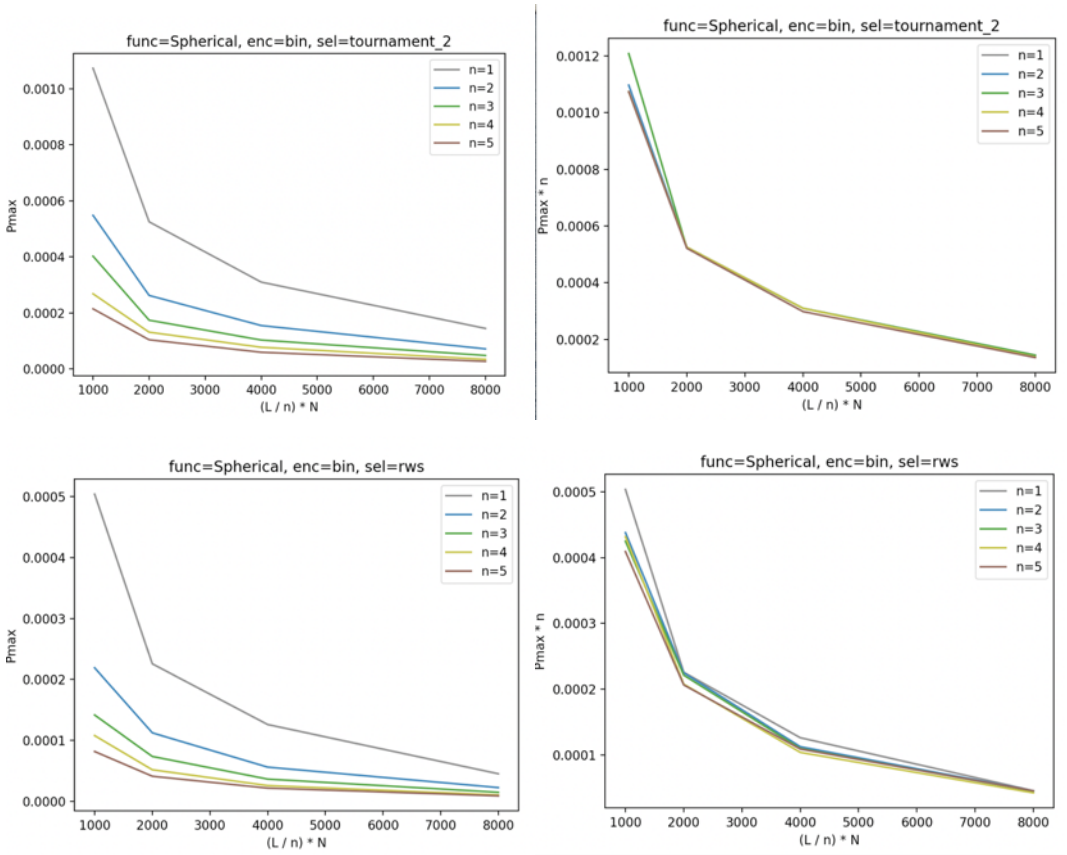
Щоб перевірити це припущення, проведемо такі експерименти: для функції Еклі, сферичної та функції x^4 візьмемо значення вже підібране в

однакових умовах для розмірності n=1. Запустимо алгоритм, описаний в пункті 2.1 для розмірностей $n \in \{2,3,4,5\}$, взявши за початкову точку підібране Pmax поділене на n.

Набір параметрів та результати цих експериментів можна знайти в додатку Е, вибравши рядки з dim_n > 1.

Маємо такі результати для сферичної функції та різних типів відбору:

```
SELECT (L/dim_n) * N,  Pmax, Pmax * n
FROM values
WHERE {condition}
```



