

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕТИЧНОГО
АЛГОРИТМУ РОЗМІЩЕННЯ ДЖЕРЕЛ ЖИВЛЕННЯ

В РОЗПОДІЛЕНИХ МЕРЕЖАХ

RESEARCH AND SOFTWARE IMPLEMENTATION OF GENETIC ALGORITHM
FOR PLACING POWER SUPPLIES IN DISTRIBUTED NETWORKS

Виконав: студент(ка) 2 курсу, групи КНТ-220м
Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Поповський О.Д.

(прізвище та ініціали)

Керівник Зайко Т.А.

(прізвище та ініціали)

Рецензент Коцур М.І.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Поповського Олексія Дмитровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація генетичного алгоритму розміщення джерел живлення в розподілених мережах. Research and Software Implementation of Genetic Algorithm for Placing Power Supplies in Distributed Networks

керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) “06” грудень 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Зайко Т.А., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “1” жовтня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка структури програми	6 тиждень	Розділ 2
4	Розробка програми	7-9 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	10 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
7	Захист роботи	12 тиждень	

Студент(ка)

(підпис) Поповський О.Д.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Зайко Т.А.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
88 с., 20 рис., 14 табл., 4 дод., 19 джерел.

ДИСКРЕТНА ОПТИМІЗАЦІЯ, ПРОГРАМНІ ЗАСОБИ, ГЕНЕТИЧНИЙ
МЕТОД, ОПТИМАЛЬНЕ РОЗТАШУВАННЯ ЕЛЕМЕНТІВ, СЕРЕДОВИЩЕ
РОЗРОБКИ.

Об'єкт дослідження – процес пошуку оптимального розташування елементів.

Предмет дослідження – генетичні методи розв'язання завдань оптимального розташування елементів.

Метою роботи є дослідження та реалізація генетичного алгоритму пошуку оптимального розміщення джерел живлення в розподілених мережах.

Матеріали, методи та технічні засоби: C# як мова програмування, Microsoft Visual Studio як середовище розробки.

Результати. Розроблено програмне забезпечення, призначене для пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації.

Практична цінність роботи полягає у створенні забезпечення для розв'язання практичної задачі пошуку оптимального розміщення джерел живлення в розподілених мережах.

Висновки. Розроблено програмні засоби, які на основі застосування генетичних алгоритмів можуть розв'язувати завдання пошуку оптимального розміщення елементів.

Галузь використання – інтелектуальні системи для проєктування електричних мереж.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 88 pages, 20 figures, 14 tables, 4 appendixes, 19 sources.

DISCRETE OPTIMIZATION, SOFTWARE, GENETIC METHOD, OPTIMAL LOCATION OF ELEMENTS, DEVELOPMENT ENVIRONMENT.

The object of research is the process of finding the optimal location of elements.

The subject of the research is genetic methods for solving problems of optimal arrangement of elements.

The purpose of this work is to study and implement a genetic algorithm for placing power supplies in distributed networks.

Materials, methods and technical tools: programming language C#, integrated development environment Microsoft Visual Studio.

Results. The software to find the optimal location of power supplies in distributed networks based on genetic optimization has been developed.

The practical value of the work lies in the providing a solution to the practical problem of finding the optimal location of power supplies in distributed networks.

Conclusions. Software tools that can solve the problem of finding the optimal placement of elements based on the usage of genetic algorithms has been developed.

Scope of use – intelligent systems for designing electrical networks.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	11
1.1 Поняття про електричні мережі	11
1.2 Задача побудови раціональної конфігурації розподільної електричної мережі	14
1.3 Аналіз методів розв'язання завдань оптимального розташування елементів	16
1.4 Висновки за розділом 1	22
2 Матеріали і методи	24
2.1 Вибір мови програмування	24
2.2 Вибір середовища розробки ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах	27
2.3 Визначення цільової функції для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах ..	29
2.4 Розробка генетичного методу для пошуку оптимального розміщення джерел живлення в розподілених мережах	31
2.5 Структурна схема програми	39
2.6 Висновки за розділом 2	40
3 Опис програми	42
3.1 Призначення й умови застосування програми	42
3.2 Характеристики програми	43
3.3 Звернення до програми	44
3.4 Початкові та вихідні дані	44
3.5 Повідомлення програмісту	44
3.6 Висновки за розділом 3	45
4 Експлуатація, тестування та експериментальне дослідження програми	46
4.1 Призначення й умови застосування програми	46

4.2 Виконання програми.....	46
4.3 Призначення програми	47
4.4 Повідомлення оператору	47
4.5 Інтерфейс програми	48
4.6 Тестування та експериментальне дослідження програми	51
4.7 Висновки за розділом 4	53
Висновки.....	54
Перелік джерел посилання	56
Додаток А Технічне завдання.....	58
Додаток Б Опис програми	62
Додаток В Текст програми	66
Додаток Д Слайди презентації	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ГА – генетичний алгоритм;

ДЖ – джерела живлення;

ДО – дискретна оптимізація;

ЛП – лінійне програмування;

НЛП – нелінійне програмування;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ТЗ – технічне завдання;

УДП – узагальнене диз'юнктивне програмування.

ВСТУП

Важливим елементом життєзабезпечення фізичних та юридичних осіб є електричні мережі, через які поставляється електрична енергія до споживачів. Проте існуючі електричні мережі є досить застарілими та недостатньо розумними, щоб відповідати сучасним вимогам. Електрична мережа повинна бути достатньо розумною, щоб мати можливість працювати з тисячами генераторів різних технологій та розмірів, а також забезпечувати електроенергію мільйонам споживачів надійним та стійким способом. Тому дуже актуальним завданням є модифікація електричних мереж, у тому числі шляхом оптимального розміщення їх елементів [1]–[7].

Для пошуку оптимального розташування джерел живлення в розподілених мережах ефективно можуть застосовуватися методи генетичної оптимізації, які відносяться до дискретних методів та не висувають вимог до цільового критерію [8]–[11].

Таким чином, актуальною є розробка програмного забезпечення для пошуку оптимального розташування джерел живлення в розподілених мережах на основі генетичної оптимізації.

Для досягнення поставленої мети роботи необхідно розв'язати такі завдання:

- дослідити проблемну область побудови раціональної конфігурації розподільних електричних мереж;
- проаналізувати сучасні методи пошуку оптимального розташування елементів;
- розробити удосконалений генетичний метод для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- розробити структуру програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;

- розробити програмне забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- протестувати розроблену програму для розв’язання задачі пошуку оптимального розміщення джерел живлення в розподілених мережах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття про електричні мережі

Поеднання різних електричних елементів або компонентів, які пов'язані будь-яким чином, називається електричною мережею [1]-[7].

Електричні мережі є найскладнішими і найбільшими мережами та системами у світі. У всьому світі електромережі є застарілі та недостатньо розумні, щоб відповідати вимогам 21 століття. Навіть у розвинених країнах мільйони людей залишаються без електричного живлення протягом декількох годин щодня. Останнім часом кількість серйозних відмов, спричинених погодою, швидко зростає, головним чином тому, що більшість компонентів електричної мережі розташовані ззовні та не досить оптимально, що обумовлює необхідність реконструкції таких мереж з обов'язковим пошуком оптимального розташування їх структурних елементів. Навіть у країнах із сучасною електричною мережею досить часто виникають проблеми з їх функціонуванням. Електрична мережа повинна бути достатньо розумною, щоб мати можливість працювати з тисячами генераторів різних технологій та розмірів, а також забезпечувати електроенергію мільйонам споживачів надійним та стійким способом. Тому дуже актуальним завданням є модифікація електричних мереж, у тому числі шляхом оптимального розміщення їх елементів [1]-[7].

Проаналізуємо різні типи схем підключення електричних елементів. Схема, яка містить багато електричних елементів, таких як резистори, конденсатори, котушки індуктивності, джерела струму та джерело напруги (як змінного, так і постійного струму), називається складною мережею. Такі мережі неможливо легко розв'язати за допомогою простого закону Ома чи закону Кірхгофа. Тобто ми вирішуємо ці схеми специфічними методами, наприклад, за допомогою теореми Нортона, теореми Тевеніна, теореми суперпозиції тощо. Електрична схема представляє собою замкнутий контур,

що дає зворотний шлях для струму. Провідний шлях, по якому може протікати струм, називається ланцюгом [1]-[7].

Існують різні типи електричних ланцюгів, такі як: серія ланцюга, послідовно-паралельна схема, схема зірка-дельта, резистивна схема, індуктивна схема, ємнісний контур, резистивно-індуктивний контур (ланцюг RL), резистивно-ємнісний контур (RC-схема), ємнісно-індуктивний контур (LC схема), лінійна схема, нелінійна схема, односторонні схеми, двосторонні схеми, активна схема, пасивна схема, відкрита схема, коротке замикання [1]-[7].

У ланцюгових схемах всі електричні елементи (джерела напруги або струму, котушки індуктивності, конденсатори, резистори тощо) з'єднані послідовно, тобто існує лише один шлях для електрики, що рухається, і жодна інша гілка не складається в цій ланцюзі [1]-[7].

У паралельних контурах всі електричні елементи (джерела напруги або струму, котушки індуктивності, конденсатори, резистори тощо) з'єднані паралельно, тобто існує багато шляхів для проходження електрики, і мінімальні гілки в цій схемі дві [1]-[7].

Якщо елементи схеми послідовно з'єднані в одних частинах і паралельні в інших, це буде послідовно-паралельна схема. Іншими словами, це комбінація послідовних і паралельних ланцюгів. У схемі зірки-дельти електричні елементи з'єднані таким чином, що не визначені в термінах конфігурації серії, паралелі або серії паралельних. Такі схеми можна вирішити за допомогою різних трансформацій [1]-[7].

Однією з ключових особливостей інтелектуальної мережі є технологія автоматизації, яка дозволяє утиліті регулювати та керувати мільйонами пристроїв з центрального місця. Концепція інтелектуальної мережі існує вже багато років, але для нас важливо зрозуміти, що це насправді означає, і як інтелектуальна мережа може сприяти надійному, стійкому та економічному енергопостачанню. Ще однією ключовою, але складною особливістю концепції інтелектуальної мережі є залучення споживача через двосторонній

зв'язок між постачальником електроенергії та електроприладами замовника. Майбутня електрична мережа також повинна мати можливість обробляти велику кількість електромобілів, які будуть підключені до мережі для зарядки та розрядки. Електромобіль можна легко та зручно інтегрувати в електричну мережу. Електромобіль представляє новий попит на електроенергію протягом періодів зарядки, а також може грати важливу роль як розподілені накопичувальні пристрої, які можуть подавати електроенергію назад до мережі [1]-[7].

Поточні електромережі орієнтовані на виробництво, що вимагає масивної мережі передачі та розподілу для передачі електроенергії у великі центри навантаження. Такі підходи працювали століттями, особливо за регульованою вертикально інтегрованою моделлю. Але відокремлення передачі від виробництва, залишаючи останнє приватним інвесторам, створило вакуум. Наприклад, на деяких ринках є занепокоєння тим, що недостатньо інвестицій йде на передачу електроенергії, або що планування передачі та генерації стало роздвоєним, що має негативні наслідки для обох.

До недавнього часу інвестиції в мережі передачі енергії, як правило, відставали від інвестицій у генерацію, роблячи більший акцент на ролі управління перевантаженнями у розробці ринку, ніж це було деінде. Однак, останнім часом знову зосереджено увагу на передачі електроенергії, де є кілька ініціатив щодо вдосконалення інвестицій та ринків передачі. Майже у всіх взаємопов'язаних мережах потужність передачі є недостатньою в періоди пікового попиту, так само, як і потужність виробництва, що впливає на здатність операторів ринку пропонувати найменш витратні варіанти виробництва для задоволення місцевих навантажень, змушуючи їх розходитися від так званого необмеженого оптимального потоку. Коли передача обмежена, для обслуговування локальних навантажень часто потрібно використовувати більш дорогу локальну генерацію, навіть коли дешевші ресурси доступні десь у мережі. Це тягне за собою значні витрати, які в кінцевому рахунку передаються споживачам через підвищення цін. Зі

збільшенням визнання важливості вузьких місць передачі, оператори ринку розробили різні методи, щоб краще використовувати обмежену потужність передачі та заохочувати як навантаження, так і генерацію до визнання обмежень мережі. Місцеве або вузлове ціноутворення, є одним із інструментів, розроблених у відповідь на цю проблему. Незважаючи на те, що спочатку не всі сприймали цю концепцію, деякі ринки перероблені з метою її включення. Такі концепції, як права власності на передачу, аукціони на передачу та форвардний ринок для розподілу обмеженої пропускної спроможності, також розвинулись для вирішення цих складних питань [1]-[7].

Тому важливим є завдання розробки методів та засобів планування ефективного розташування різних елементів в електричних джерелах, що дозволить більш ефективно забезпечити зберігання та розподіл електричної енергії в різні проміжки часу (як в пікові години навантаження на мережу, так і в звичайний час).

1.2 Задача побудови раціональної конфігурації розподільної електричної мережі

В цілому завдання вибору рішень при побудові раціональної конфігурації розподільної електричної мережі включає в себе наступні підзадачі [1]-[7]:

- оптимальне розташування підстанцій;
- оптимізація прокладки ліній з урахуванням обмежень на місцевості;
- оптимальне закріплення споживачів за підстанціями;
- оптимальний вибір потужності підстанцій;
- вибір оптимального числа трансформаторів на підстанціях.

