НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.415.25

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмна система інтелектуального формування стартових популяцій»

Виконав:
студент ІІ курсу, групи ІТ-04мп
Рибніков Владислав Ігорович _____

Керівник:
професор, д.т.н.,
Сидоров Микола Олександрович _____

Рецензент:
доцент кафедри ІСТ, к.т.н.,
Шимкович Володимир Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Рибнікову Владиславу Ігоровичу

1. Тема дисертації «Програмна система інтелектуального формування стартових популяцій», науковий керівник дисертації Сидоров Микола Олександрович, д.т.н, професор, затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об’єкт дослідження – програмне забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів

4. Предмет дослідження – підходи, методи та моделі створення і супроводження програмного забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; розробка методу створення програмного забезпечення систем інтелектуального формування стартових популяцій; дослідження ефективності розробленого методу до створення програмного забезпечення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 2 плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
НК			
Перевірка на співпадіння	Лісовиченко О. І., доцент		

9.Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Формування проблематики	06.09.2021	
2	Аналіз існуючих рішень	06.10.2021	
3	Постановка задачі	16.10.2021	
4	Розробка методу	28.10.2021	
5	Розробка ПЗ	15.11.2021	
6	Виконання експериментальних досліджень	24.11.2021	
7	Оформлення пояснювальної записки	25.11.2021	
8	Подання дисертації на попередній захист	25.11.2021	
9	Подання дисертації на захист	26.12.2021	

Студент Владислав РИБНІКОВ

Науковий керівник Микола СИДОРОВ

РЕФЕРАТ

Розмір пояснювальної записки становить 90 аркушів, містить 29 ілюстрацій, 27 таблиць та 4 додатки.

Актуальність теми. У роботі розглянуто проблему створення методу для розробки програмного забезпечення для формування початкових популяцій у генетичних алгоритмах, показано основні особливості існуючих рішень проблеми, їх переваги та недоліки. Виявлено потребу в удосконаленні методу до розробки генетичних алгоритмів, в тому числі для формування початкових поколінь на основі компонентного підходу.

Мета дослідження. Основною метою роботи є покращення існуючих методів розробки програмного забезпечення генетичних алгоритмів за допомогою надання можливості до інтелектуального формування початкових популяцій за рахунок використання компонентного підходу.

Об’єкт дослідження: програмне забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів

Предмет дослідження: підходи, методи та моделі створення і супроводження програмного забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів

Для реалізації поставленої мети **сформульовані наступні завдання**:

- аналіз проблеми та існуючих рішень;
- розробка методу створення програмного забезпечення систем інтелектуального формування стартових популяцій;
- дослідження ефективності розробленого методу до створення програмного забезпечення.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано метод до розробки програмного забезпечення генетичних алгоритмів, що на відміну від існуючих, надає можливість інтелектуального задання початкових поколінь а також надає можливість задання всіх інших операторів генетичного алгоритму. Результат досягнутий шляхом розробки методу на основі компонентного підходу.

Практичне значення отриманих результатів полягає в тому, що реалізований в роботі метод для розробки програмного забезпечення генетичних алгоритмів, може виражати майже будь-який генетичний алгоритм за рахунок використання компонентного підходу, а також дозволятиме задавати інтелектуальний метод для формування стартових популяцій. Дана система може бути використана для вирішення задач з великою кількістю обмежень, як наприклад задача по формуванню тестових даних, розв'язання якої було розглянуто у роботі.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на Першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології»(SoftTech-2021) – м. Київ.

Публікації. Наукові положення дисертації опубліковані в: першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22–26 листопада 2021р. – С.16 – 20.

Рибніков В. І. Застосування компонентно-орієнтованого програмування при проектуванні генетичних алгоритмів // Матеріали Першої Всеукраїнської науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології»(SoftTech-2021) – м. Київ. НТУУ «КПІ ім. Ігоря Сікорського», 22-26 листопада 2021 р.

Ключові слова: ГЕНЕТИЧНІ АЛГОРИТМИ, ПОЧАТКОВІ ПОПУЛЯЦІЇ, КОМПОНЕНТНИЙ ПІДХІД.

ANNOTATION

The size of the explanatory note is 90 sheets, contains 29 illustrations, 27 tables and 4 appendices.

Topicality. The paper considers the problem of creating a method for software development for the formation of initial populations in genetic algorithms, and shows the main features of existing solutions to the problem, their advantages and disadvantages. The need to improve the method for the development of genetic algorithms, including for the formation of initial generations based on a component approach.

The aim of the study. The main purpose of the work is to improve the existing methods of software development of genetic algorithms by enabling the intellectual formation of primary populations through the use of a component approach.

Object of research: software system of starting populations intellectual formation in genetic algorithms

Subject of research: approaches, methods and models of creation and maintenance of software system of starting populations intellectual formation in genetic algorithms

To achieve this goal, the **following tasks** were formulated:

- analysis of the problem and existing solutions;
- development of a method of creating software for systems of intelligent formation of starting populations;
- study of the effectiveness of the developed method to create software.

The scientific novelty of the results of the master's dissertation is the proposed method for software development of genetic algorithms, which, unlike existing ones, provides the ability to intelligently generate the initial populations and also provides the ability to specify all other operators of genetic algorithms. The result was achieved by developing a method based on a component approach.

The practical value of the obtained results is that the method implemented in the work for software development of genetic algorithms can express almost any genetic algorithm through the use of a component approach, and will allow to specify an intelligent method for the formation of starting populations. This system can be used to solve problems

with many constraints, such as the problem of generating test data, the solution of which was considered in this paper.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the First All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies" (SoftTech-2021) - Kyiv.

Publications. The scientific provisions of the dissertation published in: the first All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies" (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference materials. - Kyiv. - 2021. November 22-26, 2021. - P.16 – 20.

Rybnikov V. I. Application of component-oriented programming in the design of genetic algorithms // Proceedings of the First All-Ukrainian scientific-practical conference of young scientists and students "Software Engineering and Advanced Information Technologies" (SoftTech-2021) - Kyiv. NTUU "KPI them. Igor Sikorsky ", November 22-26, 2021.

Keywords: GENETIC ALGORITHMS, INITIAL POPULATIONS, COMPONENT APPROACH.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GA - Генетичний алгоритм.

EA - Еволюційний алгоритм.

PSO (Particle swarm optimization) - метод рою частинок.

.NET - платформа розробки програмного забезпечення розроблена компанією Microsoft, яка включає в себе такі мови програмування як C#, F#, Visual Basic та інші.

NFC (Number of function calls) - метрика, яка показує кількість викликів функції за час роботи алгоритму.

OP (Opposite population) - Протилежна популяція.

DE (Differential evolution) - Алгоритм диференціальної еволюції.

API (Application programming interface) - Прикладний програмний інтерфейс, набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення

UML (Unified Modeling Language) - уніфікована мова моделювання, що застосовується у парадигмі об'єктно-орієнтованого програмування.

ВСТУП

Протягом останніх років спостерігається стрімкий ріст інтересу до галузі, яка називається генетичними алгоритмами (GA). Ця сфера перебуває на стадії колосального зростання популярності, про що свідчить збільшення кількості конференцій, семінарів та доповідей з цього приводу.

У дослідженнях машинного інтелекту має сенс вивчати два природні архетипи навчання: мозок та еволюцію. Генетичний алгоритм — це така ж точна модель еволюції, як штучна нейронна мережа — точна модель мозку. Причина, чому ми обираємо генетичні алгоритми як тему нашого дослідження, є подвійною. По-перше, процеси природної еволюції та природної генетики були висвітлені століттям величезного прогресу біології та молекулярної біології. По-друге, завдяки своїй високій надійності генетичні алгоритми виявилися перспективною технікою для багатьох програм оптимізації, проектування, автоматизованого управління та машинного навчання.

Основними етапами будь-якого генетичного алгоритму є: формування початкових поколінь, обчислення функції відповідності осіб популяції, селекція (тобто вибір найбільш відповідних індивідів), схрещування та мутація, а потім формування наступного покоління та повторення цих етапів до досягнення потрібного результату. Таким етап як селекція, схрещування та мутація виділяється досить вагома роль у більшості досліджень в цій галузі, проте дослідження формування початкових поколінь також має досить великий вплив на швидкодію та точність генетичного алгоритму.

Початкова популяція відіграє важливу роль в евристичних алгоритмах, таких як GA, оскільки допомагає зменшуватися час, необхідний цим алгоритмам для досягнення прийнятного результату. Крім того, це може вплинути на якість остаточної відповіді, заданої еволюційними алгоритмами. У цій роботі ми розробимо програмне забезпечення що дозволяє генерувати початкові покоління у генетичних поколіннях інтелектуально, тобто відповідно до задачі, яку вирішує генетичний

алгоритм. Також програмний код системи буде відкритим (Open Source), що дозволяє легко розширити таку систему іншими алгоритмами генерування або іншим функціоналом. Тому об’єктом дослідження буде програмне забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів, а предметом дослідження: підходи, методи та моделі створення і супроводження програмного забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів. Завдяки подібному дослідженню можна буде значно пришвидшити генетичні алгоритми та застосувати це у генеруванні сценаріїв тестування, що, в свою чергу, дозволить значно пришвидшити та автоматизувати процеси тестування програмного забезпечення, а також збору вимог до нього. Така система може бути використана бізнес аналітиками, або тестувальниками програмного забезпечення для опису Use-Case сценаріїв, а також в подальшому розширена інженерами програмного забезпечення для генерації коду, що дозволить автоматизувати процеси тестування.

1 АНАЛІЗ СИСТЕМ ІНТЕЛЕКТУАЛЬНОГО ФОРМУВАННЯ СТАРТОВИХ ПОПУЛЯЦІЙ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Для визначення актуальності проблематики програмного забезпечення систем інтелектуального формування стартових популяцій генетичних алгоритмів потрібно проаналізувати існуючі дослідження в даній галузі. Для цього застосуємо метод так званого систематичного огляду літератури (Systematic mapping study). Систематичний огляд літератури це така форма вторинного дослідження, що застосовує чітко визначену послідовність кроків - рис. 1.1 для аналізу та інтерпретації всіх існуючих джерел інформації.

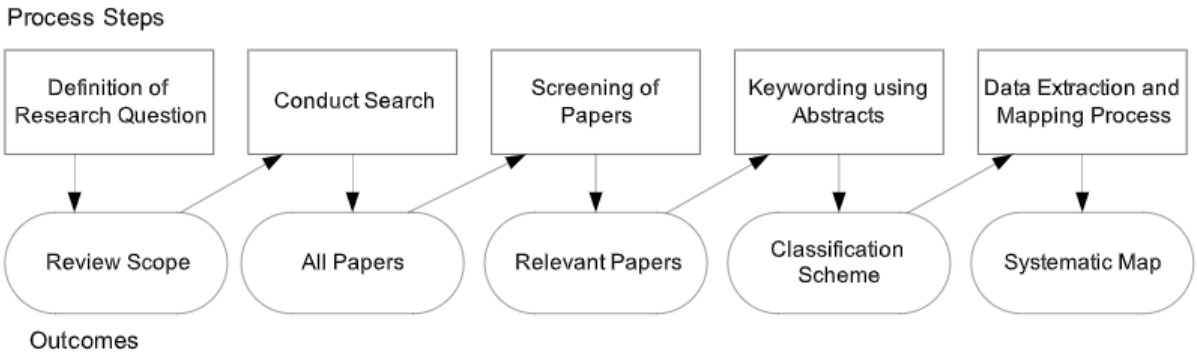


Рисунок 1.1 - Етапи систематичного огляду літератури

Отже, можна виділити такі основні етапи аналізу:

1) визначення питань обраного дослідження, тобто опис предметної області;
2) пошук первинних досліджень;
3) перегляд джерел та їх опрацювання;
4) класифікація за допомогою ключових слів;
5) виділення потрібних даних та їх візуалізація.

1.1 Опис предметної області. Визначення основних дослідницьких питань

В основі дослідження програмних систем інтелектуального формування стартових популяцій, лежить дослідження побудови генетичних алгоритмів в цілому а також систем що використовують генетичні алгоритми. Отже дослідження можна

умовно розділити на дві частини – алгоритмічну (математичні методи побудови генетичних алгоритмів та генерування стартових популяцій) та програмну (реалізація програмного забезпечення для реалізації та застосування генетичних алгоритмів та інтелектуальної ініціалізації стартових популяцій).

Для початку потрібно виділити основні дослідницькі питання при розробці програмної системи автоматизованого формування стартових популяцій, що дозволить правильно сформувати запити для пошуку первинних досліджень:

- ДП1: які основні методи до формування початкових поколінь у програмному забезпеченні на базі генетичних алгоритмів?;
- ДП2: як застосовуються генетичні алгоритми при розробці програмного забезпечення?;
- ДП3: які методи побудови програмної системи інтелектуального формування стартових популяцій?;
- ДП4: які основні вимоги до програмної системи формування стартових популяцій у генетичних алгоритмах?;
- ДП5: які програмні засоби потрібно застосувати для побудови систем з використанням генетичних алгоритмів?;
- ДП6: які основні характеристики програмних систем генерування стартових популяцій?.

Дані питання сформовано з оглядом як на алгоритмічну, так і на програмну складову роботи. Потрібно розглянути інтелектуальне формування стартових популяцій та генетичні алгоритми в цілому як явище, а потім спираючись на ці дані розглянути методи та засоби побудови програмної системи для формування початкових популяцій. Далі, на основі сформованих питань потрібно провести пошук первинних джерел.

1.2 Аналіз існуючих рішень

Для пошуку інформації та відповідей на поставлені дослідницькі питання застосуємо такі ресурси, як: IEEE Xplore Digital Library, ACM, Springer та інші

важливі репозиторії наукових робіт. Ключовими словами для пошуку первинних джерел будуть наступні: “intellectual populations formatio”, “genetic algorithm”, “initial populations”, “software”, “programming system”. Скористаємося можливостями розширеного пошуку та сформуємо наступні пошукові запити, відповідно до теми роботи:

- "genetic algorithm" AND ("initial" OR "starting") AND ("populations" OR "population");
- "genetic algorithm" AND ("software" OR "program");
- ("program" OR "system") AND ("initial" OR "starting") AND ("populations" OR "population") AND "genetic algorithm";
- “programming system” AND “genetic algorithm”;
- ("genetic algorithm" OR “GA”) AND ("automated system" OR "software") AND ("initial population" OR "starting population").

Для сформованих пошукових запитів, отримаємо наступні результати пошуку сформовані у вигляді таблиці – табл 1.1. Можливо помітити тенденції пошуку для алгоритмічної складової формування початкових поколінь.

Важливо відзначити, що формування пошукових запитів було направлене не тільки на пошук програмних рішень, а й на пошук алгоритмічних рішень, так чи інакше імплементованих у вигляді програмного забезпечення, або ж застосовуваних при розробці та проектуванні програмного забезпечення.

Таблиця 1.1 - Кількість публікацій за пошуковими запитами

Пошукова стрічка	К-ть знайдених публікацій
"genetic algorithm" AND ("initial" OR "starting") AND ("populations" OR "population")	764
"genetic algorithm" AND ("software" OR "program")	808
("program" OR "system") AND ("initial" OR "starting") AND ("populations" OR "population") AND "genetic algorithm"	251
“programming system” AND “genetic algorithm”	8
("genetic algorithm" OR “GA”) AND ("automated system" OR "software") AND ("initial population" OR "starting population")	38

Застосуємо наступну класифікацію по категоріям робіт:

1) evaluation research – В дослідженні приводиться оцінка роботи методу та його практичне значення;
2) validation Research – Досліджуваний метод є новим, тому проводиться оцінка доцільності його застосування на практиці;
3) solution proposal – Досліджуваний метод є новим рішенням існуючої проблеми, або ж розширенням іншого методу, наводиться аргументація його застосування, іноді підкріплена незначними прикладами;
4) philosophical papers – Досліджуваний метод є новим та розглядається як можлива концепція вирішення проблеми;
5) opinion papers – Робота виражає суб’єктивну думку по вирішенню проблеми за допомогою розглянутого методу;
6) experience Papers – Робота являється пояснення щодо практичного застосування певного методу.

Основні галузі в досліджуваних роботах – це:

1) genetic algorithms – роботи в яких використовуються та покращуються генетичні алгоритми;
2) starting population formation – роботи в яких досліджується питання формування початкових поколінь у генетичних алгоритмах, а також описуються варіанти їх модифікації для покращення роботи алгоритму;
3) software systems applied genetic algorithm – роботи в яких досліджується робота програмних систем, що використовують генетичні алгоритми;
4) performance improvement in software (using GA) – роботи в яких розглядається питання покращення швидкості роботи програмних систем за допомогою використання генетичних алгоритмів;
5) starting population software system – роботи в яких розглядається розробка програмних засобів формування початкових поколінь у генетичних алгоритмах.

Проаналізуємо джерела відповідно до галузей, виділивши їх кількість - табл. 1.2, що дозволить виявити основні тенденції в наукових публікаціях, а також систематизувати знайдену інформацію.

Таблиця 1.2 - Розподіл публікацій за галузями

Назва галузі	Публікації	Загальна кількість
Genetic algorithms	1, 2, 3, 4, 5, 6, 11	7
Starting population formation	7, 8, 10, 12, 14, 15, 16	7
Software systems applied genetic algorithm	6, 20, 9	3
Performance improvement in software (using GA)	2, 7, 9	3
Starting population generation software system	13	1

Можемо помітити, що найбільшу частоту мають публікації пов’язані з генетичними або еволюційними алгоритмами та методи формування стартових

популяцій, проте серед релевантних джерел що розглядають створення програмної системи формування стартових популяцій – було виявлено досить мало прикладів, що може свідчити про те що ця тема недостатньо висвітлена в наукових публікаціях.

Також досить низькі показники мають галузі автоматизованого програмування, питань продуктивності та алгоритми апроксимації, але вони найменше відносяться до теми дослідження. Серед розглянутих публікацій також були теми, які не відносяться на пряму до теми дослідження, проте були корисними для пошуку посилань які в них використовуються.

Для того щоб виявити тенденції по категоріям для кожної з цих галузей, опишемо такий розподіл у вигляді таблиці – табл. 1.3.

Таблиця 1.3 - Розподіл публікацій по категоріям

категорія	галузь				
	Genetic algorithms	Starting population formation	Software systems applied genetic algorithm	Performance improvement	Starting population software system
Evaluation research	0	0	1	1	1
Validation Research	2	3	1	0	0
Solution proposal	3	2	1	0	0
Philosophical papers	3	1	0	0	
Opinion papers	0	0	0	0	0
Experience Papers	0	0	0	0	0

Відповідно до таблиці візуалізуємо отримані результати у вигляді діаграми, для того, щоб показати основні тенденції досліджень даної предметної області відповідно до визначених галузей та категорій (рис. 1.2). Потрібно формувати діаграму розподілу по галузям та категорії а також, окремо, діаграми.