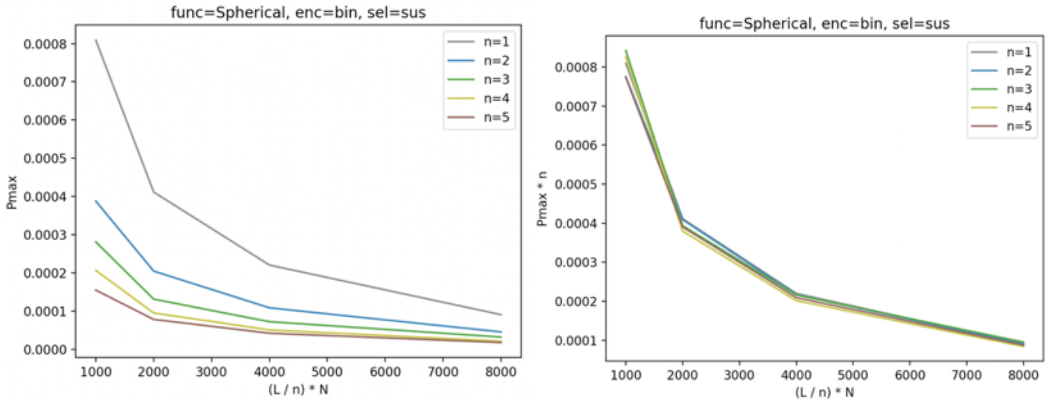
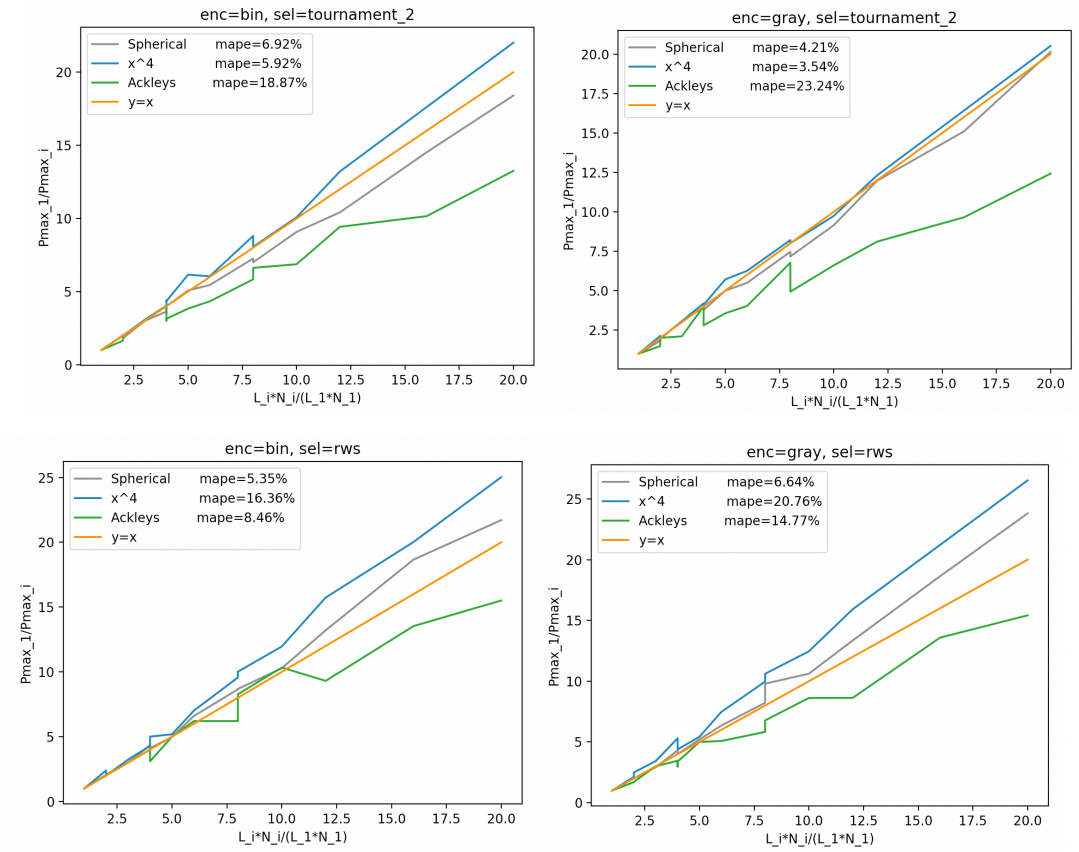


*Рисунок 3.4 Графіки залежностей Pmax від (L/dim_n)*N – зліва, та Pmax*n від (L/dim_n) * N – справа.*

Також обрахуємо похибки. Для цього побудуємо такий графік: по осі Ox маємо $\frac{L_n N_n}{L_1 N_1}$, а на осі Oy маємо $\frac{Pmax_1}{Pmax_n}$ для $\forall n \in \{1,2,3,4,5\}$. SQL-запит наведено в додатку Ж.

Порівняємо такий графік з $y = x$ та порахуємо відхилення MAPE.

$$MAPE(y, y') = \frac{1}{n}\sum\left(\frac{|y - y'|}{y} * 100\%\right)$$



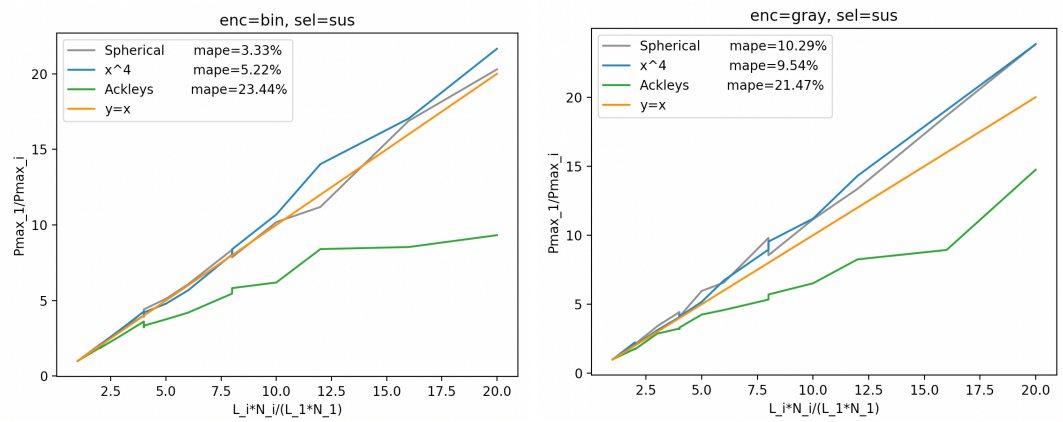


Рисунок 3.5 Графіки залежностей $\frac{Pmax_1}{Pmax_n}$ *від* $\frac{L_nN_n}{L_1N_1}$ *для різних тестових функцій.*

Бачимо, що для сферичної функції значення MAPE не перевищує 10%, для функції x^4 бачимо велику похибку для рулетки, а для функції Еклі на всіх графіках похибку 10-20%.