На даному етапі представляє інтерес саме завдання оптимального розміщення декількох джерел живлення однакового типорозміру в

розподільній електричній мережі методами генетичного програмування і одночасного закріплення споживачів за обраними джерелами живлення [1]-[7].

Нехай в декартовій системі координат (на площині) задана система точок с координатами $\{x_i, y_i\}_{i=1}^n$, які будемо називати надалі точками споживання електроенергії. Множина $I = \{1, 2, \dots, n\}$ представляє собою множину пунктів споживання електроенергії [1]-[7].

Кожному споживачеві зіставляється деяка вага S_i , що дорівнює споживаній цим споживачем потужності. Множина $\{S_i\}_{i=1}^n$ однозначно визначає обсяги споживання. Таким чином, задана також сумарна споживана потужність $S_{\text{розр}}$ усіма споживачами в розглянутій задачі (1.1):

$$S_{\text{розр}} = \sum_{i=1}^n S_i. \quad (1.1)$$

Нехай також в декартовій системі координат задана система точок з координатами $\{x'_j, y'_j\}_{j=1}^m$ – можливих місць розміщення джерел живлення, де m кількість цих можливих місць; апріорі передбачається, що це число m свідомо більше практично необхідного числа місць розміщення ВП.

Задамо типовий ряд використовуваних для вирішення завдання джерел електричної потужності (джерел живлення, генераторів). Тут використовуємо наступну угоду: вважаємо, що джерело живлення сам виробляє електричну енергію і умовно вважається в нашій задачі генератором [1]-[7]. У даній постановці вважаємо, що потужності всіх ДЖ однакові і рівні S_z . Очевидно, що для сумарної потужності, що $S_z^{\text{сум}}$ має виконуватися нерівність (1.2):

$$S_z^{\text{сум}} \geq S_{\text{розр}}. \quad (1.2)$$

З урахуванням усього вищевикладеного маємо наступну оптимізаційну задачу: потрібно вибрати найбільш економічний варіант розміщення ДЖ з

урахуванням витрат на доставку електроенергії до споживачів, причому слід вибрати оптимально наступні параметри [1]-[7]:

- місця розміщення джерел живлення із запропонованих m можливих місць розміщення;
- для кожного споживача визначити, за яким джерелом живлення він буде закріплений.

1.3 Аналіз методів розв'язання завдань оптимального розташування елементів

Як відзначено вище, розв'язання завдання побудови раціональної конфігурації розподільної електричної мережі пов'язано з необхідністю застосування методів дискретної оптимізації [12]. Проаналізуємо такі методи

Для розв'язання завдання оптимізації цільової функції при певних умовах можуть застосовуватися методи лінійного та цілочисельного програмування. Ці методи часто використовуються для розв'язання завдань розміщення, планування, складання розкладу та ін. Причиною частого використання таких методів є те, що моделі включають у більшості випадків дискретні подання часу в поєднанні з досить простими моделями продуктивності. Для випадку, коли не задіяні дискретні змінні u , проблема зводиться до проблеми лінійного програмування (ЛП). Це спеціальний клас опуклих задач оптимізації, для яких оптимальне рішення лежить у вершині багатогранника, що визначається нерівностями [12]. Розв'язання задач ЛП значною мірою спирається на симплексний алгоритм, хоча останнім часом методам внутрішніх точок приділяється підвищена увага для вирішення дуже великих задач через їх поліноміальну складність. Інші методи в значній мірі покладаються на симплексні гілки на основі ЛП та зв'язані методи, що складається з переліку характеристик, де підзадачі ЛП вирішуються на кожному вузлі та усуваються на основі обмежуючих властивостей. Ці методи вдосконалюються за допомогою технік різання площин, які забезпечують

більш жорсткі нижні межі для оптимуму. Варто зазначити, що оскільки задачі ЛП є NP-повноцінними, при вирішенні задач з великою кількістю змінних завжди можна потрапляти в обмеження часу, особливо якщо розрив інтегральності (різниця між оптимальною цілочисельною функцією та рішенням, знайденим за допомогою ЛП) є досить великим [12]-[14].

Методи нелінійного програмування (НЛП) мають перевагу перед моделями ЛП, оскільки вони здатні обробляти нелінійні функції і в основному використовуються для оптимізації в режимі реального часу. Ці моделі включають лише вбудовані безперервні змінні і є досить обмежувальними для планування та прогнозування, хоча вони є досить широко використовуваними. Задачі НЛП відповідають проблемам безперервної оптимізації, які полягають у знаходженні оптимуму (мінімуму) нелінійної цільової функції [12].

За умови, що функції є безперервними та диференційованими, і дотримуються певних обмежувальних кваліфікацій, місцеве оптимальне рішення задачі НЛП дається умовами Каруша-Кун-Таккера. Рішення проблем НЛП спирається або на послідовний алгоритм квадратичного програмування (SQP), або на метод зменшеного градієнта. Більшість методів НЛП визначають загальний оптимум, якщо проблема є опуклою (тобто опуклими є цільова функція та обмеження). Коли функції НЛП не є опуклими, загальний оптимум не може бути гарантований. Одним із варіантів такого підходу є спроба модифікувати проблему, як правило, за допомогою експоненціальних перетворень, хоча кількість випадків, коли це можливо, є досить малою. Альтернативно, можна використовувати жорсткі глобальні методи оптимізації, які за останні кілька років досягли значного прогресу. Ці методи припускають, що в задачі присутні спеціальні структури, такі як білінійні, лінійні дробові та увігнуті відокремлювані функції. Незважаючи на те, що це може здатися досить обмежувальним, алгебраїчні моделі завжди зводяться до цих структур, за умови, що вони не включають тригонометричні функції. До методів нелінійного програмування в якійсь мірі можуть відноситися, так звані, неточні методи, такі як модельований відпал та генетичні алгоритми, які

не роблять жодних припущень щодо функцій, але не можуть гарантувати чіткі рішення за скінченну кількість часу. Крім того, ці методи не формують проблему як математичну програму, оскільки вони включають методи процедурного пошуку, які, в свою чергу, вимагають певного типу дискретизації, а порушення обмежень вирішується за допомогою спеціальних штрафних функцій [12]-[14].

Оптимізація на основі логіки – полягає у формуванні та вирішенні проблем дискретної оптимізації за допомогою логічних моделей та методів оптимізації. Ці методи, що полегшують формулювання проблем та часто зменшують комбінаторний пошук, починають мати значний вплив при плануванні завдань та розв'язанні інших завдань дискретної оптимізації. Двома основними методами є узагальнене програмуване роз'єднання та програмування обмежень [12]-[14].

Узагальнене диз'юнктивне програмування (УДП), основна ідея якого полягає у використанні булевих і неперервних змінних та формулюванні задачі з цільовою функцією з урахуванням трьох типів обмежень: глобальних нерівностей, які не залежать від дискретних рішень; роз'єднань, які є умовними обмеженнями, що включають оператор АБО; чисті логічні обмеження, які включають лише булеві змінні. Цільова функція включає деякий доданок $f(x)$ для безперервних змінних (наприклад, експлуатаційні витрати) та витрати c_k , які залежать від дискретного вибору. Умови типу рівностей/нерівностей $g(x) \leq 0$ повинні виконуватися незалежно від дискретних умов, а інші нерівності $h_{jk}(x) \leq 0$ – це умовні обмеження, які повинні бути виконані, коли відповідна логічна змінна виконується. Проблема УДП являє собою розширення диз'юнктивного програмування, яке в минулому використовувалось як основа для виведення площин різання для алгебраїчної задачі. Цікаво відзначити, що будь-яку проблему УДП можна переформулювати як проблему НЛП. Це досягається переформулюванням роз'єднань за допомогою опуклої трансформації корпусу або з використанням великої кількості обмежень, а пропозиційні логічні твердження

переформульовані як лінійні нерівності. Для лінійного випадку проблемного УДП, і коли відсутні логічні обмеження, запропоновано метод відхилення та зв'язку, який не покладається на обмеження та заснований безпосередньо на рівняннях роз'єднань [12]-[14].

Для нелінійного випадку задачі УДП розроблено алгоритми, які покладаються на отримання алгебраїчного опису опуклої оболонки нелінійних опуклих нерівностей. Переформулювання призводять до жорстких задач НЛП, тоді як алгоритми загалом використовують методи гілок і границь та зв'язані методи, де розгалуження виконується на диз'юнкціях. Для випадку мережевих процесів запропоновано логічний алгоритм зовнішнього наближення, що полягає у вирішенні підпроблем НЛП у зменшеному просторі, в якому ігноруються обмеження, які не застосовуються в диз'юнкціях, за допомогою чого можна покращити ефективність та надійність. У цьому методі основні задачі НЛП відповідають опуклому корпусу лінеаризації нелінійних рівностей. Крім того, для ініціалізації головної проблеми необхідно вирішити декілька підпроблем НЛП, щоб загальні умови в диз'юнкціях виконувалися. Штрафи також можуть бути додані для обробки ефекту неопуклостей [12]-[14].

Також для розв'язання проблем УДП використовується програмування обмежень, що представляє собою засіб оптимізації на основі логіки, основна ідея якого полягає у використанні компактних мов для висловлення проблем оптимізації з точки зору змінних, які є безперервними, цілими та / або булевими, та обмежень, бути поданим в алгебраїчній формі, як диз'юнкції або як умовні логічні твердження. Додатково можуть підтримуватися спеціальні неявні функції, обмеження для присвоєння різних значень цілочисельним змінним. Замість того, щоб покладатися на традиційні методи математичного програмування, тут виконується пошук по дереву за допомогою неявного перерахування. Дерево, як правило, перераховується за допомогою глибинного першого пошуку, при якому нижня межа задається частковими рішеннями, а верхня межа найкращим можливим рішенням. На кожному з вузлів у пошуку дерев розповсюдження обмежень виконується шляхом

зменшення змінних. Це передбачає для фактичного зменшення меж у випадку безперервних змінних та / або доменів у випадку дискретних змінних. Перший використовує процедури затягування меж для лінійних та монотонних функцій, тоді як остання виконується або методами висновку, або спеціальними процедурами [12].

Для розв'язання завдань дискретної оптимізації ефективно можуть використовуватися генетичні алгоритми (ГА). ГА є метаявристикою, мотивованою процесом еволюції, і належить до великого класу еволюційних алгоритмів в інформатиці та обчислювальній математиці. Ці алгоритми часто використовуються для створення високоякісних рішень для оптимізації та пошуку проблем, зосереджуючись на таких біоінспекторах, як вибір, конвергенція чи мутації. Автор генетичних алгоритмів, Джон Холланд, розробив ГА на основі еволюційної теорії Дарвіна. Еволюційні алгоритми використовуються для вирішення проблем, які ще не мають чітко визначеного ефективного рішення. Цей підхід застосовується для вирішення задач оптимізації (планування, найкоротший шлях тощо), а також у моделюванні та моделюванні, де використовується функція випадковості. ГА – це рішення популяції кандидата (відомого як люди, тварини або генотипи) проблеми оптимізації, яка розробляється в напрямку кращих варіантів. Рішення кожного кандидата має набір характеристик (гени або фенотип), які можна еволюціонувати та змінювати; як правило, рішення зображуються двійковими цифрами у вигляді рядків з нулів та одиниць, хоча також допускається інший тип кодування. Еволюція, як правило, починається із спільноти рандомізованих індивідів і є ітераційним процесом, коли популяція розглядається як метод генерації для кожного відтворення. Для кожного покоління вимірюється підготовленість кожного з населення. Однак придатність – це, як правило, значення об'єктивної ознаки, що вирішується. Коли достатньо підготовлених особин вибирають з імовірності серед існуючої популяції, і ген модифікують, щоб створити новий цикл генерації для всіх (рекомбінованих та потенційно мutowаних навання) [12]-[14]. Нове

покоління стратегій-кандидатів буде використано протягом наступного покоління процесу. Алгоритм зазвичай закінчується, коли генерується або максимальне число поколінь, або задоволення. Тому кожне наступне покоління є найбільш невідповідним для середовища населення. У межах підходів до пошуку популяції підтримуються. Кожна людина, яка представляє рішення заданого питання в обчислювальній складності. Кожен із сукупності нумерується як вектори кінцевої довжини компонентів, які вони мають. Компонент подібний до генів, і багато генів генерують хромосому. Фітнес-оцінка представлена кожному, хто має здатність людини виконувати свої обов'язки. ГА може підтримувати популяцію з певної кількості осіб разом із показниками фізичної підготовки, які вони мають. Та особина, що має хороші показники фізичної підготовки, має більше шансів на відтворення. Для відтворення вибирається будь-яка особа, яка має хороші показники фізичної підготовки, чиї спаровування та генерування доброго потомства здійснюються шляхом групування хромосом їхнього покоління. Коли народиться новонароджена дитина, кімната буде створена, оскільки чисельність населення є статичною. Незалежно від того, чи закінчиться термін дії менш придатних особин, існує ймовірність того, що будуть спроби відшукати альтернативні рішення у наступних поколіннях. Ці нові покоління забезпечують більше хороших генів, ніж у старшого покоління. Тому кожне нове покоління має гарні рішення, ніж старе покоління. Коли потомство породжується без істотних відмінностей, ніж потомство, породжене старим населенням, популяції сходяться. Перевагами ГА є: відсутність вимог до цільової функції; пошук рішення на основі великої кількості особин; стабільність роботи при незначних змінах вхідного сигналу або шумах, що виникають під час процесу; використання стохастичної стратегії [12]-[14].