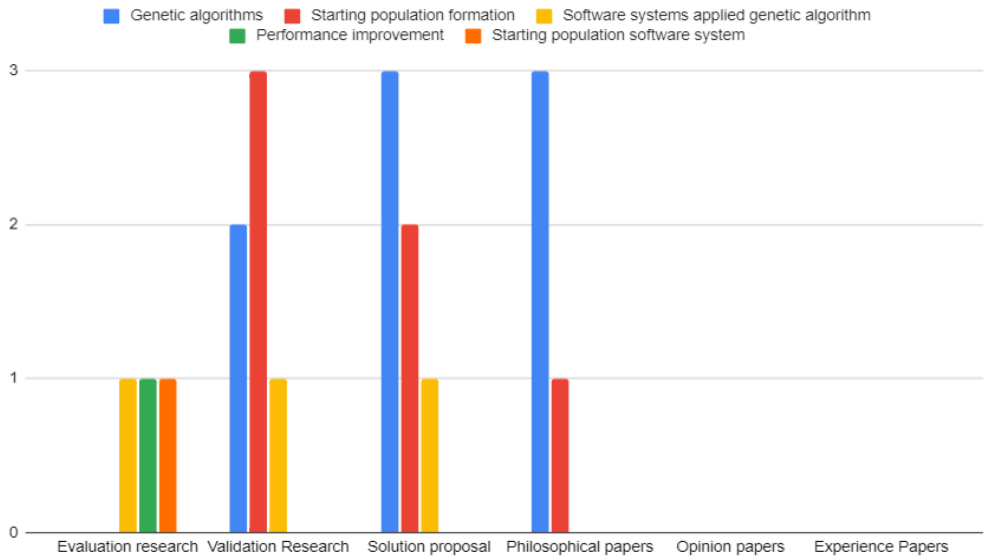


Рисунок 1.2 - Діаграма класифікації проаналізованих публікацій

Сформована діаграма дозволяє помітити значну перевагу Validation research та Solution research у галузях Genetic algorithm та Starting population formation. Також публікації в категоріях Opinion papers та Experience papers - відсутні, проте вони не представляють достатню цінність в рамках дослідження, так як дана робота сформована саме на складову впровадження програмних систем з інтелектуального формування стартових популяцій та зазвичай розглядається у дослідження типу: solution proposal, evaluation research та validation research. Розподіл за категоріями публікацій - рис. 1.3.

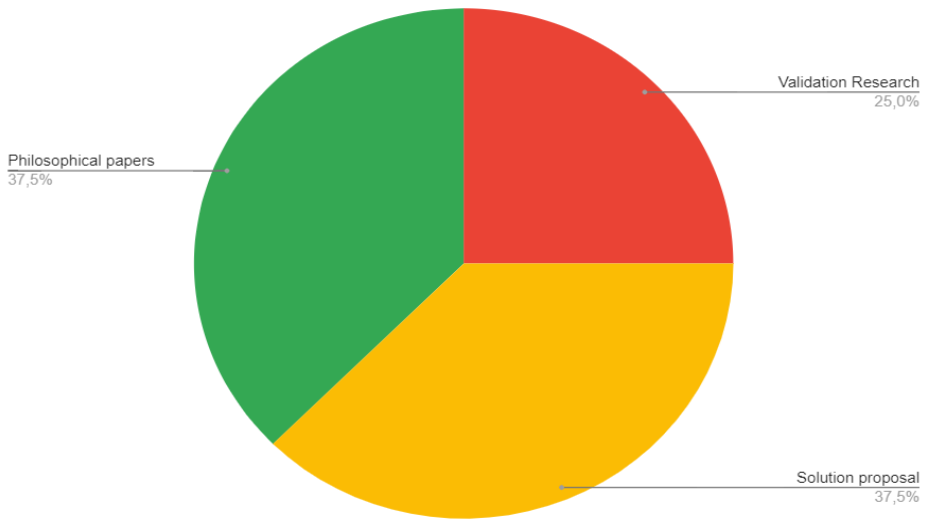


Рисунок 1.3 - Діаграма класифікації розглянутих публікацій за категоріями

Також важливими галузями публікацій для дослідження є Software systems applied genetic algorithm та Starting population formation. Так як найбільшим сектором, відповідно до діаграми є Solution proposal, тобто пропозиція конкретного рішення - програмного забезпечення, дане дослідження є актуальним. Розподіл за категоріями публікацій - рис 1.4.

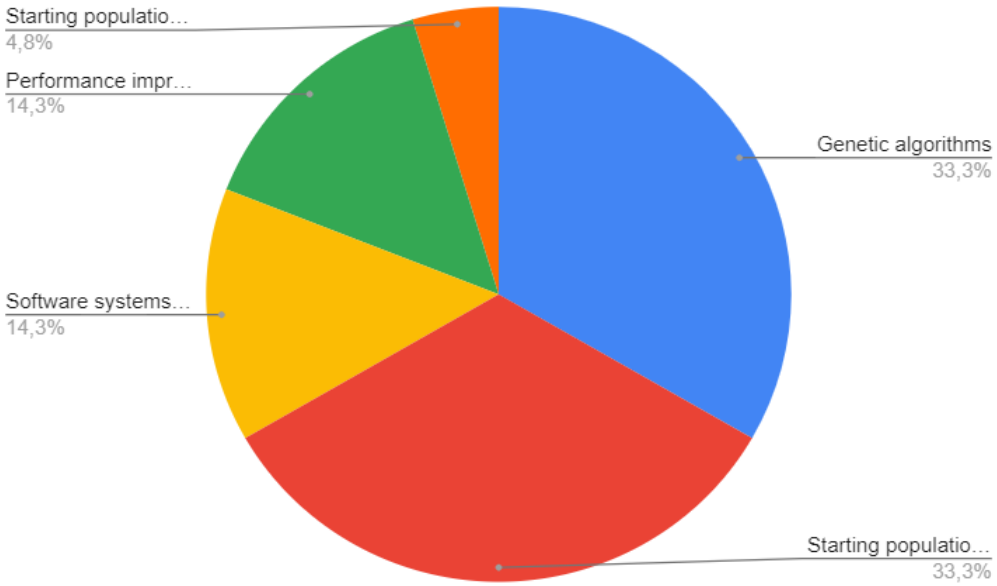


Рисунок 1.4 - Розподіл публікацій за категоріями

Дане дослідження дозволило визначити основні тенденції в науковій спільноті відносно обраної теми та виявити питання які, або недостатньо висвітлені, або ж не висвітлені взагалі. Досить мала кількість публікацій за темою Starting Population Software System, особливо в категорії Solution proposal та Validation research свідчить про актуальність обраної теми та важливість її розгляду в даній роботі. Для того щоб виявити конкретні питання до розроблюваної системи, потрібно розглянути та проаналізувати вже конкретні рішення які приводяться в обраних публікаціях. Виявивши їх слабкі та сильні сторони, можна буде сформувати функціональні вимоги до системи.

1.3 Аналіз існуючих рішень

Для того щоб визначити ключові переваги та недоліки існуючих рішень потрібно проаналізувати обрані публікації. Для початку розглянемо генетичні та еволюційні алгоритми як явище.

Для вирішення нелінійної комплексної оптимізації були введені еволюційні алгоритми (EA). Найбільш усталені та широко використовувані еволюційні алгоритми – це генетичні алгоритми (GA), метод рою частинок (PSO) та диференціальна еволюція (DE).

Генетичний алгоритм (GA) — це еволюційний алгоритм, заснований на природному виборі, розроблений Джоном Холландом. Це пошук наближених рішень задач оптимізації та пошуку методом комп'ютерного моделювання. Метою GA є досягнення кращих результатів шляхом відбору, схрещування та мутації. Основними складовими генетичних алгоритмів – є ген, хромосома та популяція. Потім за допомогою відбору, кросоверу та мутації визначають популяцію, яка дозволяє отримати оптимальне рішення.

Ідея відбору полягає в тому, щоб вибрати найкращі рішення переважно відповідно до функції придатності. Функція придатності $f(Td)$ визначається над вибіркою генетичного алгоритму і вимірює якість представленого рішення. Кросовер і мутація використовуються для створення популяції другого покоління рішень із відібраних. Кросовер може змінювати популяцію від покоління до покоління шляхом рекомбінації «батьківських» рішень. Мутація змінює деякий ген одного рішення, щоб уникнути локальних оптимальних рішень.

Оптимізація рою частинок (PSO)-один з біоінспірованих алгоритмів, і пошук простого оптимального рішення у просторі розв'язань є простим. Він відрізняється від інших алгоритмів оптимізації таким чином, що потрібна лише цільова функція, і вона не залежить від градієнта або будь -якої диференціальної форми цілі. Він також має дуже мало гіпер параметрів.

Алгоритм диференціальної еволюції (DE) – це евристична техніка глобальної оптимізації, заснована на популяції, яка проста для розуміння, проста у впровадженні, надійна та швидка. Еволюційні параметри безпосередньо впливають на продуктивність алгоритму диференціальної еволюції. Налаштування параметрів управління є глобальною поведінкою і на даний момент немає загальної теорії дослідження для контролю параметрів у процесі еволюції.

У даній роботі буде запропоновано адаптивний метод регулювання параметрів, який може динамічно регулювати параметри управління відповідно до стадії еволюції. Експерименти з оптимізації високо-розмірних функцій показали, що покращений алгоритм має більш потужні можливості глобального дослідження та більшу швидкість зближення.

Кожен з цих методів має свої особливості, сильні та слабкі сторони; але довгий час обчислень - це загальний недолік для всіх схем на основі популяцій, особливо коли простір для вирішення важко описати та дослідити. Багато зусиль уже було зроблено для прискорення конвергенції цих методів. Проте, більшість цих робіт зосереджені на впровадженні або вдосконалені роботи кросоверів і операцій мутації а також покращення механізмів відбору. Хоча ініціалізація початкових популяцій може також значно вплинути на швидкість конвергенції, а також на якість кінцевого рішення. Але є лише незначні згадки про дослідження в цій області. Мааранен та ін. запровадили квазівипадкову ініціалізацію популяції для генетичних алгоритмів [15].

Представлені результати показали, що запропонований метод ініціалізації може покращити якість остаточних рішень та поліпшити швидкість конвергенції. З іншого боку, генерація квазівипадкових послідовності – складніша, і її перевага зникає для задач вищого розміру.

В первинних джерелах також розглядається саме інтелектуальне формування початкових популяцій, засноване на введенні функцій обмежень на вхідні значення відповідно до проблеми, що вирішується [24]. Такий підхід є досить дієвим для складних задач оптимізації коли важливо підібрати відповідні початкові дані для роботи алгоритму. В такому разі вводять ряд обмежень та формують початкове

покоління відповідно до цих обмежень (інтелектуально) на першій ітерації роботи, а на інших підбирати рішення, що знаходяться “поруч” з вже підібраними. Саме такий підхід є одним з найдієвіших для роботи генетичних алгоритмів для складних задач оптимізації.

Серед розглянутих публікацій досить велику увагу приділяють саме конкретним алгоритмам та їх реалізації, проте лише одиниці сфокусовані на розробці програмного рішення, що дозволить, автоматизовано сформувати початкові покоління відповідно до задачі та надаватиме можливість задавати параметри та тестувати рішення.

Цінність такого рішення буде досить високою для студентів та науковців. Це дозволить спростити вивчення генетичних алгоритмів в цілому, тестувати алгоритми формування стартових популяцій серед запропонованого набору, а також дозволить реалізувати власні алгоритми завдяки відкритому програмному коду застосунку.

Також серед систем, що розглядають проектування та розробку програмного забезпечення для генетичних алгоритмів, були представлені реалізації на базу компонентно-орієнтованого підходу, що досить ефективно застосовується для вирішення проблеми формування початкових поколінь.

1.4 Постановка задачі

Метою роботи є покращення роботи програмного забезпечення на базі генетичних алгоритмів за рахунок розробки програмної системи, що використовує метод інтелектуального формування початкових поколінь при побудові генетичних алгоритмів. При чому, ключовими для дослідження є саме методи та засоби формування даної програмної системи.

Для досягнення мети дослідження необхідно вирішити наступні задачі:

1) проаналізувати застосування генетичних алгоритмів при розробці програмного забезпечення;
2) модифікувати існуючі рішення методом інтелектуального формування початкових поколінь;

3) розробити архітектуру програмної системи на основі компонентного підходу;
4) реалізувати програмну систему інтелектуального формування початкових поколінь;
5) проаналізувати застосувати розробленої системи для генерування тестових даних при тестуванні програмного забезпечення;
6) провести оцінку ефективності рішення та заміри продуктивності роботи алгоритму;
7) провести маркетингове дослідження впровадження розроблення програмної системи.

Так як об’єктом дослідження - є розробка програмного забезпечення системи інтелектуального формування початкових популяцій, для забезпечення реалізації всіх поставлених задач - потрібно сформувати вимоги для програмного забезпечення, що буде розроблене в ході дослідження.

Створена програмна система повинна відповідати наступним вимогам:

- можливість задання різних методів генерування початкових поколінь;
- інтуїтивно зрозумілий інтерфейс;
- можливість налаштування та використання системи для вирішення різних;
- можливість модифікації програмної реалізації системи;
- покращення швидкодії генетичного алгоритму завдяки інтелектуальному формуванні початкових популяцій.

Отже, проаналізувавши існуючі рішення та сформувавши основні задачі дослідження, потрібно перейти до огляду теоретичної бази формування початкових популяцій для генетичних алгоритмів, а також спроектувати архітектуру програмної системи, що буде реалізовувати даний алгоритм.

Висновки до розділу

В ході огляду літератури, було виявлено предметну область дослідження та сформовано основні дослідницькі питання для програмної системи інтелектуального

формування початкових поколінь за допомогою методу структурованого пошуку літератури. За допомогою даного методу, було також визначено основу для алгоритмічної та програмної реалізації.

Було виявлено та проаналізовано основні недоліки та переваги існуючих рішень, а також сформовано план для подальшої роботи. Основою для дослідження програмно системи формування початкових поколінь за алгоритмічної точки зору стало інтелектуальне формування поколінь за допомогою введення функцій обмежень. З точки зору розробки архітектури програмного забезпечення було обрано компонентний підхід до розробки. Також було визначено актуальність роботи та сфери застосування.

Основними сферами застосування програмної системи формування початкових популяцій, є вирішення задач з великою кількістю обмежень, коли застосування генетичного алгоритму з випадково сформованою початковою популяцією - є неефективним.

Програмна система інтелектуального формування початкових поколінь дозволить значно пришвидшити генетичні алгоритми. Завдяки низькій досліджуваності питання та популяризацією генетичних алгоритмів, дана робота є досить актуальною в наш час.

В результаті аналізу первинних джерел та пошуку існуючих рішень була сформована постановка задачі та описано основні етапи, щодо її реалізації. Також, були сформовані вимоги до програмної системи, що буде спроектована та розроблена в ході роботи.

2 ОГЛЯД ПІДХОДУ ДО ІНТЕЛЕКТУАЛЬНОГО ФОРМУВАННЯ СТАРТОВИХ ПОПУЛЯЦІЙ У ГЕНЕТИЧНИХ АЛГОРИТМАХ

Для розгляду методів та засобів проектування систем інтелектуального формування початкових поколінь, потрібно на сам перед розглянути методи та засоби проектування генетичних алгоритмів а також вирішення проблем оптимізації.

Методи стохастичної оптимізації – це алгоритми, в яких робиться випадковий вибір напрямку пошуку рішення на кожній ітерації алгоритму до оптимального. Алгоритми стохастичної оптимізації швидко розвивалися протягом останнього десятиліття та стали «стандартними в галузі» підходами для вирішення проблем складної оптимізації.

Завдання оптимізація - знайти мінімальне або максимальне значення для функції та її розташування, тоді як проблеми проектування мають відповідати деяким критеріям ефективності. Проте, завжди можна замінити будь-яку задачу проектування на проблему оптимізації. Налаштування параметрів системи змінюється на пошук розташування такого значення у множині вхідних даних, що оптимізує роботу системи.

Проблеми оптимізації можуть бути формалізовані наступним чином: вектор $x \in X$ де X - підмножина R^k, цільова функція - $f(x) = (f1(x), f2(x), \ldots, fk(x)) \in R^k$. Тоді, задача - мінімізувати функцію $f(x)$ так щоб вона задовольняла умовам $G(x) > 0$. Щоб уникнути розв’язування складних нелінійних рівнянь і обчислення другої похідної, щоб дізнатися, чи є точка локальним мінімумом, локальним максимумом, було розроблено багато алгоритмів оптимізації пошуку.

Один з підходів — використання стохастичного пошуку. Якщо немає обмежень на час виконання та вартості, найкраще рішення завжди можна знайти за допомогою повного пошуку. Внаслідок залежності розмірності, простір пошуку експоненціально збільшується з розмірністю. Сучасні алгоритми евристичного пошуку засновані на припущенні, що є хороші рішення швидше за все, близькі до інших відомих рішень, ніж до випадково вибраних рішень. Основна ідея евристичних алгоритмів пошуку

полягає лише в пошуку шляхів, які мають тенденцію вести до мети а не шукати весь простір. За рахунок мінімізації простору пошуку, алгоритми евристичного пошуку можуть знаходити рішення набагато швидше, ніж випадковий пошук. Для будь-якого алгоритму евристичного пошуку, потрібна функція оцінки, щоб визначити, наскільки хороший шлях вирішення. Ця додаткова оцінка вирішить, яким буде наступний шлях пошуку на наступній ітерації.

Генетичні алгоритми (GA), спочатку введені Холландом (1975), є біологічно-мотивованими стохастичними та популяційними методами пошуку, побудованими на принципах природного відбору і генетичної рекомбінації. Привабливість GA обумовлена їх простотою і надійністю, а також їх здатністю знаходити хороші рішення для складних проблем глобальної оптимізації високої розмірності, з якими дуже важко впоратися іншим методам.

Одна з основних характеристик GA, яка відрізняє їх від інших пошукових методів – це їх здатність обробляти сукупність потенційних рішень, а не змінювати одне значення. Виконуючи оптимізацію одночасно на різних регіонах у проблемній області, GA пропонує більші шанси виявити хороші результати. Генетичні алгоритми отримують здатність розпізнавати тенденції до оптимальних рішень поєднуючи принципи виживання найсильніших із випадковим обміном інформацією.

Реалізація генетичного алгоритму починається з генерування популяції можливих рішень, що зазвичай відбувається шляхом випадкового відбору. В даній роботі буде розглянуто інтелектуальний метод формування початкових поколінь, а також розроблено програмну систему, що дозволить оптимізувати застосування генетичних алгоритмів.

2.1 Огляд генетичних алгоритмів

При розробці програмної системи інтелектуального формування стартових популяцій, важливо розглянути основні складові генетичних алгоритмів, так як це на пряму впливає на розробку програмних компонентів, а також на архітектуру програмного забезпечення в цілому.

Виділяють такі основні етапи, як:

1) формування початкових популяцій - сукупностей, що складаються з усіх ймовірних рішень даної задачі. Найпопулярнішим прийомом ініціалізації є використання випадкових двійкових рядків;
2) обчислення функції відповідності осіб популяції, що призначає оцінку придатності кожній особі, яка додатково визначає ймовірність бути обраною для відтворення. Чим вищий показник придатності, тим вище шанси бути обраним для розмноження;
3) селекція (тобто вибір найбільш відповідних індивідів). На цій фазі відбираються особи для відтворення потомства. Відібрані особини потім розташовуються парами по два для посилення розмноження;
4) схрещування та мутація. Цей етап передбачає створення дитячої популяції. Алгоритм використовує оператори варіації, які застосовуються до батьківської сукупності;
5) визначення відповідності рішення та завершення роботи алгоритму. При цьому вводиться функція, що розраховує відповідність рішення та визначає умову завершення алгоритму.

Тоді алгоритм, в загальному випадку, виглядає наступним чином – рис 2.1.

На всіх етапах своєї роботи, генетичний алгоритм оперує наступними основними компонентами:

1) ген - набір значень над якими працює алгоритм (числа, символи та навіть складні структури);
2) хромосома - гени що формують певне рішення генетичного алгоритму;
3) популяція - група хромосом в певному поколінні (на певному етапі роботи алгоритму).

Дані компоненти можливо описати наступним чином - рис. 2.2.

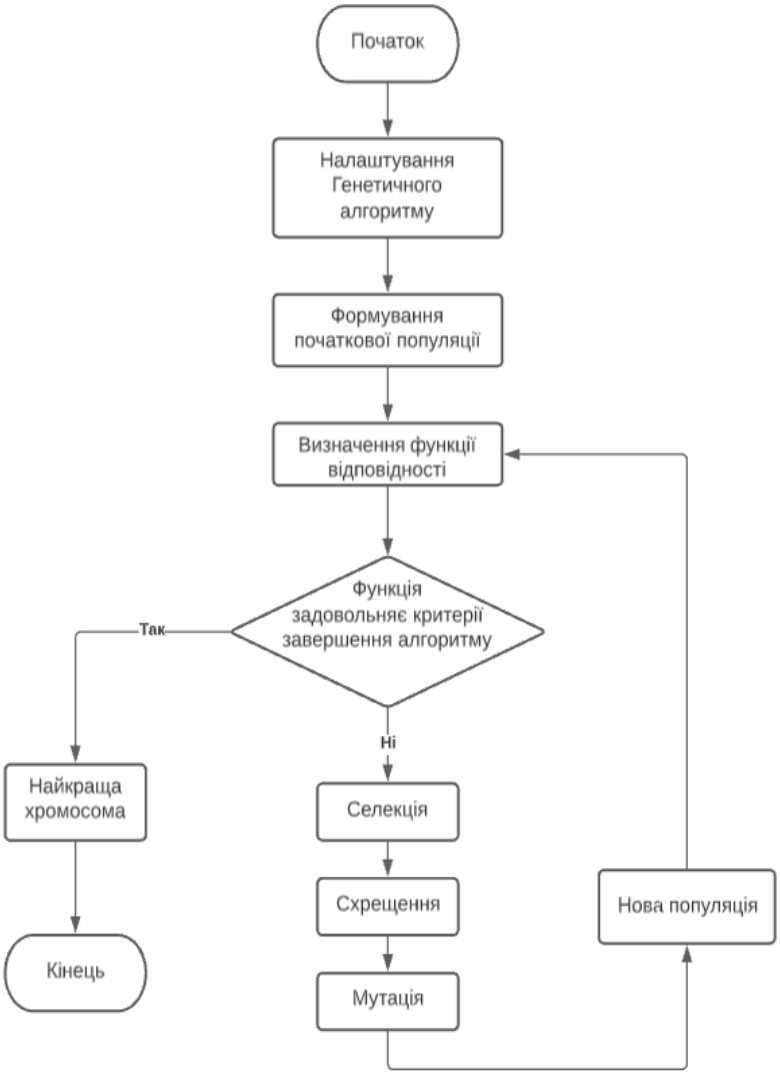


Рисунок 2.1 - Генетичний алгоритм у загальному випадку

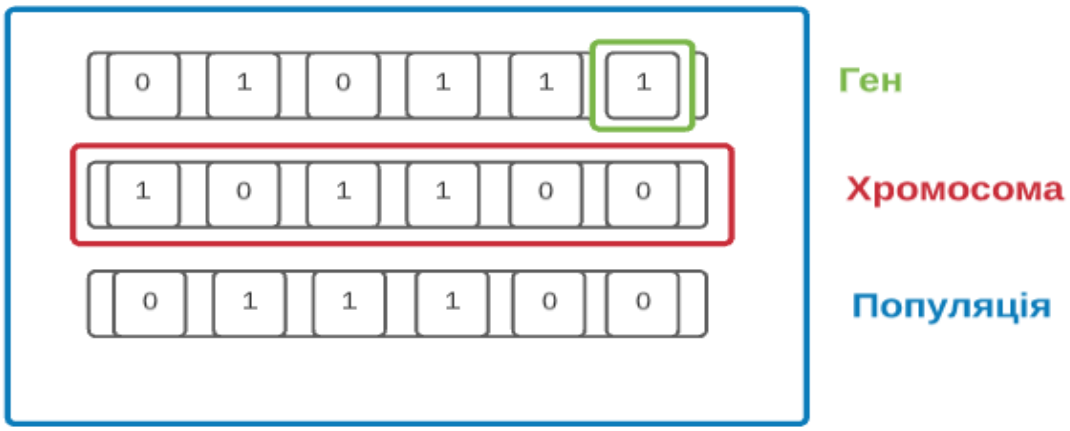


Рисунок 2.2 - Основні складові генетичного алгоритму

Виділяють наступні кроки генетичних алгоритмів, що можуть повторюватись:

1) ініціалізація;
2) визначення відповідності;
3) селекція;
4) кросинговер;
5) мутація;
6) визначення критерію завершення.

2.1.1 Ініціалізація стартової популяції

Робота генетичного алгоритму починається з визначення початкового набору індивідів, що називаються популяцією. Кожен такий індивід, являє собою можливе вирішення проблеми, що розглядається. Також, кожен індивід характеризується параметрами - набором генів, що його описують. Зазвичай підбір початкової популяції у генетичних алгоритмах відбувається випадковим чином, проте такий підхід - не завжди є оптимальним.

2.1.2 Визначення функції відповідності

Визначення функції відповідності (або ж фітнес-функції) - визначає наскільки певний індивід популяції (рішення) - наближене до оптимального рішення проблеми. При цьому кожному індивіду призначає числове значення фітнес функції, що визначає вірогідність того, що індивід буде доданий до наступного покоління. Зазвичай за найбільше значення такої функції - беруть 1. Важливим моментом при описі функції відповідності, являється правильний опис правил предметної області, що впливають на її значення.

2.1.3 Селекція хромосом

Селекція хромосом – основний етап генетичного алгоритму, так як він відповідає за вибір підходящих індивідів популяцій, що впливає загальний прогрес алгоритму на шляху до знаходження найоптимальнішого рішення. Селекція

спирається на значення фітнес-функції, чим більший показник, тим більша вірогідність того що індивід популяції буде обраний до наступного покоління.

Одними за найпоширеніших є:

- метод рулеткового відбору;
- метод елітарного відбору;
- метод часткової заміни;
- метод ранжування;
- метод Больцмана;
- метод усічення.

Розглянемо для прикладу метод рулеткового відбору. Виразимо індивідів популяції у наступному вигляді: $X_1^{(0)}, X_2^{(0)}, \ldots, X_i^{(0)}, \ldots, X_n^{(0)}$, $X_i^{(0)} = (x_{i1}^{(0)}, x_{i2}^{(0)}, \ldots, x_{id}^{(0)})$. Для того щоб обрахувати значення функції відповідності, всіх індивідів популяцій сортують в зворотньому порядку. Після цього, припустимо що $\beta \in (0,1)$, обрахуємо значення фітнес-функції для кожного індивіда, що розраховується як:

$$eval(\underline{X_i}^{(0)}) = \beta(1 - \beta)^{i-1} \quad i = 1,2,\ldots,n$$

Де, $eval(\underline{X_i}^{(0)})$ - значення і-того члена популяції і $\beta \in (0,1)$ - параметр, значення якого зазвичай обирається в межах 0.01 і 0.3. Тоді принцип роботи рудеткового методу заключається в наступному:

Вірогідність вибору і-того члена популяції під час селекції виражається наступним рівнянням:

$$P_i = \frac{eval(\underline{X_i}^{(0)})}{\sum_{i=1}^{n} eval(\underline{X_i}^{(0)})}$$

Припускаючи:

$$PP_0 = 0$$

$$PP_i = \sum_{j=1}^{i} P_i, i = 1,2,\ldots,n$$

Рулетковий метод проходить ітерації до n разів, і на кожній ітерації генерується випадкове число $\eta_k \in (0,1)$, коли таке випадкове число задовольняє умові: $PP_{i-1} \leq \eta_k < PP_i$, тоді такий індивід популяції проходить далі до кросинговеру.

2.1.4 Оператор кросинговеру хромосом

Кросовер є найважливішою фазою в генетичному алгоритмі. Для кожної пари батьків, які будуть спаровуватися, точка кросинговеру вибирається випадковим чином всередині генів.

Припустимо, що у кросовері беруть участь два індивіда - $X = \{x_i, i = 1, K\}$ та $Y = \{y_j, j = 1, K\}$.

Тоді, всередині кожної з хромосом ми випадковим чином вибираємо точку кросинговеру, що ділить їх на 2 частини - рис. 2.3. Внаслідок чого, відбувається обмін такими частини між цими двома хромосомами.

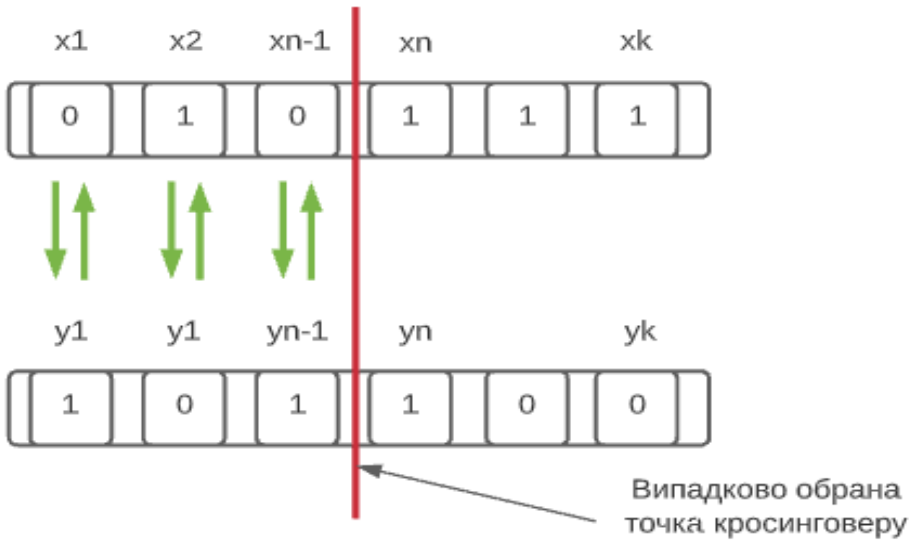


Рисунок 2.3 - Операція кросинговеру

2.1.5 Оператор мутації хромосом

У певних нових нащадків, які утворилися, деякі їх гени можуть бути піддані мутації з низькою випадковою ймовірністю. Це означає, що деякі біти в бітовому рядку можна перевернути

.Важливо відзначити, що вірогідність операції мутації - значно нижча чим вірогідність кросинговеру, адже вона спрямована на збільшення різноманіття хромосом перед майбутніми операціями кросинговеру.

На рис. 2.4 - зображено мутації одразу декількох генів, а саме зміну їх значень на протилежні.

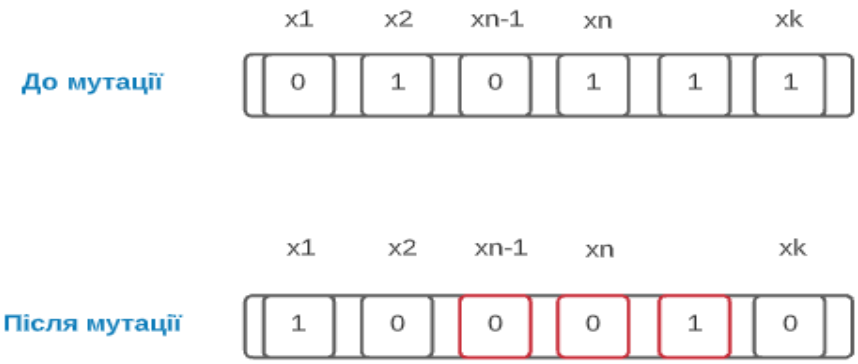


Рисунок 2.4 - Операція мутації хромосоми

2.1.6 Визначення критерію завершення роботи алгоритму

Алгоритм завершується, якщо популяція зблизилася (не дає потомства, яке значно відрізняється від попереднього покоління). Тоді кажуть, що генетичний алгоритм надав набір рішень нашої проблеми.

2.1.7 Підходи до формування початкових популяцій

Одним з можливих рішень для формування початкових поколінь – є квазівипадкова ініціалізація стартових популяцій. Розглянемо основні деталі роботи алгоритму.

Багато практичних задач оптимізації є “неопуклими” і містять кілька локальних оптимумів. Не втрачаючи загальності, ми можемо обмежити наше дослідження проблемами мінімізації. Можна розглянути проблеми глобальної оптимізації, де

безперервна дійснозначна цільова функція f мінімізована над можливою областю $S \subset R^n$. Допустима область S визначається за допомогою верхніх і нижніх границь для кожної зі змінних $x(i),\ i = 1,\ldots,n$. Рішення $\underline{x}$ називається локальним мінімумом, якщо існує сусід $\underline{x}$ такий, що $f(\underline{x}) < f(x)$, для всіх x у цьому околиці. Глобальний мінімум x* є найменшим з усіх локальних мінімумів у S.

Традиційні методи оптимізації досліджують сусідство поточного рішення і вибирають лише нові рішення, які значно знижують правильність рішення. Це зазвичай призводить до стагнації на місцевому мінімумі, що може бути далеко не так глобальний мінімум. Тому для оптимального рішення потрібні методи глобальної оптимізації. [13]

Зазвичай у генетичних алгоритмах використовують підхід випадкової генерації стартових популяцій, що підходить для більшості задач, але це значно збільшує час роботи та знижує оптимальність рішення.

В основі методу квазівипадковий ініціалізації популяцій лежать квазвипадкові послідовності. У квазігенетичних алгоритмах ми частково замінюємо псевдовипадкові числа квазі випадковими послідовностями. У цьому розділі ми обговорюємо псевдовипадкові числа та квазівипадкові послідовності загалом і вказуємо на деякі їх відмінності. Псевдовипадкові числа детерміновані, але вони намагаються імітувати незалежну послідовність справжніх випадкових чисел. Звичайні генератори псевдовипадкових чисел включають, серед іншого, лінійний конгруентний, квадратичний конгруентний, інверсивний конгруентний, паралельний лінійний конгруентний, адитивні конгруентні генератори регістрів Фібоначчі з відставанням і зворотним зв’язком Крім того, існують численні модифікації та комбінації цих генераторів.

На відміну від псевдовипадкових чисел, точки в квазівипадковій послідовності не імітують справжні випадкові точки, але замість цього намагайтеся охопити можливу область оптимальним чином. Квазі випадкові генератори генерують не числа, а послідовності точок у потрібному вимірі. Точки в квазівипадковій

послідовності взаємопов’язані так, як вони мають певні «знання» про розташування попередніх точок у послідовності. (рис. 2.5)

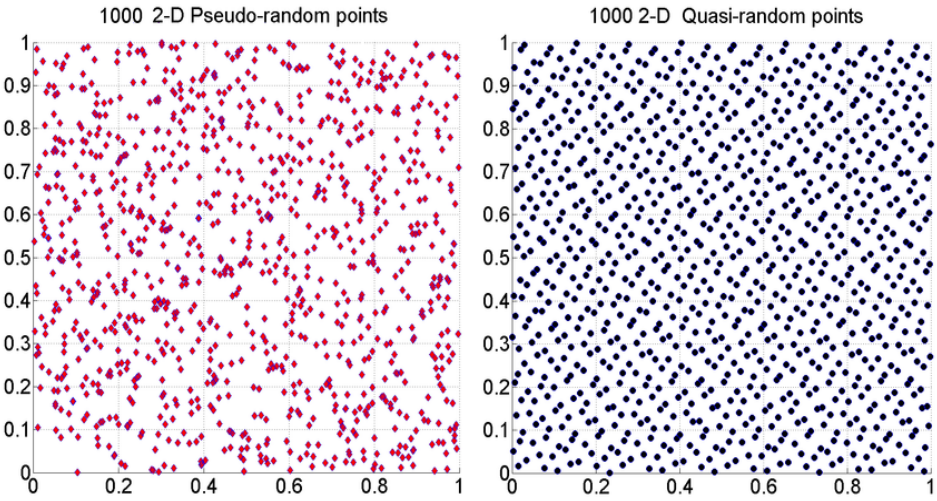


Рисунок 2.5 - Псевдо-випадкові та квазі-випадкові послідовності

У квазігенетичних алгоритмах ми використовуємо як псевдовипадкові числа, так і квазівипадкові послідовності де це доречно. Використовуються псевдовипадкові числа, які імітують справжні випадкові числа у всіх генетичних операціях, при цьому квазівипадкові послідовності, які імітують точки з досконалим рівномірним розподілом, використовуються лише при ініціалізації початкових послідовностей. Цей поділ природний, оскільки генетичні операції (відбір, кросовер і мутація) передбачають випадковість, але бажано, щоб точки в початковій сукупності мали хороший рівномірний розподіл.

2.2 Формування алгоритму інтелектуального генерування стартових популяцій

Проблеми оптимізації з багатьма обмеженнями досить поширені в інженерії та наукових застосунках. Деякі традиційні методи мають обмеження у вирішенні проблем оптимізації, включаючи режим роботи однієї точки обмеження обчислювальної ефективності, маючи слабку здатність глобального пошуку, що призводить до локально-оптимальних рішень. Найбільші переваги алгоритмів інтелектуальної оптимізації – це алгоритмічна простота, можливість багатоточкових паралельних обчислень, можливості глобального пошуку, а також цільова функція та умови обмеження. В останні роки інтелектуальні алгоритми оптимізації швидко

розвиваються розширюється та успішно застосовується в сферах системного контролю, планування виробництва, штучному інтелекті, розпізнаванні образів, плануваннях шляху тощо. Найчастіше використовуються алгоритми інтелектуальної оптимізації в: генетичних алгоритмах, алгоритмах мурашиних колоній, алгоритмах пошуку табу, оптимізації рою частинок та алгоритмі пошуку хижаків.

Загальною рисою алгоритмів інтелектуальної оптимізації є паралельність, тобто одночасне проведення незалежного пошуку з кількох точок. Ось чому необхідна початкова популяція. У разі необмежених задач оптимізації або задач з відносно невеликою кількістю обмежень, генерувати початкову сукупність легко, наприклад, випадковим чином генеруючи початкову популяцію, проте існують інші методи для створення початкової сукупності, що значно покращуючи продуктивність алгоритмів інтелектуальної оптимізації у випадках коли кількість обмежень досить велика. Коли існує багато обмежень, методи, що оперують випадковими величинами працюють не дуже добре. Для задач з великою кількістю обмежень потрібно ввести алгоритм, що буде генерувати значення початкових популяцій відповідно до цих обмежень.

У більшості задач оптимізації реального світу змінні зустрічаються в кінцевому діапазоні що описується певними обмеженнями. Вирішення таких проблем називаються задачами оптимізації з обмеженнями. Математична модель обмежених оптимізацій полягає в наступному:

$$g_j(X) \geq 0, j = 1,2,\ldots,l$$
$$h_i(X) \geq 0, j = 1,2,\ldots,m,$$

де $X = (x1, x2, \ldots, x3)$ - n-розмірна величина, або індивід у популяції, а g_j та h_i - функції обмеження що визначені на області значень X та визначають обмеження що застосовуються до всіх індивідів.

Тоді маємо наступні визначення:

1) визначення 1: індивід що задовольняє всі умови функцій обмежень - являється підходящим індивідом;

2) визначення 2: всі індивіди, що створені за допомогою алгоритму інтелектуального формування початкових поколінь - формують початкову популяцію;
3) визначення 3: індивід популяції в рамках регіону відповідності (не за границею відповідності або ж на її межі) називається внутрішньою точкою або строгою внутрішньою точкою популяції.

При використанні інтелектуальних алгоритмів оптимізації для вирішення задач обмеженої оптимізації - задача генерування початкової сукупності ділиться на два кроки. Перший крок полягає у використанні генетичного алгоритму для отримання першої внутрішньої точки початкової популяції; другий крок полягає у створенні решти можливих індивідуумів початкової популяції за ітераціями. Даний метод розбиває проблему початкової сукупності генерування на два етапи:

1) задача генерування першої внутрішньої точки перетворюється на розв’язання n задач без обмежень оптимізації за методом внутрішньої точки;
2) потім генетичний алгоритм, який використовується для вирішення цієї задачі оптимізації без обмежень, отримує першу початкову внутрішню точка. Це ключ до подальшого створення початкової популяції.

В такому випадку, перевагами над методами заснованими на випадковому формуванні початкових популяцій, є більш відповідна вибірка, що зменшує кількість ітерацій генетичного алгоритму до досягнення потрібної оптимальності рішення.

При використанні інтелектуальних алгоритмів оптимізації для вирішення обмеженої оптимізації, задача генерування початкової сукупності ділиться на два кроки. Перший крок полягає у використанні генетичного алгоритму для отримання першої внутрішньої точки початкової популяції; другий крок полягає у створенні решти можливих індивідуумів початкового населення за ітераціями. Запропонований метод розбиває проблему початкової сукупності генерування на два етапи: задача генерування першої внутрішньої точки перетворюється на розв’язання n задач без обмежень оптимізації за методом внутрішньої точки, а потім генетичний Алгоритм,

який використовується для вирішення цієї задачі оптимізації без обмежень, отримує першу початкову внутрішню точку. Це ключ до подальшого створення початкової популяції. Опишемо набір кроків, необхідних для реалізації алгоритму інтелектуального формування початкових популяцій.

1) Припустимо, що розмір початкової популяції - буде m. Тоді m індивідів будуть згенеровані випадковими чином та і-тий індивід буде: $X_i^{(k)} = (x_{i1}^{(k)}, x_{i2}^{(k)}, \ldots, x_{id}^{(k)})$. При цьому, $a_i = (a_{i1}, a_{i2}, \ldots, a_{id})$ та $b_i = (b_{i1}, b_{i2}, \ldots, b_{id})$ - верхня та нижня границі індивіда відповідно;
2) Для кожного індивіда вектори S_k та T_k , будуть згенеровані відповідно до:

$$T_k = \{j | g_j(X_i^{(k)}) > 0, 1 \leq j \leq l\}$$
$$S_k = \{j | g_j(X_i^{(k)}) \leq 0, 1 \leq j \leq l\}$$

Де T_k - вектор індексів, що задовольняють умовам обмежень, а вектор S_k - набір індексів, що не задовольняють умовам;

3) Повторити ітерації допоки, вектор S_k - не буде порожнім.

Після цих кроків, сформована початкова популяція подається на вхід генетичному алгоритму, з обраними операторами селекції, кросинговеру та мутації, відповідно до задачі, що вирішується. Зазвичай правила описані для формування початкового покоління також відображаються в операторі селекції, тому сформована початкова популяція з більшою вірогідністю буде обрана для кросинговеру та мутації. Таким чином, буде зменшено кількість ітерацій, необхідних для знаходження рішення генетичного алгоритму, а отже, можна припустити, що загальна швидкість роботи алгоритму буде зменшена а точність - покращена.

2.3 Застосування компонентного підходу розробки системи

Для забезпечення усіх функціональних і нефункціональних вимог до ПЗ було обрано компонентно-орієнтований підхід до розробки, що заснований на побудові незалежних компонентів та дозволяє змінювати функціональні частини алгоритми без зміни загального програмного рішення.

Компонентно-орієнтоване програмування вимагає як систем, які підтримують цей підхід, так і програмістів, які дотримуються його дисципліни та основних принципів. Однак часто важко відрізнити справжній принцип від простої особливості технології компонентів, що використовується. Оскільки допоміжні технології стають потужнішими, безсумнівно, програмна інженерія розширить своє розуміння того, що є компонентно-орієнтованим програмуванням, і охопить нові ідеї, а основні принципи продовжуватимуть розвиватися.

Тим не менш, Найважливіші принципи компонентно-орієнтованого програмування включають:

- розділення інтерфейсу та реалізації;
- бінарна сумісність;
- мовна незалежність;
- прозорість розташування;
- управління паралельністю;
- контроль версій;
- безпека компонентів.

Основний принцип компонентно-орієнтованого програмування полягає в тому, що основним блоком програми є двійково-сумісний інтерфейс. Інтерфейс надає абстрактне визначення контракту між клієнтом і об’єктом. Цей принцип контрастує з об’єктно-орієнтованим поглядом на світ, який ставить об’єкт, а не його інтерфейс, у центр. Інтерфейс – це логічне угруповання визначень методів, яке діє як контракт між клієнтом і постачальником послуг. Кожен постачальник може надати власну інтерпретацію інтерфейсу, тобто свою власну реалізацію. Інтерфейс реалізований бінарним компонентом чорного ящика, який повністю інкапсулює його внутрішні особливості. Цей принцип відомий як відділення інтерфейсу від реалізації.

Програмний компонент – це одиниця композиції з інтерфейсами, визначеними контрактом, і явними залежностями від контексту. Щоб зробити компонент доступним, необхідно визначити будь-який необхідний або наданий інтерфейс, і для того, щоб його використовувати, необхідно створити необхідну відповідність інших

компонентів зазначеним інтерфейси. В якості моделювання було обрано UML2.0 мову для опису архітектури компонентів (рис. 2.6), де зовнішнє уявлення (діаграма компонентів) зображує компоненти як чорні скриньки, що показують їх необхідні контракти для взаємодії з іншими компонентами.

Рисунок 2.6 - Зображення компоненту

Щоб використовувати компонент, клієнту потрібно лише знати визначення інтерфейсу (тобто контракт на обслуговування) і мати можливість отримати доступ до бінарного компонента, який реалізує цей інтерфейс. Цей додатковий рівень опосередкованості між клієнтом і об’єктом дозволяє замінити одну реалізацію інтерфейсу іншою, не впливаючи на код клієнта. Для використання нової версії клієнта не потрібно перекомпілювати. Іноді клієнта навіть не потрібно вимикати, щоб виконати оновлення. За умови, що інтерфейс незмінний, об’єкти, що реалізують інтерфейс, можуть вільно розвиватися, а нові версії можна вводити плавно та легко. Для реалізації функціональних можливостей, які обіцяє інтерфейс усередині компонента, використовуються традиційні об’єктно-орієнтовані методології, але отримані ієрархії класів зазвичай простіші та легші в управлінні.

Зазвичай, розробка за допомогою компонентів потребую визначення певного підходу до проектування доменної моделі для правильного визначення основних компонентів системи. Зазвичай розробка починається з аналізу та визначення предметної області. Після цього, формується архітектура програмного забезпечення та описуються основні програмні компоненти системи. Потім, для кожного компоненту також застосовуються аналіз та детальний опис його структури. Таким

чином, ітеративно визначивши структуру кожного компоненту, можливо спроектувати програмну систему на базі компонентів - рис 2.7.

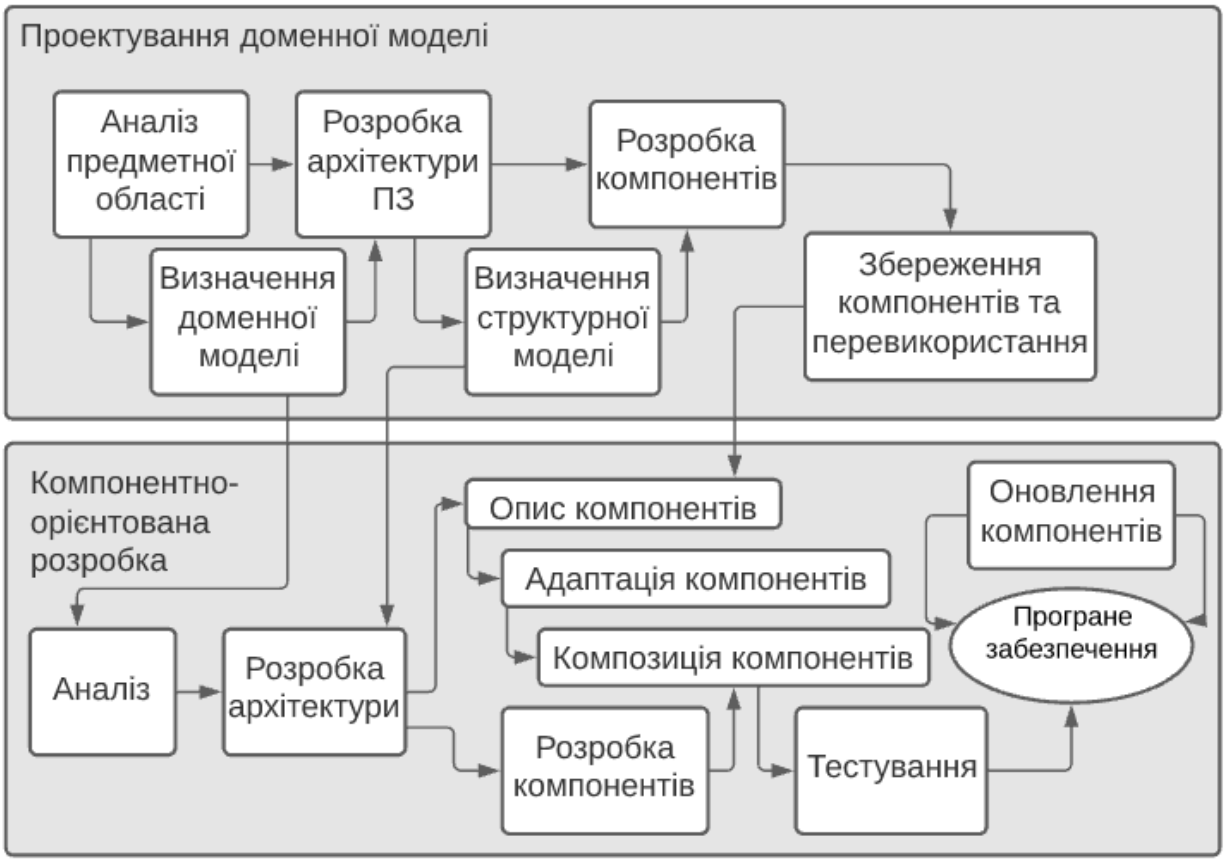


Рисунок 2.7 - Розробка програмного забезпечення на основі компонентного підходу.

На рисунку помітна важливість саме проектування предметної області над розробкою. В основі формування програмних компонентів лежить її структурний аналіз а також аналіз системи в цілому.

2.3.1 Визначення основних компонентів генетичного алгоритму

Відповідно до визначеного раніше компонентно-орієнтованого підходу – в попередніх розділах було проаналізовано предметну область, а саме розробка генетичних алгоритмів.

При проектуванні системи за допомогою компонентно-орієнтованого підходу, перш за все, потрібно визначити основні компоненти системи та методи їх реалізації. В попередніх розділах вже було виділено основні оператори або ж етапи генетичного

алгоритму. Компоненти будь майже в точності відображати структуру основних етапів генетичного алгоритму.

Опишемо такі основні компоненти - табл. 2.1.

Таблиця 2.1 - Компоненти генетичного алгоритму та методи їх реалізації

Компонент	Метод реалізації
Кодування рішення	двійкове, реальне
Формування початкової популяції	Випадкове, задане, квазі-випадкове, інтелектуально сформоване
Відбір	турнір, рулетковий
Кросинговер	Двійковий: одноточковий, двоточковий; Реальний: AX, BLX-α, BLX -α − β, WHX.
Мутація	Двійкова: один або декілька бітів.

В такому випадку можливо схематично зобразити вибір конкретних компонентів для побудови генетичного алгоритму з заданим набором параметрів - рис. 2.8.

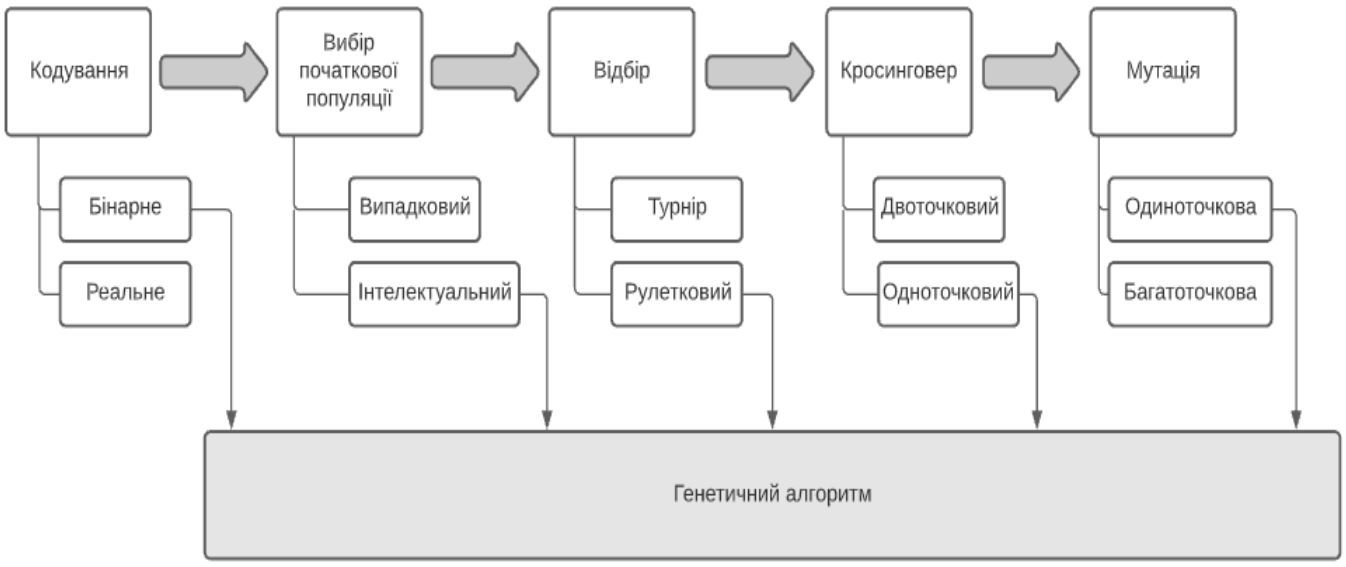


Рисунок 2.8 – Набір компонентів для формування генетичного алгоритму

Висновки до розділу

На даному етапі роботи було розглянуто основні складові генетичних алгоритмів та принципи їх роботи. Також було детально розглянуто етап побудови

початкових популяцій та обґрунтовано його важливість для вирішення задач з великою кількістю обмежень. Було розглянуто методи випадкового та квазі-випадкового формування початкових поколінь. Також, було сформовано метод інтелектуального формування початкових популяцій та його переваги над вище-згаданими методами.

Було сформовано метод інтелектуального формування початкових популяцій, що буде реалізований програмно в наступних розділах роботи.

Також було розглянути компонентний підхід до розробки системи, що реалізує генетичний алгоритм. Даний підхід дозволить описати потрібну структуру системи інтелектуального формування початкових поколінь.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО ФОРМУВАННЯ СТАРТОВИХ ПОПУЛЯЦІЙ

Один з основних етапів побудови системи інтелектуального формування початкових популяцій є проектування та розробка програмного забезпечення. При цьому важливо враховувати особливості генетичних алгоритмів розглянуті в попередньому розділі. Враховуючи ці особливості, було обрано компонентно-орієнтовану архітектуру, а також мову програмування C#, яка надає як можливості об’єктно-орієнтованого програмування для побудови незалежних, слабко-зв’язаних компонентів, так і можливості функціонального програмування, для опису математичних операцій.

В даному розділі буде розглянуто основні функціональні та нефункціональні вимоги до ПЗ, його архітектуру та методи до його побудови.

3.1 Опис функціоналу програмного забезпечення

Відповідно до розглянутої у попередньому розділі структури алгоритму ми маємо наступні компоненти (блоки):

- компонент формування хромосоми;
- компонент побудови початкової популяції;
- компонент визначення відповідності;
- компонент відбору популяції;
- компонент кросинговеру;
- компонент мутації;
- компонент визначення завершення роботи.

Основний функціонал системи зосереджений навколо компоненту побудови початковий популяції використовуючи інтелектуальне задання правил предметної області.

Функціональні вимоги до ПЗ:

- можливість задання правил до інтелектуального формування початкових поколінь;
- можливість формування хромосоми для генетичного алгоритму;
- можливість вибору методу відбору популяцій;
- можливість задання методу мутації;
- можливість задання методу кросинговеру.

Нефункціональні вимоги до ПЗ:

- швидкість роботи алгоритму;
- точність роботи;
- якість програмного забезпечення.

Для того щоб забезпечити було обрано мову програмування C# а також .NET5.0 для виконання коду. Враховуючи об’єктно-орієнтовані можливості для опису інтерфейсів та компонентів додатку та функціональні можливості для опису математичної моделі, можливо буде забезпечити всі функціональні та нефункціональні

3.2 Проектування архітектури програмної системи генерування стартових популяцій

Давши визначення компонентному підходу до проектування системи а також описавши основні компоненти генетичного алгоритму, ми можемо сформувати архітектуру системи за допомогою компонентів. При цьому крім зазначених раніше компонентів, потрібно описати компонент самого генетичного алгоритму, що буде формувати основний алгоритм роботи системи та оркеструвати роботу інших компонентів. Отже, в центрі системи інтелектуального формування початкових популяцій буде реалізація компоненту роботи генетичного алгоритму, який буде залежати від реалізації компонентів формування початкової популяції, селекції, кросинговеру та мутації, що необхідні для повноцінної роботи генетичного

алгоритму, а для інтелектуального формування початкових популяцій буде реалізовано метод, який буде відповідати заданому контракту поведінки.

Описавши компоненти генетичного алгоритму таким чином, ми можемо уявити рішення майже будь-якої задачі. Навіть якщо, не вистачає конкретної реалізації певного компоненту генетичного алгоритму, ми легко можемо визначити власний відповідно до його контракту.

Відповідно до описаних компонентів, що являються основними компонентам генетичного алгоритму у загальному випадку, потрібно описати компоненти програмної системи, що будуть відображати їхню функціональність.

Визначимо компоненти програмного забезпечення генетичного алгоритму та сформуємо діаграму компонентів системи - рис. 3.1.

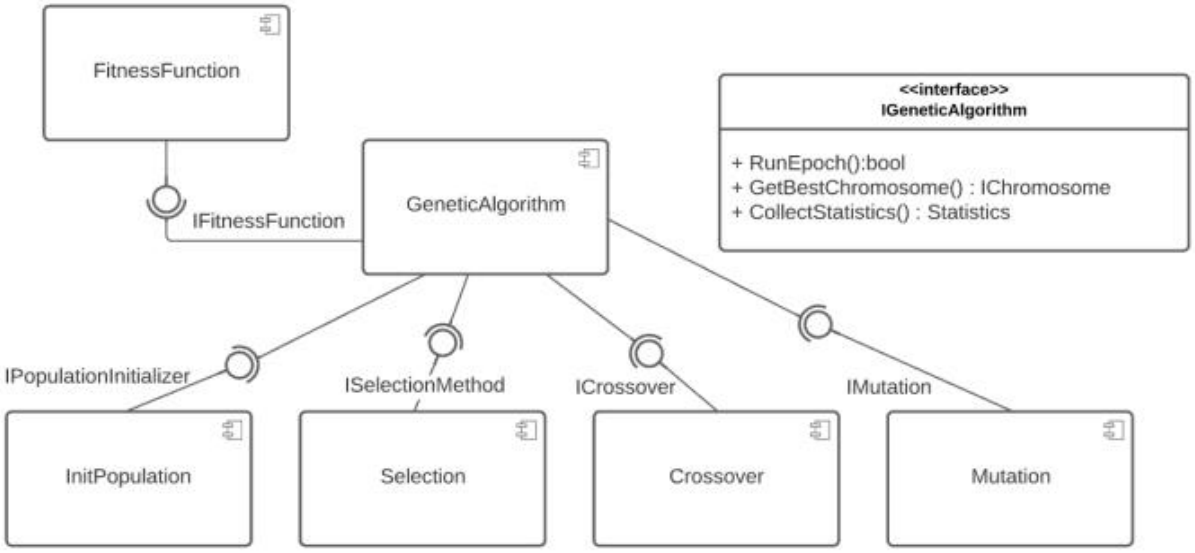


Рисунок 3.1 - Опис компонентів системи

Відповідно до діаграми компонентів, сформованої раніше, опишемо основні інтерфейси генетичного алгоритму.

Отже основними інтерфейсами системи будуть:

- IGeneticAlgorithm - Реалізація загальної роботи алгоритму. В даному випадку реалізація компоненту буде залишатись незмінною та буде залежати від інших компонентів;
- IFitnesFunction - Реалізація функції відповідності;

- IPopulationInitializer - Реалізація методу генерування початкової популяції. В рамках дослідження буде розглянуто 2 реалізації - випадкову ініціалізацію та інтелектуальну;
- ISelectionMethod - Метод вибору підходящих для наступного покоління індивідів популяції;
- ICrossover - Метод кросинговеру;
- IMutation - Метод мутації.

Повну діаграму класів зображено у додатку Б.

Також, система буде надавати функціональність для налаштування генетичного алгоритму а також методу формування початкових популяцій.

Опишемо Use-Case діаграму користувача системи - рис. 3.2.

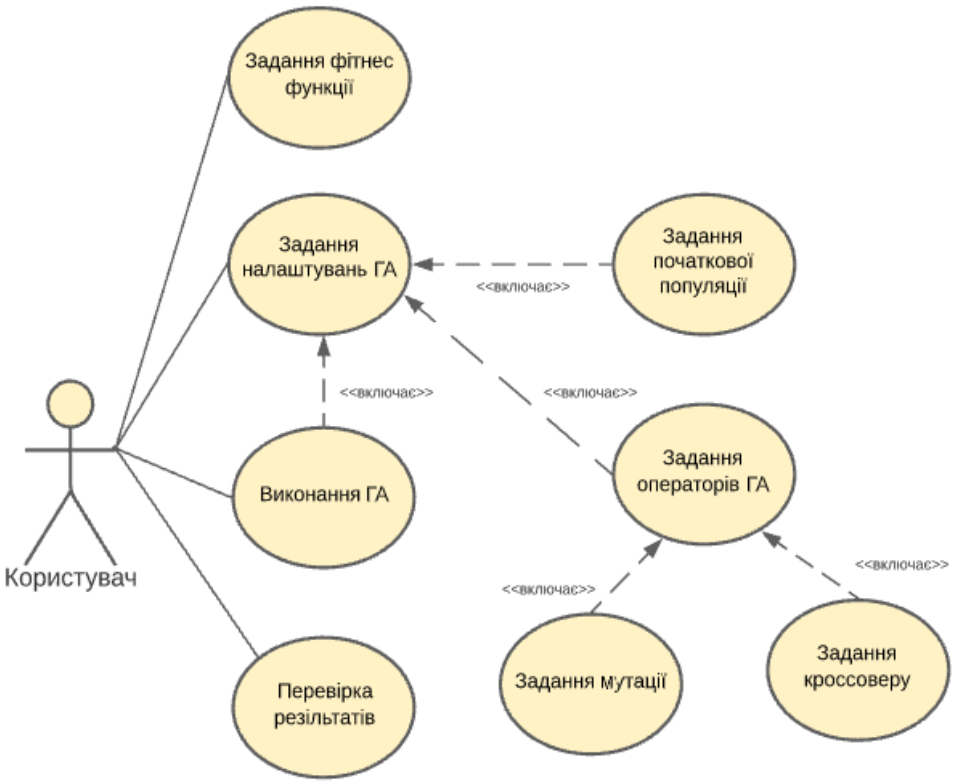


Рисунок 3.2 - UML діаграма використання розробленої системи

Розглянуту ідею представлення генетичного алгоритму у вигляді чітко визначених компонентів, можливо досить вдало масштабувати до засобу візуального програмування, коли кожному такому компоненту відповідає UI компонент, з яким може взаємодіяти користувач. В такому випадку налаштування генетичного

алгоритму для вирішення конкретної задачі можливо буде не лише програмно, імплементуючи певні інтерфейси а й через графічний інтерфейс. Реалізувавши такі компоненти у вигляді незалежних модулі, можна було б досягти їх незалежного розгортання, що дозволяло б розробляти та модифікувати генетичні алгоритми значно швидше.

3.3 Структура основних програмних компонентів

Для розробки програмного забезпечення системи інтелектуального формування стартових популяцій, необхідно детально описати структуру кожного компоненту системи.

3.3.1 Компонент формування початкових популяцій

Вибір початкової популяції - ключовий для даного дослідження. Тому, розглядається та проектується програмна реалізація декількох компонентів для формування початкових популяцій. Юзкейс діаграма користувача при заданні алгоритму формування початкової популяції виглядає наступним чином - рис 3.3.

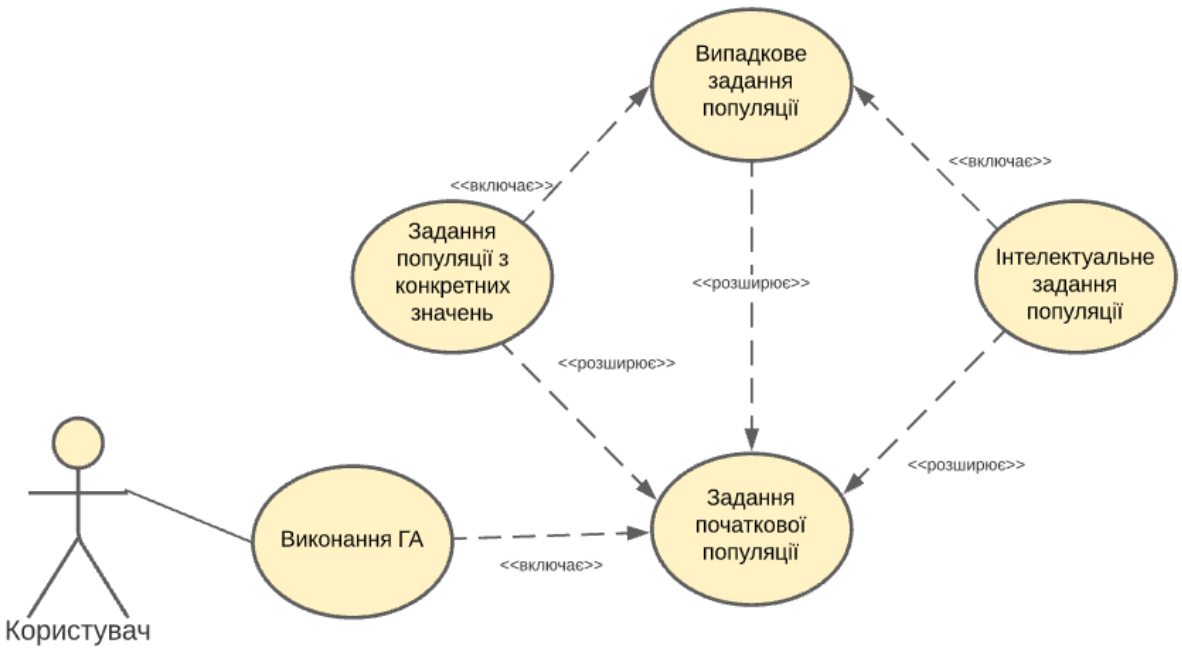


Рисунок 3.3 - UML діаграма вибору алгоритму формування початкової популяції.

Для першого етапу початкової ініціалізації зазвичай використовують метод випадкового генерування генів у популяції. Реалізація такого підходу мовою програмування C#, може виглядати наступним чином (рис. 3.4).

```
1 usage  new *
private TPopulation Generate<TPopulation, TGen>(
    Func<IEnumerable<TGen>, TPopulation> formPopulation,
    Func<int, TGen> generateGen,
    int length)
{
    var genes = new List<TGen>(length);
    for (var i = 0; i < length; i++)
    {
        var genNumber:int = Random.Next(0, _allSteps.Length);
        genes.Add(item:generateGen(genNumber));
    }

    return formPopulation(genes);
}
```

Рисунок 3.4 - Випадкове генерування стартових популяцій

Для даної реалізації, опишемо інтерфейс компонента генерації початкових популяцій у загальному випадку – Рис. 3.5.

```
5 usages  2 inheritors  new *
public interface IInitialPopulationGenerator<TGen, TPopulation>
{
    2 usages  2 implementations  new *
    TPopulation Generate(
        Func<IEnumerable<TGen>, TPopulation> formPopulation,
        Func<int, TGen> generateGen, int length);
}
```

Рисунок 3.5 – Опис інтерфейсу генерації початкових поколінь

В даному випадку, метод приймає типізовані функцій для створення генів та популяцій, що дозволяє зобразити їх не тільки в числовому вигляді, а й у вигляді будь-якого складного типу C#.

Реалізація компоненту для інтелектуального формування початкових популяцій буде відповідати описаному контракту, проте на вхід буде приймати набір допустимих значень, а також набір функцій обмежень. При кожній ітерації алгоритму, обирається випадковим чином індивід і заданого переліку допустимих значень, після чого перевіряються всі умови. Якщо індивід задовольняє всім умовам - він входить до початкової популяції. Алгоритм роботи - зображено на рис. 3.6. Повний код реалізації компоненту знаходиться в Додатку Б.

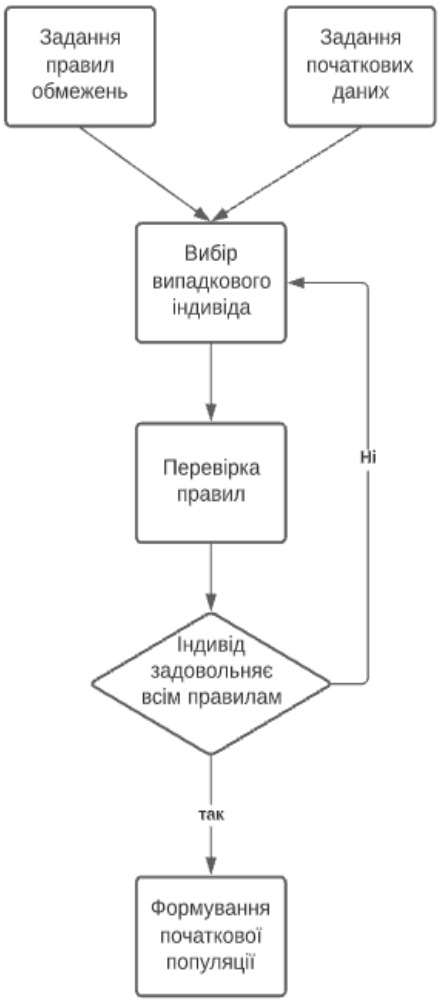


Рисунок 3.6 - Алгоритм інтелектуального формування початкової популяції

3.3.2 Компонент функції відповідності

Компонент задання функції відповідності, також досить важливий для роботи генетичного алгоритму, адже саме від його вибору, залежить рішення генетичного алгоритму. У випадку системи, що розробляється, компонент задання функції відповідності буде представляти собою інтерфейс, що визначає лише один метод - Evaluate - рис. 3.7.

```
public interface IFitnessFunction
{
  /// <summary>Evaluates chromosome.</summary>
  /// <param name="chromosome">Chromosome to evaluate.</param>
  /// <returns>Returns chromosome's fitness value.</returns>
  /// <remarks>The method calculates fitness value of the specified
  /// chromosome.</remarks>
  double Evaluate(IChromosome chromosome);
}
```

Рисунок 3.7 - Визначення програмного інтерфейсу компоненту функції відповідності

В даному випадку, компонент, що реалізує відповідний інтерфейс, повинен визначити метод, в якому на основі значення хромосоми порахувати значення відповідності для переходу до наступного покоління. Чим вище значення - тим більша вірогідність переходу.

3.3.3 Компонент селекції

Компонент селекції відповідає за вибір наступного покоління базований на раніше обрахованих значення в компоненті визначення відповідності. Раніше вже були описані варіанти реалізації алгоритмів селекцій, проте важливо визначити структуру компоненту у вигляді контракту для визначення та підтримки нових реалізацій - рис. 3.8.

```
public interface ISelectionMethod
{
  /// <summary>Apply selection to the specified population.</summary>
  /// <param name="chromosomes">Population, which should be filtered.</param>
  /// <param name="size">The amount of chromosomes to keep.</param>
  /// <remarks>Filters specified population according to the implemented
  /// selection algorithm.</remarks>
  void ApplySelection(List<IChromosome> chromosomes, int size);
}
```

Рисунок 3.8 - Визначення програмного інтерфейсу компоненту селекції

В даному випадку, на вхід компонент приймає набір всіх хромосом з розрахованими значеннями відповідності та розмір популяції в конкретному поколінні. Залежно від обраної реалізації, робота алгоритму буде заснована на виборі (зазвичай випадковому) індивідів для наступної популяції відсортованих за значеннями фітнес функції.

3.3.4 Компонент кросинговеру

Компонент кросинговеру разом з компонентом мутації формують математичну основу роботи генетичного алгоритму. Вони допомагають підтримувати різноманітність індивідів в конкретній популяцій, тому правильний вибір реалізації цих компонентів також є досить важливим.

Компонент кросинговеру - повинен приймати на вхід дві хромосоми та змінювати значення їх генів - рис. 3.9.

```
public  interface ICrossover
{
    /// <summary>
    /// Applies crossover to <see cref="left"/> <see cref="IChromosome"/>
    /// and <see cref="right"/> <see cref="IChromosome"/>
    /// </summary>
    /// <param name="left"></param>
    /// <param name="right"></param>
    void Apply(IChromosome left, ICrossover right);
}
```

Рисунок 3.9 - Визначення програмного інтерфейсу компоненту кросинговеру

3.3.5 Компонент мутації

Визначення компоненту мутації досить схоже на визначення компоненту кросинговеру, проте в перетвореннях генів бере участь лише одна хромосома, в якій випадковим чином відбувається заміна одного або декількох генів.

Реалізація програмного компоненту - виглядає наступним чином - рис. 3.10.

```
public interface IMutation
{
    /// <summary>
    /// Applies mutation to specified <see cref="IChromosome"/>
    /// </summary>
    /// <param name="chromosome"></param>
    void Apply(IChromosome chromosome);
}
```

Рисунок 3.10- визначення програмного інтерфейсу компоненту мутації

Висновки до розділу

Генерування початкових популяцій випадково - підходить для більшості задач, що вирішуються за допомогою генетичних алгоритмів, проте іноді такий спосіб може бути не оптимальним, тому можемо ввести поняття інтелектуального формування початкових популяцій. В такому випадку початкові покоління можливо буде сформувати відповідно до задачі алгоритму.

В даному розділі було визначено метод до побудови системи а також описано її архітектуру та програмну реалізацію. Було розглянути компонентний підхід до розробки та наведено переваги його використання.

Запропонований підхід до розробки генетичних алгоритмів за допомогою незалежних компонентів, дозволяє значно пришвидшити розробку та модифікацію генетичних алгоритмів а також налаштувати незалежне розгортання таких

компонентів. Побудована архітектура значно покращує гнучкість програмного рішення, так як дозволяє змінювати окремі компоненти без зміни основної логіки ПЗ.

Також, таке рішення можливо було б застосувати для розробки застосунку візуального програмування генетичних алгоритмів, що надавало би можливості вирішення конкретних проблем за допомогою зміни набору компонентів у графічному редакторі.

Було наведено приклади архітектури за допомогою компонентів, а також сформовано діаграму класів та розглянуто ефективність такого підходу для розробки генетичних алгоритмів.

4 ЗАСТОСУВАННЯ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО ФОРМУВАННЯ СТАРТОВИХ ПОПУЛЯЦІЙ

В даному розділі буду наведено приклади застосування системи для вирішення конкретних задач та доведено ефективність запропонованого рішення. Буде продемонстровано роботу графічного додатку для моделювання роботи алгоритму для вирішення задач оптимізації, апроксимації та обчислення часових рядів, а також вирішено реальну задачу формування тестових сценаріїв з використанням інтелектуального формування початкових популяцій.

4.1 Приклад графічного інтерфейсу для моделювання роботи алгоритму

Для демонстрації та моделювання роботи алгоритму, було розроблено графічний інтерфейс, що містить такі приклади застосування системи:

- апроксимація функцій;
- обчислення числових рядів;
- оптимізація функцій.

Також для кожного прикладу створено набори вхідних даних у форматі CSV, що можуть бути завантажені та оброблені системою. При запуску застосунку - доступне меню вибору прикладу для моделювання роботи - Рис. 4.1.

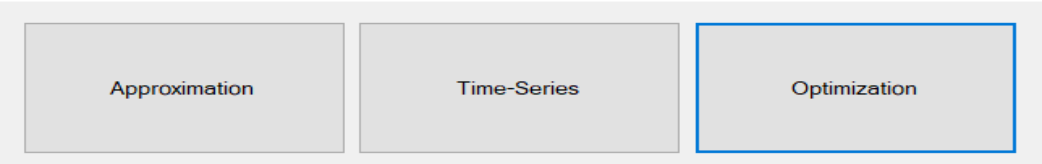


Рисунок 4.1 - Вибір типу моделювання роботи системи

Після вибору конкретного типу моделювання роботи, буде відкрито форму з можливістю для налаштування параметрів генетичного алгоритму, а також специфічних для даного прикладу параметрів. Також систему надаватиме можливість завантаження вхідних даних, які можуть бути обрані або ж із запропонованого набору прикладі, або створені користувачем.

Розглянемо форму апроксимації функції - Рис. 4.2. Оберемо метод селекції, розмір популяції - 100 та кількість ітерацій - 1000.

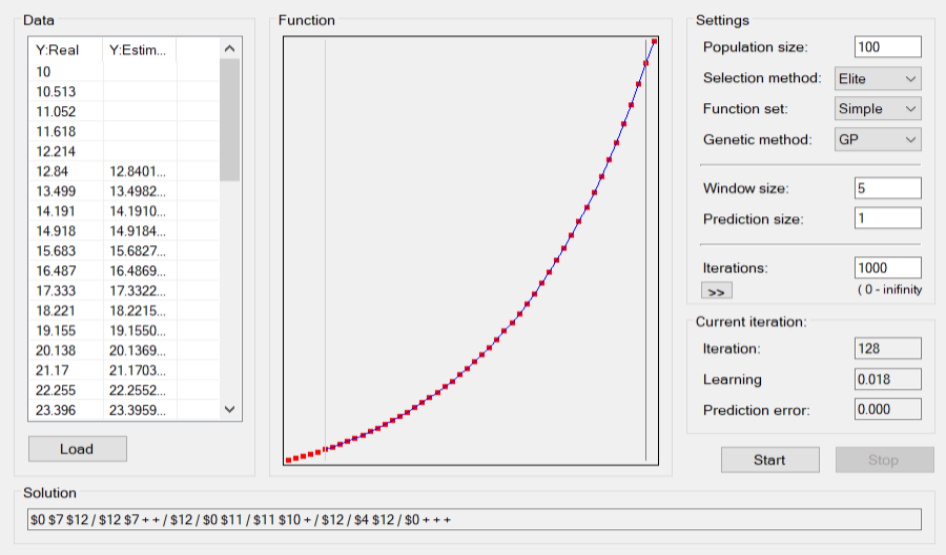


Рисунок 4.2 - Моделювання роботи алгоритму апроксимації функцій

Можемо помітити досить точне обчислення генетичного алгоритму та низьку кількість помилок при навчанні алгоритму. Прогрес роботи алгоритму можливо спостерігати на кожній ітерації алгоритму, а також, є можливість зупинити роботу алгоритму на будь-якому етапі його роботи.

Розглянемо роботу алгоритму на прикладі оптимізації функції - Рис. 4.3. Задамо кількість початкової популяції - 40, довжину хромосоми - 32, оберемо ранковий метод селекції, а тип оптимізації - знаходження максимуму.

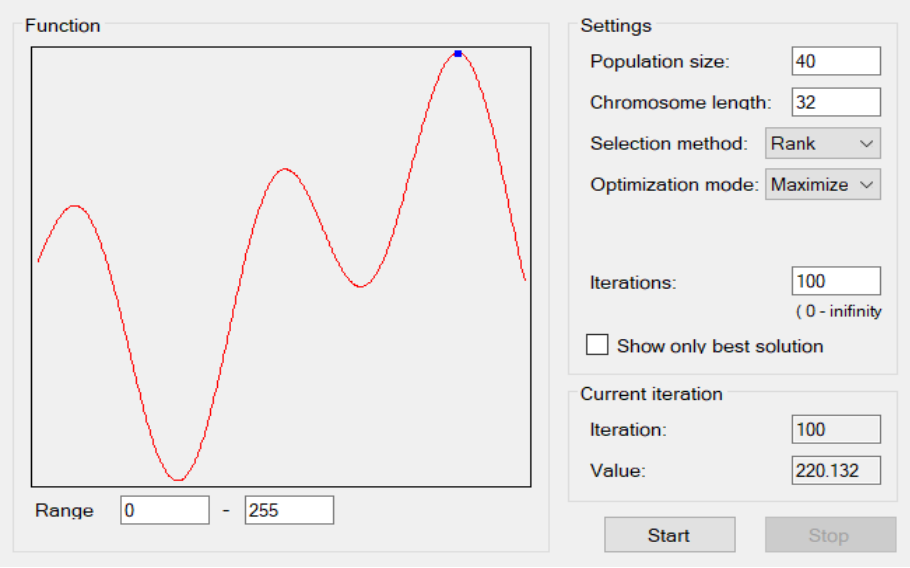


Рисунок 4.3 - Моделювання роботи алгоритму для оптимізації функцій

В даному випадку, також, можемо помітити точно розрахований максимум функції за відносно малу кількість ітерацій роботи алгоритму, що свідчить про правильність обраних параметрів GA.

Розглянемо розширені можливості генетичного алгоритму при моделювання числових рядів на прикладі зростаючої синусоїди. На формі розширених налаштувань - Рис. 4.4, можливо задавати максимальну початкову глибину дерева та максимальну.

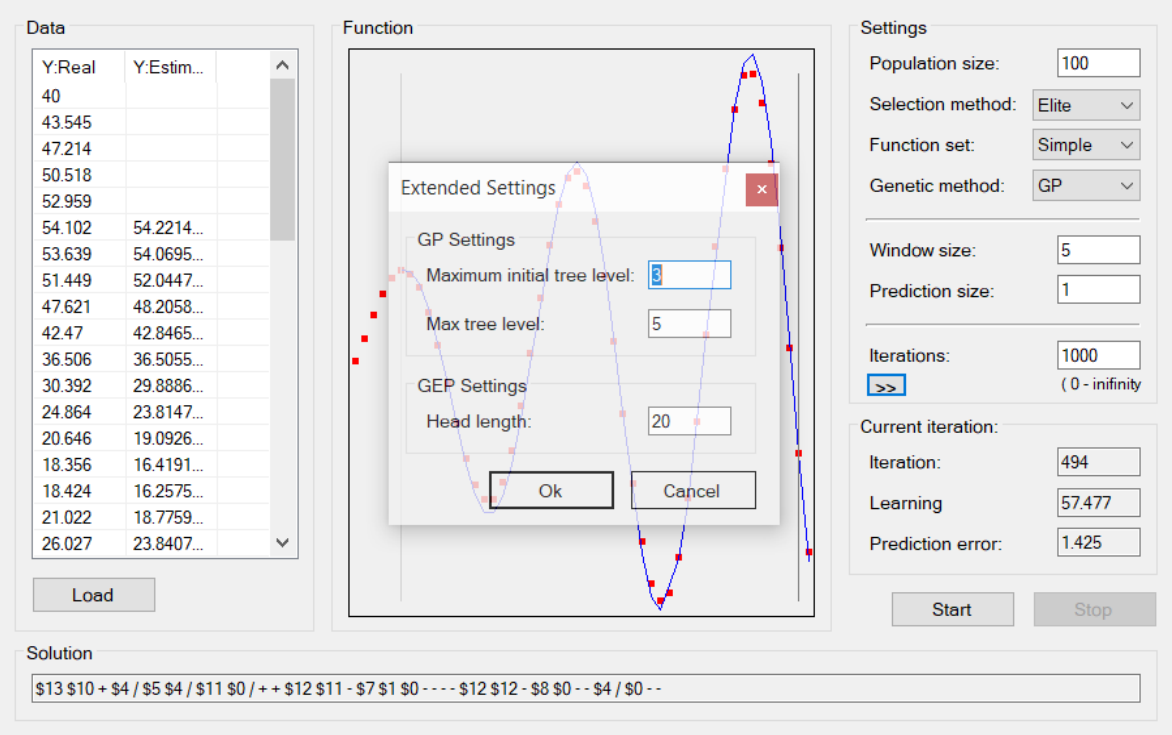


Рисунок 4.4 - Розширені налаштування на прикладі апроксимації функції зростаючої синусоїди

4.2 Застосування розробленої системи для генерування тестових сценаріїв

Для обґрунтування важливості застосування розробленої системи, розглянемо задачу формування тестових даних на основі UML діаграми станів. Дана задача є досить важливою при розробці та підтримці якості програмного забезпечення та досить часто є не автоматизованим процесом. Для того щоб автоматизувати процес опису тестових сценаріїв, можна скористатися генетичними алгоритмами. Так як початкові популяції для такої задачі досить обмежені, потрібно скористатися підходом до інтелектуального формування початкових поколінь.

Тестування в основному проводиться як на програмному забезпеченні для ПК, так і в інтернет додатках для тестування архітектури клієнта та сервера. Тестування програмного забезпечення є одним з основних методів для досягнення високої якості програмного забезпечення. Тестування програмного забезпечення проводиться для виявлення несправностей, які викликають помилки в коді програмного забезпечення. Однак тестування програмного забезпечення займає досить багато часу. Воно може споживати до 50% ресурсів розробки програмної системи. Тестування програмного забезпечення також можна визначити як процес перевірка та валідація програмного забезпечення для перевірки відповідності як технічним, так і бізнес-вимогам.

Зазвичай перед початком тестування та навіть перед початком розробки програмного забезпечення описуються бізнес вимоги та формуються діаграми переходів.

Тому, для початку опишемо загальний сценарій конкретного використання певного додатку, як наприклад форма реєстрації користувача на веб порталі.

Опишемо сценарій скориставшись спеціальним синтаксисом для побудови Sequence діаграм - Рис. 4.5.

```
title Registration

actor User
participant Validation
participant Agreement Form
participant Main Page
participant Error Page

User->Validation:Enter form

alt password is valid
Validation->Agreement Form:info
else password is invalid
Validation-xError Page:info
end

alt agreement is signed
Agreement Form->Main Page:info
else password is invalid
Agreement Form-xError Page:info
end
```

Рисунок 4.5 - Опис UML сценарію

Такий текстовий формат запису Sequence діаграми описує логіку основних переходів - participant та користувачів - actor.

В даному випадку описано логіку переходів користувача між певними сторінками системи, змінюючи її стан.

В результаті отримаємо наступну діаграму - Рис. 4.6.

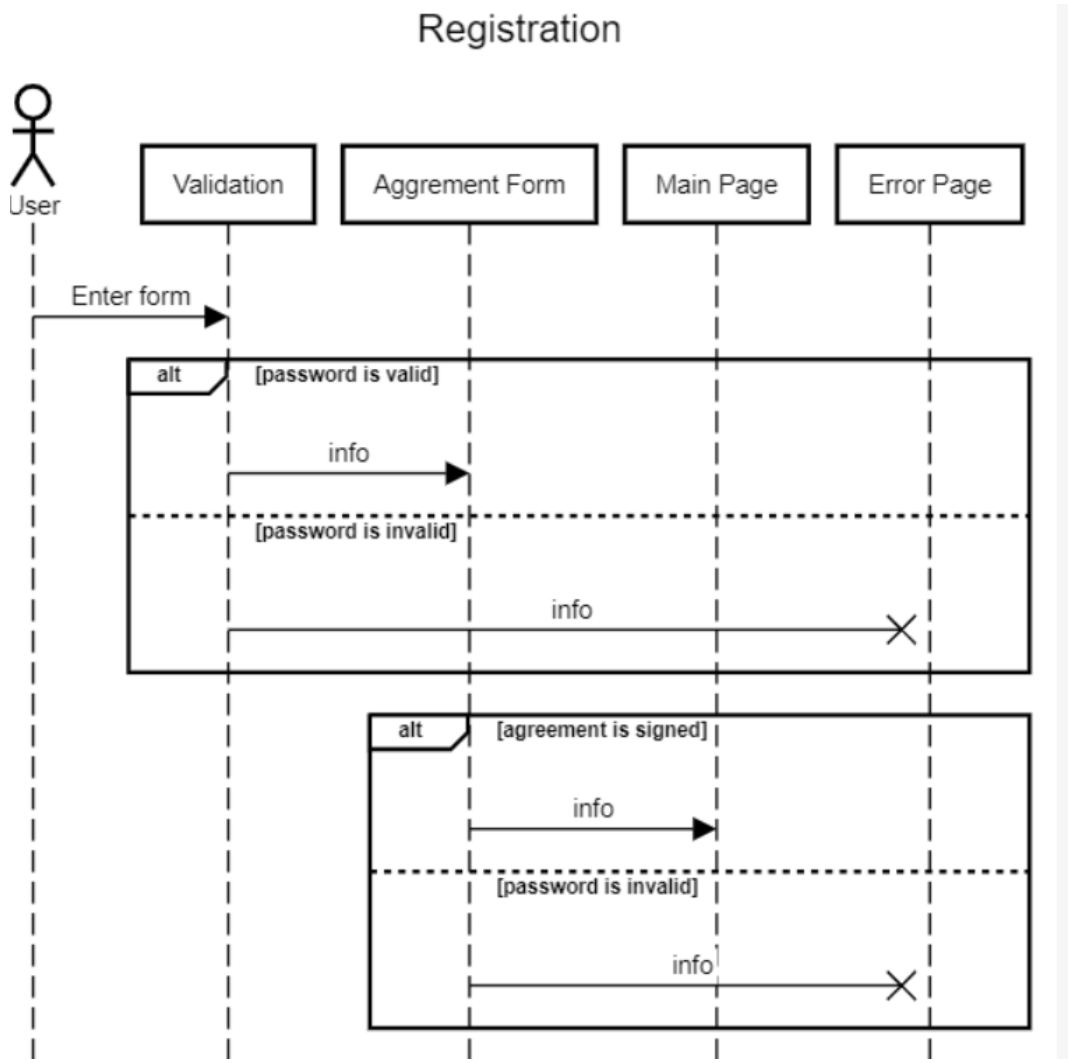


Рисунок 4.6 - Діаграма переходів користувача

Можливо помітити три сценарії, два з яких є негативними та один позитивний. Для того щоб автоматизувати процес визначення сценаріїв, можливо, досить легко, створити системи за допомогою визначених раніше програмних компонентів, з використання компоненту інтелектуального формування початкових популяцій. Тоді сформована таким чином система, на вхід буде приймати UML діаграму у текстовому

форматі, синтаксис якого був описаний раніше. На виході, система буде повертати набір сценарії тестування.

Віддавши програмі по генерування тестових сценаріїв на базі генетичного алгоритму сформований код Sequence діаграми на вхід - на виході отримуємо такі сценарії для тестування згенеровані системою.

Таблиця 4.1 – Згенеровані сценарії

	Тип сценарію	Згенерований сценарій
1	негативний	Given user start validation When user enter valid password And user agreement is declined Then user go to error page
2	негативний	Given user start validation When user enter invalid password Then user go to error page
3	позитивний	Given user start validation When user enter valid password And user agreement is signed Then user go to main page

Отримані сценарії можливо відразу використовувати для створення сценаріїв для тестування (Test Cases), а також, можливо, в автоматизованому тестуванні, так як, в даному випадку тестові сценарії описані на синтаксисі, схожому на мову Gherkin, що використовується для автоматизації .

Таким чином, можна переконатися, що генетичні алгоритми, а також інтелектуальне формування стартових популяцій, може досить вдало застосовуватися на практиці.

Автоматизація тестування програмного забезпечення - є стрімко напрямком в інженерії програмного забезпечення, що досить стрімко розвивається останнім часом.

Важливість використання подібних засобів для автоматизації роботи тестувальників та бізнес аналітиків зараз є дуже високою.

Використання генетичних алгоритмів та системи інтелектуального формування початкових поколінь, дозволяє досить швидко та точно отримати тестові сценарії на основі UML діаграм.

4.3 Аналіз отриманих результатів

В ході дослідження було протестовано роботу програмної системи по формуванню початкових популяцій для оптимізації та апроксимації функцій, а також розглянуто генерування тестових сценаріїв. Для вирішення задачі по генеруванню тестових сценаріїв було отримано 3 сценарії використавши інтелектуальний підхід до генерації стартових популяцій. В даному випадку - це всі можливі варіанти, відповідно до описаної раніше UML діаграми. Це свідчить про коректність роботи алгоритму.

Дану задачу також було розв’язано за допомогою випадкової ініціалізації популяцій. Було проведено порівняння результатів отриманих за допомогою двох реалізацій компонентів формування початкових популяцій:

1) випадкове формування початкової популяції;
2) інтелектуальне формування початкової популяції.

Отримані результати продемонстровано на Рис. 4.7.

```
BenchmarkDotNet=v0.13.1, OS=Windows 10.0.19042.1288 (20H2/October2020Update)
Intel Core i7-8565U CPU 1.80GHz (Whiskey Lake), 1 CPU, 8 logical and 4 physical cores
.NET SDK=5.0.101
  [Host]     : .NET 5.0.1 (5.0.120.57516), X64 RyuJIT
  DefaultJob : .NET 5.0.1 (5.0.120.57516), X64 RyuJIT


|      Method |     Mean |    Error |    StdDev |
|------------ |---------:|---------:|----------:|
|      Random | 423.4 ns | 62.08 ns | 183.05 ns |
| Intelligent | 244.1 ns |  4.56 ns |   4.27 ns |
```

Рисунок 4.7 - Порівняння часу роботи алгоритмів генерування тестових сценаріїв за допомогою випадкової та інтелектуальної ініціалізації початкових популяцій

В даному випадку, можна помітити приріст в швидкодії алгоритму для формування тестових сценаріїв при тестуванні програмного забезпечення. Завдяки використанню компонента інтелектуального формування початкових поколінь, можливо досягти підвищення точності та швидкодії.

Висновки до розділу

В даному розділі було доведено актуальність застосування системи інтелектуального формування початкових популяцій, а також генетичних алгоритмів, на практиці. Було розглянути графічну систему моделювання генетичних алгоритмів на практиці, а також розв'язано задачу генерування тестових сценаріїв.

Було продемонстровано приріст в швидкодії алгоритму а також точність отриманих результатів, що свідчить про переваги застосування сформованої програмної системи.

Програмна система по інтелектуальному формуванню початкових популяцій може бути використана у багатьох галузях та задачах рішення яких не може бути описане у вигляді чіткого алгоритму, або ж таке, що потребує складних обчислень. За допомогою розробленої системи, можливо буде формувати генетичні алгоритму для вирішення майже будь-якої задачі.

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП ПРОЕКТУ

1.1 Опис ідеї проєкту

Таблиця 5.1 — Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користучача
Створення програмної системи інтелектуального формування початкових популяцій за допомогою компонентного підходу для розв’язання задач з великою кількістю обмежень	Системи що включають застосування генетичних алгоритмів для вирішення задач.	Поліпшення якості, швидкості та зручності настроювання параметрів генетичних алгоритмів, можливість інтелектуального задання початкових популяцій при вирішенні задач.

Таблиця 5.2 — Опис ідеї стартап-проєкту

№	Техніко-економічні характери-тики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	GeneticSharp	ExtremeNumerics.Genetic			
1	Можливості формування початкових популяцій	Так	Ні	Ні			+
2	Тип архітектури	Компонентно-орієнтована	Об’єктно-орієнтована	Об’єктно-орієнтована		-	
3	Точність та швидкість роботи	Висока	Висока	Висока			+
4	Можливість паралельної обробки даних	Так	Так	Ні			+

1.2 Технологічний аудит ідеї проєкту

Таблиця 5.3 — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Обробка вхідних даних	C#, .Net5.0,	+	Загальнодоступні
2	Графічний інтерфейс моделювання роботи алгоритму	WPF	+	Загальнодоступні
3	Бібліотека алгоритмів	Accord.Net	+	Загальнодоступні
4	Моделювання та аналіз результатів	C#, .Net5.0, WPF	+	Загальнодоступні

У результаті технологічного аналізу ідеї проекту було вирішено для її реалізації використати наступний технологічний стек:

- мова програмування C#;
- середовище виконання .Net5;
- бібліотека еволюційних обчислень Accord.Net;
- бібліотека візуалізації WPF.

1.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 5.4 — Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн./ум.од	1 млн. ум. од
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі або по ринку, %	20 %

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Нестача систем для інтелектуального формування початкових популяцій	Підприємства, що використовують генетичні алгоритми, або ж розробляють додадтки, робота яких заснована на генетичних алгоритмах	Масштаби застосування, обсяги даних, технічні засоби для розгортання, фінанси	Можливості до налаштуваня кожного компоненту генетичного алгоритму та перш за все налаштування формування початкових популяцій.

Таблиця 5.6 — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Нестача технічних ресурсів	Обмежені обчислювальні потужності для розгортання та впровадження системи	Винесення системи на хмарні технології
1	Конкуренти	Наявність систем, що надають такі ж рішення для генерування початкових популяцій та/або застосовують компонентний підхід, що дозволяє реалізувати компонент формування початкових популяцій.	Розробка додаткових реалізацій компонентів, покращення візуального додатку для моделювання роботи алгоритму.
2	Кошти на розробку та підтримку продукту	Недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на перспективу; Ітеративна розробка продукту задля покрокового виведення продукту на ринок та отримання відповіді користувачів

Таблиця 5.7 — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Хмарні обчислення	Підтримка виконання алгоритму на віддалених серверах, а також підтримка рішення з боку провайдерів хмарних обчислень	Розробка нової функціональності; Перенесення системи у хмарне середовище одного з популярних провайдерів (Azure, Amazon, Google Cloud Platform)
2	Інтеграція з сервісами обробки даних	Можливість інтеграції з сервісами по обробці або перетворенню данних. Можливість підключення системи як споживача або надавача даних для інших сервісів.	Розробка нової функціональності по підтримці інтеграції з іншими системами
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконалення продукту	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів системи кешування даних.
4	Грошова винагорода за рекламу	При достатньому попиту на систему кешування даних можлива комерціалізація продукту на основі реклами задля отримання грошової винагороди для подальшого розвиту продукту та оплати заробітної плати працівникам	Точкова комерціалізація продукту; Введення реклами; Ведення додаткових коштів у проєкт задля його подальшого розвитку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: вільна конкуренція	Присутня невелика кількість постачальників. Домінуючих конкурентів немає через галузеву специфіку. Невелика кількість конкурентів та домінуючих конкурентів немає	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники;
2	Рівень конкурентної боротьби: глобальний	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вихід на ринок збуту продукту з клієнто-необхідною функціональністю; Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів; Надання бета-версій продукту.
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків \ продуктів	Надання зручного, інтуїтивно зрозумілого інтерфейсу; Підтримка всім відомих методів взаємодії з середовищем розробки; Наявність документації та он-лайн підтримки.
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція – конкуренція між товарами одного виду.	Впровадження функціональності яка відсутня у товарів-замінників; Спрощення інтерфейсів; Надання підтримки.
5	Характер конкурентних переваг: цінова та не цінова	Цінові переваги – точкова комерціалізація; Не цінова – надання функціональності, що відсутня у товарах-замінниках.	Надання платних ліцензій лише на критично важливу функціональність для клієнта з певним строком підтримки, що зазначена у відповідній ліцензії; Впровадження унікальної функціональності.
6	За інтенсивністю: марочна	Наявність унікального знаку що відрізняє даний продукт від продуктів-замінників	Впровадження власної назви та власного знаку.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Клієнти
	GeneticSharp	ExtremeNumerics.Genetic	Розробники програмного забезпечення з використанням генетичних алгоритмів
Висновки	Контролюють значну частину ринку, проте мають певний ряд недоліків, в тому числі неможливість вибору алгоритму формування стартової популяції, а також, не достатньо гнучкий метод налаштування.	Поки ще постачальник - не є досить популярним так як має певний ряд недоліків, а також обмежену модель ліцензування.	Так як, клієнтами є розробники програмного забезпечення - важливо надати можливість гнучкого налаштування системи.

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо вирішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею [23].

Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Універсальність розрахунку	Універсальна, та висока достовірність отриманих результатів, яка підтверджується якісними математичними розрахунками.
2	Простота реалізації	Система реалізована за допомогою компонентів, що робить її використання та модифікацію досить простими.
3	Час роботи алгоритму	За рахунок удосконалення алгоритму формування початкових популяцій - час роботи генетичного алгоритму сформованого за допомогою системи - є відносно зменшим.
4	Можливість модифікації	За рахунок використання компонентів систему можливо досить легко модифікувати та розширювати.
5	Додаткові можливості	Система потенційно може бути адаптована для вирішення конкретних задач.

Завдяки реалізації такої функціональності, буде забезпечена достатня конкуренція на ринку, так як описані фактори - є ключовими на даному ринку.

Також, потрібно проаналізувати сильні та слабкі сторони конкурентів, відносно до сформованих факторів.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін системи кешування мало змінних даних

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Висока якість	18	-	-	-		1	1	2
2	Оптимальне співвідношення ціни і якості	16	-	-	-		1	1	1
3	Можливість модифікаці	20	-	-	-		1	1	2
4	Додаткові можливості	14	-	-	-		1	1	2
5	Час роботи алгоритму	14	-	-	-		1	1	1

Таблиця 5.12 — SWOT аналіз стартап-проєкту

Сильні сторони (S): - якість, - універсальність, - точність	Слабкі сторони (W): - необхідність інвесторів, - дешеві замінники
Можливості (O): - удосконалення	Загрози (T): - додаткові витрати, - розвиток конкурентів

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	2-3 місяці
2	Реклама	Залучення власних коштів для реклами товару	1-2 місяці
3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	2-3 тижні
4	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

1.4 Розроблення ринкової стратегії проєкту

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Програмні системи що впроваджують та використовують генетичні алгоритми	Висока готовність сприйняти товар. Оскільки існує потреба в якісній програмній системі, що дозволятиме інтелектуально формувати початкові популяції.	Попит високий через те, що рівень ціни та якості є кращим ніж у конкурентів	Конкуренція в сегменті досить низька.	Досить висока складність входу в сегмент, так як потребуються знання в формування та розробці генетичних алгоритмів

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є працівники IT сфери, IT компанії в цілому та будь-які підприємства котрі використовують програмні продукти побудовані на мові програмування Java, та використовують реляційні бази даних. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу, оскільки для підприємств, IT працівників та IT компаній у цілому надається стандартизований продукт з можливістю розширення функціональності за домовленістю (відповідно до ліцензії).

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	Зниження ступеню замінності товару; Прихильність клієнтів; Відмітні властивості товару; Відмітні характеристики товару;	Стратегія диференціації

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-заміннники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

Таблиця 5.17 — Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
1	Можливості повного налаштування систем, а саме, можливість задання методу формування початкової популяції	Стратегія диференціації	Компонентний підхід до розробки, можливість задання методу інтелектуального формування початкових популяцій	Простота реалізації та використання, швидкість роботи, можливість повного налаштування

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

1.5 Розроблення маркетингової програми стартап-проєкту

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Можливість налаштування	Пропонується компонентний підхід до налаштування генетичного алгоритму, що забезпечує повне налаштування всіх операторів генетичного алгоритму.	Більшість існуючих рішень засновані на використанні звичайного об’єктно-орієнтованого підходу, та не надають можливості до повного налаштування системи.
2	Швидкість роботи	Відносно збільшена швидкість роботи.	Використанню методу інтелектуального формування початкових популяцій
3	Простота у використанні	Простота використанні за рахунок компонентів.	Використання компонентів для задання генетичного алгоритму.

Таблиця 5.19 — Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Програмна система інтелектуального формування початкових популяцій</td></tr>
<tr><td rowspan="6">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>Можливість налаштування</td><td>Нм</td><td>Вр/Тх/Тл</td></tr>
<tr><td>Швидкість роботи</td><td>Нм</td><td>Вр/Тх/Тл</td></tr>
<tr><td>Простота у використанні</td><td>Нм</td><td>Вр/Тх/Тл</td></tr>
<tr><td>Кількість компонентів</td><td>М</td><td>Вр/Тх/Тл/Е</td></tr>
<tr><td colspan="3">Форма товару: у вигляді програмної бібліотеки, а також десктопному застосунку.</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару</td></tr>
<tr><td colspan="3">Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент</td></tr>
</table>

Таблиця 5.20 — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
100 у.е.	Немає аналога	1000 у. е.	90-150 у. е.

Таблиця 5.21 — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Орієнтація на регулярні поставки	Встановлення контактів із споживачами та підтримка їх. Формування попиту. Проведення маркетингової діяльності	без посередників	Серед IT компаній

Таблиця 5.22 — Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Орієнтація на регулярні поставки	Формальні/неформальні канали комунікацій	Швидкість, простота у використанні та можливість налаштування	Інформування споживачів; Стимулювання продажу;	Максимальна швидкість та гнучкість, що дозволить вирішити будь-яку задачу!

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

В п’ятому розділі описано стратегії та підходи з розроблення стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проєкту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проєкту було встановлено що проєкт є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проєкту та доведено доцільність подальшої імплементації проєкту.

ВИСНОВКИ

У ході виконання магістерської дисертації було розглянуто питання пов’язані з методами та засобами розробки програмної системи формування початкових популяцій для генетичних алгоритмів.

На основі даних, отриманих в процесі аналізу, сформульовано задачу, що полягає у розробці програмного забезпечення для формування початкових популяцій, використовуючи компонентний підхід до розробки програмного забезпечення.

Для розробки програмної системи було використано мову програмування C# та платформу .Net Framework. Для формування роботи генетичного алгоритму було застосовано бібліотеку Accord.Net. Для візуалізації роботи системи, було використано фреймворк для розробки десктопних застосунків - WPF. Все перераховане є широко вживаними та безкоштовними інструментам, що дозволяють ефективно вирішувати задачу розробки програмної системи, що імплементує генетичні алгоритми.

Розроблена модель програмної системи за допомогою визначення основних програмних компонентів генетичного алгоритму:

- компонент формування початкової популяції;
- компонент визначення функції відповідності;
- компонент селекції;
- компонент кросинговеру;
- компонент мутації.

Розроблено програмне забезпечення, що реалізує дані програмні компоненти, з застосуванням підходів об’єктно-орієнтованого та компонентно-орієнтованого програмування.

Проведений маркетинговий аналіз стартап-проєкту. Проведено опис основної ідеї проєкту програмної системи формування стартових популяцій, технологічний аудит ідеї проєкту, аналіз ринкових можливостей запуску стартап-проєкту. Розроблено ринкову стратегію проєкту та маркетингову програму стартап-проєкту.

Результати роботи над магістерською дисертацією опубліковані в статті на конференції SoftTech 2021.

Наукова новизна одержаних результатів магістерської дисертації полягає в удосконаленні існуючих програмних рішень для реалізації генетичних алгоритмів, основними відмінностями над якими є:

- введення можливості інтелектуального формування початкових популяцій;
- компонентний підхід до реалізації генетичного алгоритму.

Практична значущість одержаних результатів полягає у розробці програмної системи, що може бути застосована для вирішення задач з великою кількістю обмежень, як наприклад система формування тестових сценаріїв, реалізації якої продемонстрована в роботі.

ПЕРЕЛІК ПОСИЛАНЬ

1) Effects of the nature of the starting population on the properties of Rugate filters designed with the genetic algorithm / P.L. Swart; A.P. Kotze, B.M. Lacquet // Journal of Lightwave Technology. - 2000. - N18. - pp. 853 - 859. - Режим доступу: https://ieeexplore.ieee.org/document/848398

2) Distributed GAs with case-based initial populations for real-time solution of combinatorial problems / Takashi Kawabe, Masaki Suzuki, Taro Matsumaru // 2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS). - 2015. - Режим доступу: https://ieeexplore.ieee.org/document/7009509

3) Genetic programming with multiple initial populations generated by simulated annealing / Takuya Mototsuka, Akira Hara, Jun-ichi Kushida // 2013 IEEE 6th International Workshop on Computational Intelligence and Applications (IWCIA). - 2013. - Режим доступу: https://ieeexplore.ieee.org/document/6624797

4) A Genetic Algorithm Convergence and Models for Eigenstructure Assignment via Linear Quadratic Regulator / Takuya Mototsuka, Akira Hara, Jun-ichi Kushida // IEEE Latin America Transactions. - pp. 1-9. - 2008. - Режим доступу: https://ieeexplore.ieee.org/document/4461626

5) The Genetic Algorithm Fractal / Jenny Juliany, Michael D. Vose. // Evolutionary Computation. - pp. 165-180. 1994. - Режим доступу: https://ieeexplore.ieee.org/document/6792023

6) Flexible job shop scheduling based on multi-population genetic-variable neighborhood search algorithm / Liang Xu, Weiping Sun, Huang Ming // 2015 4th International Conference on Computer Science and Network Technology (ICCSNT) - pp. 19-20 - 2015. - Режим доступу: https://ieeexplore.ieee.org/document/7490745

7) Combining approximation algorithm with genetic algorithm at the initial population for NP-complete problem / Hajar Razip, M. Nordin Zakaria // 2017 IEEE 15th Student Conference on Research and Development (SCOReD) - pp. 13-14. - 2017. - Режим доступу: https://ieeexplore.ieee.org/document/8305413

8) An Adaptive Genetic Algorithm Based on Multi-population Parallel Evolutionary and Variable Population Size / Jianxin Chen, Qing Liu, Junqin Huang, Yun Hou // 2010 Second WRI Global Congress on Intelligent Systems - pp. 258-262. - 2010. - Режим доступу: https://ieeexplore.ieee.org/document/5708756

9) Load balancing task scheduling based on Multi-Population Genetic Algorithm in cloud computing / Bei Wang, Jun Li // 2016 35th Chinese Control Conference (CCC), 2016, pp. 5261-5266. - 2016. режим доступу: https://ieeexplore.ieee.org/document/7554174

10) Investigation on selection schemes and population sizes for genetic algorithm in unmanned aerial vehicle path planning / Kai Yit Kok, Parvathy Rajendran, Ruslan Rainis // 2015 International Symposium on Technology Management and Emerging Technologies (ISTMET) - pp. 6-10. - 2015. - Режим доступу: https://ieeexplore.ieee.org/document/7358990

11) Preparing initial population of genetic algorithm for region growing parameter optimization / Sándor Szénási, Zoltán Vámossy, Miklós Kozlovszky //2012 4th IEEE International Symposium on Logistics and Industrial Informatics - pp. 47-54. - 5-7 Sept. 2012 - Режим доступу: https://ieeexplore.ieee.org/document/6319460

12) An Artificial Life and Genetic Algorithm based on optimization approach with new selecting methods / Chen Yang, Hao Ye, Jing-Chun Wang // Proceedings. International Conference on Machine Learning and Cybernetics - pp. 684-688. - 2002 - Режим доступу: https://ieeexplore.ieee.org/document/1174434

13) An Educational Genetic Algorithms Learning Tool / Ying-Hong Liao, Chuen-Tsai Sun // IEEE - Режим доступу: https://www.ewh.ieee.org/soc/es/May2001/14/Begin.htm

14) An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TS / Yong Deng, Yang Liu, Deyun Zhou // Mathematical Problems in Engineering, vol. 2015 - Article ID 212794. - Режим доступу: https://www.hindawi.com/journals/mpe/2015/212794/

15) Quasi-random initial population for genetic algorithms / H. Maaranen, K. Miettinen, M.M. Mäkelä // Computers & Mathematics with Applications - Volume 47, Issue 12. - 2004. - Режим доступу: https://www.sciencedirect.com/science/article/pii/S0898122104840240

16) A Heuristic Method to Generate Better Initial Population for Evolutionary Methods / E. Khaji, A. S. Mohammadi // Graduate Students, Complex Adaptive Systems Group, School of Applied Physics, University of Gothenburg, Sweden. - Режим доступу: https://arxiv.org/ftp/arxiv/papers/1406/1406.4518.pdf

17) A novel population initialization method for accelerating evolutionary algorithms / Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M. A. Salama // Medical Instrument Analysis and Machine Intelligence Research Group, Faculty of Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada - Режим доступу: https://www.sciencedirect.com/science/article/pii/S0898122107001344

18) Adaptive Randomness: A New Population Initialization Method / Weifeng Pan, Kangshun Li, Muchou Wang // Mathematical Problems in Engineering 2014 - Режим доступу: https://www.hindawi.com/journals/mpe/2014/975916/

19) Effects of population initialization on differential evolution for large scale optimization / B. Kazimipour, X. Li and A. K. Qin // 2014 IEEE Congress on Evolutionary Computation (CEC). - pp. 2404-2411. - 2014. - Режим доступу: https://ieeexplore.ieee.org/document/6900624

20) A foundational study on the applicability of genetic algorithms to software engineering problems. Hsinyi Jiang, Carl K. Chang, Dan Zhu, Shuxing Cheng // 2007 IEEE Congress on Evolutionary Computation. - pp. 2210-2219. - 2007. - Режим доступу: https://ieeexplore.ieee.org/document/4424746

21) A Cluster Based Approach for Task Scheduling across Multiple Programming Systems / Hongliang Lu, Jiannong Cao, Shailey Chawla // 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC) - pp. 222-229. - 2016. - Режим доступу: https://ieeexplore.ieee.org/document/7904294

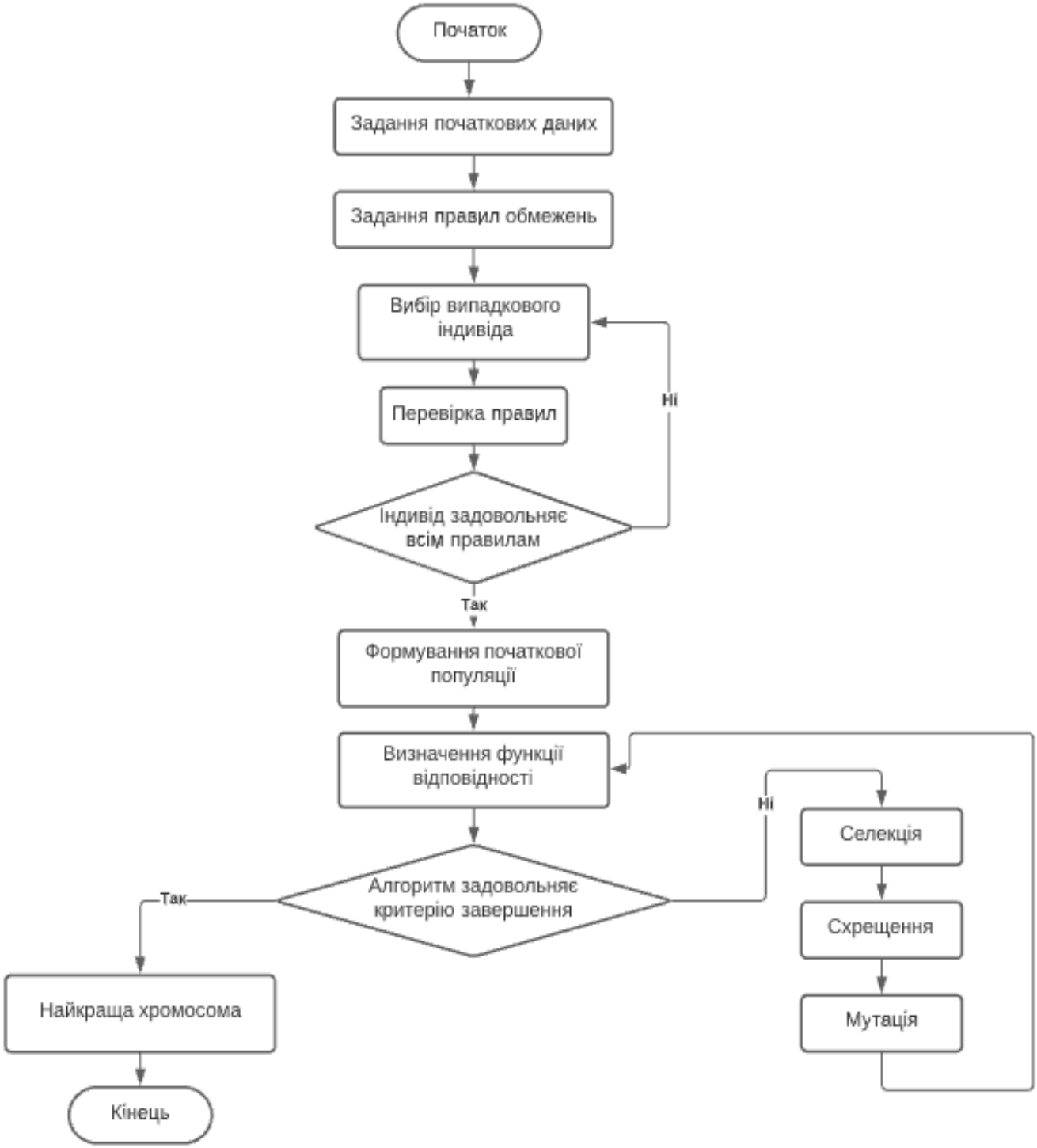
22) Evolutionary Algorithms in Engineering Applications / Dipankar Dasgupta, Zbigniew Michalewicz. - Springer, Berlin, Heidelberg. - 1997. - 555 p. - ISBN 978-3-642-08282-5.

23) Applications of Genetic Algorithm in Software Engineering, Distributed Computing and Machine Learning / Samriti Sharma. //International Journal of Computer Applications & Information Technology. - Режим доступу: https://www.researchgate.net/publication/322488040_Applications_of_Genetic_Algorithm_in_Software_Engineering_Distributed_Computing_and_Machine_Learning

24) A Method of Initial Population Generation of Intelligent Optimization Algorithms for Constrained Global Optimization / Jiquan Wang, Okan K. Ersoy, Xinxin Chen // International Journal of Hybrid Information Technology. - pp.47-56. - 2017. - Режим доступу: https://gvpress.com/journals/IJHIT/vol10_no6/5.pdf

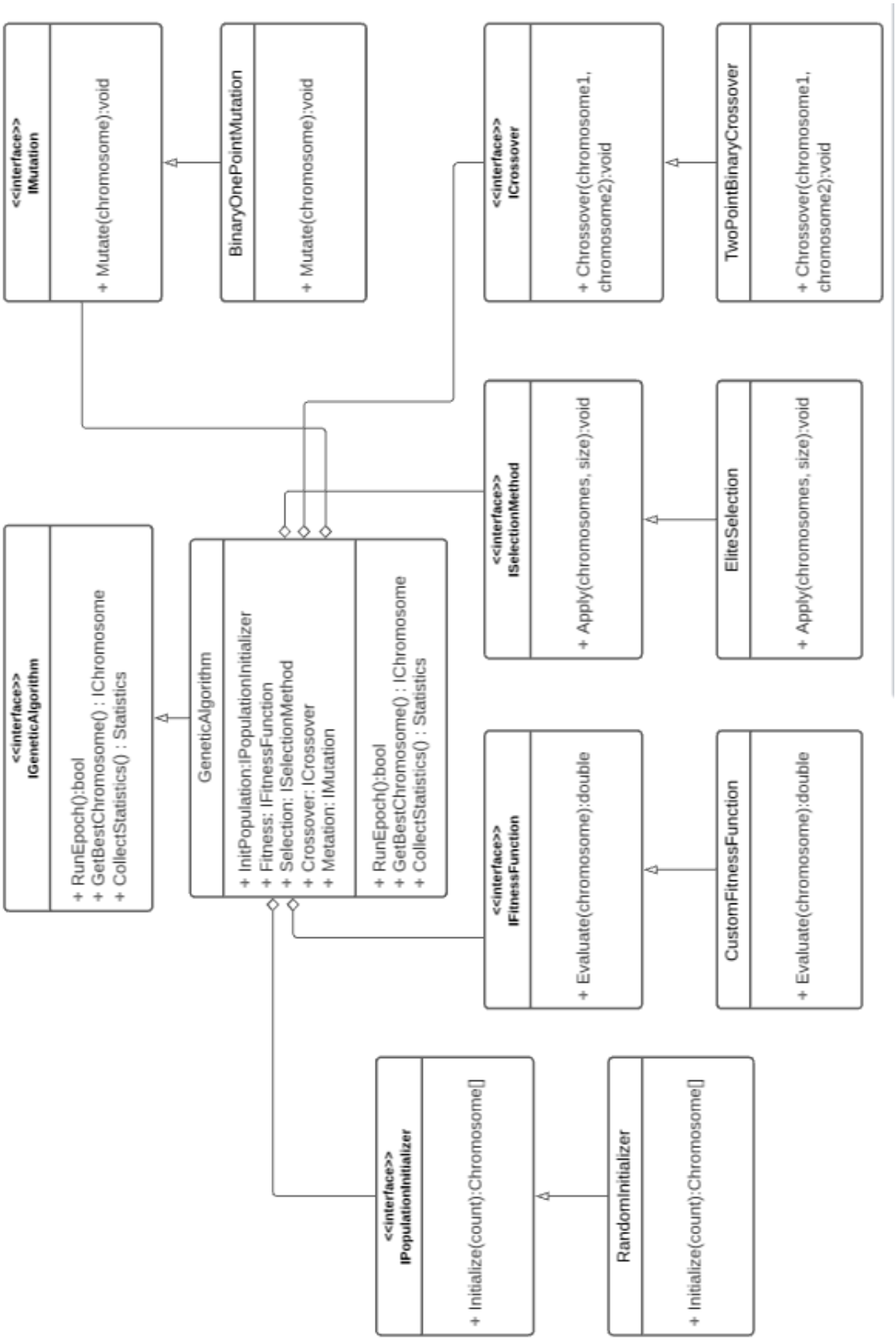
ДОДАТОК А

Алгоритм роботи системи



ДОДАТОК Б

Діаграма класів



ДОДАТОК В

Лістинг коду

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Accord.Genetic;

namespace TestCaseGenerator
{
    public  interface ICrossover
    {
        /// <summary>
        /// Applies crossover to <see cref="left"/> <see
cref="IChromosome"/>
        /// and <see cref="right"/> <see cref="IChromosome"/>
        /// </summary>
        /// <param name="left"></param>
        /// <param name="right"></param>
        void Apply(IChromosome left, ICrossover right);
    }

    public interface IMutation
    {
        /// <summary>
        /// Applies mutation to specified <see cref="IChromosome"/>
        /// </summary>
        /// <param name="chromosome"></param>
        void Apply(IChromosome chromosome);
    }

    public class ScenarioRules : IDomainRules<ScenarioChromosome>
    {
        private readonly string[] _initialSteps;
        private readonly string[] _finalSteps;

        public ScenarioRules(string[] initialSteps, string[]
finalSteps)
        {
            _initialSteps = initialSteps;
            _finalSteps = finalSteps;
        }

        public bool Apply(ScenarioChromosome chromosome)
        {
            return
_initialSteps.Contains(chromosome.Scenario.Steps.FirstOrDefault().From
)
```

```
                &&
_finalSteps.Contains(chromosome.Scenario.Steps.LastOrDefault().To);
        }
    }

    public struct GenIndex
    {
        public bool IsStart { get; private set; }
        public bool IsEnd { get; private set; }
        public int Value { get; set; }

        public GenIndex(int index)
        {
            Value = index;
            IsStart = false;
            IsEnd = false;
        }

        public static GenIndex Start => new GenIndex(0) {IsStart =
true};

        public static GenIndex End(int index) => new GenIndex(index)
{IsEnd = true};
    }

    public interface IInitialPopulationGenerator<TGen, TPopulation>
    {
        TPopulation Generate(
            Func<IEnumerable<TGen>, TPopulation> formPopulation,
            Func<int, TGen> generateGen, int length);
    }

    public class RandomGenerator<TGen, TPopulation> :
IInitialPopulationGenerator<TGen, TPopulation>
    {
        private readonly Range _allowedValues;
        private readonly Random _random = new Random();

        public RandomGenerator(Range allowedValues)
        {
            _allowedValues = allowedValues;
        }

        public TPopulation Generate(Func<IEnumerable<TGen>,
TPopulation> formPopulation, Func<int, TGen> generateGen, int length)
        {
            var genes = new List<TGen>(length);
            for (var i = 0; i < length; i++)
            {
                var genNumber =
_random.Next(_allowedValues.Start.Value, _allowedValues.End.Value);
```

```
                genes.Add(generateGen(genNumber));
            }

            return formPopulation(genes);
        }
    }

    public class IntelligentGenerator<TGen, TPopulation> :
IInitialPopulationGenerator<TGen, TPopulation>
    {
        private readonly Range _allowedValues;
        private readonly Func<(TGen Gen, GenIndex Index), bool>
_constraints;
        private readonly Random _random = new Random();

        public IntelligentGenerator(Range allowedValues, Func<(TGen
Gen, GenIndex Index), bool> constraints)
        {
            _allowedValues = allowedValues;
            _constraints = constraints;
        }

        public TPopulation Generate(
            Func<IEnumerable<TGen>, TPopulation> formPopulation,
            Func<int, TGen> generateGen,
            int length)
        {
            var genes = new List<TGen>(length);
            for (var i = GenIndex.Start; i.Value < length;
                i = i.Value == length - 2 ? GenIndex.End(i.Value+1) :
new GenIndex(i.Value+1))
            {
                TGen gen = default;
                do
                {
                    var genNumber =
_random.Next(_allowedValues.Start.Value, _allowedValues.End.Value);
                    gen = generateGen(genNumber);
                } while (!_constraints.Invoke((gen, i)));

                genes.Add(gen);
            }

            return formPopulation(genes);
        }
    }

    public struct Step
        {
            public string From { get; set; }
```

```
            public string To { get; set; }

            public string Action { get; set; }

            public Step(string @from, string to, string action)
            {
                From = @from;
                To = to;
                Action = action;
            }

            public override string ToString()
            {
                return $"({From}->[{Action}]->{To})";
            }
        }

        public class Scenario
        {
            public LinkedList<Step> Steps { get; set; }


            public Scenario(IEnumerable<Step>steps)
            {
                Steps = new LinkedList<Step>(steps);
            }

            public override string ToString()
            {
                return string.Join("->", Steps.Select(x =>
x.ToString()));
            }

            public string CucumberScenario()
            {
                var actor = Steps.First.Value.From.ToLower();

                var sb = new StringBuilder($"Given {actor} " +

$"{Steps.First.Value.Action.ToLower()} " +

$"{Steps.First.Value.To.ToLower()} \n");

                sb.AppendLine($"When {actor}
{Steps.First.Next.Value.Action}");
                if (Steps.Count > 2)
                {
                    foreach (var s in Steps.Skip(2).Take(Steps.Count -
2))
                    {
                        sb.AppendLine($"And {actor} {s.Action}");
```

```
                    }
                }

                sb.AppendLine($"Then {actor} go to
{Steps.Last.Value.To.ToLower()}");

                return sb.ToString();
            }
        }

        public class ScenarioChromosome : ChromosomeBase
        {
            public static Random Random = new Random();

            private readonly Step[] _allSteps;
            public readonly string[] InitialStages;
            public readonly string[] FinalStages;

            private readonly Lazy<int[]> _initialSteps;
            private readonly Lazy<int[]> _finalSteps;

            private readonly int _maxScenarioLength;
            private readonly IInitialPopulationGenerator<Step,
Scenario> _initialPopulationGenerator;
            public Scenario Scenario { get; private set; }

            public ScenarioChromosome(
                Step[] allSteps,
                string[] initialStages,
                string[] finalStages,
                int maxScenarioLength,
                IInitialPopulationGenerator<Step, Scenario>
initialPopulationGenerator)
            {
                _allSteps = allSteps;
                InitialStages = initialStages;
                FinalStages = finalStages;

                _initialSteps = new Lazy<int[]>(()
                    => _allSteps.Select((x, i) => (Value: x, Index:
i))
                        .Where(s =>
InitialStages.Contains(s.Value.From))
                        .Select(x => x.Index)
                        .ToArray());

                _finalSteps = new Lazy<int[]>(()
                    => _allSteps.Select((x, i) => (Value: x, Index:
i))
                        .Where(s => FinalStages.Contains(s.Value.To))
                        .Select(x => x.Index)
```

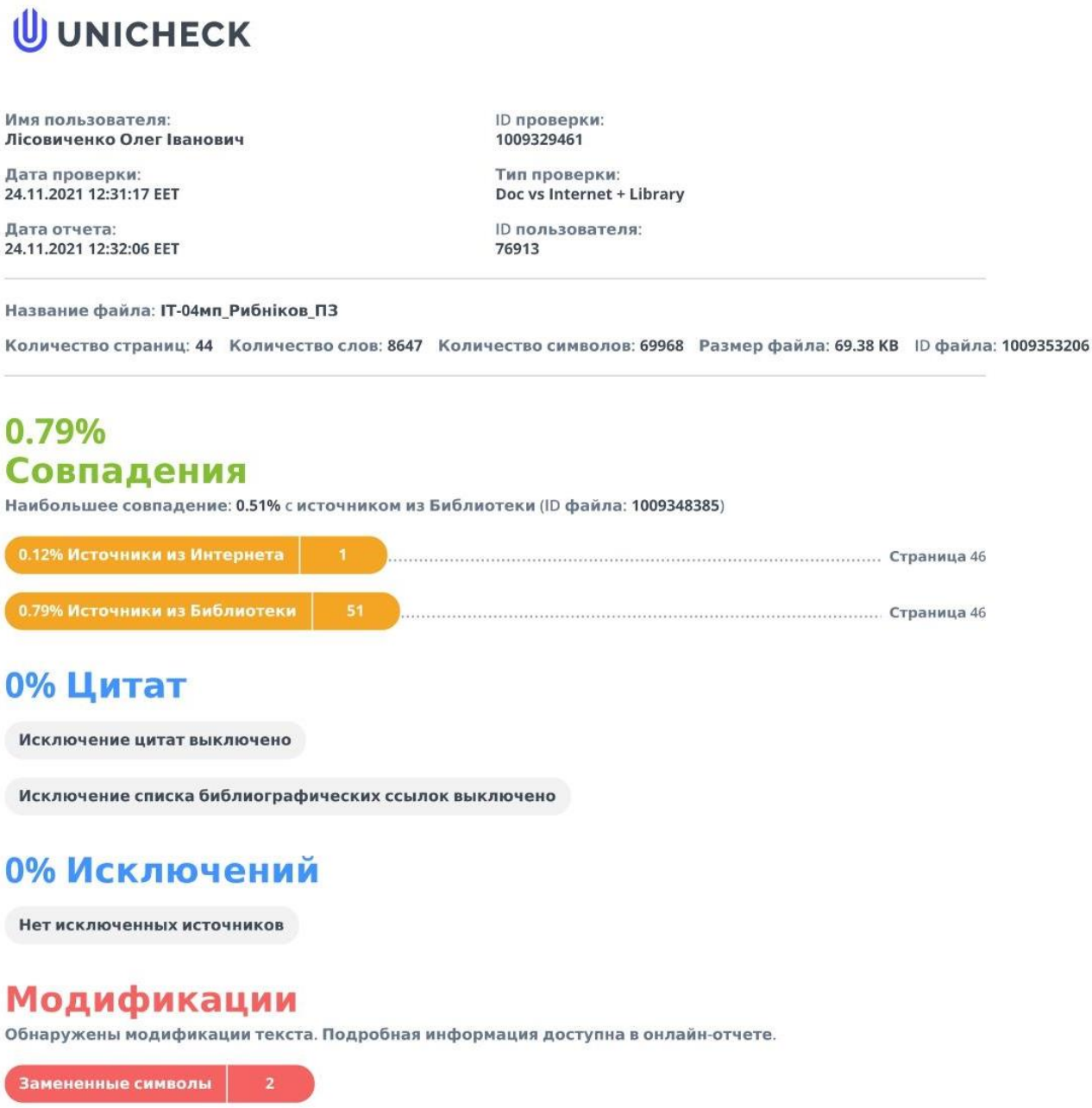
```
                        .ToArray());

                _maxScenarioLength = maxScenarioLength;
                _initialPopulationGenerator =
initialPopulationGenerator;
                Generate();
            }

            public ScenarioChromosome(
                Scenario scenario,
                Step[] allSteps,
                string[] initialStages,
                string[] finalStages,
                int maxScenarioLength,
                IInitialPopulationGenerator<Step, Scenario>
initialPopulationGenerator)
            {
                Scenario = scenario;
                _allSteps = allSteps;
                InitialStages = initialStages;
                FinalStages = finalStages;
                _maxScenarioLength = maxScenarioLength;
                _initialPopulationGenerator =
initialPopulationGenerator;
            }

            public override void Generate()
            {
                var length = Random.Next(2, _maxScenarioLength);

                Scenario = _initialPopulationGenerator.Generate(
                    steps => new Scenario(steps),
                    n => _allSteps[n], length);
            }

            public void GenerateIntelligent()
            {
                var length = Random.Next(2, _maxScenarioLength);

                var l = new LinkedList<Step>();
                var firstIndex = _initialSteps.Value[Random.Next(0,
_initialSteps.Value.Length - 1)];
                l.AddFirst(_allSteps[firstIndex]);

                Scenario = _initialPopulationGenerator.Generate(
                    steps => new Scenario(steps),
                    n => _allSteps[n], length);
            }


            public override IChromosome CreateNew()
```

```
            {
                var c = new ScenarioChromosome(
                    _allSteps, InitialStages, FinalStages,
_maxScenarioLength, _initialPopulationGenerator);
                c.Generate();
                return c;
            }

            public override IChromosome Clone()
            {
                return new ScenarioChromosome(Scenario, _allSteps,
InitialStages, FinalStages, _maxScenarioLength,
_initialPopulationGenerator);
            }

            public override void Mutate()
            {
                var stepIndex = Random.Next(0, _allSteps.Length);
                Scenario.Steps.RemoveFirst();
                Scenario.Steps.AddLast(_allSteps[stepIndex]);
            }

            public override void Crossover(IChromosome pair)
            {
                var otherChromosome = pair as ScenarioChromosome;
                var thisLastStep = Scenario.Steps.Last.Value;
                if (Scenario.Steps.Last.Previous.Value.To !=
thisLastStep.From)
                {
                    Scenario.Steps.RemoveLast();
                }

                var otherLastStep =
otherChromosome.Scenario.Steps.Last.Value;
                if
(otherChromosome.Scenario.Steps.Last.Previous.Value.To !=
otherLastStep.From)
                {
                    otherChromosome.Scenario.Steps.RemoveLast();
                }

                if (Scenario.Steps.Last.Value.To ==
otherLastStep.From)
                {
                    Scenario.Steps.AddLast(otherLastStep);
                }

                if (otherChromosome.Scenario.Steps.Last.Value.To ==
thisLastStep.From)
                {
```

```
otherChromosome.Scenario.Steps.AddLast(thisLastStep);
                }
            }
        }

        public class ScenarioFitnessFunction : IFitnessFunction
        {
            public double Evaluate(IChromosome chromosome)
            {
                double score = 1;
                if (chromosome is ScenarioChromosome sc)
                {
                    if (sc.Scenario.Steps.Count == 0) return 0;
                    if
(!sc.InitialStages.Contains(sc.Scenario.Steps.First.Value.From)) score
+= 2;
                    if
(!sc.FinalStages.Contains(sc.Scenario.Steps.Last.Value.To)) score +=
2;

                    var l = sc.Scenario.Steps;
                    var node = l.First;
                    while (node is {Next: {} next})
                    {
                        if
(sc.InitialStages.Contains(next.Value.From)) score += 1;
                        if (node.Value.To != next.Value.From) score +=
2;
                        node = next;
                    }

                    return Math.Pow(score, -1);
                }

                return 0;
            }
        }
```

ДОДАТОК Г

Результати перевірки роботи на співпадіння



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009329461

Дата проверки:
24.11.2021 12:31:17 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
24.11.2021 12:32:06 EET

ID пользователя:
76913

Название файла: ІТ-04мп_Рибніков_ПЗ

Количество страниц: 44 Количество слов: 8647 Количество символов: 69968 Размер файла: 69.38 KB ID файла: 1009353206

0.79%
Совпадения

Наибольшее совпадение: 0.51% с источником из Библиотеки (ID файла: 1009348385)

0.12% Источники из Интернета 1 ..... Страница 46

0.79% Источники из Библиотеки 51 ..... Страница 46

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы 2