По всім наборам функції, точності, методу відбору та типу кодування середнє MAPE складає **11.3%** .

Отже, ці графіки показують, що дійсно підхід, коли для конкретної задачі ми спершу підбираємо оптимальне значення Pmax на малих розмірностях, а потім використовуємо це значення для обрахунку оптимального Pmax для більших розмірностей, може давати досить гарні результати.

3.2.3 Вплив методу кодування

Також є припущення, що спосіб кодування може впливати на час до збіжності популяції. Зокрема, існують твердження [2], що з кодуванням Грея популяція досягає збіжності за меншу кількість ітерацій.

Розглянемо графіки залежності Pmax від L*N для обох способів кодування:

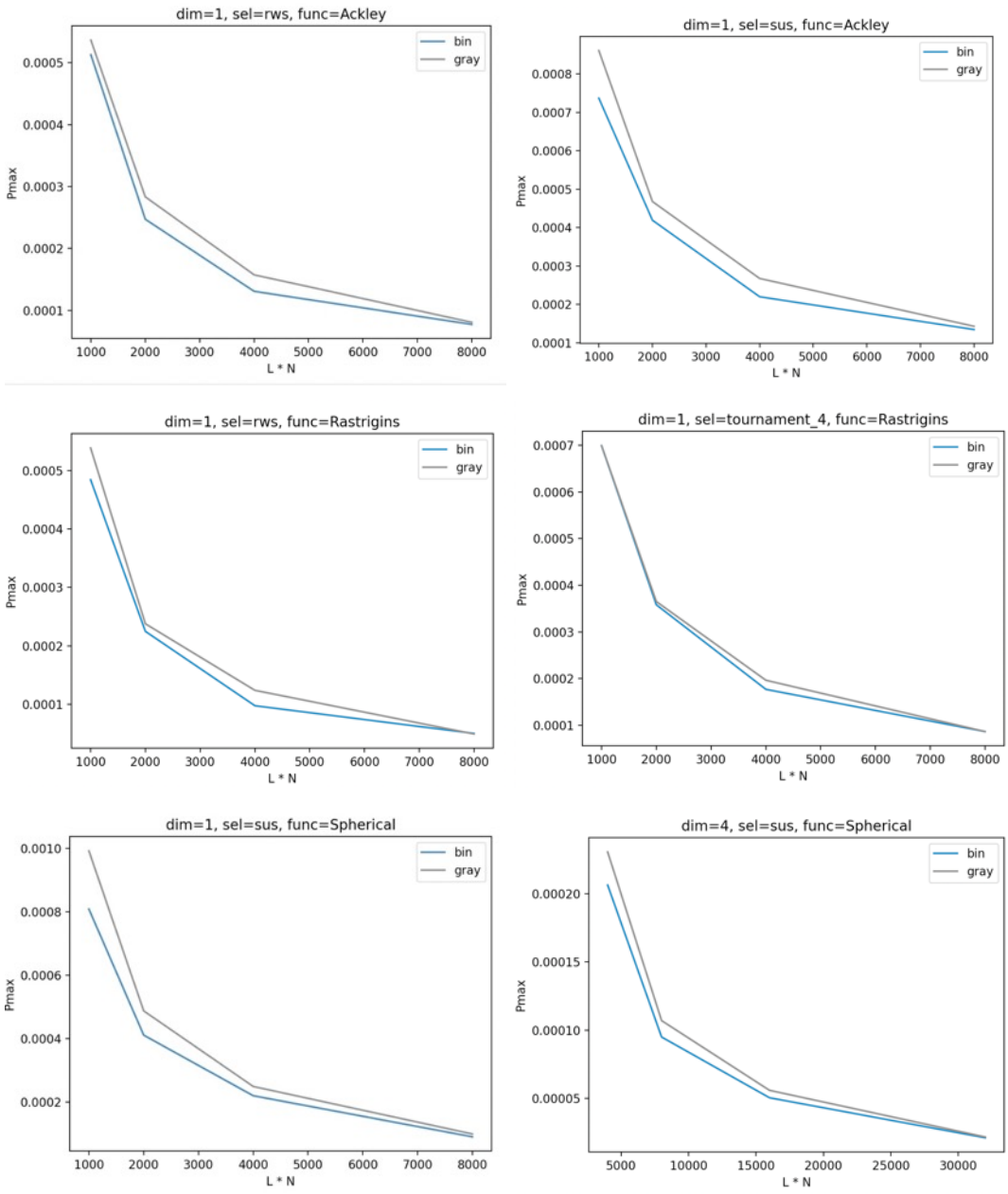


*Рисунок 3.6 Графіки залежностей Pmax від L * N для бінарного кодування та кодів Грея.*

Бачимо, що графік Pmax за кодування Грея лежить вище ніж графік з бінарним кодуванням.

Пояснити той факт, що значення Pmax більші в умовах кодування Грея можна тим, що в нашому алгоритмі, ми підбираємо такі значення Pmax, що збіжність за Pmax * 120% вже має втрачатись (популяція може збігатись, але за більше ніж MAX_ITER = 20000 ітерацій). Тобто

збігаючись за таку ж кількість ітерацій, що і в умовах бінарного кодування, популяція в умовах кодування Грея може толерувати більше значення Pmax.