Результати аналізу різних методів дискретної оптимізації, що можуть застосовуватися для розв'язання завдання побудови раціональної конфігурації розподільної електричної мережі, наведено у таблиці 1.1.

Таблиця 1.1 – Аналіз методів дискретної оптимізації

Критерій порівняння	Метод		
	Лінійне програмування	Нелінійне програмування	Генетичні методи
Вимоги щодо лінійності цільової функції	+	—	—
Наявність вимог щодо неперервності цільової функції	+	+	—
Вимоги щодо диференційованості цільової функції	+	+	—
Тип пошуку	Детермінований	Детермінований	Стохастичний
Висока часова складність	—	+—	+

Таким чином, в результаті аналізу різних методів дискретної оптимізації, прийнято рішення як базис використовувати генетичні методи [8]-[11], [14] для розв’язання завдання побудови раціональної конфігурації розподільної електричної мережі.

1.4 Висновки за розділом 1

Проаналізовано процес проектування мереж джерел живлення. Визначено, що для розробки програмного забезпечення розміщення джерел живлення в розподілених мережах необхідно розв’язувати завдання дискретної оптимізації.

Досліджено та проаналізовано різні методи дискретної оптимізації. Прийнято рішення як базис використовувати генетичні методи для розв’язання завдання побудови раціональної конфігурації розподільної електричної мережі.

Визначено основні вимоги, що повинно надавати програмне забезпечення побудови раціональної конфігурації розподільної електричної мережі:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);
- наявність засобів для відображення результатів роботи програми.

2 МАТЕРІАЛИ І МЕТОДИ

У попередньому розділі було проведено аналіз та дослідження процесу та методів розв'язання завдань дискретної оптимізації. У цьому розділі міститься опис прийнятих рішень щодо проектування та розробки генетичного алгоритму та програмного забезпечення для розміщення джерел живлення в розподілених мережах.

2.1 Вибір мови програмування

Для створення програмного забезпечення для розміщення джерел живлення в розподілених мережах використовується мова програмування C#, яка є об'єктно-орієнтованою мовою програмування, розробленою в 2000 році Андерсом Хейлсбергом в Microsoft як суперник мові Java, на яку вона досить схожа [15]-[17].

C # є мовою загального призначення, розробленою для створення програм на платформі Microsoft, і для роботи якої потрібна платформа .NET у Windows. C # часто розглядають як гібрид, який використовує найкраще з C та C ++ для створення справді модернізованої мови. Хоча платформа .NET підтримує кілька інших мов кодування, C# швидко стала однією з найпопулярніших. C# можна використовувати для створення майже будь-якого програмного забезпечення, але особливо часто її використовують при створенні настільних застосунків та ігор Windows. C # також може використовуватися для розробки веб-додатків і стає все більш популярною і для мобільних розробок. Крос-платформні інструменти, такі як Xamarin, дозволяють використовувати програми, написані на C#, майже на будь-яких мобільних пристроях [15]-[17].

C # широко використовується для створення ігрових застосунків за допомогою ігрового механізму. C# є дуже популярним інструментом для

створення цих додатків, і тому робить чудовий вибір для будь-якого програміста, який сподівається проникнути в індустрію розробки ігор, або для тих, хто цікавиться віртуальною реальністю [15]-[17].

C # має велику кількість функцій, які полегшують навчання. Це мова високого рівня, яка є порівняно легкою для читання, де багато найскладніших завдань абстраговано, тому програміст не повинен про них турбуватися. Наприклад, управління пам'яттю вилучається з відповідальності користувача та обробляється схемою збору сміття .NET [15]-[17].

C # є мовою статичного типу, тому код перевіряється перед перетворенням на застосунок. Це полегшує пошук помилок, що може бути особливо корисним для початківців. Незважаючи на те, що синтаксис C# є більш послідовним та логічним, ніж C++, тексти великих програм програмістам-початківцям даються не зовсім легко. Проте для розробки ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах мова C# може бути ефективно використана [15]-[17].

Оскільки ця мова є досить потужною, гнучкою та добре підтримуваною, вона швидко стала однією з найпопулярніших доступних мов програмування. На сьогодні це четверта за популярністю мова програмування, приблизно третина усіх розробників регулярно її використовують. Це також третя за величиною спільнота на ресурсі StackOverflow (яка була побудована за допомогою C#) з понад 1,1 мільйонами тем [15]-[17].

Таким чином, мова C# є досить досконалою мовою програмування, яка підтримується розробниками та має велику кількість добре реалізованих зручних функцій для створення графічного інтерфейсу, роботи з базами даних, веб-застосунками, математичними обчисленнями, реалізації паралельних обчислень, а також функцій узгодження шаблонів, повернення багатозначних значень, роботи з іменованими параметрами, класами тощо. Крім того, C # та .NET в наш час є крос-платформними, з власною компіляцією та відкритим кодом, та дозволяє розробляти велику кількість програмних застосунків: від

настільного програмного забезпечення та мобільних застосунків до 3D-ігор та складних програм [15]-[17].

Наведемо таблицю з порівняльним аналізом мов програмування (таблиця 2.1), що можуть бути використані для створення ПЗ для розміщення джерел живлення в розподілених мережах.

Таблиця 2.1 – Порівняння мов програмування для створення ПЗ для розміщення джерел живлення в розподілених мережах

Критерій порівняння	C#	Java	C++
Зручність використання для розміщення джерел живлення в розподілених мережах	+	±	±
Можливість розробки кросплатформного програмного забезпечення	±	+	+
Наявність засобів проєктування інтерфейсу користувача	+	±	–
Підтримка засобів створення ПЗ на основі процедурного програмування	+	–	+
Засоби створення ПЗ з використанням об'єктно-орієнтованої парадигми	+	+	+
Наявність засобів роботи з базами даних	+	+	±

На основі проведеного аналізу, для розробки програми для розміщення джерел живлення в розподілених мережах було обрано мову C#.

2.2 Вибір середовища розробки ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах

Для створення програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах доцільно обрати середовище розробки програмного забезпечення Microsoft Visual Studio, що являє собою множину інструментів для створення ПЗ [18]-[19].

Microsoft Visual Studio, як інтегроване середовище розробки, є програмою, яка сприяє розробці програмних застосунків. Microsoft Visual Studio призначено для охоплення всіх завдань програмування в одній програмі. Тому Microsoft Visual Studio пропонує центральний інтерфейс, що включає всі інструменти, необхідні розробнику [18]-[19].

Редактор коду середовища розробки Microsoft Visual Studio – це інструменти, що являє собою це текстовий редактор, призначений для написання та редагування вихідного коду. Редактори вихідного коду відрізняються від текстових, оскільки вони покращують або спрощують написання та редагування коду [18]-[19].

Компілятор у середовищі розробки Microsoft Visual Studio є інструментом, який перетворює вихідний код, написаний зрозумілою для читання / запису мовою, у форму, що виконується комп'ютером [18]-[19].

Налагоджувач у середовищі розробки Microsoft Visual Studio використовується під час тестування для налагодження прикладних програм [18]-[19].

Інструменти автоматизації побудови автоматизують загальні завдання розробника [18]-[19].

Крім того, корисними є функції браузера класів, що являють собою інструмент для вивчення та посилення на властивості об'єктно-орієнтованої ієрархії класів [18]-[19].

Переглядач об'єктів у середовищі розробки Microsoft Visual Studio використовується для перевірки об'єктів, створених у запусненій програмі [18]-[19].

Діаграма ієрархії класів дозволяє програмісту візуалізувати структуру об'єктно-орієнтованого коду програмування [18]-[19].

Порівняльний аналіз середовищ розробки, які можуть бути використані для створення програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах, наведено у таблиці 2.2.

Таблиця 2.2 – Порівняний аналіз різних середовищ розробки для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах

Критерій порівняння середовищ розробки	Microsoft Visual Studio	Eclipse	NetBeans
1	2	3	4
Зручність використання для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах	+	+	—
Наявність засобів автоматизації створення інтерфейсу користувача	+	+	+

Продовження таблиці 2.2

1	2	3	4
Механізми для роботи з репозиторіями документів	+	+	—
Швидкість модифікації та модернізації створюваних програм	+	—	+
Платформонезалежність	+	+	+

Виходячи з аналізу середовищ розробки (таблиця 2.2), середовищем розробки для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах було обрано Microsoft Visual Studio.

2.3 Визначення цільової функції для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах

Цільова функція, мінімум якої будемо знаходити, може бути представлена у вигляді:

$$Z = \min(\sum_{i \in I} \sum_{j \in J} 3'_{ij} S_i l_{ij}). \quad (2.1)$$

Тут передбачається, що функція витрат на передачу електроенергії залежить від величини переданої потужності S і від відстаней від джерел живлення до споживачів l_{ij} . $3'_{ij}$ – питомі приведені витрати для передачі одиниці потужності на одиницю відстані. При цьому приймаємо припущення,

що існуюча електрична мережа не обмежує передачу потужності від джерел живлення (ДЖ) до споживачів.

Відстань від ДЖ до споживача може обчислюватися по одній з двох можливих метрик [12]-[14]:

- $l_{ij} = |x_i - x'_j| + |y_i - y'_j|$ – метрика Вебера;
- $l_{ij} = \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2}$ – метрика Евкліда.

Тут, у обох випадках, (x_i, y_i) – координати пункту споживання; (x'_i, y'_i)

- координати можливого пункту розміщення джерела.

Будемо надалі вважати, що маємо радіальну електричну мережі і що питомі приведені витрати $z'_{ij} = 1$, тобто цільова функція (2.1) приймає наступний вигляд [2]-[4]:

$$Z = \min(\sum_{i \in I} \sum_{j \in J} S_i l_{ij}). \quad (2.2)$$

У даній постановці завдання може інтерпретуватися як завдання дискретної оптимізації. У класичному випадку це завдання вирішують за допомогою методів комбінаторного аналізу. У спеціальній літературі часто зустрічається поняття комбінаторної задачі або задачі комбінаторного пошуку, проте важко знайти достатньо загальне визначення, яке покривало б все розмаїття завдань такого роду. Буде розглянуто комбінаторне завдання фіксованої розмірності [12]-[14].

Дано n кінцевих множин $U_1, U_2, \dots, U_m$ (безлічі значень змінних) і безліч значень параметрів P . Задана також функція обмежень $G(X, p) = G(x_1, x_2 \dots x_n, p) \rightarrow (0, 1)$, що описує область допустимих значень змінних $x_1, x_2 \dots x_n$ при значенні параметра $p \in P$.

Потрібно для заданих вихідних даних $p \in P$ побудувати і далі можлива одна з трьох постановок задачі [12]-[14]:

- будь-який набір значень $x_1, x_2 \dots x_n$ такий, що $G(X, p) = 1$ (завдання пошуку);

- всі набори значень $x_1, x_2 \dots x_n$ такі, що $G(X, p) = 1$ (завдання перерахування);
- такий набір значень $x_1, x_2 \dots x_n$, що $G(X, p) = 1$ і задана цільова функція $F(X, p) = 1$ приймає мінімальне значення (завдання оптимізації).

Вже згадане вище завдання відноситься до завдання дискретної оптимізації, розв'язання якого, як відзначено вище, може бути виконано за допомогою генетичних алгоритмів [8]-[11].

2.4 Розробка генетичного методу для пошуку оптимального розміщення джерел живлення в розподілених мережах

Будь-які дискретні обчислення вимагають попереднього аналізу трудомісткості рішення вихідної задачі і використовуваних алгоритмів її рішення (табл. 1.1). Завдання зазвичай оцінюються з точки зору розміру, тобто загальної кількості різних варіантів, серед яких потрібно знайти оптимальне рішення, а алгоритми оцінюються з точки зору складності. Виходячи з вищенаведеної концепції, можна вирішити завдання відноситься до класу задач великої розмірності. Так, при розгляді декількох десятків можливих варіантів електропостачання і близько 60-70 можливих місць розміщення генераторів можна сформулювати кілька десятків мільярдів можливих варіантів вирішення завдання [12]-[14].

Крім цього, другою відмінною рисою даного завдання є те, що виникає необхідність вирішення оптимізаційної задачі шляхом оптимізації одночасно відразу кілька параметрів, тобто дана задача відноситься до класу задач багатопараметричної оптимізації [12]-[14].

Третьою особливістю завдання, яка значно ускладнює її рішення, є те, що цільову функцію неможливо уявити в аналітичному вигляді, для кожного можливого варіанту вирішення завдання її доводиться обчислювати, використовуючи досить складний алгоритм, тобто цільова функція задається алгоритмічно [12]-[14].

Всі методи дискретної оптимізації можна умовно розділити на точні і наближені. До точних методів належать: метод повного перебору, метод неявного перебору, метод гілок і меж, метод динамічного програмування та ін. Щоб зрозуміти всю «красу» точних методів розв'язання задач комбінаторної оптимізації, зупинимося на характеристиці методу повного перебору [12]-[14].

Повний перебір всіх планів дозволяє напевно вирішити задачу. Інша справа, що для цього може знадобитися неприйнятно багато часу. Тільки тому і існує розгалужена теорія комбінаторних задач, основна мета якої – розробка та аналіз ефективних, тобто досить швидких алгоритмів для різних окремих випадків комбінаторних задач. Проте, перебір планів залишається найбільш універсальним методом вирішення. Якщо він не завжди придатний для практичних цілей, то корисний для дослідження задач, для порівняння з наближеними алгоритмами і т.п. [12]-[13].

