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

_____Інститут електроенергетики_____
(інститут)
_____факультет інформаційних технологій_____
(факультет)
Кафедра_____інформаційних технологій та комп'ютерної інженерії _____
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра___
(бакалавра, спеціаліста, магістра)

Студента _____Бережной Андрій Анатолійович_____
(ПІБ)
академічної групи _____126м-20-1_____
(шифр)
спеціальності _____126 «Інформаційні системи та технології»_____
(код і назва спеціальності)
за освітньо-професійною програмою _____

_____«Інформаційні системи та технології»_____
(офіційна назва)
на тему _____Підвищення ефективності складання розкладу руху поїздів_____
_____Укрзалізниці методами штучного інтелекту_____
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Галушко О.М.			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2022

ЗАТВЕРДЖЕНО:
завідувач кафедри
___інформаційних технологій___
_____та комп'ютерної інженерії___
(повна назва)
_____ _Гнатушенко В.В._
(підпис) (прізвище, ініціали)
«_____»_____ 2022_року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____магістр _____
(бакалавра, спеціаліста, магістра)

студенту _____Бережному А.А. **академічної групи** ____126м-20-1___
(прізвище та ініціали) (шифр)

спеціальності _____126 «Інформаційні системи та технології»_____

за освітньою-професійною програмою _____

_____«Інформаційні системи та технології»_____

на тему _____Підвищення ефективності складання розкладу руху поїздів_____
_____Укрзалізниці методами штучного інтелекту_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 10.12.2021 № 1036-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз стану області рішення задачі	01.09.2021 - 26.09.2021
Розділ 2	Моделі та методи розв'язання задачі	27.09.2021- 31.10.2021
Розділ 3	Розробка методики застосування генетичного алгоритму до розв'язання задачі складення розкладу руху транспорту	01.11.2021- 02.01.2022

Завдання видано _____ _____Коротенко Г.М._____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____1.10.2020 р._____

Дата подання до екзаменаційної комісії ____17.01.2022 р.____

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

РЕФЕРАТ

Пояснювальна записка: 66 стор., 18 рис., 4 додатки, 27 джерел.

Об'єкт дослідження: розклад руху залізничного транспорту.

Предмет дослідження: можливість застосування генетичних алгоритмів для складання розкладу залізничного транспорту.

Мета кваліфікаційної роботи: вивчення застосування методів машинного навчання для складення розкладу руху залізничного транспорту.

Кваліфікаційна робота присвячена актуальній задачі вивчення застосування методів штучного інтелекту для розв’язання NP – повних задач.

Наукова новизна виконаної роботи полягає в розробці методики застосування алгоритмів штучного інтелекту до процесу складання розкладу руху залізничного транспорту з метою підвищення його якості.

Пояснювальна записка містить опис математичних методів, що використовуються для складання розкладів, обґрунтування та опис обраного алгоритму та результати випробувань на тестових даних.

Список ключових слів: АЛГОРИТМИ, МАШИНЕ НАВЧАННЯ, СКЛАДЕННЯ РОЗКЛАДУ, ГЕНЕТИЧНІ АЛГОРИТМИ, МАТЕМАТИЧНА МОДЕЛЬ, СЕЛЕКЦІЯ, ПРИРОДНИЙ ДОБІР, МУТАЦІЯ.

ABSTRACT

Explanatory note: 66 pages, 18 figures, 4 applications, 27 sources.

Object of research: Railway traffic schedule.

Subject of research: Possibility of application of genetic algorithms, drawing up of the schedule of railway transport.

Study of the application of machine learning methods for scheduling railway traffic.

The qualification work is devoted to the current problem of studying the application of artificial intelligence methods to solve NP-complete problems.

The scientific novelty of the performed work is the development of methods for applying artificial intelligence algorithms to the process of scheduling railway transport in order to improve its quality.

The explanatory note contains a description of the mathematical methods used to compile the schedules, substantiation and description of the chosen algorithm and the results of tests on test data.

Keyword list: ALGORITHMS, MACHINE LEARNING, SCHEDULE, GENETIC ALGORITHMS, MATHEMATICAL MODEL, SELECTION, NATURAL SELECTION, MUTATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MIP – Mixed Integer Problem – Змішана ціла задача.

NP – Non-Deterministic Polynomial – Недетермінований поліном.

PESP – Periodic Event Scheduling Problem – Проблема планування періодичних подій.