Спробуємо також провести прогони із однаковим значенням Pmax для бінарного кодування та кодів Грея. Зафіксуємо:

- Функція: сферична, n=3, точність = 3 знаки;
- L = 60;
- N = 200;
- Ініціалізація: uniform;
- Відбір: sus;
- Pmax: 0.0000586

Проведемо по 100 прогонів за таких умов для бінарного кодування та кодування Грея. Результати цього експерименту можна переглянути в додатку К. Отримуємо такі результати:

- З бінарним кодуванням
  Середня збіжність: **6814.57** ітерацій
  Медіана: **5875** ітерацій

- З кодуванням Грея:
  Середня збіжність: **5315.43** ітерацій
  Медіана: **4568** ітерацій

Побудуємо гістограму – розподіл кількості ітерацій до збіжності:

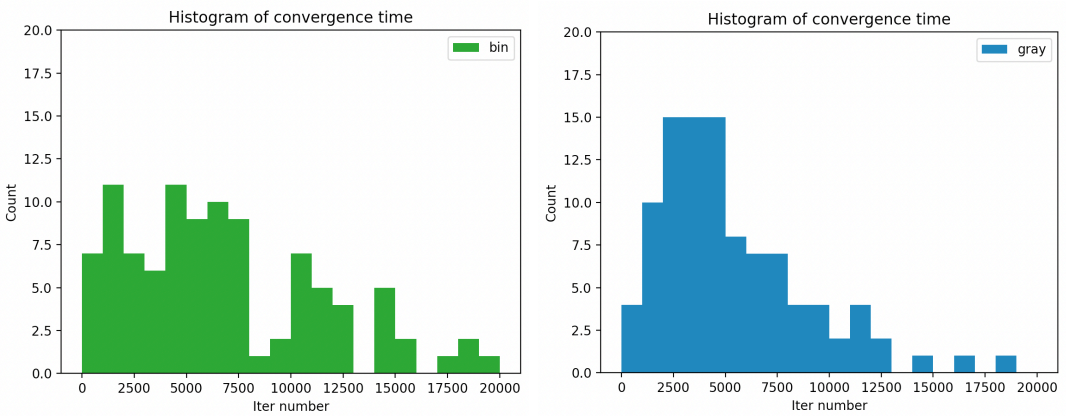


Рисунок 3.7 Гістограма по кількості ітерацій до збіжності для бінарного кодування та кодів Грея.

Зобразимо ці ж дані на одному графіку, з'єднавши точки – вершини стовпців:

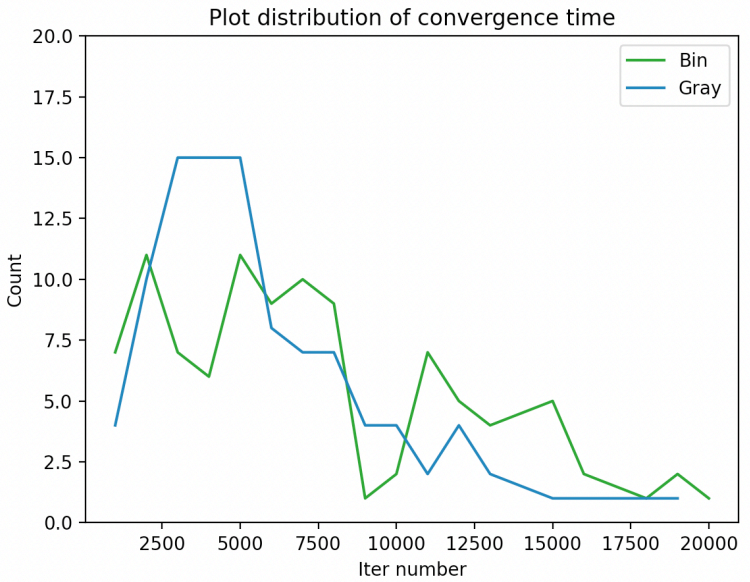


Рисунок 3.8 Полігон частот по кількості ітерацій до збіжності для бінарного кодування та кодів Грея.

А також діаграму кумулятивної суми:

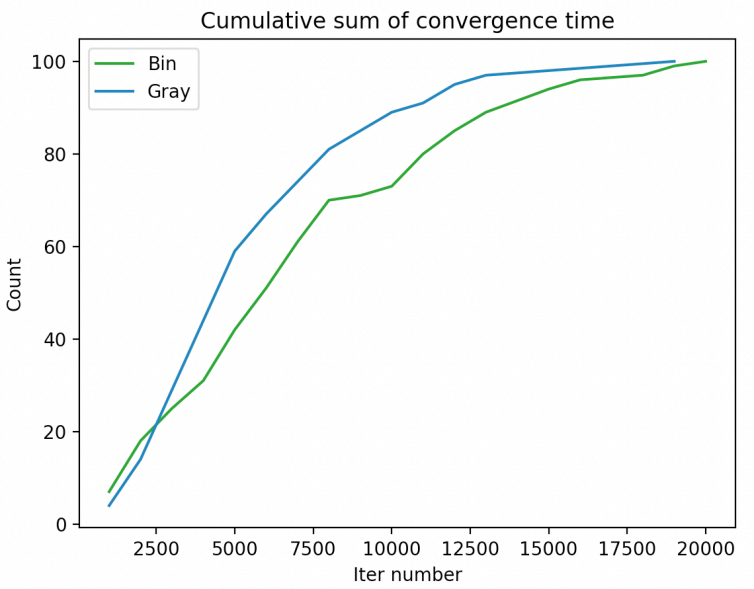


Рисунок 3.7 Кумулятивна сума кількості ітерацій до збіжності для бінарного кодування та кодів Грея.