Вважається, що конкретний алгоритм виконує вичерпний перебір, якщо можна гарантувати, що не пропущений жоден план, який міг би вплинути на результат. [12]-[13].

Найбільш часто для організації перебору планів використовується схема під назвою перебір з поверненням. Перебір планів завдання можна представити як обхід дерева перебору. Розмір дерева перебору може бути дуже великим. Досить часто при цьому можливий такий ефект: комбінаторна задача для малої розмірності вирішується досить просто, але при збільшенні розмірності швидко стає практично нерозв'язною. Цей ефект отримав назву комбінаторного вибуху [12]-[13].

Логічно зробити висновок, що повний перебір планів є вельми небажаним способом вирішення комбінаторних завдань, свого роду крайнім засобом при відсутності більш практичних алгоритмів. Слід використовувати будь-яку можливість, що дозволяє або істотно скоротити перебір з урахуванням специфіки конкретної задачі, або взагалі, якщо це можливо, відмовитися від перебору і використовувати інші методи вирішення [12]-[13].

Методи перебору і все їх удосконалення мають один, але дуже серйозний недолік: час їх роботи експоненціально зростає при збільшенні розмірності задачі. Для вирішення практичних завдань це в більшості випадків є неприйнятним. Інших же підходів, придатних відразу ж для всіх завдань комбінаторного пошуку, немає. Значить можна сподіватися тільки на алгоритми, що враховують специфіку конкретних завдань [12]-[13].

Алгоритми оптимізації, для яких є нетривіальні оцінки можливого відхилення рішення від оптимуму, називаються наближеними або субоптимальними [12]-[13].

Однак не завжди вдається оцінити похибка методу. Цілком типова ситуація, коли використовується алгоритм дає досить пристойні рішення, проте немає ніяких гарантій, що ці рішення близькі до оптимальних. Алгоритми, засновані на нестрогих міркуваннях «здорового глузду» і не мають ніяких гарантій близькості до оптимальних рішень, називаються евристичними алгоритмами [12]-[13].

Однією з різновидів евристичних алгоритмів є популярні останнім часом генетичні алгоритми (ГА). По суті генетичні алгоритми є оригінальною різновидом алгоритмів випадкового пошуку з послідовним поліпшенням. Дослідження показали, що внесення в такі методи елементів детермінованості дає значне поліпшення показників. Детермінованість цих методів полягає в моделюванні природних процесів відбору, розмноження і успадкування, що відбуваються за суворо визначеними правилами, при цьому основним є закон еволюції: «виживає найсильніший», який забезпечує поліпшення знаходять рішення. Іншим важливим фактором ефективності еволюційних обчислень є моделювання розмноження і успадкування. Розглянуті варіанти рішень можуть за певним правилом породжувати нові рішення, які будуть наслідувати кращі риси своїх «предків» [14].

Отже й у підтвердження аналізу у розділі 1, застосуємо для вирішення вищесформульованого завдання методи генетичного програмування.

Для будь-якого генетичного алгоритму виділяються чотири основних етапи [8]-[11]:

- 1) формування початкової популяції;
- 2) синтез нових хромосом (оператори схрещування і мутації);
- 3) цілеспрямована зміна знову отриманих хромосом (оператори інверсії);
- 4) селекція поточної популяції.

Перший етап побудови генетичного алгоритму для вирішення даного завдання полягає у виборі можливої кодування рішення, тобто в побудові хромосоми певної довжини, кожен ген в якій займає певну позицію і має певну довжину. Довжина кожного з генів, а також довжина всієї хромосоми буде прямо залежати від множин J , I . Задамося одним варіантом електропостачання, тобто кількістю однакових ДЖ, які покриють повністю потреби споживачів електричної енергії. Нехай це буде число $L1$ – число однакових ДЖ, що беруть участь в електропостачанні даного району. Задамося також типорозміром ДЖ [2]-[11].

Геометрично будь-яку хромосому можна представити у вигляді нитки з нанизаними на неї генами [8]-[11].

За допомогою хромосом формується в першу чергу початкова популяція [12]-[13].

Алгоритм формування початкової популяції. Для успішної роботи генетичних алгоритмів важливим є визначення правил, за якими формуватиметься популяція в початкову епоху свого існування, тобто на момент часу $t = 0$. Основна парадигма, яка буде покладена в основу цих правил, полягає в тому, що в початковій популяції повинен обов'язково бути присутнім весь генетичний матеріал завдання; тобто в нашому випадку в початковій популяції повинні обов'язково бути присутніми в якості можливих місць розміщення ДЖ всі точки з множини $J = \{1, 2, \dots, m\}$ [8]-[11].

Наведемо загальний вигляд хромосоми [8]-[11], використовуваної для вирішення даного завдання у таблиці 2.3.

Таблиця 2.3 – Загальний вигляд хромосоми

Значення	1	0	0	...	1	0
Можливі місця розташування ДЖ	1	2	3	...	(m-1)	m

Кількість одиниць в першому рядку має дорівнювати числу $L1$. Одиниця вказує на той факт, що в даному можливе місце розміщений ДЖ, нуль – ДЖ в цьому місці немає. У другому рядку зазначено, в який з можливих точок розміщений даний ДЖ [8]-[11].

Цільова функція не залежить від вартості ДЖ, так як ця складова буде однаковою. Хромосома фізично це допустиме рішення даної задачі. Значення цільової функції, обчислене для даної хромосоми, це вартість даного варіанту електропостачання [8]-[11].

Для вирішення конкретного завдання потрібно однозначно відобразити кінцеву множину варіантів на множину рядків відповідної довжини. Генетичний алгоритм за один крок робить обробку деякою популяції хромосом. Популяція $G(t)$ на кроці t являє собою кінцевий набір рядків [8]-[11]:

$$G(t) = (H_1^t, H_2^t, \dots, H_{PR}^t), \quad (2.3)$$

де PR – кількість особин (хромосом) в популяції, причому хромосоми в популяції не повинні повторюватися.

Алгоритм операції схрещування (кросинговеру). Визначальним для роботи генетичних алгоритмів є наявність ефективної операції схрещування. В якості вихідних даних маємо початкову популяцію, що складається з PR хромосом. Виберемо варіант статевого розмноження в популяції, тобто випадок, коли в створенні нової дочірньої хромосоми завжди беруть участь дві хромосоми. Виберемо в якості варіанту добору батьківських хромосом турнірний відбір [8]-[11].

Для цього застосуємо модифікований оператор кросинговеру, який дозволяє враховувати специфіку завдання розміщення джерел живлення. За

допомогою цього оператора розв'яжемо тестове завдання пошуку оптимального розташування десяти можливих місць джерел живлення, при цьому для покриття електричної потужності споживачів електричної енергії потрібно п'ять джерел живлення. Нехай маємо дві хромосоми-батьки, гени яких наведено у таблицях 2.4 та 2.5 [8]-[11].

Таблиця 2.4 – Інформація про першу батьківську хромосому (А)

Значення генів хромосоми	1	1	0	0	1	0	0	0	0	1
Місця розміщення джерел живлення	1	2	3	4	5	6	7	8	9	10

Таблиця 2.5 – Інформація про першу батьківську хромосому (Б)

Значення генів хромосоми	1	0	1	1	0	0	0	0	1	0
Місця розміщення джерел живлення	1	2	3	4	5	6	7	8	9	10

Кількість одиничних позицій в значеннях кодів хромосом є однаковою і дорівнює п'яти, за кількістю джерел живлення, що необхідно розташувати в регіоні [8]-[11]. Далі пропонується знайти однакові гени у батьківських хромосомах, які передаються до хромосом, які будуть нащадками, без будь-яких змін відповідних значень. Після цього відбувається стиснення значень батьківських хромосом батьків до ненульових елементів без урахування генів, які мають однакові значення у хромосом-батьків (таблиця 2.6).

Таблиця 2.6 – Батьківські хромосоми після перетворення

Перша хромосома-батько (А)			
Значення генів хромосоми	1	1	1
Місця розміщення джерел живлення	2	5	10
Друга хромосома-батько (Б)			
Значення генів хромосоми	1	1	1
Місця розміщення джерел живлення	3	4	9

Далі виконується вибір точки розриву (ця точка обирається випадково). Наприклад, така точка може знаходитися між другим та третім генами батьківських хромосом. Після цього відбувається схрещування, тобто заміна відповідних частин хромосом у місці розриву, внаслідок цього отримуються такі хромосоми нащадки (таблиця 2.7).

Таблиця 2.7 – Отримані хромосоми-нащадки

Перша хромосома-нащадок (В)										
Значення генів хромосоми	1	1	1	1	0	0	0	0	1	0
Місця розміщення джерел живлення	1	2	3	4	5	6	7	8	9	10
Друга хромосома-нащадок (Г)										
Значення генів хромосоми	1	0	0	0	1	0	0	0	0	1
Місця розміщення джерел живлення	1	2	3	4	5	6	7	8	9	10

Таким чином, шляхом передачі хромосомам-нащадкам однакових генів досягається ефект передачі нащадкам значень генів, що відповідають хромосомам-рішенням з кращими значеннями цільової функції [8]-[11].

У запропонованому методі вирішено використовувати відомий оператор мутації [8]-[11], що забезпечує класичний природний відбір. Однак замість комбінування батьківських якостей мутація вносить випадкові зміни в одну з хромосом. Після кожного схрещування для кожної з знову одержані хромосом формуємо «ознака мутації»: генеруємо випадкове число від 0 до 1; якщо це число менше, ніж коефіцієнт мутації, то запускаємо процедуру мутації для цієї хромосоми [8]-[11]. Ця процедура полягає в наступному [8]-[11]:

- 1) випадково обирається ген з ненульовим значенням [8]-[11];
- 2) виконується заміна значення цього гена на значення будь-якого іншого ненульового гена, обраного випадковим чином.

Оператор інверсії [8]-[11] змінює характер зв'язків між компонентами хромосоми. Він бере хромосому, випадковим чином вибирає в ній дві точки

розриву і має в своєму розпорядженні в зворотному порядку елементи, що потрапили між точками розриву [8]-[11].

Оператор селекції [8]-[11] формує нове покоління з хромосом з кращими значеннями цільової функції Z . Він знищує більшу частину популяції і освіжає генетичний матеріал, поповнюючи популяцію великою кількістю нових членів. В результаті виконання оператора селекції розмір популяції нового покоління знову стає рівним PR [8]-[11].

При реалізації генетичного алгоритму в даній постановці доводиться багаторазово реалізовувати евристичний алгоритм оптимального закріплення споживачів. Перш ніж обчислити значення цільової функції для даного варіанту рішення, необхідно закріпити кожен споживач за своїм джерелом електричної потужності [2]-[11].

В якості критеріїв зупинення використовуються відомі критерії [8]-[11], зокрема обмеження на максимальне число епох функціонування алгоритму, або визначення його збіжності шляхом порівняння пристосованості популяції на кількох епохах і зупинки процесу пошуку оптимального рішення при стабілізації цього параметра [8]-[11].

Таким чином, запропоновано генетичний метод пошуку оптимального розміщення джерел живлення в розподілених мережах, що використовує модифікований оператор схрещування, який дозволяє враховувати специфіку розв'язуваного завдання та передбачає формування нових хромосом-рішень таким чином, щоб вони задовольняли умовам задачі. Це дозволяє в процесі створення нової популяції уникати генерації хромосом-рішень, що є недопустимими (такими, що не відповідають умовам задачі), що, у свою чергу, зменшує час пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного пошуку.

2.5 Структурна схема програми

На основі ТЗ з метою розробки програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму, виконаємо розробку структури відповідного програмного забезпечення.

Структурну схему програми пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу наведено на рис. 2.1.

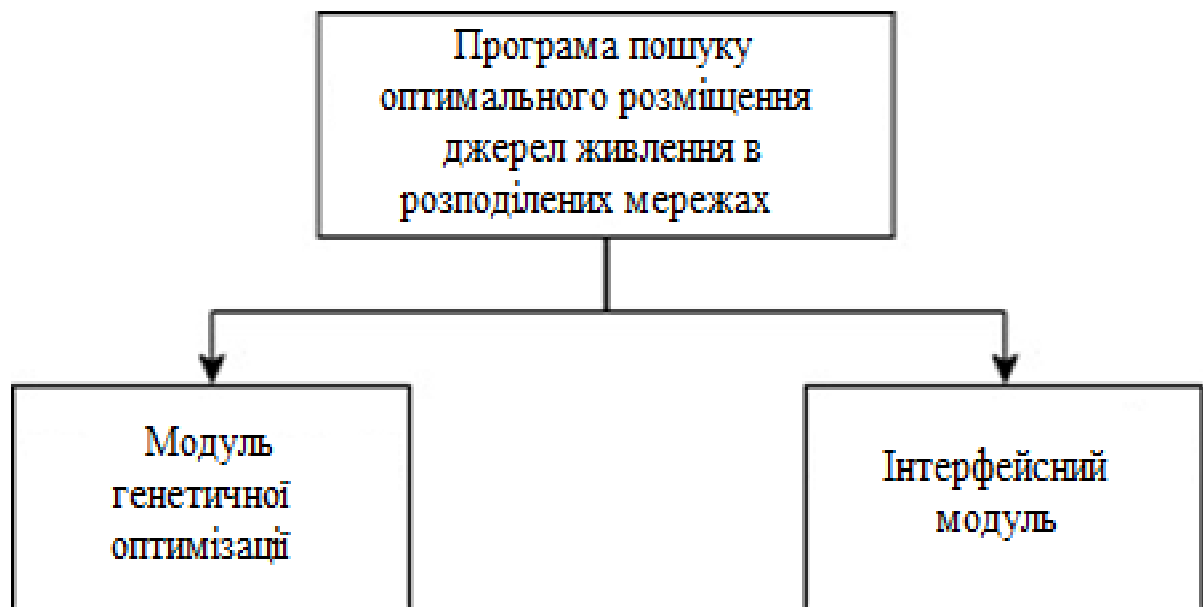


Рисунок 2.1 – Структурна схема програми

Як видно з рисунку 2.1, програма пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу має ієрархічну структуру. Програма складається з двох функціональних модулів:

- модуля генетичної оптимізації;
- інтерфейсного модуля.

Модуль генетичної оптимізації призначений для пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. В цьому модулі реалізується механізм генетичної оптимізації (формування множини рішень-хромосом, функції відбору, схрещування та мутації хромосом та формування нового покоління на основі нових хромосом та наявних хромосом-батьків, функції перевірки критеріїв зупинення генетичної оптимізації).

Інтерфейсний модуль призначений для організації взаємодії користувача з розробленою програмою пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу.

2.6 Висновки за розділом 2

Досліджено та обрано мову програмування C# для розробки програми пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. Як середовище розробки для створення ПЗ обрано Microsoft Visual Studio. Вибір цих засобів обґрунтовується зручністю розробки за їх допомогою.

Запропоновано генетичний метод пошуку оптимального розміщення джерел живлення в розподілених мережах, що використовує модифікований оператор схрещування, який дозволяє враховувати специфіку розв'язуваного завдання та передбачає формування нових хромосом-рішень таким чином, щоб вони задовольняли умовам задачі. Це дозволяє в процесі створення нової популяції уникати генерації хромосом-рішень, що є недопустимими (такими, що не відповідають умовам задачі), що, у свою чергу, зменшує час пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного пошуку.

З метою створення програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму, виконано розробку структури відповідного

програмного забезпечення. Відповідно до розробленої структури програма має ієрархічну структуру та складається з двох функціональних модулів: модуля генетичної оптимізації та інтерфейсного модуля.

3 ОПИС ПРОГРАМИ

Описано розроблене програмне забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму.

3.1 Призначення й умови застосування програми

Програмний продукт вирішує задачу комбінаторної оптимізації на прикладі задачі розташування джерел живлення і призначений для використання будь-якими суб'єктами (фізичними особами, підприємствами або фірмами), що мають справу з названим питанням для вирішення задач зниження логістичних витрат.

Програма забезпечує виконання наступних функцій:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);
- наявність засобів для відображення результатів роботи програми.

Для експлуатації ПП необхідні такі програмно-технічні засоби:

- обсяг ОЗП – не менш 512 Мб;
- комп'ютер під керуванням операційної системи сімейства WINDOWS;
- жорсткий диск з вільним простором не менше 150 Мб;
- монітор для виведення результатів роботи програми та взаємодії з користувачем;
- клавіатура;
- маніпулятор миша.

Для роботи програми повинна бути встановлена програмна платформа .NET Framework версії 4.5.1 або вище.

Функціональні характеристики розробленої програмної системи – згідно з технічним завданням, наведеним у додатку А. Опис програми та текст програми наведено у додатках Б та В, відповідно.

3.2 Характеристики програми

Час роботи програми пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму залежить від конфігурації комп'ютера, на якому вона виконується.

Програма пошуку оптимального розміщення джерел живлення в розподілених мережах має ієрархічну структуру і складається з двох функціональних модулів:

- модуля генетичної оптимізації;
- інтерфейсного модуля.

Модуль генетичної оптимізації призначений для пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. В цьому модулі реалізується механізм генетичної оптимізації (формування множини рішень-хромосом, функції відбору, схрещування та мутації хромосом та формування нового покоління на основі нових хромосом та наявних хромосом-батьків, функції перевірки критеріїв зупинення генетичної оптимізації).

Інтерфейсний модуль призначений для організації взаємодії користувача з розробленою програмою пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу.

Програма пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму відстежує коректність даних, що вводяться користувачем програми. У випадку

отримання некоректних даних на екран видається повідомлення про виникнення виняткової ситуації та її характер.

3.3 Звернення до програми

Для завантаження програми пошуку оптимального розміщення джерел живлення в розподілених мережах потрібно скопіювати відповідний файли ПЗ з носія на комп'ютер.

Для початку роботи з програмою пошуку оптимального розміщення джерел живлення в розподілених мережах необхідно запустити файл GAL.exe на виконання.

3.4 Початкові та вихідні дані

Вхідними даними до програми пошуку оптимального розміщення джерел живлення в розподілених мережах є числа, що описують параметри генетичного алгоритму відповідного модуля, значення яких вводяться користувачем з клавіатури у відповідні поля під час роботи з програмою пошуку оптимального розміщення джерел живлення, та опис позицій.

Вихідним даними програми є координати локації оптимального розташування джерел живлення за зазначеною кількістю топологій.

3.5 Повідомлення програмісту

Розроблений програмний продукт гарантує стійке функціонування. У разі виникнення виняткових ситуацій у програмі передбачено відображення відповідних повідомлень, текст і опис яких наведено в табл. 3.1.

Таблиця 3.1 – Повідомлення оператору

Текст повідомлення	Опис виняткової ситуації
Задайте размер популяции	Кількість особин у популяції повинна бути більша за нуль
Задайте максимальное число поколений	Максимальна кількість поколінь (генерацій дочірніх елементів) повинна бути більша за нуль
Задайте процент вероятности мутации	Ймовірність мутації дочірнього повинна приймати значення від 0 до 100 процентів
Задайте размер рабочей группы	Розмір робочої групи повинен приймати значення від 2 до розміру популяції-1
Задайте вероятность использования соседа при создании начальной популяции	Визначте ймовірність використання сусіда при створенні початкової популяції в діапазоні від 0 до 95 процентів
Укажите seed RNG	Вкажіть параметр «затравки» генератора випадкових чисел відмінний від 0

3.6 Висновки за розділом 3

Розроблено програмне забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації.

Описано призначення й умови застосування ПЗ, характеристики програми, вхідні та вихідні дані розробленого програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

ПЗ вирішує задачу пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації.

Програма забезпечує виконання таких функцій:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);
- наявність засобів для відображення результатів роботи програми.

Для експлуатації ПЗ необхідні такі програмно-технічні засоби:

- ПК на апаратній платформі x64 під керуванням ОС сімейства WINDOWS (Windows XP/7/8/10);
- жорсткий диск з вільним простором не менше 100 Мб;
- обсяг ОЗП – не менш 512 Мб;
- монітор з роздільною здатністю екрану 1280×1024 або вище;
- маніпулятор миша;
- клавіатура.

Для роботи програми повинна бути встановлена програмна платформа .NET Framework версії 4.5.1 або вище.

4.2 Виконання програми

Для завантаження програми пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації потрібно скопіювати файли програми з носія на комп'ютер у відповідну папку.

Для початку роботи з програмою пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації необхідно запустити відповідний файл на виконання.

4.3 Призначення програми

Програмний продукт вирішує задачу дискретної оптимізації на прикладі задачі розташування джерел живлення і призначений для використання будь-якими суб'єктами (фізичними особами, підприємствами або фірмами), що мають справу з названим питанням для вирішення задач зниження логістичних витрат.

Програма забезпечує виконання наступних функцій:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);
- наявність засобів для відображення результатів роботи програми.

4.4 Повідомлення оператору

Повідомлення оператору відповідають повідомленням, зазначеним в таблиці 3.1.

Приклади діалогових вікон із повідомленнями про виникнення виняткової ситуації наведено на рисунках 4.1, 4.2.

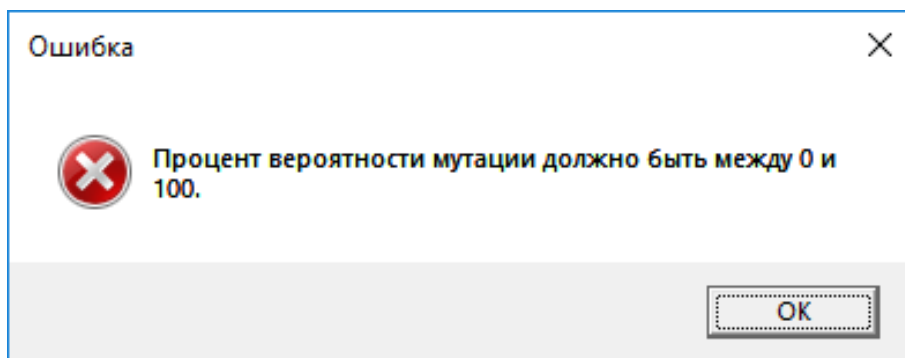


Рисунок 4.1 – Повідомлення про виняткову ситуацію

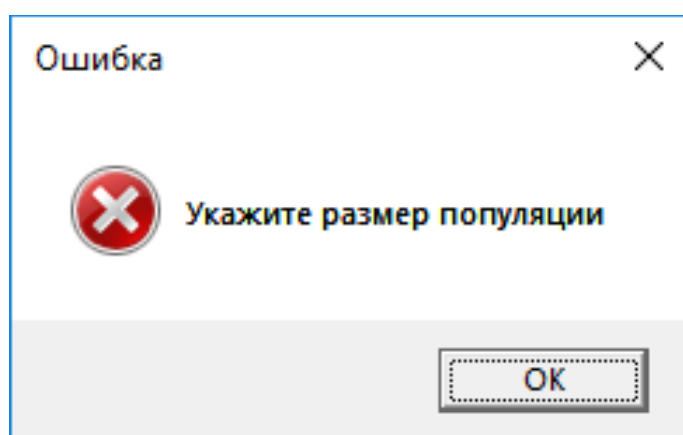


Рисунок 4.2 – Повідомлення про виняткову ситуацію

4.5 Интерфейс программы

Після запуску програми пошуку оптимального розміщення джерел живлення в розподілених мережах з'являється головне вікно.

Загальний вигляд головного вікна програми представлено на рис 4.3.

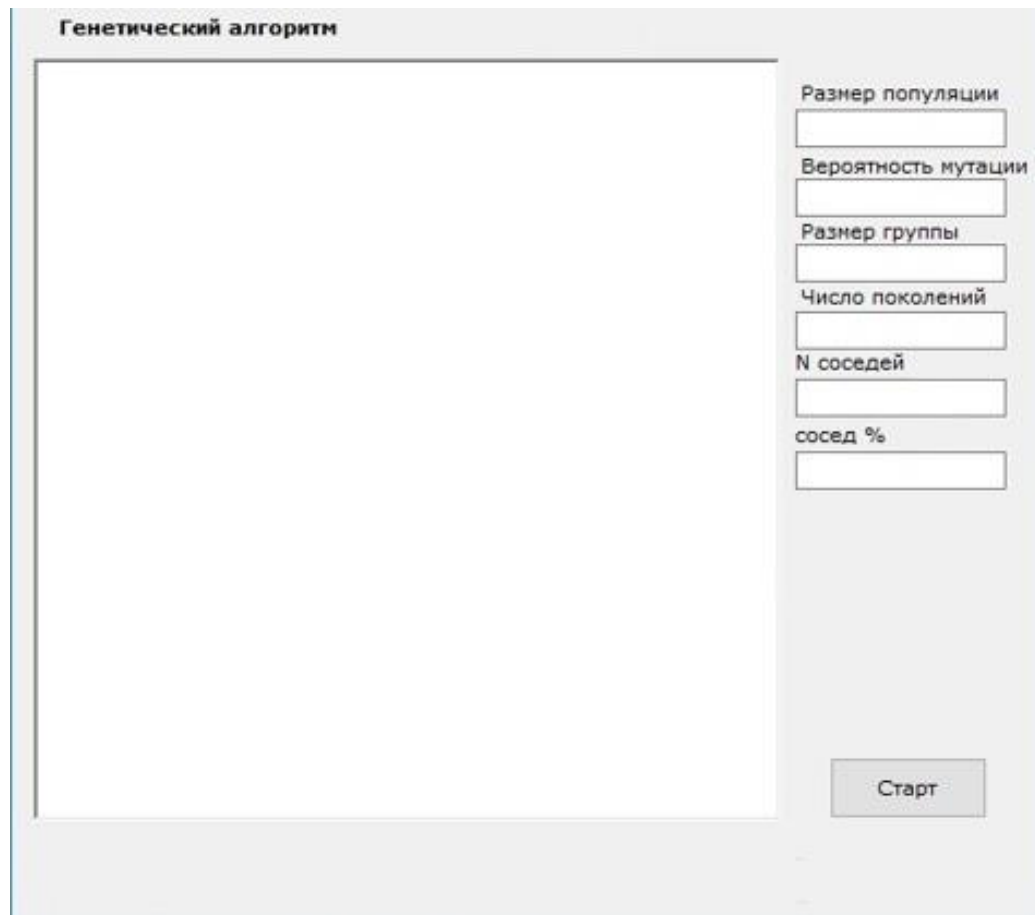


Рисунок 4.3 – Головне вікно програми

Призначення керуючих елементів вікна:

- поле «Размер популяции» дозволяє вказати розмір популяції, що будуть вдосконалюватися генетичним алгоритмом;
- поле «Вероятность мутации» дозволяє вказати ймовірність мутації дочірнім популяціям;
- поле «Размер группы» дозволяє вказати розмір групи, яка зберігається у масиві, до якого випадковим чином обираються елементи популяції, після чого він сортується;
- поле «Число поколений» дозволяє вказати максимальне число поколінь (генерацій дочірніх популяцій);
- поле «N соседей» дозволяє вказати число локацій-сусідів, що є найближчими до поточного міста;

- поле «сосед %» дозволяє вказати ймовірність вибору наступної локації зі списку його локацій-сусідів при формуванні мапи початкової популяції;
- кнопка «Старт» дозволяє розпочати розв'язок задачі за вказаними користувачем даними.

Після натискання кнопки «Старт» користувачу пропонується вказати координати джерел живлення в розподілених системах (рис. 4.4).

	X	Y	S
▶ V1, TP1			
V1, TP2			
V1, TP3			
V2, TP1			
V2, TP2			
V2, TP3			
V3, TP1			
V3, TP2			
V3, TP3			
V4, TP1			
V4, TP2			
V4, TP3			
*			

	TP1	TP2	TP3
▶			
*			

Рисунок 4.4 – Вікно для заповнення координат локацій та потужності споживачів

Призначення керуючих елементів вікна:

- поле «Кількість споживачів» дозволяє вказати кількість споживачів за трьома топологіями;
- кнопка «Сформувати значення розташування джерел» запускає розрахунок координат розташування та потужності джерел живлення з використанням генетичного алгоритму у трьох топологіях.

4.6 Тестування та експериментальне дослідження програми

Розв'яжемо задачу розміщення трансформаторів на території деякого регіону.

Як вихідні дані можна використовувати такі:

- 1) місця або координати розташування споживачів;
- 2) можливі навантаження споживачів електричних енергії;
- 3) тип джерел живлення електричної енергії;
- 4) можливі місця розташування джерел електричної енергії.

Знаючи місця розміщення споживачів електричної енергії, а також місця розміщення центрів джерел живлення та навантаження, можна обчислити мінімальне значення функції приведених витрат.

У таблиці 4.1 наведено дані для розрахунку.

Таблиця 4.1 – Вихідні дані для розрахунку

№	Координата X (м)	Координата Y (м)	Потужність S (кВА)
1	135	139	1050
2	492	420	2100
3	581	191	1150
4	250	215	1050
5	340	244	2100
6	552	382	330
7	572	351	1050
8	691	160	2100
9	752	170	330
10	754	185	1150

Будемо використовувати початкові параметри генетичного пошуку такі: кількість хромосом початкової множини рішень - 70, кількість ітерацій - 100.

Графік зміни значень цільової функції в залежності від номеру епохи популяції генетичного пошуку наведено на рисунку 4.5.

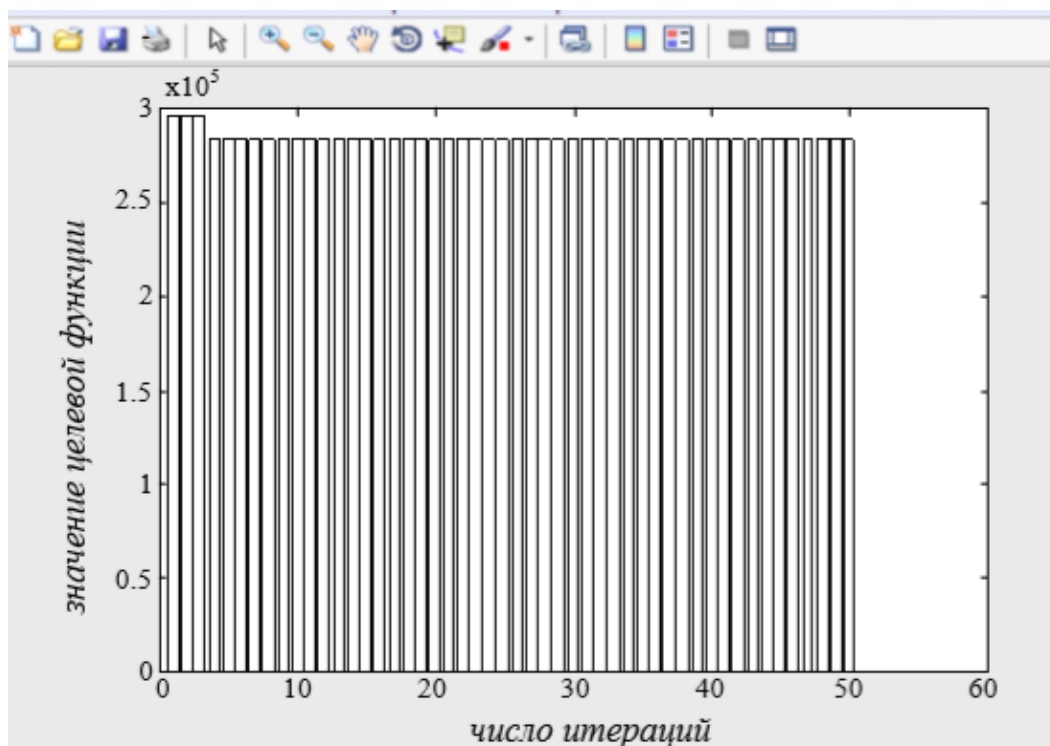


Рисунок 4.5 – Графік зміни цільової функції від номера ітерації

На рисунку 4.6 наведено візуалізація розташування джерел живлення.

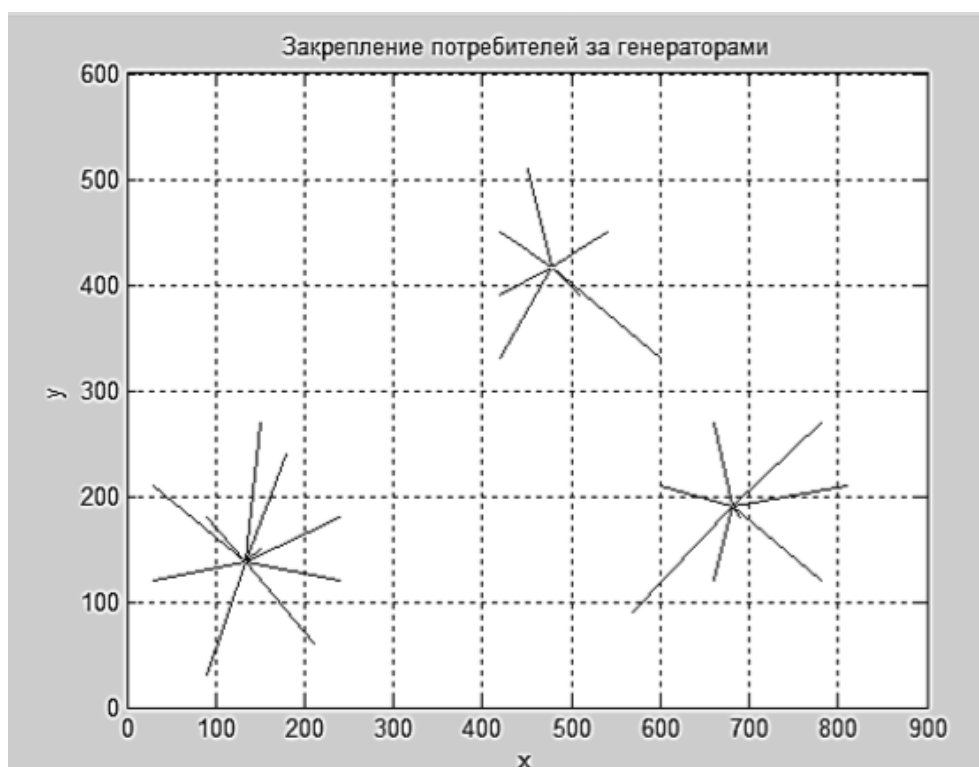


Рисунок 4.6 – Візуалізація розташування джерел живлення

Після апробації роботи запропонованого генетичного алгоритму отримано наступні результати:

- збігаються місця оптимального розміщення ТП, знайдені аналітично і за допомогою програми, що реалізує генетичний алгоритм для тестового завдання;
- практично збігається значення цільової функції, обчисленої аналітично, з результатом, який видає програма;
- результати роботи розробленої програми повністю збігаються з оптимальним закріпленням в тестовому прикладі.

4.7 Висновки за розділом 4

Наведено результати експлуатації, тестування та експериментального дослідження розробленої програми для пошуку оптимального розміщення джерел живлення в розподілених мережах.

Проведені експерименти показали, що розроблена програма дозволяє розв'язувати завдання пошуку оптимального розміщення джерел живлення в розподілених мережах на основі використання генетичних методів.

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи магістра було досліджено процес та методи пошуку оптимального розташування елементів.

Розв'язано наступні завдання:

- досліджено проблемну область побудови раціональної конфігурації розподільних електричних мереж;
- проаналізовано сучасні методи пошуку оптимального розташування елементів;
- розроблено удосконалений генетичний метод для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- виконано проєктування структури програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- розроблено програмне забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- виконано тестування розробленої програми для розв'язання задачі пошуку оптимального розміщення джерел живлення в розподілених мережах.

Досліджено та проаналізовано різні методи дискретної оптимізації. Прийнято рішення як базис використовувати генетичні методи для розв'язання завдання побудови раціональної конфігурації розподільної електричної мережі.

Визначено основні вимоги, що повинно надавати програмне забезпечення побудови раціональної конфігурації розподільної електричної мережі:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної

кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);

- наявність засобів для відображення результатів роботи програми.

Запропоновано генетичний метод пошуку оптимального розміщення джерел живлення в розподілених мережах, що використовує модифікований оператор схрещування, який дозволяє враховувати специфіку розв'язуваного завдання та передбачає формування нових хромосом-рішень таким чином, щоб вони задовольняли умовам задачі. Це дозволяє в процесі створення нової популяції уникати генерації хромосом-рішень, що є недопустимими (такими, що не відповідають умовам задачі), що, у свою чергу, зменшує час пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного пошуку.

Досліджено та обрано мову програмування C# для розробки програми пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. Як середовище розробки для створення ПЗ обрано Microsoft Visual Studio. Вибір цих засобів обґрунтовується зручністю розробки за їх допомогою.

З метою створення програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму, виконано розробку структури відповідного програмного забезпечення. Відповідно до розробленої структури програма має ієрархічну структуру та складається з двох функціональних модулів: модуля генетичної оптимізації та інтерфейсного модуля.

Розроблено програмне забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах на основі генетичної оптимізації.

Результати тестування показали, що розроблений програмний продукт може використовуватися за призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Meier, A. Electric power systems: a conceptual introduction [Text] / Alexandra Meier. – Hoboken: A Wiley-Interscience publication, 2006. – 321 p.
2. Blume, S. W. Electric Powersystem Basics. For the Nonelectrical Professional [Text] / Steven W. Blume. – Hoboken: A Wiley-Interscience publication, 2007. – 69 p.
3. Voropai, N. Electric Power System Transformations: A Review of Main Prospects and Challenges [Electronic Resource] / N. Voropai. – Access mode: <https://www.mdpi.com/1996-1073/13/21/5639/pdf>.
4. Воропай, Н.И. Иерархическое моделирование при обосновании развития электроэнергетических систем [Текст] / Н.И. Воропай // Математика в приложениях, 2003. – №4. – С. 24–27.
5. Khator, S.K. Power distribution planning: A review of models and issues [Text] // S.K. Khator // IEEE Transactions on Power Systems. – Vol. 12. – P. 1151–1159.
6. Свеженцева, О.В. Решение задачи оптимального закрепления множества потребителей за источниками питания методом комбинаторного анализа [Текст] / О.В. Свеженцева // Техничко-економические проблемы развития регионов. – 2006. – № 3. – С. 134–140.
7. Кудрин, Б.И. Электроснабжение промышленных предприятий [Текст] / Кудрин Б.И. – М.: Интермент Инжиниринг, 2006. – 670 с.
8. Chambers, L.D. The Practical Handbook of Genetic Algorithms [Text] / L.D. Chambers. – Fl.: CRC Press, 2000. – 520 p.
9. Nissen, V. Brief Introduction to Evolutionary Algorithms from the Perspective of Management Science [Text] / V. Nissen // Innovative Research Methodologies in Management. – 2018. – P. 165–210.
10. Gen, M. Genetic algorithms and engineering design [Text] / M. Gen, R. Cheng. – NJ : John Wiley and Sons, 1997. – 352 p.

11. Smith, S. Genetic and Evolutionary Computation [Text] / S. Smith, S. Cagnoni. – John Wiley and Sons, 2011. – 250 p.

12. Grossmann, I.E. Discrete Optimization Methods and their Role in the Integration of Planning and Scheduling [Electronic Resource] / I. E. Grossmann, S. A. Heever, and I. Harjunkski. – Access mode: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.8702&rep=rep1&type=pdf>.

13. Martin, A.D. A Review of Discrete Optimization Algorithms [Electronic Resource] / A.D. Martin, K.M. Quinn. – Access mode: <https://cpb-us-w2.wpmucdn.com/sites.wustl.edu/dist/4/1744/files/2019/01/A-Review-of-Discrete-Optimization-Heuristics-vlayf0.pdf>.

14. Alam, T. Genetic Algorithm: Reviews, Implementations, and Applications [Electronic Resource] / Tanweer Alam. – Access mode: https://d197for5662m48.cloudfront.net/documents/publicationstatus/38950/preprint_pdf/04a57dec4f996e2f2fcbf4f39c1d8414.pdf.

15. Троелсен, Э. С# и платформа .NET. Библиотека программиста [Текст] / Э. Троелсен. – Санкт-Петербург : Питер, 2007. – 800 с.

16. C# programming guide / [Electronic Resource] – Access mode : <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/>.

17. Mkhitarian, A. Why Is C# Among The Most Popular Programming Languages in The World? [Electronic Resource] / Armina Mkhitarian. – Access mode: <https://medium.com/sololearn/why-is-c-among-the-most-popular-programming-languages-in-the-world-ccf26824ffcb>.

18. What Is An Integrated Development Environment (IDE)? [Electronic Resource] / Armina Mkhitarian. – Access mode: <https://www.veracode.com/security/integrated-development-environment>.

19. Visual Studio / [Electronic Resource] – Access mode: <https://visualstudio.microsoft.com/ru/>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може застосовуватися для розв’язання завдань дискретної оптимізації, пов’язаних з пошуком оптимального розміщення елементів складних систем.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація генетичного алгоритму розміщення джерел живлення в розподілених мережах», затверджене наказом Національного університету «Запорізька політехніка» № 435 від 05 листопада 2021 р.

A.2 Призначення розробки

Програмний продукт призначений для вирішення задачі комбінаторної оптимізації на прикладі задачі розташування джерел живлення і призначений для вирішення задач зниження логістичних витрат.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати виконання наступних функцій:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);

- наявність засобів для відображення результатів роботи програми.

A.3.2 Вимоги до інтерфейсу програми

Програмне забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму повинно мати графічний інтерфейс користувача для можливості взаємодії з оператором.

A.3.3 Вимоги до надійності

Програмне забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму повинно відстежувати коректність і формат вхідних та вихідних даних.

A.3.4 Умови експлуатації

Експлуатація програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму повинна здійснюватися відповідно до експлуатаційної документації на розроблене програмне забезпечення.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму необхідні такі апаратні засоби:

- обсяг ОЗП – не менш 512 Мб;
- комп'ютер під керуванням операційної системи сімейства WINDOWS;

- жорсткий диск з вільним простором не менше 150 Мб;
- монітор для виведення результатів роботи програми та взаємодії з користувачем;
- клавіатура;
- маніпулятор миша.

A.3.6 Вимоги до маркування і пакування

Програмне забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму може бути записана на флеш-носії.

A.3.7 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму аналогічні вимогам, пропонованим до носія, на якому записана програма.

A.4 Вхідні та вихідні дані

Вхідними даними програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму, є числа, що описують параметри генетичного алгоритму відповідного модуля, значення яких вводяться користувачем з клавіатури у відповідні поля під час роботи з програмою пошуку оптимального розміщення джерел живлення, та опис позицій.

Вихідним даними програми є координати локації оптимального розташування джерел живлення за зазначеною кількістю топологій.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програмне забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму розроблена з використанням мови C#.

Б.2 Функціональне призначення

Функціональним призначенням програмного забезпечення є пошук оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму.

Програмне забезпечення дозволяє виконувати такі функції:

- задавання координат локацій, обраних для розміщення джерел живлення;
- введення початкових параметрів генетичного алгоритму (ймовірнісних характеристик, розміру популяції особин, максимальної кількості поколінь, введення ймовірності вибору наступної локації зі списку локацій-сусідів при формуванні мапи початкової множини рішень);
- наявність засобів для відображення результатів роботи програми.

Б.3 Опис логічної структури

Програма пошуку оптимального розміщення джерел живлення в розподілених мережах має ієрархічну структуру і складається з двох функціональних модулів:

- модуля генетичної оптимізації;
- інтерфейсного модуля.

Модуль генетичної оптимізації призначений для пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. В цьому модулі реалізується механізм генетичної

оптимізації (формування множини рішень-хромосом, функції відбору, схрещування та мутації хромосом та формування нового покоління на основі нових хромосом та наявних хромосом-батьків, функції перевірки критеріїв зупинення генетичної оптимізації).

Інтерфейсний модуль призначений для організації взаємодії користувача з розробленою програмою пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу.

Б.4 Використані технічні засоби

Для експлуатації програми пошуку оптимального розміщення джерел живлення в розподілених мережах необхідні такі програмно-технічні засоби:

- обсяг ОЗП – не менш 512 Мб;
- комп'ютер під керуванням операційної системи сімейства WINDOWS;
- жорсткий диск з вільним простором не менше 150 Мб;
- монітор для виведення результатів роботи програми та взаємодії з користувачем;
- клавіатура;
- маніпулятор миша.

Б.5 Виклик і завантаження

Для завантаження програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму необхідно скопіювати програмні файли з носія на комп'ютер у відповідну папку.

Для початку роботи з програмним забезпеченням пошуку оптимального розміщення джерел живлення в розподілених мережах за

допомогою генетичного алгоритму необхідно запустити відповідний файл на виконання.

Б.6 Вхідні та вихідні дані

Вхідними даними до програми пошуку оптимального розміщення джерел живлення в розподілених мережах є числа, що описують параметри генетичного алгоритму відповідного модуля, значення яких вводяться користувачем з клавіатури у відповідні поля під час роботи з програмою пошуку оптимального розміщення джерел живлення, та опис позицій.

Вихідним даними програми пошуку оптимального розміщення джерел живлення в розподілених мережах є координати локації оптимального розташування джерел живлення за зазначеною кількістю топологій.

ДОДАТОК В
Текст програми

```
// Файл loc.cs

using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.IO;
using System.Globalization;

namespace Tsp
{
    public class Loc : List<Loc>
    {
        public void CalculateLocDistances( int numberOfCloseLoc )
        {
            foreach (Loc loc in this)
            {
                loc.Distances.Clear();
                for (int i = 0; i < Count; i++)
                {

loc.Distances.Add(Math.Sqrt(Math.Pow((double)(loc.Location.X -
this[i].Location.X), 2D) +
Math.Pow((double)(loc.Location.Y - this[i].Location.Y), 2D)));

                }

            }

            foreach (Loc loc in this)
            {
                loc.FindClosestLoc(numberOfCloseLoc);
            }
        }

        public void OpenLocList(string fileName)
        {

```

```

        DataSet locDS = new DataSet();
        try
        {
            this.Clear();
            locDS.ReadXml(fileName);
            DataRowCollection loc = locDS.Tables[0].Rows;
            foreach (DataRow loc in loc)
            {
                this.Add(new Loc(Convert.ToInt32(loc["X"],
CultureInfo.CurrentCulture), Convert.ToInt32(loc["Y"],
CultureInfo.CurrentCulture)));
            }
        }
        finally
        {
            locDS.Dispose();
        }
    }
}

```

// Файл locs.cs

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;

namespace Tsp
{
    public class Loc
    {
        public Loc(int x, int y)
        {
            Location = new Point(x, y);
        }
    }
}

```

```

private Point location;
public Point Location
{
    get
    {
        return location;
    }
    set
    {
        location = value;
    }
}
private List<double> distances = new List<double>();
public List<double> Distances
{
    get
    {
        return distances;
    }
    set
    {
        distances = value;
    }
}
private List<int> closeLoc = new List<int>();
public List<int> CloseLoc
{
    get
    {
        return closeLoc;
    }
}
public void FindClosestLoc( int numberOfCloseLoc )
{
    double shortestDistance;

```

```

        int shortestLoc = 0;
        double[] dist = new double[Distances.Count];
        Distances.CopyTo(dist);
        if (numberOfCloseLoc > Distances.Count - 1)
        {
            numberOfCloseLoc = Distances.Count - 1;
        }
        closeLoc.Clear();
        for (int i = 0; i < numberOfCloseLoc; i++)
        {
            shortestDistance = Double.MaxValue;
            for (int locNum = 0; locNum < Distances.Count;
locNum++)
            {
                if (dist[locNum] < shortestDistance)
                {
                    shortestDistance = dist[locNum];
                    shortestLoc = locNum;
                }
            }
            closeLoc.Add(shortestLoc);
            dist[shortestLoc] = Double.MaxValue;
        }
    }
}

```

Файл Pos.cs

```

using System;
using System.Collections.Generic;
using System.Text;

```

```

namespace Tsp
{
    public class Pos : List<Link>
    {

```

```

public Pos(int capaloc)
    : base(capaloc)
{
    resetPos(capaloc);
}
private double fitness;
public double Fitness
{
    set
    {
        fitness = value;
    }
    get
    {
        return fitness;
    }
}
private void resetPos(int numberOfLoc)
{
    this.Clear();

    Link link;
    for (int i = 0; i < numberOfLoc; i++)
    {
        link = new Link();
        link.Connection1 = -1;
        link.Connection2 = -1;
        this.Add(link);
    }
}
public void DetermineFitness(Loc loc)
{
    Fitness = 0;
    int lastLoc = 0;
    int nextLoc = this[0].Connection1;
    foreach (Link link in this)

```

```

    {
        Fitness += loc[lastLoc].Distances[nextLoc];
        if (lastLoc != this[nextLoc].Connection1)
        {
            lastLoc = nextLoc;
            nextLoc = this[nextLoc].Connection1;
        }
        else
        {
            lastLoc = nextLoc;
            nextLoc = this[nextLoc].Connection2;
        }
    }
}

private static void joinLoc(Pos Pos, int[] locUsage, int
loc1, int loc2)
{
    if (Pos[loc1].Connection1 == -1)
    {
        Pos[loc1].Connection1 = loc2;
    }
    else
    {
        Pos[loc1].Connection2 = loc2;
    }

    if (Pos[loc2].Connection1 == -1)
    {
        Pos[loc2].Connection1 = loc1;
    }
    else
    {
        Pos[loc2].Connection2 = loc1;
    }

    locUsage[loc1]++;

```

```

        locUsage[loc2]++;
    }
    private static int findNextLoc(Pos parent, Pos child, Loc
locList, int[] locUsage, int loc)
    {
        if (testConnectionValid(child, locList, locUsage,
loc, parent[loc].Connection1))
        {
            return parent[loc].Connection1;
        }
        else if (testConnectionValid(child, locList,
locUsage, loc, parent[loc].Connection2))
        {
            return parent[loc].Connection2;
        }

        return -1;
    }

    private static bool testConnectionValid(Pos Pos, Loc
locList, int[] locUsage, int loc1, int loc2)
    {
        if ((loc1 == loc2) || (locUsage[loc1] == 2) ||
(locUsage[loc2] == 2))
        {
            return false;
        }
        if ((locUsage[loc1] == 0) || (locUsage[loc2] == 0))
        {
            return true;
        }
        for (int direction = 0; direction < 2; direction++)
        {
            int lastLoc = loc1;
            int currentLoc;
            if (direction == 0)
            {

```

```

        currentLoc = Pos[loc1].Connection1;
    }
    else
    {
        currentLoc = Pos[loc1].Connection2;
    }
    int PosLength = 0;
    while ((currentLoc != -1) && (currentLoc != loc2)
&& (PosLength < locList.Count - 2))
    {
        PosLength++;
        if (lastLoc != Pos[currentLoc].Connection1)
        {
            lastLoc = currentLoc;
            currentLoc = Pos[currentLoc].Connection1;
        }
        else
        {
            lastLoc = currentLoc;
            currentLoc = Pos[currentLoc].Connection2;
        }
    }
    if (PosLength >= locList.Count - 2)
    {
        return true;
    }
    if (currentLoc == loc2)
    {
        return false;
    }
}
return true;
}

public static Pos Crossover(Pos parent1, Pos parent2, Loc
locList, Random rand)
{

```

```

Pos child = new Pos(locList.Count);
int[] locUsage = new int[locList.Count];
int loc;
int nextLoc;
for (loc = 0; loc < locList.Count; loc++)
{
    locUsage[loc] = 0;
}
for (loc = 0; loc < locList.Count; loc++)
{
    if (locUsage[loc] < 2)
    {
        if (parent1[loc].Connection1 ==
parent2[loc].Connection1)
        {
            nextLoc = parent1[loc].Connection1;
            if (testConnectionValid(child, locList,
locUsage, loc, nextLoc))
            {
                joinLoc(child, locUsage, loc,
nextLoc);
            }
        }
        if (parent1[loc].Connection2 ==
parent2[loc].Connection2)
        {
            nextLoc = parent1[loc].Connection2;
            if (testConnectionValid(child, locList,
locUsage, loc, nextLoc))
            {
                joinLoc(child, locUsage, loc,
nextLoc);
            }
        }
        if (parent1[loc].Connection1 ==
parent2[loc].Connection2)

```

```

        {
            nextLoc = parent1[loc].Connection1;
            if (testConnectionValid(child, locList,
locUsage, loc, nextLoc))
            {
                joinLoc(child, locUsage, loc,
nextLoc);
            }
        }
        if (parent1[loc].Connection2 ==
parent2[loc].Connection1)
        {
            nextLoc = parent1[loc].Connection2;
            if (testConnectionValid(child, locList,
locUsage, loc, nextLoc))
            {
                joinLoc(child, locUsage, loc,
nextLoc);
            }
        }
    }

    for (loc = 0; loc < locList.Count; loc++)
    {
        if (locUsage[loc] < 2)
        {
            if (loc % 2 == 1)
            {
                nextLoc = findNextLoc(parent1, child,
locList, locUsage, loc);
                if (nextLoc == -1)
                {
                    nextLoc = findNextLoc(parent2, child,
locList, locUsage, loc); ;
                }
            }
        }
    }

```

```

    }
    else
    {
        nextLoc = findNextLoc(parent2, child,
locList, locUsage, loc);
        if (nextLoc == -1)
        {
            nextLoc = findNextLoc(parent1, child,
locList, locUsage, loc);
        }
    }
    if (nextLoc != -1)
    {
        joinLoc(child, locUsage, loc, nextLoc);
        if (locUsage[loc] == 1)
        {
            if (loc % 2 != 1)
            {
                nextLoc = findNextLoc(parent1,
child, locList, locUsage, loc);
                if (nextLoc == -1)
                {
                    nextLoc =
findNextLoc(parent2, child, locList, locUsage, loc);
                }
            }
            else
            {
                nextLoc = findNextLoc(parent2,
child, locList, locUsage, loc);
                if (nextLoc == -1)
                {
                    nextLoc =
findNextLoc(parent1, child, locList, locUsage, loc);
                }
            }
        }
    }
}

```

```

        if (nextLoc != -1)
        {
            joinLoc(child, locUsage, loc,
nextLoc);

        }
    }
}

for (loc = 0; loc < locList.Count; loc++)
{
    while (locUsage[loc] < 2)
    {
        do
        {
            nextLoc = rand.Next(locList.Count);
        } while (!testConnectionValid(child, locList,
locUsage, loc, nextLoc));
        joinLoc(child, locUsage, loc, nextLoc);
    }
}
return child;
}

public void Mutate(Random rand)
{
    int locNumber = rand.Next(this.Count);
    Link link = this[locNumber];
    int tmpLocNumber;
    if (this[link.Connection1].Connection1 == locNumber)
    {
        if (this[link.Connection2].Connection1 ==
locNumber)
        {
            tmpLocNumber = link.Connection2;

```

```

        this[link.Connection2].Connection1
=link.Connection1;
        this[link.Connection1].Connection1 =
tmpLocNumber;
    }
    else
    {
        tmpLocNumber = link.Connection2;
        this[link.Connection2].Connection2 =
link.Connection1;
        this[link.Connection1].Connection1 =
tmpLocNumber;
    }
}
else
{
    if (this[link.Connection2].Connection1 ==
locNumber)
    {
        tmpLocNumber = link.Connection2;
        this[link.Connection2].Connection1 =
link.Connection1;
        this[link.Connection1].Connection2 =
tmpLocNumber;
    }
    else
    {
        tmpLocNumber = link.Connection2;
        this[link.Connection2].Connection2 =
link.Connection1;
        this[link.Connection1].Connection2 =
tmpLocNumber;
    }
}
int replaceLocNumber = -1;
do

```

```

    {
        replaceLocNumber = rand.Next(this.Count);
    }
    while (replaceLocNumber == locNumber);
    Link replaceLink = this[replaceLocNumber];
    tmpLocNumber = replaceLink.Connection2;
    link.Connection2 = replaceLink.Connection2;
    link.Connection1 = replaceLocNumber;
    replaceLink.Connection2 = locNumber;

    if (this[tmpLocNumber].Connection1 ==
replaceLocNumber)
    {
        this[tmpLocNumber].Connection1 = locNumber;
    }
    else
    {
        this[tmpLocNumber].Connection2 = locNumber;
    }
}
}
}

```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА
РЕАЛІЗАЦІЯ ГЕНЕТИЧНОГО АЛГОРИТМУ
РОЗМІЩЕННЯ ДЖЕРЕЛ ЖИВЛЕННЯ В
РОЗПОДІЛЕНИХ МЕРЕЖАХ**

Виконав
ст. гр. КНТ-220м

О.Д. Поповський

Керівник
к.т.н., доцент

Т.А. Зайко

1

Рисунок Д.1 – Слайд 1

Об'єктом дослідження є процес пошуку оптимального розташування елементів.

Предмет дослідження – генетичні методи розв'язання завдань оптимального розташування елементів.

Мета роботи – дослідження та реалізація генетичного алгоритму пошуку оптимального розміщення джерел живлення в розподілених мережах.

2

Рисунок Д.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

- дослідити проблемну область побудови раціональної конфігурації розподільних електричних мереж;
- проаналізувати сучасні методи пошуку оптимального розташування елементів;
- розробити удосконалений генетичний метод для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- розробити структуру програмного забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- розробити програмне забезпечення для пошуку оптимального розміщення джерел живлення в розподілених мережах;
- протестувати розроблену програму для розв'язання задачі пошуку оптимального розміщення джерел живлення в розподілених мережах.

3

Рисунок Д.3 – Слайд 3

МЕТОДИ ОПТИМІЗАЦІЇ

Критерій порівняння	Метод		
	Лінійне програмування	Нелінійне програмування	Генетичні методи
Вимоги щодо лінійності цільової функції	+	–	–
Наявність вимог щодо неперервності цільової функції	+	+	–
Вимоги щодо диференційованості цільової функції	+	+	–
Тип пошуку	Детермінований	Детермінований	Стохастичний
Висока часова складність	–	+–	+

4

Рисунок Д.4 – Слайд 4

ГЕНЕТИЧНІ МЕТОДИ



[Субботін С. О. Ітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейронних мереж: монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник; під заг. ред. С.О. Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375 с.]

5

Рисунок Д.5 – Слайд 5

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

Критерій порівняння	C#	Java	C++
Зручність використання для розміщення джерел живлення в розподілених мережах	+	+-	+-
Можливість розробки кросплатформного програмного забезпечення	+-	+	+
Наявність засобів проектування інтерфейсу користувача	+	+-	-
Підтримка засобів створення ПЗ на основі процедурного програмування	+	-	+
Засоби створення ПЗ з використанням об'єктно-орієнтованої парадигми	+	+	+
Наявність засобів роботи з базами даних	+	+	+-

6

Рисунок Д.6 – Слайд 6

ПОРІВНЯЛЬНИЙ АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння	Microsoft Visual Studio	Eclipse	NetBeans
Зручність використання для створення ПЗ для пошуку оптимального розміщення джерел живлення в розподілених мережах	+	+	–
Наявність засобів автоматизації створення інтерфейсу користувача	+	+	+
Механізми для роботи з репозиторіями документів	+	+	–
Швидкість модифікації та модернізації створюваних програм	+	–	+
Платформено-незалежність	+	+	+

7

Рисунок Д.7 – Слайд 7

ГЕНЕТИЧНИЙ МЕТОД ПОШУКУ ОПТИМАЛЬНОГО РОЗМІЩЕННЯ ДЖЕРЕЛ ЖИВЛЕННЯ

Хромосома А, перший з батьків

Значення	1	0	1	0	0	1	0	0	1	1
Можливі місця розташування ДЖ	1	2	3	4	5	6	7	8	9	10

Хромосома Б, другий з батьків

Значення	1	0	0	1	0	1	1	1	0	0
Можливі місця розташування ДЖ	1	2	3	4	5	6	7	8	9	10



Перетворення хромосом-батьків

Хромосома А			
Значення	1	1	1
Можливі місця розташування ДЖ	3	9	10
Хромосома Б			
Значення	1	1	1
Можливі місця розташування ДЖ	4	7	9



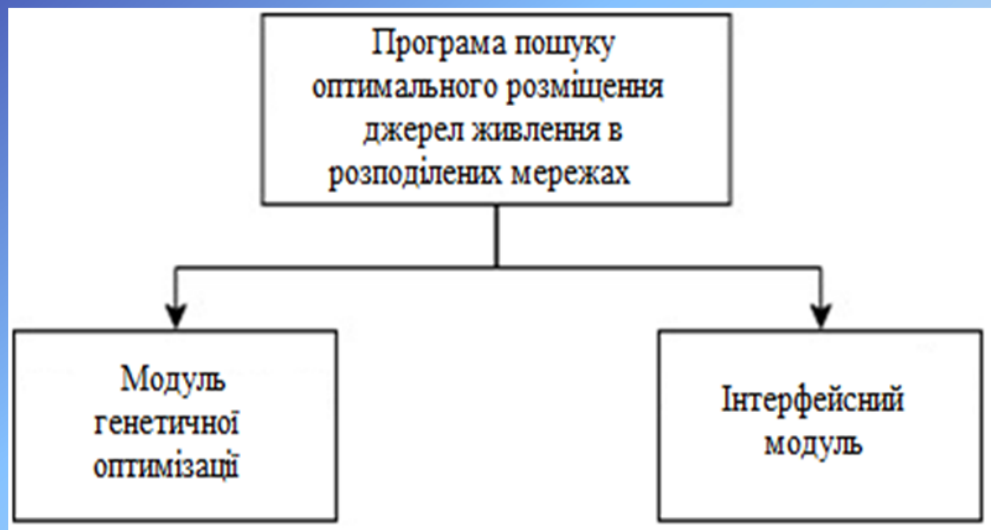
Генерація хромосом-нащадків

Хромосома В										
Значення	1	0	1	0	0	1	1	1	0	0
Можливі місця розташування ДЖ	1	2	3	4	5	6	7	8	9	10
Хромосома Г										
Значення	1	0	0	1	0	1	0	0	1	1
Можливі місця розташування ДЖ	1	2	3	4	5	6	7	8	9	10

8

Рисунок Д.8 – Слайд 8

СТРУКТУРНА СХЕМА ПРОГРАМИ



9

Рисунок Д.9 – Слайд 9

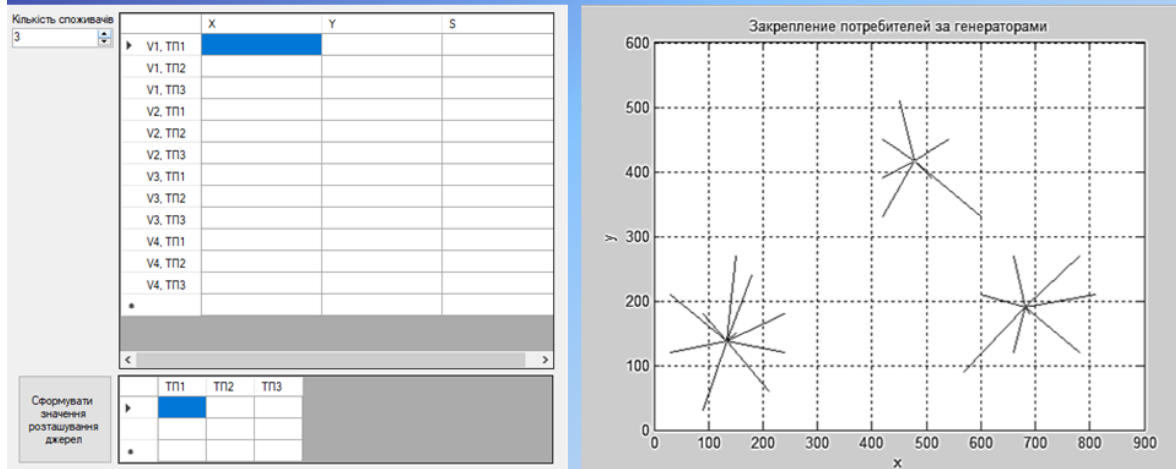
ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Головне вікно

The screenshot shows the main window of the "Генетический алгоритм" (Genetic algorithm) program. The window has a title bar with the text "Генетический алгоритм". On the right side, there are several input fields for configuration parameters: "Размер популяции" (Population size), "Вероятность мутации" (Mutation probability), "Размер группы" (Group size), "Число поколений" (Number of generations), "N соседей" (N neighbors), and "сосед %" (Neighbor %). Below these fields is a "Старт" (Start) button. The main area of the window is a large empty rectangle, likely for visualizing the algorithm's progress.

10

Рисунок Д.10 – Слайд 10

ПРИКЛАД РОБОТИ З ПРОГРАМОЮ



Вікно для заповнення координат локацій та потужності споживачів

Графічне подання закріплення споживачів за генераторами

11

Рисунок Д.11 – Слайд 11

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було досліджено процес та методи пошуку оптимального розташування елементів.

Досліджено та проаналізовано різні методи дискретної оптимізації. Прийнято рішення як базис використовувати генетичні методи для розв'язання завдання побудови раціональної конфігурації розподільної електричної мережі.

Наукова новизна роботи полягає у тому, що запропоновано генетичний метод пошуку оптимального розміщення джерел живлення в розподілених мережах, що використовує модифікований оператор схрещування, який дозволяє враховувати специфіку розв'язуваного завдання та передбачає формування нових хромосом-рішень таким чином, щоб вони задовольняли умовам задачі. Це дозволяє в процесі створення нової популяції уникати генерації хромосом-рішень, що є недопустимими (такими, що не відповідають умовам задачі), що, у свою чергу, зменшує час пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного пошуку.

12

Рисунок Д.12 – Слайд 12

ВИСНОВКИ

Досліджено та обрано мову програмування C# для розробки програми пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного методу. Як середовище розробки для створення ПЗ обрано Microsoft Visual Studio. Вибір цих засобів обґрунтовується зручністю розробки за їх допомогою.

З метою створення програмного забезпечення пошуку оптимального розміщення джерел живлення в розподілених мережах за допомогою генетичного алгоритму, виконано розробку структури відповідного програмного забезпечення. Відповідно до розробленої структури програма має ієрархічну структуру та складається з двох функціональних модулів: модуля генетичної оптимізації та інтерфейсного модуля.

Результати тестування показали, що розроблений програмний продукт може використовуватися за призначенням.

13

Рисунок Д.13 – Слайд 13