ПЗП – проблема з’єднання потягів.

ПРОП – проблема регулярного огляду потягів.

РРТ – розклад руху транспорту.

ВСТУП

Актуальність роботи. У сучасному світі транспортні комунікації набувають все більш вагомого значення. Це пов’язано з особливостями процесів економічної глобалізації. Ця глобалізація вимагає якісного та своєчасного транспортування сировини, витратних матеріалів та виготовленої продукції цілим світом, а також великої кількості пасажиропотоків.

Світова транспортна мережа постійно збільшується, при чому найбільш популярні різновиди транспорту складають один одному достатньо гостру конкуренцію. Останнім часом основний грузопотік та відповідні пасажирські потоки, де вартість перевезень відносно невелика, припадає на залізничний та морський види транспорту. Таким чином, якщо порівняти час, що витрачається на перевезення, то залізничний транспорт виглядає найбільш швидшим та привабливішим. Але так може бути тільки у випадку, якщо залізниця працює за розкладом, що сприяє найшвидшому та найдешевшому перевезенню пасажирів та вантажів. Тож задача складення оптимального розкладу для залізничного транспорту набуває особливо вагомого значення.

Задача складення оптимального розкладу є добре відомою математичною проблемою що належить до класу невизначено поліноміальних задач (NP-повні задачі), а складність розв’язання відповідних проблем у цьому класі задач складає O(e^n). З цього опису випливає, що для якісного та відносно швидкого корегування розкладу треба мати відповідно якісний програмний код, заснований на обґрунтованому алгоритмічному розв’язанні перебірної задачі. Завдяки появі методів машинного навчання у математиків та програмістів з’явився новий інструментарій для розв’язання задач такого роду. Серед можливих рішень одне з чільних місць, завдяки гнучкості та простоті, займають генетичні алгоритми.

У роботі створено програмний застосунок мовою Python в середовищі PyCharm, що реалізує генетичний алгоритм розв'язання задачі.

Об’єктом досліджень є розклад руху залізничного транспорту.

Предметом досліджень є можливість застосування генетичних алгоритмів для складання розкладу залізничного транспорту.

Мета роботи - вивчення застосування методів машинного навчання для складення розкладу руху залізничного транспорту.

Кваліфікаційна робота присвячена актуальній задачі вивчення застосування методів штучного інтелекту для розв’язання NP – повних задач.

Методи дослідження. При виконанні роботи були застосовані такі механізми генетичних алгоритмів, як: схрещування, турнірний відбір та мутації.

Постановка задачі. В даній роботі поставлено наступні завдання:

– дослідити математичні методи та засоби, що призначені для розв’язання задач складання розкладів руху залізничного транспорту у відповідних мережах;

– обґрунтувати вибір способу розв’язання повних перебірних задач з метою пошуку локального мінімуму за умови бюджетування часу проведення розв’язання;

– обрати метод розв’язання та визначити найкращі параметри його застосування з урахуванням наявних даних.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

1.1. Загальний огляд предметної області

Планування розкладу залізничного транспорту є одним з етапів в класичному ієрархічному процесі планування перміщення поїздів (потягів) залізничного транспорту залізничними мережами і, тому, це робить дане питання широко досліджуваною темою в літературі, де нині багато зусиль зосереджено на вдосконаленні методів розв’язання відповідних задач оптимізації [1-3]. Таким чином, проблема складення розкладу руху поїздів є важливою сферою дослідження операцій через складність самої проблеми, яка виникає, у тому числі, з різними іншими видами транспорту.

Розклад руху будь яких одиниць будь якого транспорту може бути однією з тем, які мають значний вплив на якість сприйняття послуг користувачами відповідних сфер обслуговування. Побудова розкладу руху транспортних одиниць відграє важливу роль у розробці річного плану обслуговування залізничних перевезень в Україні відповідними структурними одиницями Укрзалізниці. Розв`язання задачі складання розкладу руху потягів передбачає знаходження такого розкладу, який має відповідати як комерційним потребам, так і певним обмеженням, пов’язаним з пропускною здатністю та безпекою власне залізничної мережі відповідної підпорядкованості.

Поєднання разом всіх перерахованих вище завдань робить створення РРТ складним і трудомістким процесом. Хоча більшість підходів у літературі пропонують точні моделі розв’язування, як тільки вони застосовуються на практиці, одразу стає зрозуміло що вони не можуть вирішити проблему протягом розумного проміжку часу. Причиною цьому є широке використання

евристичних методів або методів релаксації. Алгоритми планування можуть забезпечити надійний і безконфліктний розклад, враховуючи такі фактори, як новий попит, затори, наявну інфраструктура та затримки через погодні умови, технічні несправності, технічне обслуговування тощо.

Існує безліч оптимізаційних алгоритмів для реальних завдань вибору найкращого рішення. Однак задача оптимізації в області управління залізничним транспортом характеризується складністю змінних фазового простору та обмежень у ньому, що визначаються особливістю колійної інфраструктури та технологічним процесом перевезення. Добре відомі пошукові алгоритми не завжди є оптимальними стосовно складних систем управління з великою кількістю обмежень, тому виникає необхідність побудови нових, більш досконалих алгоритмів оптимізації, що враховують технологічні особливості системи, що моделюється.

1.2. Опис проблеми

У даній роботі розглядається розклад руху поїздів на двосторонній міжміській залізниці з кількома станціями. При цьому мається на увазі, що місцеперебування та кількість потягів, доступних на кожній станції, відомі. Щодня з призначених станцій організовуються поїздки певних потягів. Для кожної поїздки також відомі час відправлення поїзда та прибуття його до станції, а також місце розташування станцій, які визначаються розкладом потягів. На рис. 1.1 наведено простий розклад потягів з трьома станціями і десятьма рейсами. Планування потягів має на меті призначити ряд поїздок за розкладом для певної групи потягів з метою мінімізації загальних витрат на експлуатацію потягів і задоволення ряду обмежень, включаючи угоди профспілок, урядові постанови та політику компанії-перевізника.

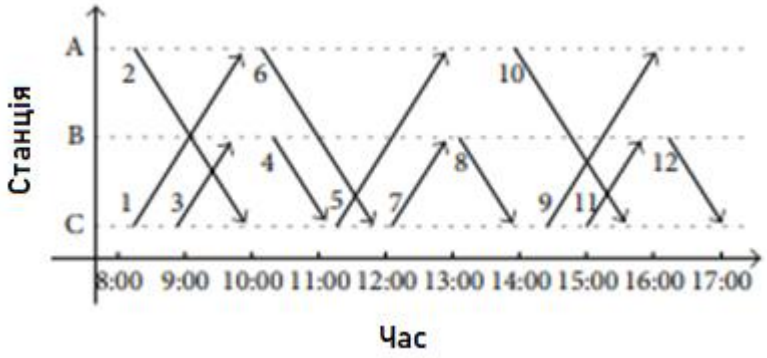


Рис 1.1. Простий розклад потягів

Для наочності необхідно врахувати наступні припущення.

(1) Будь-які дві послідовні поїздки, організовані для потяга, мають бути сплановані таким чином, щоб кінцева станція однієї збігалася з початковою станцією наступної (наприклад, поїздки 1 і 6).

(2) Будь-які дві послідовні поїздки, організовані для потяга, повинні бути сумісні за часом. Тобто потяг, який тільки прибув на станцію, не може одразу же відправитися з неї. Якщо припустити, що найкоротший час зупинки поїзду становить 15 хвилин, то можна сказати, що маршрути 4 і 5 не можуть бути сумісні.

(3) Задача розкладу руху потягів розв’язується з огляду на проміжок часу довжиною в одну добу. Для двох послідовних поїздок час відправлення однієї поїздки не повинен бути раніше часу прибуття наступної поїздки (наприклад, поїздки 1 і 2).

(4) Кількість необхідних потягів не може перевищувати встановленої максимальної кількості. На (рис 1.2) показано розклад руху потягів, що відповідає даним розкладу руху потягів на (рис 1.1). Три поїзди розраховані на 12 рейсів. Кожен рядок відповідає шляху кожного окремого потяга. Наприклад, завдання на поїздку першого потяга (№ 1) розташовані в ланцюжку з C-A, A-C, C-B, B-C і C-A. Колонки відповідають поїздкам у розкладі конкретних потягів. Треба зауважити, що кількість стовпчиків, які

відповідають різним потягам, відрізняється один від одного. Початкова точка поїздки конкретного потяга не обов’язково повинна збігатися з кінцевою.

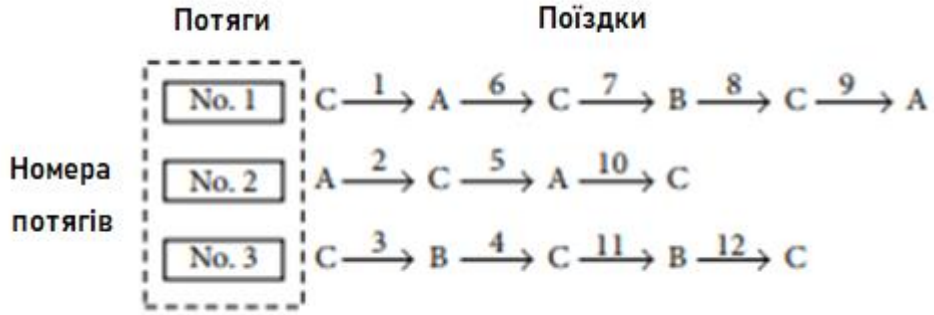


Рис 1.2. Розклад руху окремих потягів

Для розв’язання задачі з такими обмеженнями існує декілька способів, які описані нижче.

1.3. Існуючі методи для розв’язання задачі

1.3.1. Евристичний алгоритм роботи потяга на основі максимального задоволення пасажирів

Автори роботи [4] описали алгоритм оптимізації розкладу, заснований на задоволеності бажань пасажирів.

У статті розглянуто проблему оптимізації роботи потягів на станції з'єднання пасажирських ліній. Функція задоволення бажань пасажирів будується за допомогою аналізу характеристик задоволеності та кореляційних факторів. Обговорюючи поведінку вибору подорожі пасажирів, формулюється модель оптимізації на основі максимальної задоволеності пасажирів на головних станціях, а потім розробляється евристичний алгоритм, та наводиться чисельний приклад для демонстрації застосування методу, запропонованого в цій статті.

Результат показує, що запропонований в цій статті метод може ефективно вирішити проблему, і підходить для формулювання розкладу руху пасажирських потягів на одній окремій лінії.

1.3.2. Алгоритм, заснований на локальних критеріях оптимальності в разі потенційного конфлікту перетину

Проблема розкладу руху потягів — це проблема цілочисельного програмування, яка має NP-складність. На практиці таку проблему часто потрібно розв’язувати в режимі реального часу, тому найбільш бажаною є швидка евристика, яка дозволяє отримати гарне здійсненне рішення за заздалегідь визначену і кінцеву кількість кроків.

Автори роботи [5] запропонували алгоритм, який базується на локальних критеріях оптимальності у випадку потенційного конфлікту перетину розкладів руху різного призначення для вирішення конкретної проблеми розкладу руху потягів.

1.3.3. Алгоритм на основі ймовірнісного локального пошуку

Даний алгоритм приведений у роботі [6].

Локальний пошук — пошук, що здійснюється алгоритмами локального пошуку, групою алгоритмів, у яких пошук ведеться тільки на підставі поточного стану, а раніше пройдені стани не враховуються й не запам'ятовуються [6].

Основною метою пошуку є не знаходження оптимального шляху до цільової точки, а оптимізація деякої цільової функції, тому задачі, розв'язувані подібними алгоритмами, називають задачами оптимізації. Для опису простору станів у таких задачах використовують ландшафт простору станів, у цьому

представленні задача зводиться до пошуку стану глобального максимуму (або мінімуму) на даному ландшафті.

Робота над розкладом руху потягів поділяється на дві підпроблеми: проблема регулярного огляду потягів (ПРОП) і проблема з'єднання потягів (ПЗП). Таким чином проблема розкладу руху перетворюється на задачу комівояжера у мережі, яка називається мережею руху потягів, де вузли відповідають потягам, а дуги відповідають з'єднанням потягів, і на кожну дугу накладається вага, що виражає бажаність з'єднання. У представленому алгоритмі спочатку складається регулярний план перевірки, а потім виявляється Гамільтонів шлях. Якщо Гамільтонів шлях задовольняє обмеження, що стосуються щоденного огляду, він може являти собою можливий розклад руху потягів. Тому при пошуку нового Гамільтонового шляху на основі методу локального пошуку алгоритм враховує не тільки з'єднання вузлів, а й правила перевірки. На основі конструкції розроблено алгоритм апроксимації, і за допомогою експериментів з використанням фактичних даних доведено, що можна швидко отримати практичні розклади руху потягів.

1.3.4. Гібридний генетичний алгоритм

Автори роботи [7] запропонували метод фазово-регулярного планування та застосували використання регулярного інтервалу відправлення потяга та однакову довжину потяга залежно від попиту та періоду.

Було запропоновано регулярний поетапний метод планування, який розподіляє день на кілька рівних частин (часових блоків) і застосовує регулярний інтервал відправлення потяга та однакову довжину потяга для кожного періоду в умовах попиту, що залежить від періоду. У цьому дослідженні розроблено нелінійну змішану модель програмування нуль-один, яка могла б точно розрахувати час очікування пасажирів і вартість

переповненості в потязі. Гібридний генетичний алгоритм, пов’язаний з пошаровим кросинговером та операцією мутації, був розроблений для вирішення запропонованої моделі.

Привабливість генетичного алгоритму пояснюється його простотою та елегантністю як надійного алгоритму пошуку, а також його здатністю швидко знаходити хороші рішення для складних проблем великої розмірності. На практиці для вирішення запропонованої моделі призначена гібридна процедура з двошаровим каркасом.

Результати показують, що запропонований спосіб дозволяє ефективно вирішувати проблему розкладу міжміських залізничних ліній.

1.3.5. Алгоритм локального поліпшення та генетичний алгоритм

Для оптимізації розкладу пасажирських потягів у сильно завантаженому міському залізничному коридорі авторами [8] була побудована двійкова цілочисельна модель програмування, що включає події завантаження та відправлення пасажирів.

Використовуючи кумулятивні діаграми введення-виведення, автори представляють локальний алгоритм покращення для пошуку оптимальних розкладів для окремих випадків станції. Було розроблено генетичний алгоритм для вирішення проблеми з багатьма станціями за допомогою спеціального методу двійкового кодування, який вказує на відправлення або скасування потяга в будь-який можливий момент часу.

1.3.6. Табу-пошук або пошук із заборонами

Для розв’язання задачі розкладу руху поїздів на залізниці може бути використаний метод табу-пошуку. Саме такий метод використали автори відповідної роботи [9].

Пошук із заборонами або табу-пошук є реалізацією мета-алгоритму пошуку, що використовує методи локального пошуку, які використовуються для математичної оптимізації. Алгоритм створив Фред У. Гловер у 1986 і формалізував його у 1989 р.

Локальний пошук (по сусідах) для кожного потенційного рішення перевіряє сусіднє рішення з метою знаходження кращого рішення. Методи локального пошуку мають тенденцію застрягти в підоптимальних областях або плато, де багато рішень надто схожі.

Пошук із заборонами покращує продуктивність локального пошуку шляхом ослаблення його основного правила. Для початку, на кожному кроці може бути прийнято погіршення, якщо немає ніякого поліпшення. Крім того, заборони (вони ж табу) вводяться для того, щоб перешкодити пошуку за вже відвіданими рішеннями.

Реалізація пошуку з заборонами використовує структури, які описують відвідані рішення або власні набори правил. Якщо потенційне рішення було відвідано під час деякого короткого терміну або воно порушує правило, його позначають як «табу», так що алгоритм не буде розглядати рішення повторно.

Пошук із заборонами є мета-алгоритмом, який може бути використаний для вирішення завдань комбінаторної оптимізації.

Основними перевагами цього алгоритму є його простота, швидкість та гнучкість, а модель розкладу руху залізничних потягів у цій роботі є складною проблемою програмування з нульовим значенням. Таким чином, алгоритм пошуку табу можна легко використовувати.

Двовимірний метод кодування цілих чисел може бути використаний для вирішення проблеми розкладу потягів. У цьому методі рядки представляють потяги, а стовпці — поїздки. Поїздки нумеруються відповідно до часу відправлення в порядку зростання. Наприклад, у даних про рух потяга, представлених на рис.1.3, ланцюги поїздок кожного потяга можна виразити так: потяг 1: 1-6-7-8-9, потяг 2: 2-5-10 і потяг 3: 3-4-11-12.

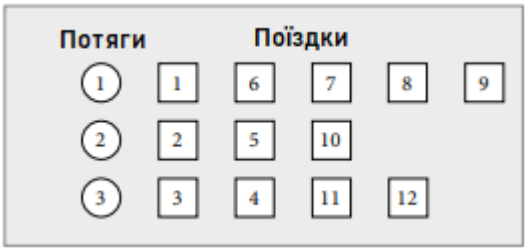


Рис 1.3. Ланцюги поїздок потягів

Початкове рішення є відправною точкою алгоритмічного пошуку. Краще початкове рішення дозволяє алгоритму швидко знайти оптимальне рішення. У процесі генерації вихідного рішення має бути виконано обмеження зсуву в часі.

Процедура роботи алгоритму така.

Крок 1. Набір поїздок, які здійснюються потягом k встановлюють $P_k = \emptyset$, для всіх k. Нехай i=1, k=1, та $a_0 = -T_0$.

Крок 2. Визначається номер поїзда $k`$ і номер послідовної поїздки $i`$ що відповідає поїздці i. Якщо $P_k = \emptyset$, то нехай $k` = k$ виконується крок 4.

В іншому випадку $i`$ = min{$a_l$ | $l \in$ {1,2,..., k}}, $k`$ ={l | $i`$=a_l, l $\in$ {1,2,..., k}}, де a_l=max{s| s $\in P_l$}, і перейти до кроку 3.

Крок 3. Обмеження зсуву в часі перевірено. Якщо $d_i - a_{i`} \geq T_0$ та = $s_i = s_{i`}$, то перейти до кроку 4. В іншому випадку, k←k+1, і перейти до кроку 2.

Крок 4. Нехай $P_{k`} = P_{k`} \cup \{i\}, i \longleftarrow i + 1$; перейти до кроку 5.

Крок 5. Якщо i > n, то алгоритм завершується, а результати отримуються. В іншому випадку перейти до кроку 2.

Структура сусідства використовує стратегії обміну поїздками та вставки між різними поїздами. Стратегію обміну поїздками можна описати так: вибирається єдина точка обміну в обох ланцюгах подорожей двох батьків. Номер подорожі цієї точки міняється між двома батьківськими організмами. Утворені організми - це діти. Наприклад, поїздка 7 потяга 1 (1-6-7-8-9) замінюється поїздкою 5 потяга 2 (2-5-10), і можна отримати нові рішення.

Стратегію вставки поїздок можна описати наступним чином: дві послідовні поїздки з потяга 1 вставляють у ланцюг поїздок потягу 2. Точка вставки залежить від часу відправлення в ланцюзі поїздок потягу 2 у порядку зростання, і два нових ланцюги поїздок за результат потягів. Наприклад, якщо маршрут 5 потяга 1 (1-5-7-8-10) вставити в ланцюг поїздки потяга 2 (2-6-9), то можна отримати нові рішення. Ланцюг відключення потяга 1 стає 1-8-10, а ланцюг відключення потяга 2 стає 2-5-7-6-9, як показано на рис. 1.4.

Рис. 1.4 Стратегія вставки поїздок

Операцію обміну або вставки поїздок неможливо виконати, якщо у відповідних батьківських генах поїздки не відповідають накладеним часовим обмеженням.

Результати розрахунків показали, що алгоритм може створювати якісні рішення для розкладу потягів. Крім того, цей метод може бути широко застосований для руху залізничних потягів, що характеризуються високою щільністю поїздок і великою кількістю потягів.

1.3.7. Генетичний алгоритм

Генетичний алгоритм (англ. genetic algorithm) - це евристичний алгоритм пошуку, який використовується для вирішення завдань оптимізації і моделювання шляхом випадкового підбору, комбінування і варіації шуканих параметрів з використанням механізмів, аналогічних природному відбору в природі [11]. Є різновидом еволюційних обчислень, за допомогою яких вирішуються оптимізаційні завдання з використанням методів природної еволюції, таких як успадкування, мутації, відбір і кросинговер [10]. Відмінною особливістю генетичного алгоритму є акцент на використанні оператора «схрещування», який проводить операцію рекомбінації рішень-кандидатів, роль якої аналогічна ролі схрещування в живій природі.

Процес природного відбору починається з вибору сильних особин з популяції. Їхнє потомство успадковує характеристики батьків і є частиною наступного покоління особин. Якщо обидва батьки сильні, то їхнє потомство буде сильнішим за батьків.

Це ітеративний процес і він завершується, коли знайдені найбільш сильні особини. Ця ідея застосовується для завдання пошуку. Розглядається набір рішень для проблеми і відбирається набір кращих з них.

Процес навчання генетичного алгоритму ділиться на 5 етапів.

1. