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

Експериментальний аналіз впливу параметра мутації на збіжність 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-3
Кобєлєв М. Д.
"___" _____ 2020 року

Київ 2020

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

ЗАТВЕРДЖУЮ
Викладач кафедри інформатики,
к-т фіз.-мат. наук, доцент
_____ Гулаєва Н.М.

„____” _____ 2019р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу
студенту Кобєлєву Михайлу Дмитровичу
факультету інформатики 3 курсу бакалаврської програми

ТЕМА: **Експериментальний аналіз впливу параметра мутації на збіжність генетичного алгоритму**

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
Розділ 1. Розробка алгоритму пошуку Pmax.
Розділ 2. Проведення обчислювальних експериментів.
Розділ 3. Аналіз результатів експериментів.
Висновки.
Список літератури
Додатки (за необхідністю)

Дата видачі „___” _____ 2019 р. Керівник_____
(підпис)
Завдання отримав _____
(підпис)

Тема: **Експериментальний аналіз впливу параметра мутації на збіжність генетичного алгоритму**

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	1.10.2019	
2.	Огляд літератури за темою роботи.	31.12.2019	
3.	Написання програмного коду для проведення досліджень.	31.01.2020	
4.	Проведення обчислювальних експериментів.	1.03.2020	
5.	Аналіз отриманих результатів експериментів з керівником	15.03.2020	
6.	Коригування роботи та проведення додаткових експериментів.	31.03.2020	
7.	Повний аналіз результатів експериментів.	15.04.2020	
8.	Написання пояснювальної роботи.	31.04.2020	
9.	Оформлення слайдів для доповіді.	11.05.2020	
10.	Захист курсової роботи	18.05.2020	

Студент _____

Керівник _____

“_____”_____

ЗМІСТ

Анотація

В роботі розроблено та реалізовано алгоритм пошуку максимального граничного значення ймовірності мутації, за якого ще відбувається збіжність у генетичному алгоритмі. Проведено ряд експериментів з різними значеннями параметрів генетичного алгоритму. Подається аналіз впливу цих параметрів на граничне значення ймовірності мутації. Зокрема, наводиться аналіз за допомогою методів машинного навчання з метою вивести залежність значення ймовірності мутації від деяких вхідних параметрів генетичного алгоритму.

Ключові слова:

Генетичні алгоритми, мутація в генетичних алгоритмах, машинне навчання, лінійна регресія.

Вступ

Одною з найважливіший проблем, пов’язаних із генетичними алгоритмами, є проблема вибору вхідних параметрів алгоритму. Від вдалого набору параметрів може суттєво змінюватися швидкість роботи алгоритму та якість знайдених розв’язків.

До складу цих параметрів, зокрема, входить параметр ймовірності мутації, ретельному дослідженню якого і присвячена ця робота.

В цій роботі розглядається тільки щільнісна мутація, тобто коли кожен ген хромосоми мутує із заданою ймовірністю[1]. Цей тип мутації є найближчим аналогом до мутаційних процесів живої природи. Можна припустити, що існує максимальне граничне значення ймоіврності мутації, при перевищенні якого відбувається руйнування інформації, а в термінах ГА не відбувається збіжність.

Метою роботи є пошук максимального значення ймовірності щільнісної мутації (Pmax), за якої спостерігається збіжність ГА.

Для досягнення мети були виділені задачі:

1. Реалізація генетичного алгоритму із можливістю зміни вхідних параметрів.
2. Проведення низки експериментів з пошуку Pmax з різними наборами параметрів, зокрема з різними:
   - довжиною хромосоми;
   - розміром популяції;
   - методами відбору (рулетка, турнір);
   - способами ініціалізації (рівномірний/нормальний розподіли, нулі у всіх локусах)

- функціями оцінювання.

3. Перевірка гіпотези існування Pmax.
4. У разі істинності гіпотези, аналіз впливу різних параметрів ГА на значення Pmax.

Актуальність теми:

На цей час інформації щодо комплексного аналізу ймовірності мутації немає. Результати, отримані в ході роботи, дозволять робити припущення щодо значення ймовірності мутації в залежності від інших параметрів генетичного алгоритму.

Об'єкт дослідження:

Збіжність генетичного алгоритму.

Предмет дослідження:

Вплив параметра «ймовірність мутації» на збіжність генетичного алгоритму.

Методи дослідження:

Експериментальний метод: згенеровано набори значень параметру мутації, за яких відбувається або не відбувається збіжність ГА.

Регресійний метод аналізу залежності граничного значення ймовірності мутації Pmax від довжини хромосоми та розміру популяції.

Структура роботи:

Робота має три розділи:

1) В першому розділі описується генетичний алгоритм і визначаються початкові параметри, на яких проводилось дослідження. Також описується алгоритм пошуку значення Pmax (значення ймовірності мутації), а також наведено спосіб реалізації цього алгоритму. Також в першому розділі приділено увагу збереженню даних експериментів. Зазначаються труднощі, які виникли під час роботи, а також шляхи їх подолання.
2) В другому розділі описується план проведення експериментів.
3) Третій розділ – аналіз результатів. В цьому розділі накопичені результати експериментів використано для аналізу залежності значення Pmax від значень вхідних параметрів. Детально розглядається вплив кожного параметра, а також зроблено регресійних аналіз залежності Pmax від довжини хромосоми L та кількості особин в популяції N.

Розділ 1. Розробка алгоритму пошуку Pmax

1.1 Опис генетичного алгоритму та його параметрів

В рамках цього дослідження був використаний такий варіант генетичного алгоритму:

1) Ініціалізація – генеруємо популяцію бінарних ланюжків з N особин.
2) Оцінювання – оцінюємо кожну особину.
3) Відбір – відбираємо N особин в батьківських пул.
4) Мутація – щільнісна, кожен ген мутує із заданою ймовірністю.
5) Якщо не виконана умова зупинки – переходимо на крок 2
6) Завершення

Надалі у роботі маються на увазі такі незмінні параметри:

- **Кодування** – двійкове;
- **Збіжність** – говоримо, що алгоритм збігається, якщо середнє здоров'я популяції протягом останніх S кроків не змінюється більше, ніж на константу EPS. В рамках роботи: S=10 і EPS=0.0001;
- **Умова зупинки** – алгоритм збігся (див. вище) або кількість ітерацій більша за MAX_ITER. В рамках роботи: MAX_ITER=20000.

Також фігурують такі параметри та можливі їх значення:

Параметр	Діапазон значень
init - Ініціалізація	1) **all_0** – значення у всіх локусах – 0; 2) **normal** – нормальних розподіл відносно довжини Гемінга до ланцюжка з 0; 3) **uniform** – рівномірний розподіл відносно можливих значень локуса (тобто ймовірність значення «1» – 0.5, і значення «0» – 0.5);

estim – Оцінювання	1) **l-hamming_d** – відстань Гемінга до нульового ланцюжка. 2) **sigma_2**, **sigma_4**, **sigma_10** – оцінювання з селективною перевагою на біт.
type – тип відбору	1) **rws** – відбір за методом рулетки [1] 2) **tournament_2**, **tournament_4**, **tournament_12** – турнірний відбір з параметрами 2, 4, 12 відповідно. [1]
L – довжина генотипу	$L \in \mathbb{N}$
N – кількість особин в популяції	$N \in \mathbb{N}$

Таблиця 1.1 Параметри генетичного алгоритму

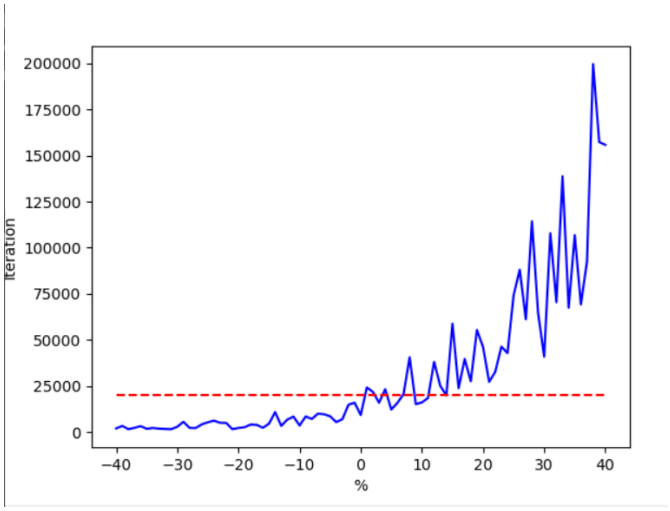
1.2 Алгоритм пошуку Pmax

Для початку, визначимо поняття значення Pmax.

Як було згадано, замале або завелике значення ймовірністі мутації може негативно вплинути на результати роботи алгоритму. Тому, введемо поняття значення Pmax, яке позначає *максимальне значення коефіцієнту застосування параметру мутації (значення ймовірності), за якого ще відбувається збіжність алгоритму в межах 20000 ітерацій*.

Одразу зауважимо, що це значення досить складно підібрати точно, через чутливість алгоритму до значення ймовірності мутації та його часову складність. На наступному графіку показано, як змінюється швидкість збіжності середнього здоров'я при зміні значення Pmax для випадку L=100, N=100. Ми взяли вже підібране раніше значення Pmax (0.0000403) для цього випадку та стали додавати та віднімати по одному відстотку ($4 * 10^{-7}$) та

зберігали номер ітерації на якій досягнуто збіжність. Повторювали по 10 разів для кожного нового значення. Потім брали максимальне значення з прогонів. В загалому розмах в 40 відсотків ($\pm 1.6 * 10^{-5}$).



Графік 1.1. Чутливість середнього часу збіжності до Pmax

Червоною лінія – позначка в 20000 ітерацій.

З графіка бачимо, що навіть дуже незначні зміни Pmax сильно впливають на збіжність алгоритму.

Через таку чутливість до цього значення, у підібрані результати ми закладаємо 10% похибки, щоб алгоритм точно збігся (тобто це означає, що отримане значення треба помножити на 0.9, і вже отриманий результат використовувати у генетичному алгоритмі).

Отже, у роботі використано такий простий алгоритм пошуку значення Pmax:

1) Ініціалізувати значення $P_m = 1 / (50 * L)$; $\delta = P_m * 0.5$;
2) Провести 10 прогонів із заданим P_m;
3) Якщо збіжність в 100% прогонів, то Pm збільшити на δ, інакше – зменшити на δ .
4) δ *= 0.5

5) Повторювати кроки 2 – 4 по 15 разів.
6) Вибрати значення P_m з останнього кроку, коли збіжність була в 100% прогонів – це значення і є шукане значення Pmax.

Для перевірки та коригування отриманого таким шляхом значення, використовуємо наступний алгоритм:

1) Проводимо 10 прогонів із Px - претендентом на значення Pmax;
2) Проводимо 10 прогонів із 1.2 * Px;
3) Проводимо 10 прогонів із 0.8 * Px;
4) Якщо збіжність у кроках 1 або 3 не є 100% - значення Px завелике для Pmax, його треба зменшити на 5-10% і спробувати знову;
5) Якщо збіжність у кроці 2 є 100% - значення Px замале для Pmax, його треба збільшити на 5-10% і спробувати знову.

1.3 Реалізація алгоритму

1.3.1 Особливості реалізації мовою Python

В останні роки Python набула великої популярності серед представників галузі Data Science через зрозумілість мови та наявність великої кількості пакетів для оптимізованої роботи з великими даними та їх візуалізації, таких як numpy, matplotlib, pandas, sklearn.

Окрім цього, Google безкоштовно надає користування платформою Google Colab, що створена для всіх, хто вивчає data science. На цій платформі можна запускати Jupyter Notebooks.

Саме з цих причин для реалізації завдань роботи була вибрана мова Python. Проте з цим вибором були пов’язані деякі труднощі.

Всі базові операції треба було реалізувати за допомогою матриць та діями над матрицями. Проблема в тому, що Python – інтерпретована мова, а значить по швидкодії дуже сильно уступає компільованим (C++). Через це треба було по максимуму використовувати модуль numpy – який по суті є Python-обгорткою над написаним на C та відкомпільованим ядром.

Проте звичні генетичні операції з синтаксисом numpy виглядають дуже дивно та нечитаємо, але це була ціна за швидкість коду. Наведемо приклад реалізації оператору мутації з модулем numpy. Всі основні оператори знаходятся в пакеті *core* в коді роботи.

```
def mutate(population: np.ndarray, px: float):
    population[np.random.rand(*population.shape) < px] ^= 1
    return population
```

(Тут і надалі `np` – псевдонім `numpy`)

З першого погляду важко зрозуміти, що робить ця одна строка коду. Але по-суті вона виконує оператор мутації. Покроково:

1) `population.shape` – поверне кортеж розмірностей популяції: `(N, L)`. Варто зауважити, що усюди популяція представлена як бінарна матриця POP[N x L].
2) `np.random.rand(*population.shape)` – в результаті дасть матрицю такої ж розмірності як і популяція (значення в комірках – випадкові від 0 до 1 за рівномірним розподілом), на приклад:

$$\underbrace{\begin{bmatrix} 0.02\ldots & 0.11\ldots & \ldots & 0.42\ldots & 0.78\ldots \\ \vdots & & \ddots & & \vdots \\ 0.34\ldots & 0.87\ldots & \ldots & 0.64\ldots & 0.51\ldots \end{bmatrix}}_{L} \Big\} N$$

3) `np.random.rand(*population.shape) < px` – також векторизована операція, яка поверне матрицю з булеанів – результатів порівнянь. Для вищенаведеного прикладу та значення px=0.3, будемо мати:

$$\begin{bmatrix} True & True & \dots & False & False \\ \vdots & & \ddots & & \vdots \\ True & False & \dots & False & False \end{bmatrix}$$

4) `population[…] –` оператор [] маючи на вхід іншу матрицю з булеанів (див. вище) та знаходячись зліва від присвоєння змінить тільки ті елементи, на місці яких у вхідній матриці стоїть True.
5) `^= 1 –` ні що інше, як побітова операція XOR, яка змінить 0 в популяції на 1, а 1 на 0.

Результатом цих всіх кроків буде те, що в популяції змутують локуси із ймовірністю px.

1.3.2 Збереження даних обчислювальних експериментів

1.3.2.1 Структура бази даних.

Оскільки більшість обчислювальних експериментів мала проводитись на платформі GoogleColab (через їх тривалість), то потрібно було вирішити проблему дистанційної бази даних. Google пропонує 200$ на 45 днів на використання на Google Cloud, де і було розгорнуто базу даних.

У ролі СКДБ була обрана досить відома open source система PostgresQL. Вона оптимізована для великого обсягу даних та зручна у

користуванні. Вона має багато вбудованих функцій для аналізу даних, а також підтримку типу масиву[2], який дуже багато використовувався в роботі.

Оскільки база даних використовувалась виключно для збереження даних експериментів, вона не була нормалізована. На приклад, структура таблички для пошуку Pmax:

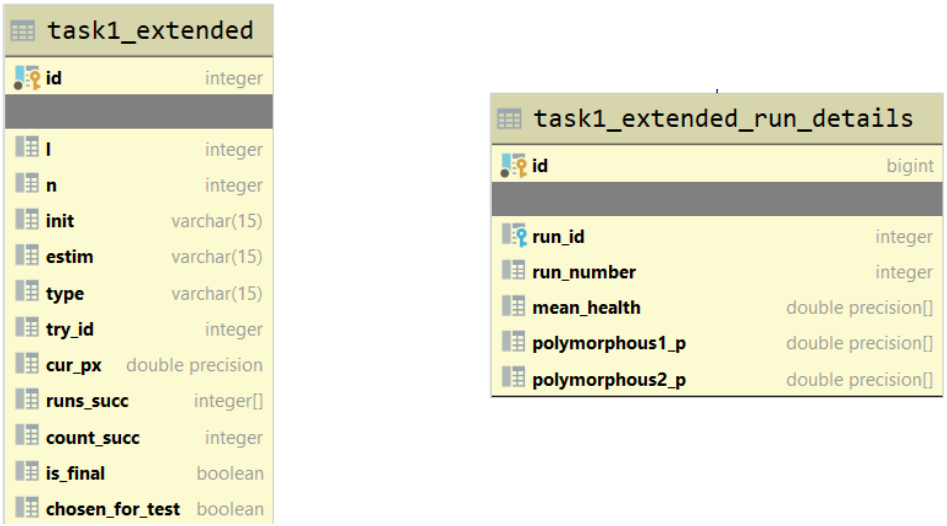


Рисунок 1.1. Основні таблиці для експериментів

`task1_extended` – в ній міститься інформація про процес підбору Pmax для кожного набору параметрів. Поля: **l** - довжина хромосоми, **n** – кількість особин в популяції, **init** – функція ініціалізації, **estim** – функція оцінювання, **type** – тип відбору, **try_id** – номер підходу (за один підхід 10 прогонів), **cur_px** – значення px для підходу, **runs_succ** – масив з номерами останньої ітерації в кожному прогоні (щоб аналізувати швидкість збігання) [2], **count_succ** – кількість прогонів зі збіжністю, **is_final** – чи це останній прогін, **chosen_for_test** – чи вибрано цей px для тестування.

`task1_extended_run_details` – використовується для відслідковування середнього здоров'я популяції в конкретному прогоні (run_id references task1_extended(id)). Поля **run_number** – номер прогону з масиву runs_succ, **mean_health** – масив із середнім здоров'ям популяції для кожної ітерації, **polymorphous1_p** – відсоток поліморфних генів (не менше 1 мутації у локусі) для кожної ітерації.

Дуже схожі таблички для зберігання результатів тестування:

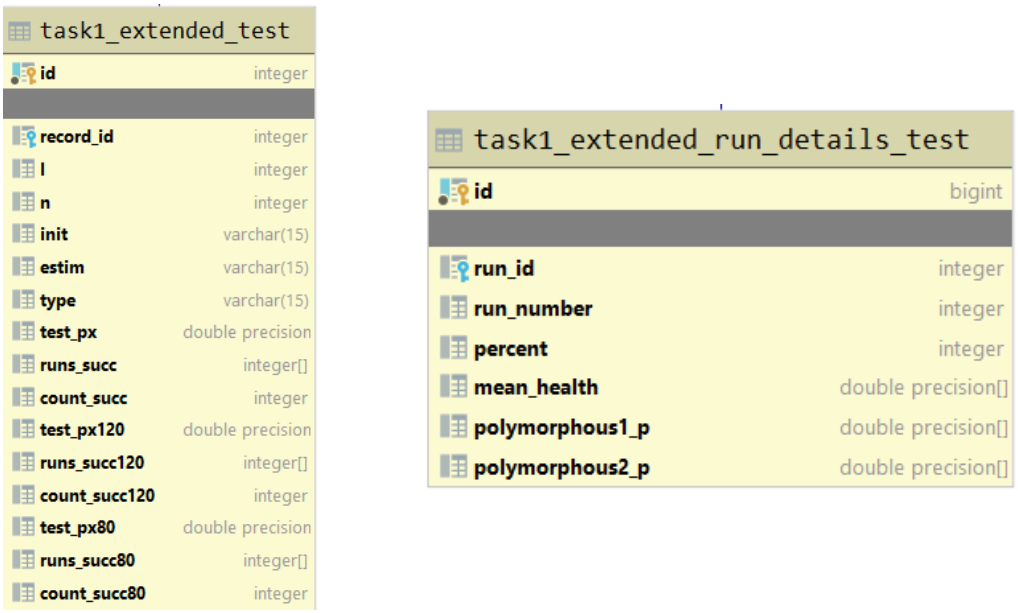


Рисунок 1.2 Додаткові таблиці для експериментів

Таблиця `task1_extended_test` подібна до `task1_extended`, тільки тепер **test_px** – значення px яке ми тестуємо, поля із закінченням 120 – випадок тестування коли ми збільшуємо px на 20%, а із закінченням 80 – зменшуємо на 20%.

Таблиця `task1_extended_run_details_test` подібна до `task1_extended_run_details`, тільки додалось поле **percent** – що означає випадок тестування (80%, 100%, 120%).

Повну структуру бази даних можна знайти в додатку А.

1.3.2.2 Маніпулювання даними.

Оскільки база даних призначалась виключно для зберігання даних експериментів, необхідності у використання ORM не було. Для додавання записів в базу використовувались INSERT SQL запити.

Для отримання інформації використано звичайні SELECT * FROM tablename; запити на діалекті PostgresQL.

Дещо цікавішими були запити для трансформації даних для візуалізації.

В наступному прикладі нам потрібно «дістати» всі значення з масиву `mean_health`, щоб зобразити їх на графіку:

```
SELECT unnest(mean_health)
FROM task1_extended_run_details
WHERE id=%s;
```

А цим запитом ми вибираємо для тестування останні значення px, за яких ще зберігалася 100% збіжність:

```
UPDATE task1_extended t1
SET chosen_for_test=true
WHERE count_succ=10 AND try_id IN  (
    SELECT MAX(try_id)
    FROM task1_extended t2
    WHERE t2.init = t1.init AND
        t2.estim = t1.estim AND
        t2.L=t1.L AND
        t2.N=t1.N AND
        t2.count_succ=t1.count_succ AND
        t2.type = t1.type
    );
```

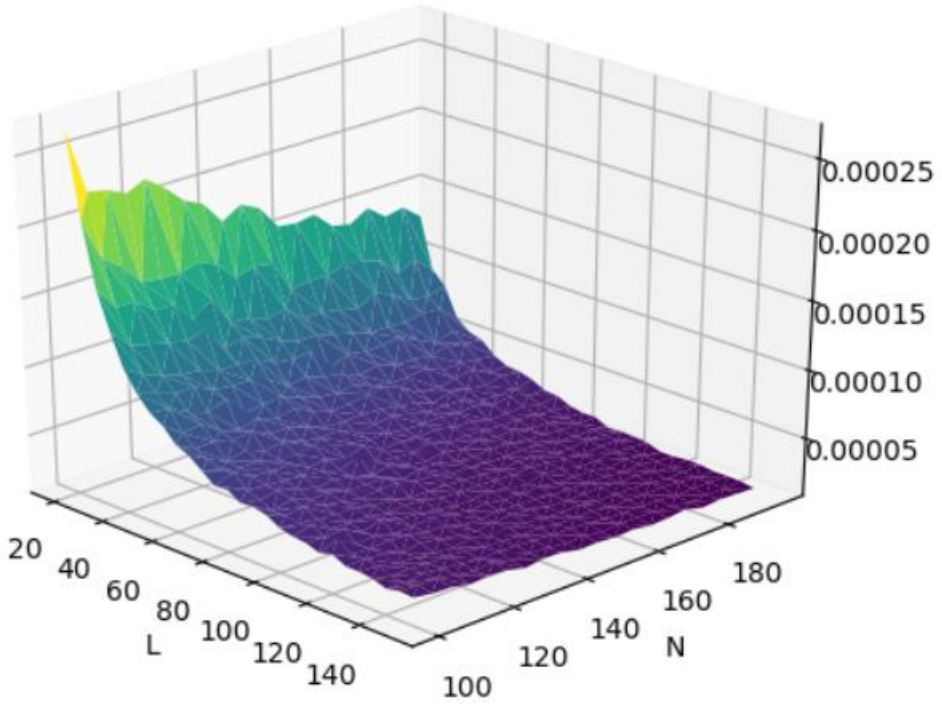
Розділ 2. Проведення обчислювальних експериментів

2.1 Експерименти з відбором за методом рулетки

Першим для дослідження був використаний такий набір вхідних параметрів: ініціалізація – **uniform**, оцінювання – **l-hamming_d**, відбір – **rws**.

Щоб прослідкувати зміну px від L та N, ми задали крок 5 для L від **20 до 150** та N від **100 до 200**. Приблизний час знаходження значень px для такого набору приблизно 6-7 годин. Приблизний час для тестування 4-5 годин.

У результаті експерименту були отримані значення Pmax для вищенаведеного набору параметрів (Додаток Б). Було побудовано графік залежностей px (вісь Oz) від L (вісь Ox) та N (вісь Oy):

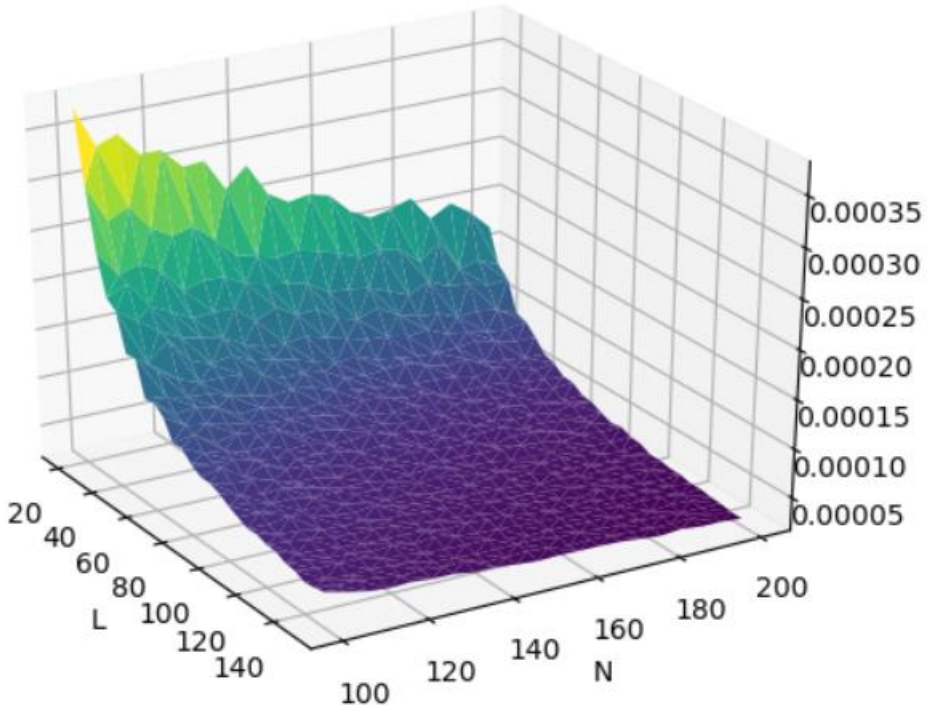


Графік 2.1 Залежність Pmax від L та від N. Рулетка.

2.2 Експерименти з турнірним відбором

Аналогічний експеримент було проведено для такого набору параметрів: ініціалізація – **uniform**, оцінювання – **l-hamming_d**, відбір – **tournament_2**, L – **20 – 150** (крок 5), N – **100 – 200** (крок 5) .

Експеримент так само зайняв приблизно 12 годин обчислювального часу. За його результатами було сформовано список значень (додаток В) для цих вхідних параметрів, а також побудовано графік залежностей, px - вісь Oz, L - вісь Ox та N - вісь Oy:



Графік 2.2 Залежність Pmax від L та від N. Турнірний відір.

2.3 Експерименти з ширшим діапазоном значень вхідних параметрів

З попередніх експериментів вже можна зробити висновок, що на поведінку Pmax критично впливають значення N та L, та не сильно впливає тип відбору.

Проте, ми маємо також знайти значення для інших параметрів, щоб бути впевненими, що вони значно не впливають на Pmax, або – в протилежному випадку – впливають і ми їх теж маємо брати до уваги при аналізі.

Отже, в цьому експерименті, ми маємо знайти значення Pmax для наступного набору параметрів:

- Ініціалізація
  - **all_0**
  - **uniform**
  - **normal**
- Оцінювання
  - **l-hamming_d**
  - **sigma_2**, **sigma_4**, **sigma_10**
- Відбір
  - **rws**
  - **tournament_2**, **tournament_4**, **tournament_12**

Проте, якщо б ми взяли значення L та N як в попередніх експериментах (всього 520 пар) для кожного набору параметрів (всього 48 наборів), обчислення зайняли б неприйнятно великий час. Окрім цього, для перевірки гіпотези про вплив цих параметрів на Pmax, нам достатньо декілька основних

значень L та N, які ми зможемо зафіксувати і дивитись, як змінюєтсья Pmax від відбору, на приклад, або від оцінювання.

Тому, опорними значеннями були обрані L = 10, 20, 80, 100, 200 та N = 100, 200. З усіма наборами вхідних параметрів, що були використані в ході цього експерименту можна ознайомитись в додатку Г.

Отримані значення цього експеримету можна отримати у додатку Д.

З усіма значеннями знайденими під час роботи під час усіх експериментів можна ознайомитися в додатку Е.

Розділ 3. Аналіз результатів експериментів

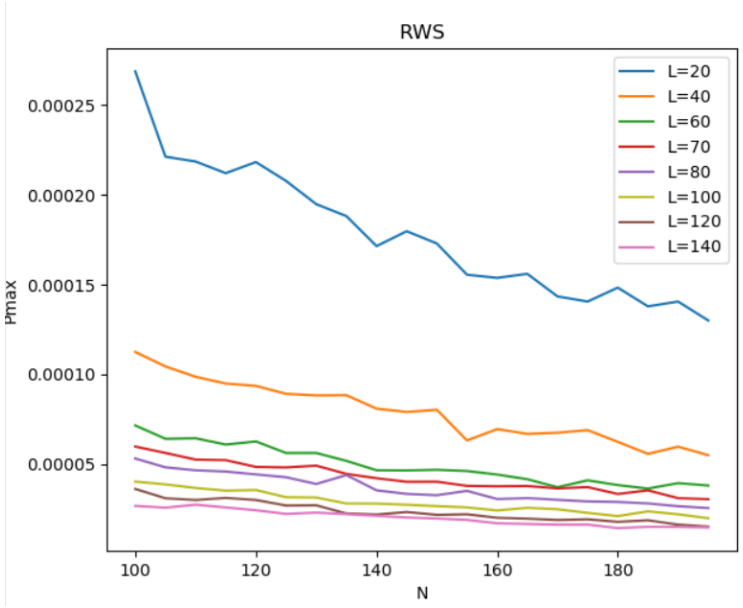
3.1 Вплив параметрів L та N на Pmax

3.1.1 Аналіз даних експериментів з використанням методу відбору rws.

3.1.1.1 Аналіз вхідних даних.

За отриманими в ході експериментів значеннями, можемо побудувати графіки залежності значення Pmax від параметрів L та N для випадку відбору за методом рулетки **rws** (тут і надалі в розділі 3.1 для інших параметрів маються на увазі такі значення: ініціалізація - **uniform**, оцінювання - **l-hamming_d**).

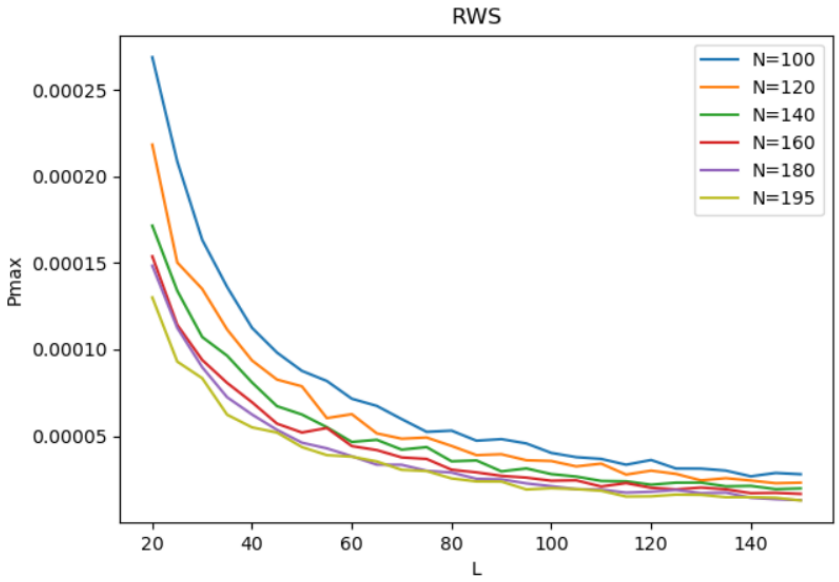
1) Фіксуємо L – та подивимось, як поводиться Pmax в залежності від N:



Графік 3.1 Залежність Pmax від N за фіксованого L. Рулетка.

Можемо бачити, що за фіксованого значення L при зростанні N значення ймовірності Pmax зменшується.

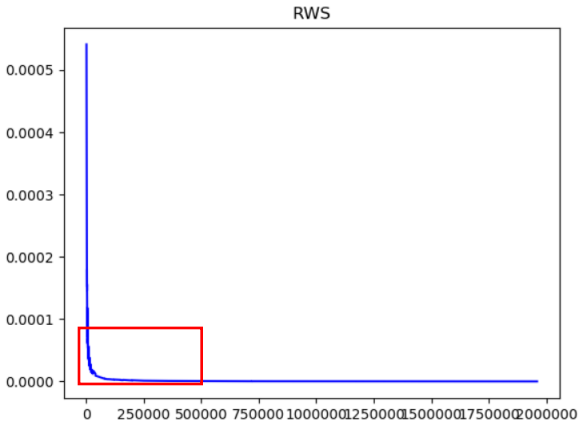
2) Тепер фіксуємо N та слідкуємо за L:



Графік 3.2 Залежність Pmax від L за фіксованого N. Рулетка.

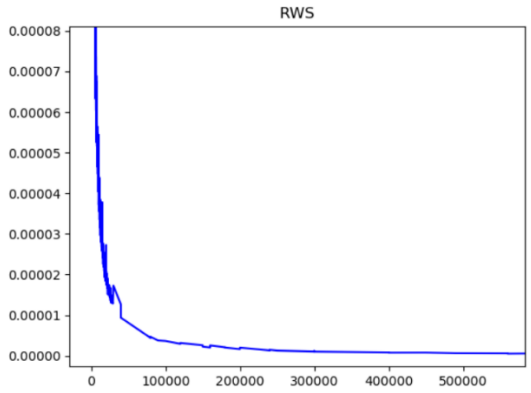
Бачимо, що за фіксованого значення N при зростанні L значення ймовірності Pmax також зменшується.

3) З огляду на попередньо побудовані графіки, можна припустити, що параметри L та N спільно впливають на значення Pmax. Побудуємо такий графік: по осі Ox – L * N, по Oy – значення Pmax:



*Графік 3.4 Залежність Pmax від L*N. Рулетка.*

Масштабуємо трохи частину виділену червоним:



*Графік 3.5 Залежність Pmax від L*N. Масштабований. Рулетка.*

4) Отримані дані також дозволяють нам зобразити темпи зміни значення Pmax зі зміною L, а саме, збудуємо графік наступним чином для фіксованого N:

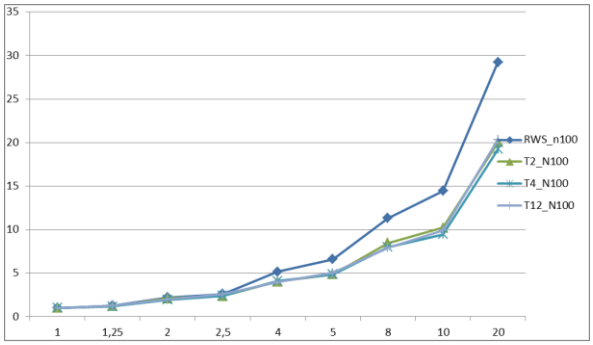
$$\left\{ \left(\frac{L_j}{L_i}, \frac{Pmax_i}{Pmax_j} \right) \middle| \; i < j \right\}$$

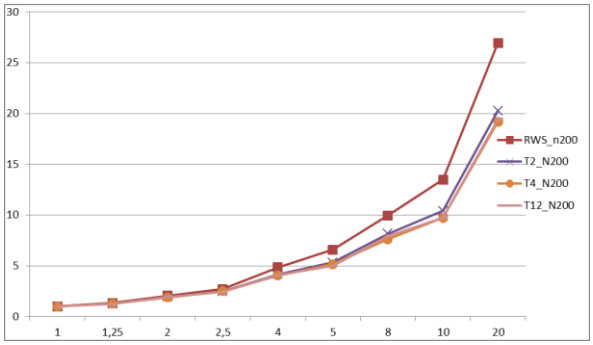
Формула 3.1 Побудова графіку 3.3

Для зручності на наступних графіках показно одразу 4 варіанти

відбору: рулетка, турнірний з параметрами 2, 4 та 12

Розглянуто випадки N = 100 та N = 200:





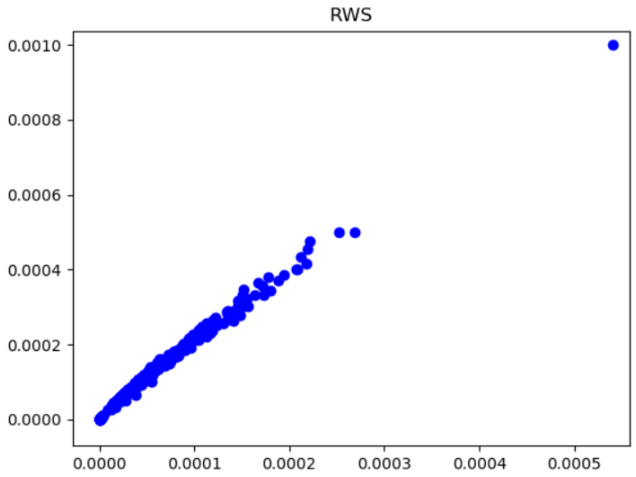
Графік 3.3 Темпи зміни значення Pmax із зміною L

З огляду на вищенаведені графіки можна зробити припущення, що наявна залежність є обернено-пропорційною, та має вигляд:

$$Pmax_{L,N} = \frac{a}{L * N} + b, \qquad a, b \in R$$

Формула 3.2 Вигляд залежності Pmax

На підтримку цієї гіпотези спробуємо побудувати графік залежності Pmax = 1 / (L * n), він повинен мати вигляд прямої лінії. Ox – 1 / (L*N), Oy – Pmax.



*Графік 3.6 Залежність Pmax від 1 / (L * N) . Рулетка.*

Дійсно, побудований графік, нагадує пряму лінію. Ми можемо спробувати апроксимувати цю залежність за допомогою регресійного аналізу.

3.1.1.2 Регресійний аналіз.

Маючи достатньо великий набір даних та впізнавши лінійну природу функції, ми можемо скористатися методами параметричної лінійної регресії, щоб апроксимувати функцію Pmax(L, N), поки що для випадку відбору за методом рулетки.

Для роботи був використаний клас LinearRegression бібілотеки sklearn [3]. Jupyter notebook, використаний для знаходження всіх рівнянь можна переглянути в Додатку Ж.

Отримане рівняння має вигляд:

$$Pmax_{L,N} = \frac{0.5108088}{L * N} - 6.64 * 10^{-6}$$

Формула 3.3

Розділивши вибірку випадково на навчальну та тестову (тестова 40% від всього набору), ми можемо проаналізувати модель на тестових значеннях. Функції оцінювання взяті з sklearn, та обчислюються за такими формулами [4]:

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|.$$

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

$\hat{\mathbf{y}}$ позначає значення, що виведені з моделі, а $\mathbf{y}$ – підібрані значення експериметальним шляхом.

Показник	Значення
MSE	$1.6*10^{-11}$
MAE	$3.08*10^{-6}$
MAE, %	7.7

Таблиця 3.1 Результати формули 3.3

Ми також можемо обрахувати середню помилку на всій вибірці для тих значень L та N, які ще дають результат більший за 0, за допомогою наступного SQL запиту:

```
WITH tmp AS (
    SELECT ABS(?a / (L * N) - ?b - final_px) /
        LEAST(?a / (L * N) - ?b , final_px) as error
    FROM final_pxs_extended
    WHERE init = 'uniform'
      AND estim = 'l-hamming_d'
      AND type = 'rws'
      AND ?a / (L * N) - ?b > 0
)
SELECT MIN(error), MAX(error), AVG(error)
FROM tmp;
```

Де ?a та ?b – параметри регресії.

Отриманий результат: min: **0.04%**, max: **106%** , avg: **7%**.

Модель дала непогані результати, проте в неї також є і мінуси. Незважаючи на те, що вільний член регресії дуже малий ($6.6*10^{-6}$), це обмежує максимальні значення L та N, для яких ми можемо передбачити значення Pmax, враховуючи те, що ймовірність мутація має бути строго вище 0:

$$\frac{0.5108088}{L * N} - 6.64 * 10^{-6} > 0$$

$$\frac{0.5108088}{L * N} > 6.64 * 10^{-6}$$

$$\frac{0.5108088}{6.64 * 10^{-6}} > L * N$$

$$L * N < \sim 77000$$

Формула 3.4

Тобто вже навіть для L = 100 та N = 1000 модель дасть від'ємний результат, що є неприпустимим.

Щоб цьому запобігти, ми можемо тренерувати модель без вільного члена. Цьому відповідає параметр *fit_intercept* лінійної регресії в sklearn [3]. На попередьому наборі вхідних параметрів, отримали таке рівняння:

$$Pmax_{L,N} = \frac{0.47591131}{L * N}$$

Формула 3.5

Аналіз моделі:

Показник	Значення
Mean squared error	$2.6*10^{-11}$

Mean absolute error	$4.14*10^{-6}$
Mean absolute error, %	10.3

Таблиця 3.2 Результати формули 3.5

Результат на всій вибірці: min: **0.004%**, max: **80%**, avg: **17.5%**

Бачимо, що модель має більшу помилку, проте ми можемо використовувати її для будь-яких значень L та N.

Також можемо бачити, що хоч сердні значення помилки не дуже високі, максимальні значення зашкалюють. Переглянувши окремо значення помилки для кожного L*N можемо впевнитися, що найвища помилка в дуже великих значення L*N (500000 і більше). Це може бути пов’язано з тим, що у навчальній вибірці значно більше значень з L*N < 20000 і вона дуже нерівномірна, як можемо бачити на наступному рисунку:

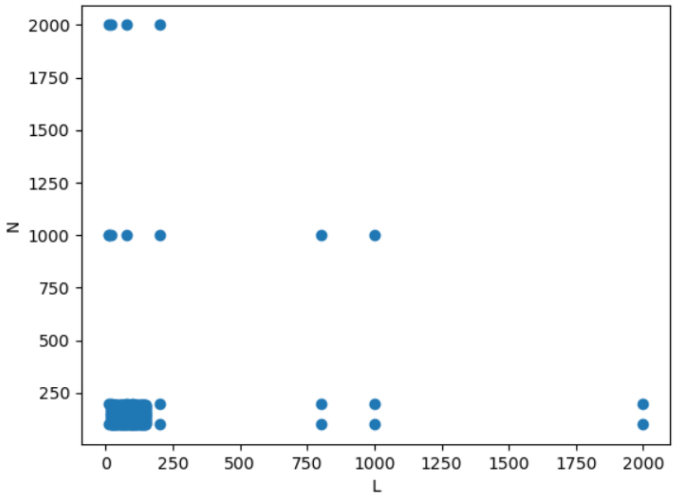


Рисунок 3.1 Датасет для рулетки

Спробуємо прибрати з вибірки значення 10 < L < 150, 100 < N < 200 та залишити базові значення. Ми значно зменшили розмір вибірки (з 450 до 50), але отримали такі більш-менш рівномірні дані:

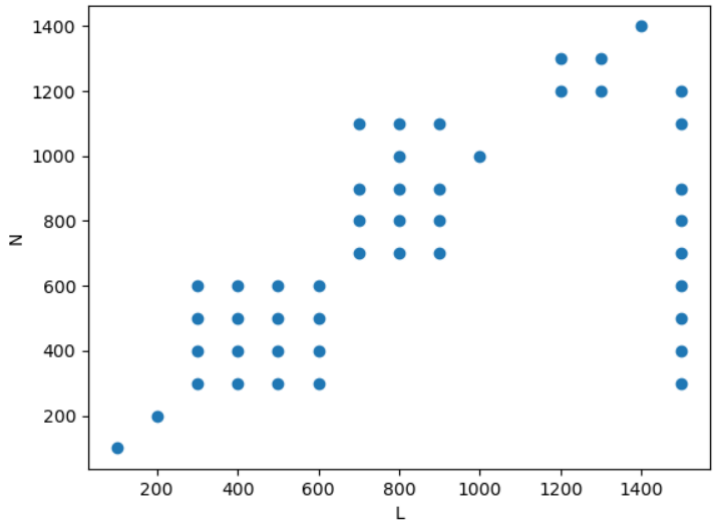


Рисунок 3.2 Рівномірний датасет для рулетки

З цієї вибірки отримуємо рівняння:

$$Pmax_{L,N} = \frac{0.39869718}{L * N}$$

Формула 3.6

Результати моделі (ми також зменшили пропорції начання/тест до 80/20 через малу кількість даних):

Показник	Значення на навчальній	Значення на тесті
MSE	$6*10^{-14}$	$5*10^{-14}$
MAE	$2*10^{-7}$	$1.8*10^{-7}$
MAE, %	10.1	9.4

Таблиця 3.3 Результати формули 3.6

Результат на всій вибірці: min: **0.02%**, max: **51%**, avg: **9.7%**

Бачимо, що ця модель дала значно кращі результати.

3.1.1.3 Перевірка отриманих результатів.

На підтвердження результатів, отриманих за допомогою методів лінійної регресії, проведемо додаткові експерименти для різних комбінацій L та N, обраховуючи значення Pmax з отриманих рівнянь.

Випадок з вільним членом (див. формулу 3.3).

Px	N	L	N*L	Чи очікується збіжність	Чи досягнуто збіжність? Якщо так – номер ітерації
5.04166822e-04 – 10%	100	10	1000	Так	{1272,1401,2909,1063,4673}
4.44389045e-05– 10%	100	100	10000	Так	{6713,3675,1212,2257,4445}
4.44389045e-05– 10%	1000	10	10000	Так	{345,145,342,1954,1683}
1.87150468e-06– 10%	60	1000	60000	Так	{112,155,131,119,70}
1.87150468e-06– 10%	1000	60	60000	Так	{213,142,189,184,136}
5.04166822e-04 – 10%	100	10	1000	Так	{1272,1401,2909,1063,4673}
5.04166822e-04 – 10%	100	100	10000	Ні	Ні
5.04166822e-04 – 10%	1000	10	10000	Ні	Ні
5.04166822e-04 – 10%	100	1000	100000	Ні	Ні
5.04166822e-04 – 10%	1000	100	100000	Ні	Ні

Таблиця 3.4 Перевірка результатів формули 3.3

В першій частині таблиці – обраховані значення для конкретних випадків L та N та протестовані на відповідних їм значенням L та N.

В другій частині, ми взяли значення, обраховане для випадку L=100 N=10 та спробували протестувати на інших значеннях цих параметрів. Як і очікувалося, збіжності в цих випадках не було.

Випадок без вільного члена (див. формулу 3.5).

Px	N	L	N*L	Чи очікується збіжність	Чи досягнуто збіжність? Якщо так – номер ітерації
4.7591131e-04 – 10%	100	10	1000	Так	{269,2238,3549,306,2765}
4.7591131e-05– 10%	100	100	10000	Так	{6713,3675,1212,2257,4445}
4.7591131e-05– 10%	1000	10	10000	Так	{5378,1917,12728,4878,904}
4.7591131e-06– 10%	100	1000	100000	Так	{455,17449,5445,2432,19065}
4.7591131e-06– 10%	1000	100	100000	Так	{2471,8036,7776,11198,4843}
4.75911310e-04 – 10%	100	10	1000	Так	{269,2238,3549,306,2765}
4.75911310e-04 – 10%	100	100	10000	Ні	Ні
4.75911310e-04 – 10%	1000	10	10000	Ні	Ні
4.75911310e-04 – 10%	100	1000	100000	Ні	Ні
4.75911310e-04 – 10%	1000	100	100000	Ні	Ні

Таблиця 3.5 Перевірка результатів формули 3.5

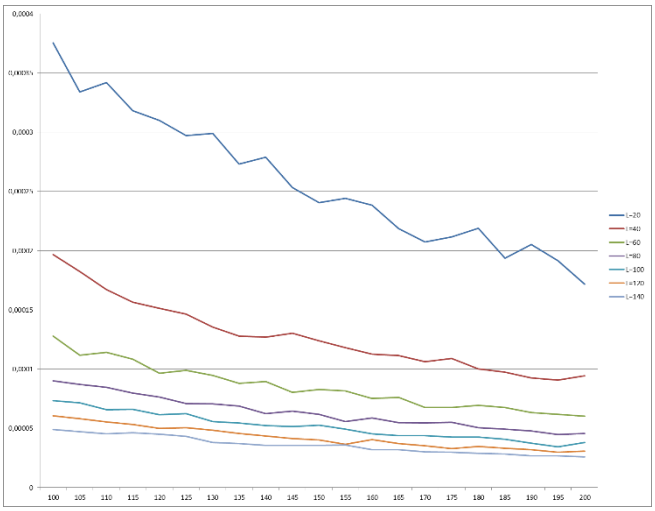
Отже, можна зробити висновок, що значення Pmax залежить від L*N лінійно.

3.1.2 Аналіз даних експериментів для турнірного відбору

3.1.2.1 Аналіз вхідних даних.

Аналогічно подібні графіки ми можемо побудувати і для випадку турнірного відбору **tournament_2**.

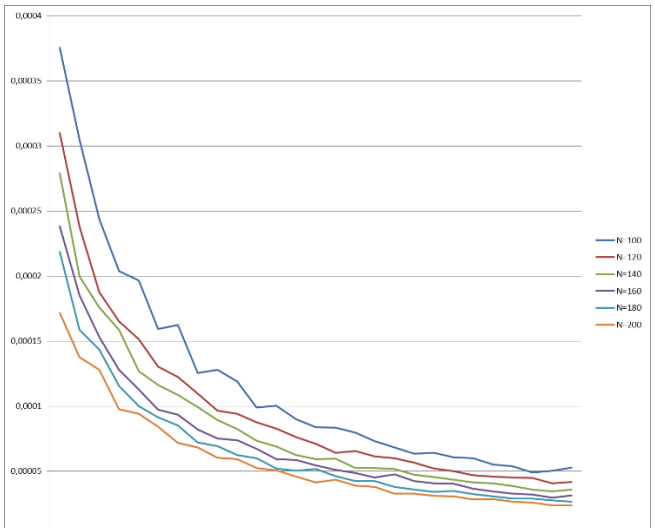
1) Фіксуємо L – та подивимось, як поводиться Pmax в залежності від N, Ox – значення N, Oy - Pmax:



Графік 3.7 Залежність Pmax від N за фіксованого L. Турнірний відбір.

Так само бачио, що за фіксованого значення L при зростанні N значення ймовірності Pmax зменшується.

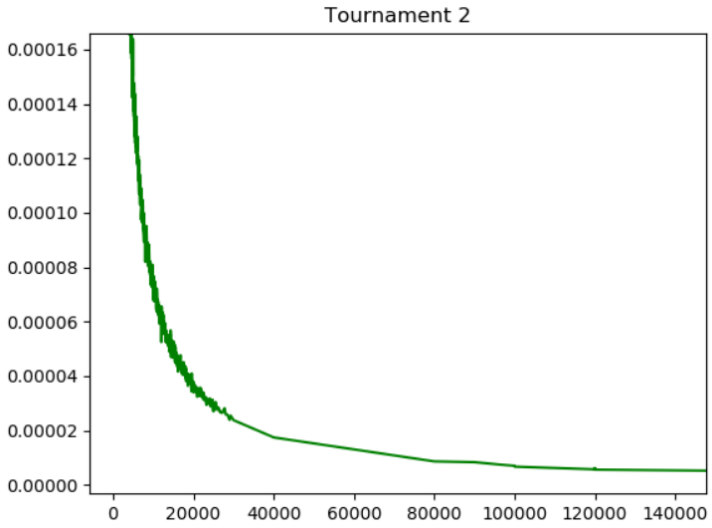
2) Тепер фіксуємо N та слідкуємо за L, Ox – значення L, Oy – Pmax:



Графік 3.8 Залежність Pmax від L за фіксованого N. Турнірний відбір.

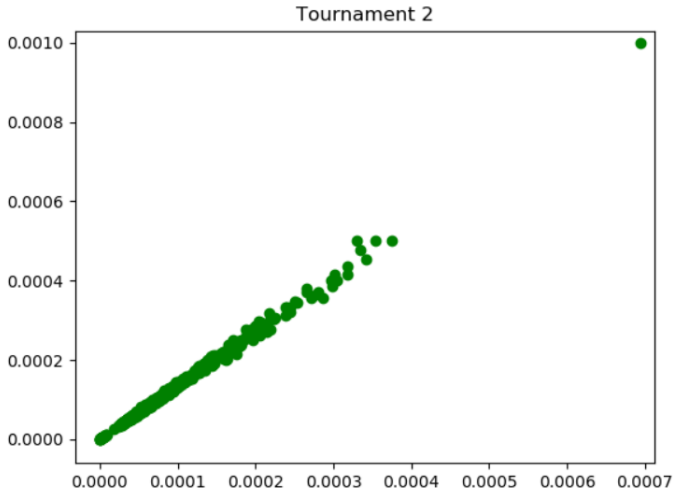
Так само при зростанні L значення ймовірності Pmax зменшується.

3) Графік залежності Pmax від L * N, Ox – L * N, Oy – Pmax:



*Графік 3.9 Залежність Pmax від L*N. Масштабований. Турнірний відбір.*

Бачимо таку саму залежність, як і у випадку з рулеткою. Побудуємо графік залежності Pmax від 1 / (L * N). Ox – 1 / (L*N), Oy – Pmax:



*Графік 3.10 Залежність Pmax від 1 / (L * N) . Турнірний відбір.*

Побудований графік так само нагадує пряму лінію, а отже ми так само спробуємо апроксимувати функцію Pmax для випадку турнірного відбору.

3.1.2.2 Регресійний аналіз.

Отже, аналогічно до попереднього варіанту отримуємо рівняння:

$$Pmax_{L,N} = \frac{0.71923526}{L * N} - 1.2 * 10^{-6}$$

Формула 3.7

Ми розділили навчальну вибірку на тренувальну та тестову (60% та 40% відповідно). Маємо такі показники:

Показник	Значення на навчальній	Значення на тестовій
MSE	$1.87*10^{-11}$	$1.98*10^{-11}$
MAE	$2.54*10^{-6}$	$2.68*10^{-6}$

MAE, %	3.19	3.34

Таблиця 3.6 Результати формули 3.7

Результат на всій вибірці: min: **0.003%**, max: **1381%**, avg: **15%**

Бачимо аномальні значення максимуму, при тому що середнє значення не дуже високе. Знову, такі аномально великі значення мають дуже великі L*N.

Також пробуємо тренувати модель без вільного члену, виходять такі результати:

$$Pmax_{L,N} = \frac{0.7251663}{L * N}$$

Формула 3.8

Показник	Значення на навчальній	Значення на тестовій
MSE	$1.94*10^{-11}$	$1.92*10^{-11}$
MAE	$2.47*10^{-6}$	$2.56*10^{-6}$
MAE, %	3.11	3.2

Таблиця 3.7 Результати формули 3.8

Результат на всій вибірці: min: **0.001%**, max: **15%**, avg: **3.2%**

Бачимо, що модель вийшла значно краща за попередню. Через це ми перконуємся, шо навіть якщо вільний член рівнння дуже малий, він відіграє дуже велику роль для великих значень L*N, де значення Pmax дуже малі.

Ми також як і у випадку з рулеткою можемо спробувати покращити результат, зробивши навчальну вибірку рівномірною. На початку маємо це:

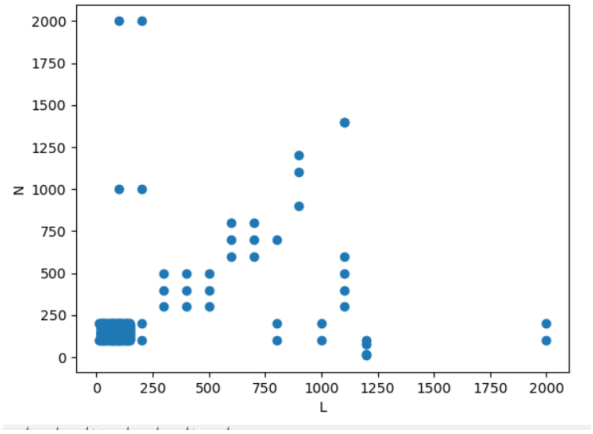


Рисунок 3.3 Датасет для турніру

Прибравши значення 10 < L < 150, 100 < N < 200 та залишивши з них основні, маємо:

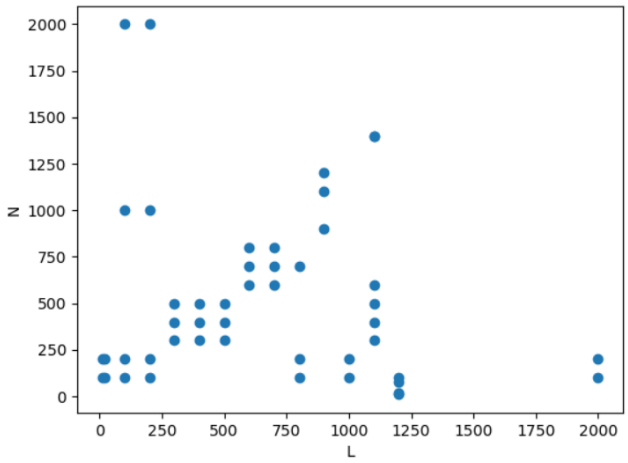


Рисунок 3.4 Рівномірний датасет для турніру

Тепер спробуємо отримати рівняння:

$$Pmax_{L,N} = \frac{0.69157678}{L * N}$$

Формула 3.9

Результати моделі (ми так само змінили пропорцію навчання/вибірка на 80/20, як і у випадку з рулеткою):

Показник	Значення на навчальній	Значення на тестовій
MSE	$1.49*10^{-11}$	$3.3*10^{-13}$

MAE	$1.53*10^{-6}$	$2.6*10^{-7}$
MAE, %	3.7	0.65

Таблиця 3.8 Результати формули 3.9

Результат на всій вибірці: min: **0.01%**, max: **17.5%**, avg: **6%**

Бачимо, що результат погіршився на всій вибірці. Це може говорити про те, що відібраних даних замало або вони мають велику похибку, щоб на їх основі будувати регресію.

3.1.2.3 Перевірка отриманих результатів.

Тепер, як і в попередньому варіанті, обрахуємо значення Pmax виходячи з отриманих рівнянь та подивимось на збіжність.

Випадок із вільним членом (див. формулу 3.7).

Px	N	L	N*L	Чи очікується збіжність	Чи досягнуто збіжність? Якщо так – номер ітерації
7.20442448e-04– 10%	100	10	1000	Так	{4042,1920,4514,1986,15307}
7.31307123e-05– 10%	100	100	10000	Так	{6399,661,5514,2170,1901}
7.31307123e-05– 10%	1000	10	10000	Так	{4832,393,1875,1875,2421}
8.39953883e-06– 10%	100	1000	100000	Так	{5540,4192,3212,7135,9084}
8.39953883e-06– 10%	1000	100	100000	Так	{6394,1485,4042,4113,1252}
7.20442448e-04– 10%	100	10	1000	Так	{4042,1920,4514,1986,15307}
7.20442448e-04– 10%	100	100	10000	Ні	Ні
7.20442448e-04– 10%	1000	10	10000	Ні	Ні
7.20442448e-04– 10%	100	1000	100000	Ні	Ні
7.20442448e-04– 10%	1000	100	100000	Ні	Ні

Таблиця 3.9 Перевірка результатів формули 3.7

Випадок без вільного члену (див. формулу 3.8)

Маємо такі результати:

Px	N	L	N*L	Чи очікується збіжність	Чи досягнуто збіжність? Якщо так – номер ітерації

7.25166297e-04– 10%	100	10	1000	Так	{1238,203,387,588,151}
7.25166297e-05– 10%	100	100	10000	Так	{2449,2112,3368,2501,888}
7.25166297e-05– 10%	1000	10	10000	Так	{765,1330,36,1563,987}
7.25166297e-06– 10%	100	1000	100000	Так	{1288,3501,10832,5886,2001}
7.25166297e-06– 10%	1000	100	100000	Так	{7730,451,6432,2408,1324}
7.25166297e-04– 10%	100	10	1000	Так	{1238,203,387,588,151}
7.25166297e-04– 10%	100	100	10000	Ні	Ні
7.25166297e-04– 10%	1000	10	10000	Ні	Ні
7.25166297e-04– 10%	100	1000	100000	Ні	Ні
7.25166297e-04– 10%	1000	100	100000	Ні	Ні

Таблиця 3.10 Перевірка результатів формули 3.8

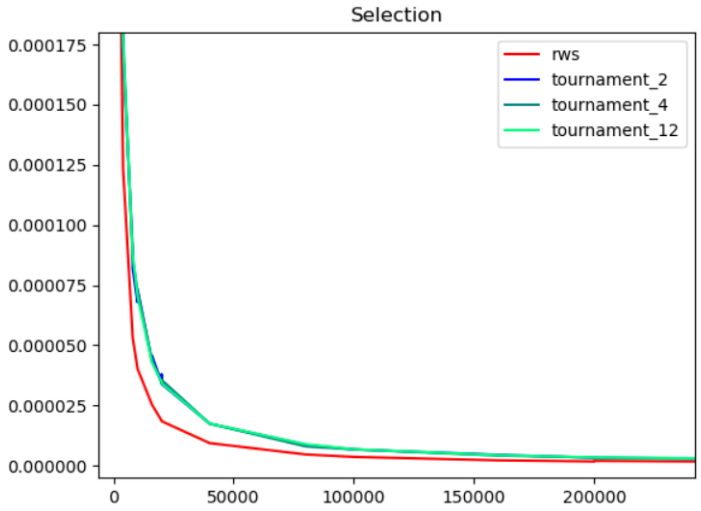
Отже, можна зробити висновок, залежність Pmax від L * N у випадку турнірного відбору так само є лінійною.

3.2 Вплив інших чинників

3.2.1 Метод відбору

Вже з вищенаведених міркувань можна зробити висновок, що і рулетка, і турнірний відбір, мають однакову лінійну природу, говорячи про значення Pmax для них, проте вони відрізняються параметрами (кутом нахилу лінії). Спробуємо порівняти ці два методи, а також з’ясувати, чи впливає параметр турніру на значення Pmax.

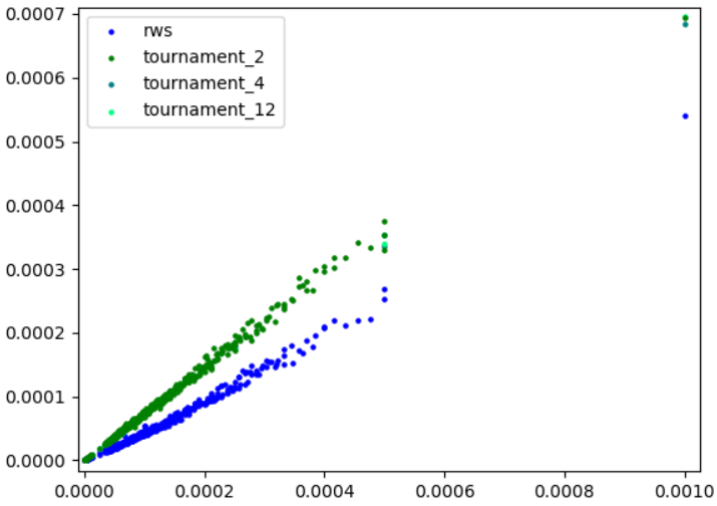
Отже, спершу побудуємо графік залежності Pmax від L * N. Ox – L * N, Oy – Pmax:



*Графік 3.11 Залежність Pmax від L*N. Різні функції відбору.*

З графіку видно, що параметр турніру суттєво не впливає на значення Pmax (відмінність можна списати на похибку, бо в одних точках значення t=2 менше, а в інших t=4).

Стосовно порівняння рулетки та турніру (з параметром 2), то можна побудувати спільний графік Pmax = 1 / (L * N) для обох варіантів. Ox – 1 / (L*N), Oy – Pmax:



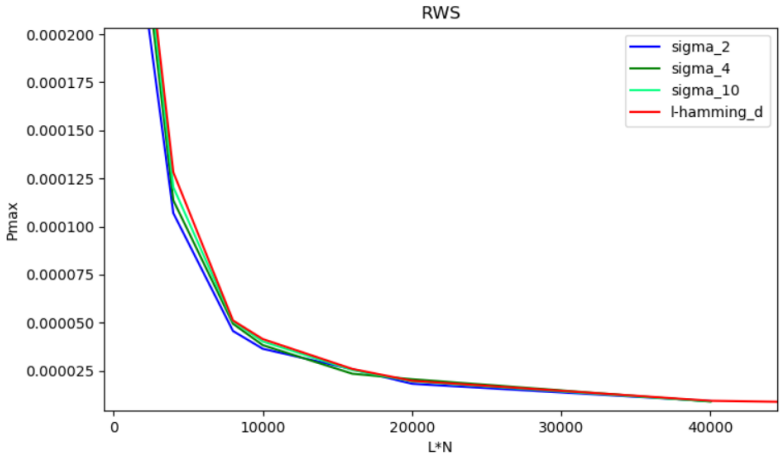
*Графік 3.12 Залежність Pmax від 1/(L*N). Різні функції відбору.*

Бачимо, що кут нахилу для рулетки принципово менший, ніж для турнірного відбору (що бачимо із рівнянь отриманих у минулому розділі).

3.2.2 Функція пристосованості

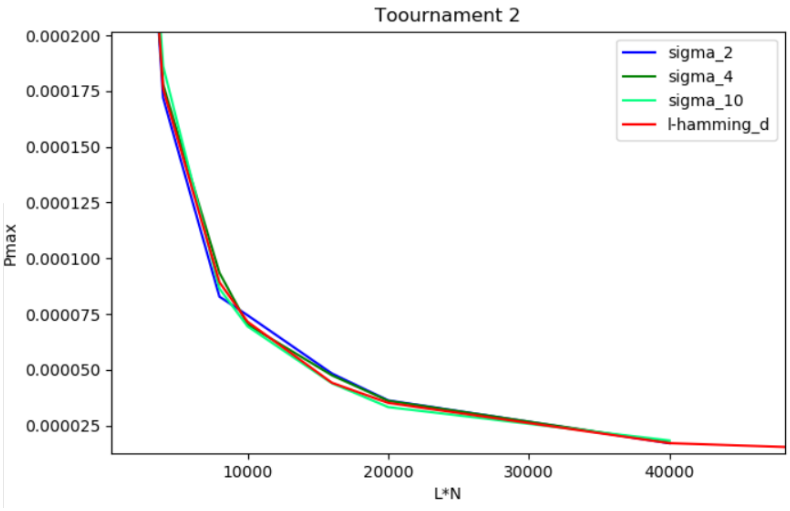
До цього у експериментах ми завжди використовували функцією **l-hamming_d** для оцінки здоров'я особини. Треба також перевірити, чи зміна функції оцінювання вплине на значення Pmax. Використаємо функції з селективною перевагою на біт **sigma_2**, **sigma_4** та **sigma_10**, як описано в розділі 1.1.

Отже, побудуємо такий графік: по осі Ox – L * N, по осі Oy – значення Pmax. Відбір - рулетка



*Графік 3.13 Залежність Pmax від L*N. Рулетка. Різні функції оцінювання.*

Та сама картина спостерігається для tournament_2 (як і для турнірного відбору з параметрами 4 і 12):



*Графік 3.14 Залежність Pmax від L*N. Турнірний відбір. Різні функції оцінювання.*

Спробуємо знайти різницю максимального та мінімального значення в кожній точці та знайти мінімум, максимум і середнє цих значень.. SQL-скрипт, використаний для пошуку цих значень, можна знайти в додатку К.

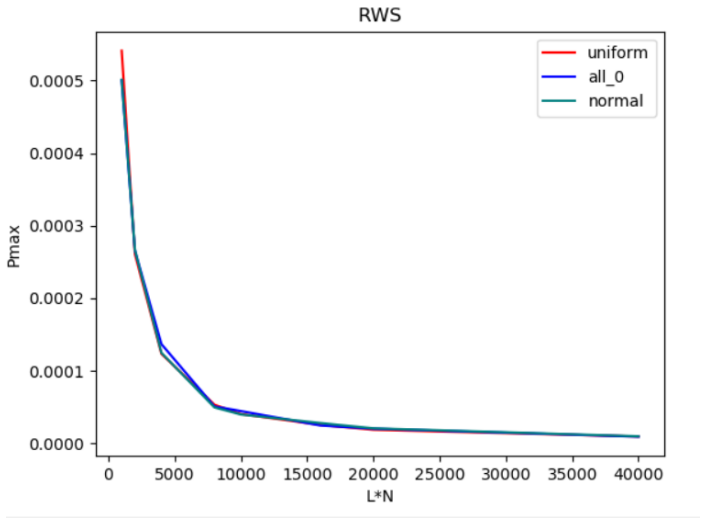
Отже, min: **3%**, max: **16%**, avg: **9.8%**.

Хоча і різниця не є дуже малою, з графіку видно, що у різних точках значення мінімуму і максимуму належать різним функціям оцінювання.

Отож з вищенаведених фактів, можна зробити висновок, що функція пристосованості не впливає на порогове значення мутації Pmax, або якщо впливає – то несуттєво.

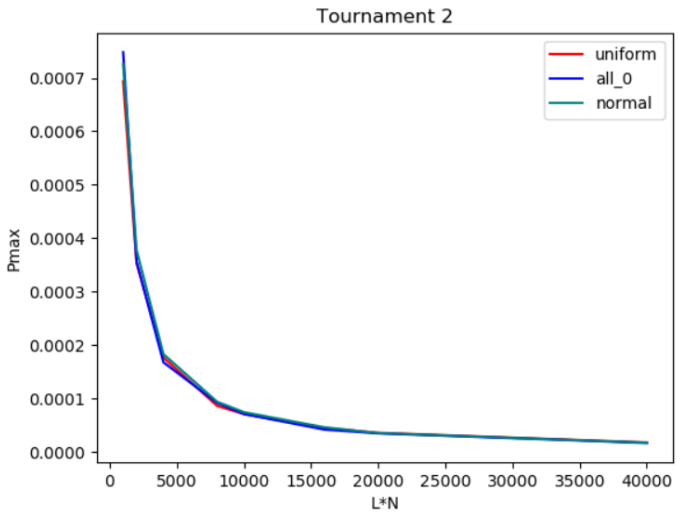
3.2.3 Початковий розподіл

Аналогічно можемо впевнитися, що початковий розподіл також не впливає на значення Pmax. Побудуємо знову графік залежності Pmax від L * N, зафіксуючи інші значення: оцінювання – **L - hamming**, відбір: **rws**.



*Графік 3.15 Залежність Pmax від L*N. Рулетка. Різний початковий розподіл.*

Аналогічний результат для турніру **tournament_2**:



*Графік 3.16 Залежність Pmax від L*N. Турнірний турнір. Різний початковий розподіл.*

Аналогічно до попереднього випадку порівняння функцій оінювання, знайдемо різинці в точках L*N:

Отже, min: **4%**, max: **16.6%**, avg: **9.5%**.

Так само бачимо, що ініціалізація або не впливає на залежність Pmax від L * N, або її вплив є незначним.

Висновки

Виходячи з усього вищенаведеного, можна зробити такі висновки.

По-перше, експерименти із генетичними алгоритмами, а саме підбір значень їх параметрів можуть займати досить значний машинний час, а тому проблеми оптимізації алгоритмів та збереження даних потребують більшої уваги.

По-друге, із результатів експериментів видно, що шукане значення Pmax зворотно пропорційно залежить від L * N – вхідних параметрів довжини хромосоми та розміру популяції. Більш того, цю залежність можна апроксимувати за допомогою лінійної регресії для значень L та N використаних у роботі ($10 < L, N < 2000$).

По-третє, різні методи відбору мають різні значення Pmax. При цьому варто зауважити, що характер самої функції (рулетка/турнір) має суттєвий вплив, а параметр функції (для турніру – 2, 4, 12) майже ніяк не впливає. Рівняння, що описують залежності Pmax у випадках рулетки та турнірного відбору мають різні коефіцієнти, а тому ці випадки треба розглядати окремо, підбираючи коефіцієнт застосування мутації в генетичному алгоритмі.

Окрім цього, можна стверджувати, що використані в роботі значення інших параметрів (ініціалізація, оцінювання) не сильно впливають, або взагалі не впливають на значення Pmax.

Підсумовуючи, можна сказати, що значення коефіцієнту застосування мутації відіграє суттєву роль у роботі генетичного алгоритму. Знайти оптимальне значення Pmax для конкретної задачі може бути нетривіальною задачею. Проте, враховуючи вищезгадані спостереження, ця задача може бути спрощена.

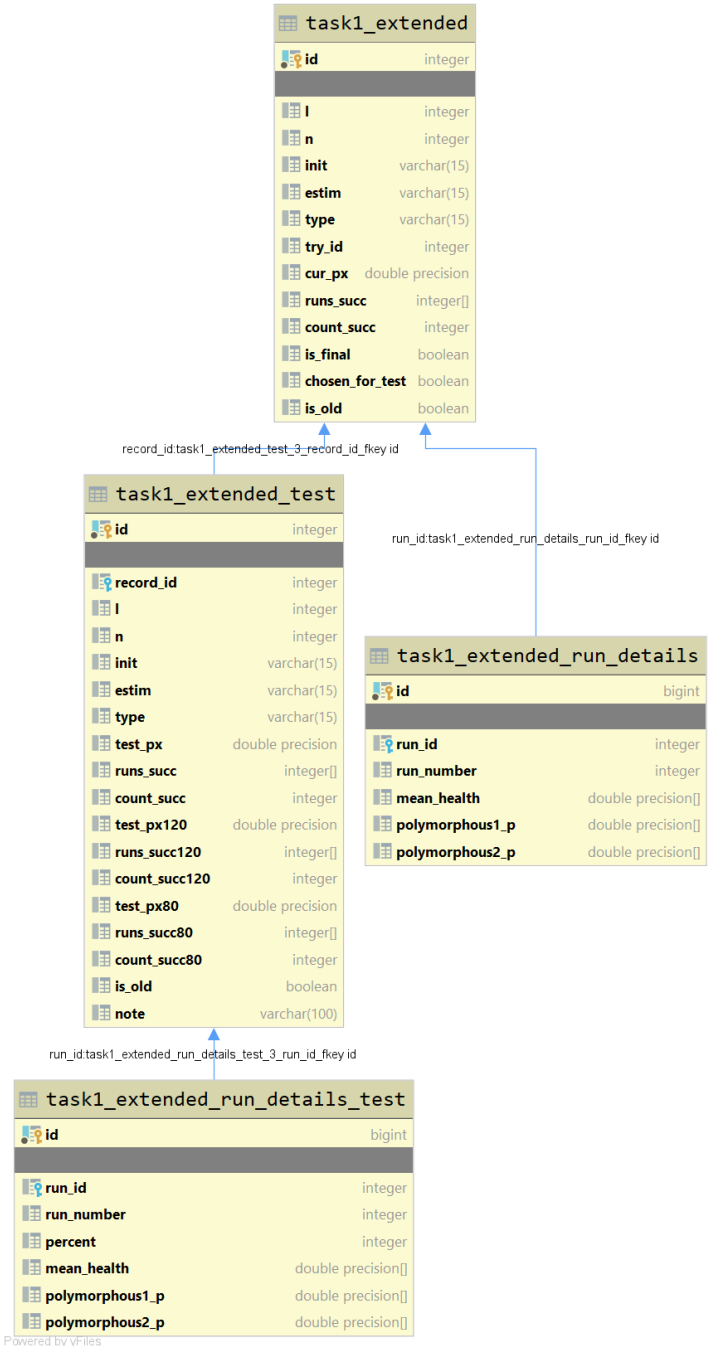
Список використаної літератури

1. Глибовець, Микола Миколайович, Еволюційні алгоритми : підручник / М. М. Глибовець, Н. М. Гулаєва ; Нац. ун-т "Києво-Могилян. акад.". - Київ : [НаУКМА], 2013. - 826 с. : іл. - Містить покажчик.
2. PostgreSQL: Documentation: 12: 8.15. Arrays [Електронний ресурс] – Режим доступу до ресурса: https://www.postgresql.org/docs/12/arrays.html
3. sklearn.linear_model.LinearRegression — scikit-learn 0.22.2 documentation [Електронний ресурс] – Режим доступу до ресурса: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
4. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.2 documentation [Електронний ресурс] – Режим доступу до ресурса: https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

ДОДАТКИ

Додаток А
(обов'язковий)

Структура бази даних



Додаток Б
(обов’язковий)

Результати експериментів з рулеткою

Назва файлу: **dodatok2.csv**

Додаток В
(обов’язковий)

Результати експериментів з турнірним відбором

Назва файлу: **dodatok3.csv**

Додаток Г
(обов'язковий)

Розширений набір вхідних параметрів

init	estim	type	L	N
all_0	l-hamming_d	rws	10, 20, 80, 100, 200	100, 200
all_0	l-hamming_d	tournament_12	10, 20, 80, 100, 200	100, 200
all_0	l-hamming_d	tournament_2	10, 20, 80, 100, 200	100, 200
all_0	l-hamming_d	tournament_4	10, 20, 80, 100, 200	100, 200
normal	l-hamming_d	rws	10, 20, 80, 100, 200	100, 200
normal	l-hamming_d	tournament_12	10, 20, 80, 100, 200	100, 200
normal	l-hamming_d	tournament_2	10, 20, 80, 100, 200	100, 200
normal	l-hamming_d	tournament_4	10, 20, 80, 100, 200	100, 200
uniform	sigma_10	rws	10, 20, 80, 100, 200	100, 200
uniform	sigma_10	tournament_12	10, 20, 80, 100, 200	100, 200
uniform	sigma_10	tournament_2	10, 20, 80, 100, 200	100, 200
uniform	sigma_10	tournament_4	10, 20, 80, 100, 200	100, 200
uniform	sigma_2	rws	10, 20, 80, 100, 200	100, 200
uniform	sigma_2	tournament_12	10, 20, 80, 100, 200	100, 200
uniform	sigma_2	tournament_2	10, 20, 80, 100, 200	100, 200
uniform	sigma_2	tournament_4	10, 20, 80, 100, 200	100, 200
uniform	sigma_4	rws	10, 20, 80, 100, 200	100, 200
uniform	sigma_4	tournament_12	10, 20, 80, 100, 200	100, 200
uniform	sigma_4	tournament_2	10, 20, 80, 100, 200	100, 200
uniform	sigma_4	tournament_4	10, 20, 80, 100, 200	100, 200

Додаток Д
(обов’язковий)

Результати експериментів із розширеним набором параметрів

Назва файлу: **dodatok5.csv**

Додаток Е
(обов’язковий)

Результати усіх експериментів

Назва файлу: **dodatok6.csv**

Додаток Ж
(обов’язковий)

Jupyter notebook, що використовується для регресійного аналізу.

Назва файлу: **regression.ipynb**

Додаток К
(обов'язковий)

SQL-скрипт для обчилення різниці значень Pmax в різних точках для різних функцій пристосованості.

```
WITH sigma_2 AS (
    SELECT L,N, final_px as px
    FROM final_pxs_extended
    WHERE estim = 'sigma_2'
      AND type = 'tournament_2'AND init <> 'normal_with_loc'
),
     sigma_4 AS (
         SELECT L,N,final_px as px
         FROM final_pxs_extended
         WHERE estim = 'sigma_4'
           AND type = 'tournament_2'AND init <> 'normal_with_loc'
     ),
     sigma_10 AS (
         SELECT L, N, final_px as px
         FROM final_pxs_extended
         WHERE estim = 'sigma_10'
           AND type = 'tournament_2' AND init <> 'normal_with_loc'
     ),
     hamm AS (
         SELECT L, N, final_px as px
         FROM final_pxs_extended
         WHERE estim = 'l-hamming_d'
           AND type = 'tournament_2' AND init <> 'normal_with_loc'
     ),
     tmp AS (
         SELECT s2.L,s2.N,
                GREATEST(s2.px, s4.px, s10.px, hamm.px),
                LEAST(s2.px, s4.px, s10.px, hamm.px),
                (GREATEST(s2.px, s4.px, s10.px, hamm.px) - LEAST(s2.px, s4.px,
s10.px, hamm.px)) /
                LEAST(s2.px, s4.px,  s10.px, hamm.px) as p
         FROM sigma_2 s2
                  INNER JOIN sigma_4 s4 ON s2.L = s4.L AND s2.N = s4.N
                  INNER JOIN sigma_10 s10 ON s2.L = s10.L AND s2.N = s10.N
                  INNER JOIN hamm ON s2.L = hamm.L AND s2.N = hamm.N
     )
SELECT MIN(p), MAX(p), AVG(p)
FROM tmp;
```