Отже, з графіків видно, що припущення має місце, і дійсно, використовуючи кодування Грея популяція в середньому швидше досягає збіжності.

Зробимо ще одну перевірку. Під час досліджень, було згенеровано багато наборів RunSet (2.2), зокрема з однаковими умовами та однаковими значеннями Px. Ми можемо порівняти швидкість збіжності та точність алгоритму за різних методів кодування з цих даних.

Зафіксуємо тестову функцію, метод відбору, L та N. Для однакових значень Px порівняємо середню кількість ітерацій до збіжності, а також найкраще здоров'я (знайдених розв'язок) окремо для бінарного кодування та кодів Грея.

Для прикладу розглянемо функцію Деба-4, відбір рулетка:

L	N	Px	Final Pmax (bin)	Final Pmax (gray)	Avg iters (bin)	Avg iters (gray)	Extremum	Best health (bin)	Best health (gray)
10	200	0.000254576	0.000338409	0.000385496	2468.9	921.8	0.9998	0.9998	0.9998
10	400	0.000099674	0.000186445	0.000195838	540.6	583.7	0.9998	0.9998	0.9998

10	400	0.000149511	0.000186445	0.000195838	1393.7	773.1	0.9998	0.9998	0.9998
10	400	0.000174430	0.000186445	0.000195838	2385.5	2189.1	0.9998	0.9998	0.9998
				Avg:	1697.175	1116.925			

- Final Pmax – значення, яке ми врешті отримали;
- Avg iters – середня кількість ітерацій до збіжності;
- Extremum - максимально можливе значення, яке може прийняти функція за таких умов та на такому відрізку.

Усі значення можна отримати в додатку Л.

Агреговані метрики по всім значенням:

- Середня кількість ітерацій по всім значенням:
  bin: **2439.16**
  gray: **2003.99**
- Середнє відхилення від екстремуму:
  bin: 0.018%
  gray: 0.0009%

Отже, бачимо, що дійсно середня кількість ітерацій залежить від методу відбору, а саме для кодів Грея вона є дещо меншою. Важко зробити висновки щодо різниці в точності алгоритму, оскільки за обох методів алгоритм знаходить розв’язок достатньо точно (відхилення від оптимуму < 0.2% та варіюється для різних тестових функцій та методів відбору).

Висновки

Підсумовуючи все вищенаведене, можна зробити наступні висновки.

По перше, бачимо, що спосіб ініціалізації популяції не сильно впливає ані на оптимальне значення мутації Pmax, ані на середню швидкість збіжності популяції. Проте проаналізувавши «крайні» випадки (коли вся популяції знаходиться в найкращій та найгіршій точках) та підхід випадкового розподілу, можемо дійти висновку, що спосіб ініціалізації може вплинути на точність знайденого розв'язку. А тому для більшості задач має сенс використовувати саме випадкову ініціалізацію, бо вона стабільно дає досить точні результати.

По друге, провівши декілька перевірок, ми впевнилися, що спосіб кодування особин має вплив на швидкість до збіжності. А саме, для кодів Грея спостерігали меншу середню кількість ітерацій до збіжності, а відтак і підібрані значення Pmax були більшими. З цього виходить, що алгоритми, які використовують коди Грея, в середньому досягатимуть збіжності дещо швидше.

По третє, хоча ми і спостерігаємо сильну зворотно пропорційну залежність Pmax від розміру популяції і довжини особин, ця залежність веде себе по-різному для різних задач і різних типів функцій. Це означає, що дати точну рекомендацію стосовно значення Pmax для конкретної задачі дуже складно, чи взагалі неможливо.

Проте провівши експерименти на більших розмірностях, ми показали, що значення Pmax для необхідної розмірності можна достатньо точно вирахувати з, на приклад, підібраного Pmax для меншої розмірності цієї ж задачі. Ми показали, що проста модель $Pmax_n = \frac{Pmax_1}{n}$ може давати достатньо непогані результати.

Підсумовуючи все, можемо сказати, що проблема підбору вхідних параметрів ГА надалі залишається досить нетривіальною. Це стосується і параметру ймовірності мутації Pmax, дослідженню якого була присвячена ця робота. Проте якщо взяти до уваги природу залежностей Pmax від інших параметрів, то для кожної конкретної задачі можна підібрати значення, яке буде давати досить точні результати.

Список використаної літератури

1. Кобєлєв М. Д. Експериментальний аналіз впливу параметра мутації на збіжність генетичного алгоритму – Курсова робота 3 р. н. .– Нац. ун-т "Києво-Могилян. акад.", Київ 2020.
2. Глибовець М. М. Еволюційні алгоритми : підручник / М. М. Глибовець, Н. М. Гулаєва ; Нац. ун-т "Києво-Могилян. акад.". - Київ : [НаУКМА], 2013. - 826 с. : іл. - Містить покажчик.
3. Overview — NumPy v1.20 Manual [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/1.20/
4. peewee — peewee 3.14.4 documentation [Електронний ресурс] – Режим доступу до ресурсу: http://docs.peewee-orm.com/en/latest/
5. Docker Pricing & Subscriptions for Individuals & Teams [Електронний ресурс] – Режим доступу до ресурсу: https://www.docker.com/pricing

ДОДАТКИ

Додаток А

(обов’язковий)

Алгоритм підбору Pmax

pm = (початкове значення)
δ = **pm** * 0.5
ПОВТОРЮВАТИ 15 разів
    Провести 10 прогонів для заданого значення **pm**
    ЯКЩО спостерігається збіжність в 100% прогонів,
        ТО збільшити **pm** += **δ**
        ІНАКШЕ зменшити **pm** -= **δ**
    δ *= 0.5
Pmax = (останнє **pm**, за якого відбувалась 100% збіжність)

Додаток Б

(обов’язковий)

Набір параметрів та результатів першої серії

Файл: `dodatok_B_first_series.csv`

Запит: `dodatok_B_query.txt`

Опис стовпців:

- Potential max – максимально можливе значення, що може набути функція за таких умов;
- Convergance_pct – відсоток збіжності (частка прогонів з 10, що збіглись);
- Avg_distance – середня відстань від фінального середнього здоров’я популяції до potential_max;
- Converge_to – масив (з 10 прогонів), для кожного прогону виведено середнє здоров’я популяції на останній ітерації;
- Max_iter, min_iter, avg_iter, median_iter – максимальне, мінімальне, середнє значення кількості ітерацій до збіжності по усіх (10) прогонах та їх медіана відповідно;
- Std_iter – стандартне відхилення кількості ітерацій до збіжності по усіх (10) прогонах;

Додаток В

(обов'язковий)

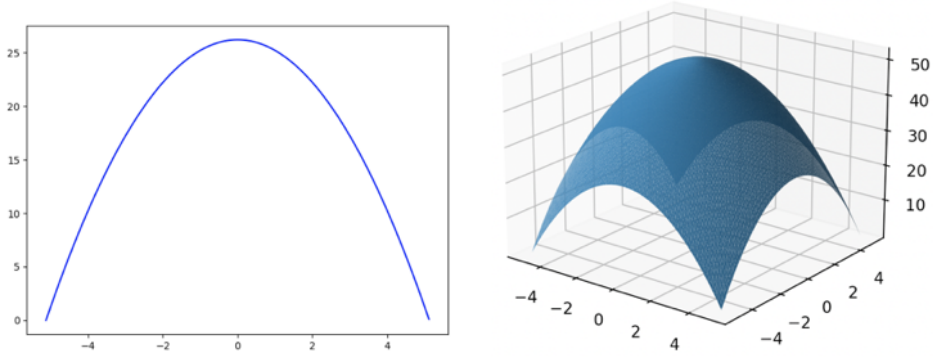
Тестові функції

Тестові функції, використані в роботі. Надано назву, формулу, а також графіки для кожної функції.

- **Сферична функція** (тестова ф-я Де Йонга)

  (на графіках позначається sphercical)

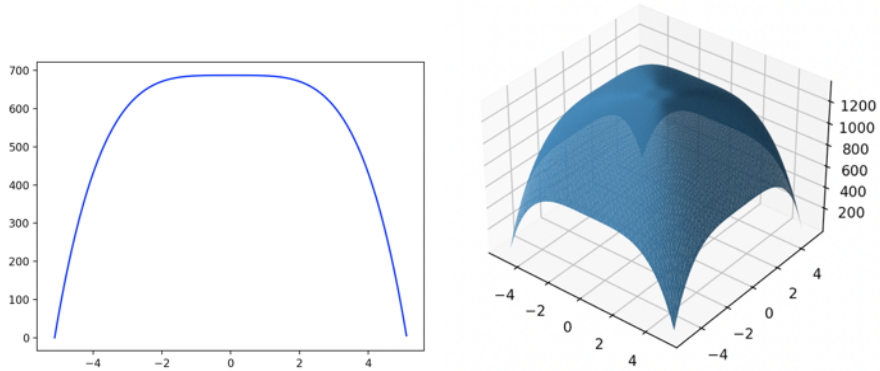
$$f(x_1, \dots, x_n) = (5.12)^2 n - \sum_{i=1}^{n} x_i^2$$



- **Функція x^4**

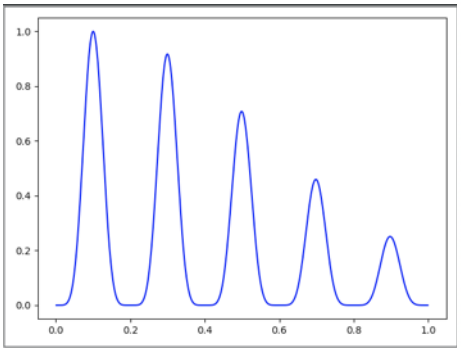
  (на графіках позначається almost_sphercical)

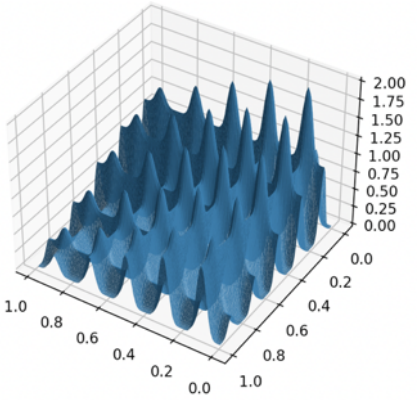
$$f(x_1, \dots, x_n) = (5.12)^4 n - \sum_{i=1}^{n} x_i^4$$



- **Функція спадаючих максимумів** (тестова функція Деба 2)
  (на графіках позначається debs)

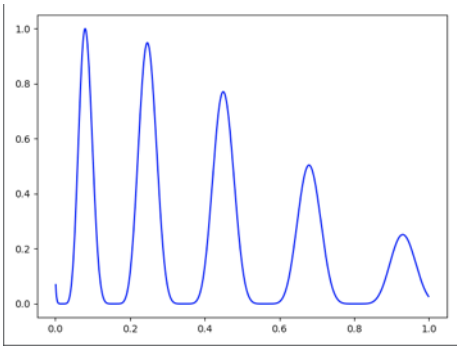
$$f(x_1, \dots, x_n) = \sum_{i=1}^{n} e^{-2(ln2)\left(\frac{x_i - 0.1}{0.8}\right)^2} * \sin^6(5\pi x)$$

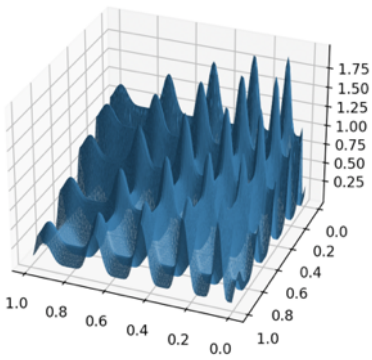




- **Функція нерівномірних спадаючих максимумів** (тестова функція Деба 4)
  (на графіках позначається uneven_debs)

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n} e^{-2(ln2)\left(\frac{x_i - 0.08}{0.854}\right)^2} * \sin^6\left(5\pi\left(x_i^{0.75} - 0.05\right)\right)$$

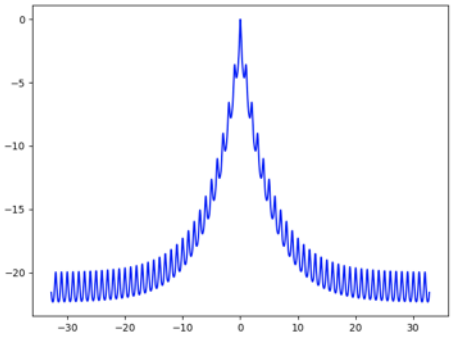
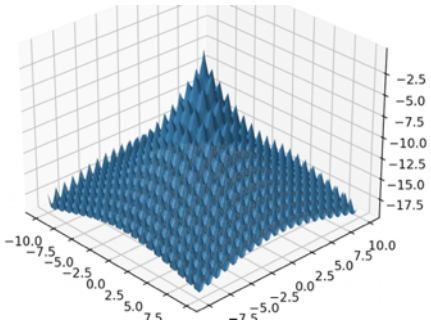




- **Функція шляху Еклі**

  (на графіках позначається ackleys)

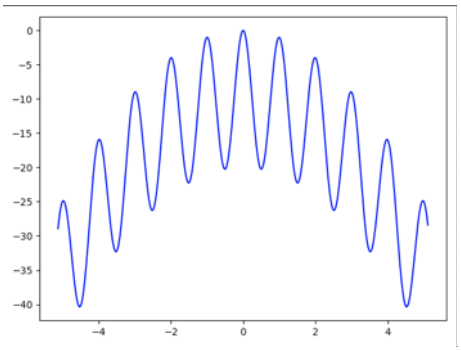
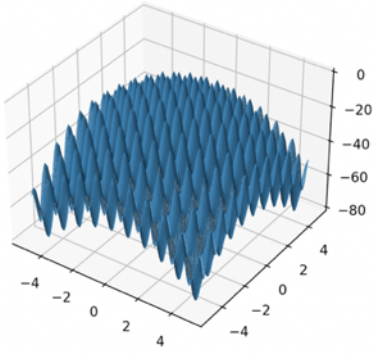
$$f(x_1, \dots, x_n) = 20 * \exp\left(-0.2 * \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) + \exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) - 20 - e$$

- **Функція Растригіна**

  (на графіках позначається rastrigins)

$$f(x_1, \dots, x_n) = \left(\sum_{i=1}^{n} 10\cos(2\pi x_i) - x^2\right) - 10n$$

Додаток Г

(довідниковий)

R-модель бази даних

Файл: `dodatok_G_R_diagram.png`

Додаток Д

(обов'язковий)

Програмна реалізація тестових функцій на numpy

```
import numpy as np


def ackleys_function(pop_decoded: np.ndarray, *args, **kwargs) -> np.ndarray:
    n = pop_decoded.shape[1]
    x_expr_1 = -0.2 * np.sqrt((pop_decoded ** 2).sum(axis=1) / n)
    x_expr_2 = np.cos(2 * np.pi * pop_decoded).sum(axis=1) / n

    return 20 * np.exp(x_expr_1) + np.exp(x_expr_2) - 20 - np.e


def spherical_function(pop_decoded: np.ndarray, a: float, *args, **kwargs) -> np.ndarray:
    n = pop_decoded.shape[1]
    return a * a * n - (pop_decoded ** 2).sum(axis=1)


def almost_spherical_function(pop_decoded: np.ndarray, a: float, *args, **kwargs) -> np.n
    n = pop_decoded.shape[1]
    return a * a * a * a * n - (pop_decoded ** 4).sum(axis=1)


def debs_function(pop_decoded: np.ndarray, *args, **kwargs) -> np.ndarray:
    n = pop_decoded.shape[1]
    x_expr = ((pop_decoded - 0.1) / 0.8) ** 2
    sum_expr = np.exp(-2 * np.log(2) * x_expr) * (np.sin(5 * np.pi * pop_decoded)) ** 6
    return sum_expr.sum(axis=1)


def uneven_debs_function_n(pop_decoded: np.ndarray, *args, **kwargs) -> np.ndarray:
    n = pop_decoded.shape[1]
    x_expr = ((pop_decoded-0.08)/0.854) ** 2
    sin_part = (np.sin(5 * np.pi * (pop_decoded ** 0.75 - 0.05))) ** 6
    sum_part = np.exp(-2 * np.log(2) * x_expr) * sin_part
    return sum_part.sum(axis=1)


def rastrigins_function(pop_decoded: np.ndarray, *args, **kwargs) -> np.ndarray:
    n = pop_decoded.shape[1]
    sum_expr = 10 * np.cos(2 * np.pi * pop_decoded) - pop_decoded**2
    return sum_expr.sum(axis=1) - 10 * n
```

Додаток Е

(обов’язковий)

Набір параметрів та результатів 2 серії експериментів

Файл: `dodatok_E_second_series.csv`

Запит: `dodatok_E_query.txt`

Опис стовпців:

- F_alias – назва тестової функції (Додаток В);
- Dim_n – розмірність;
- Accuracy_decimals – точність (кількість знаків після коми), з якою ми кодуємо особини;
- [interval_a, interval_b] – інтервал, на якому ми кодуємо особини, та відповідно шукаємо екстремум.

Додаток Ж

(обов'язковий)

SQL-запит для порівняння значень Pmax для одновимірного та багатовимірного випадків тестової функції

```
CREATE VIEW full_exp_param AS
(
SELECT f3.alias                        AS f_alias
     , f2.interval_a                   AS interval_a
     , f2.interval_b                   AS interval_b
     , f2.dim_n                        AS dim_n
     , f2.accuracy_decimals            AS accuracy_decimals
     , f.extremums
     , f.adj_extremums
     , CAST(f.extremums AS text)       AS extremums_txt
     , CAST(f.adj_extremums AS text)   AS adj_extremums_txt
     , f.extremum_val
     , f.adj_extremum_val
     , paramset.*
FROM paramset
         INNER JOIN funccase f ON f.id = paramset.func_case_id
         INNER JOIN funcparam f2 ON f2.id = f.func_param_id
         INNER JOIN function f3 ON f3.id = f.function_id
    );



SELECT t2."L" * t2."N" / (t1."L" * t1."N")
     , t1.result_pmax / t2.result_pmax
     , t2."L" * t2."N" / (t1."L" * t1."N")
FROM full_exp_param t1
     INNER JOIN full_exp_param t2 ON t1.f_alias = t2.f_alias
               AND t1.accuracy_decimals = t2.accuracy_decimals
               AND t1.sel_type = t2.sel_type
               AND t1.init = t2.init
               AND t1.encoding ->> 'type' = t2.encoding ->> 'type'
WHERE t1.f_alias = '{func_alias}'
  AND t1.sel_type = '{sel_type}'
  AND t1."encoding" ->> 'type' = '{enc}'
  AND t1."N" = 100
  AND t1.accuracy_decimals = 3
  AND t1."dim_n" = 1
  AND t1.result_pmax IS NOT NULL
  AND t2.result_pmax IS NOT NULL
ORDER BY 1;
```

Додаток К

(обов’язковий)

Результат експерименту з бінарним кодуванням та кодами Грея

Файл: `dodatok_K_bin_gray.csv`

Опис стовпців:

- F_alias – назва тестової функції (Додаток В);
- Dim_n – розмірність;
- Accuracy_decimals – точність (кількість знаків після коми), з якою ми кодуємо особини;
- Pmax– значення мутації, з яким проводився експеримент;
- Iter_num – кількість ітерацій до збіжності;
- Max_health – знайдений розв’язок;
- Extremum – максимально можливе значення, яке може набути функція за таких умов;
- Diff_pct – різниця у відсотках між знайденим розв’язком та оптимумом.

Додаток Л

(обов’язковий)

Результати порівняння однакових Px для бінарного кодування та кодів Грея

Файл: `dodatok_L_bin_gray.csv`

Запит: `dodatok_L_query.txt`

Опис стовпців:

- F_alias – назва тестової функції (Додаток В);
- Bin_res_pmax, gray_res_pmax – значення Pmax, яке ми зрештою отримали;
- Bin_avg, gray_avg – середня кількість ітерацій до збіжності;
- Potential_max - максимально можливе значення, яке може прийняти функція за таких умов та на такому відрізку.
- Bin_solution, gray_solution – знайдений розв’язок;
- Bin_diff_pct, gray_diff_pct – різниця у % між знайденим розв’язком та оптимумом.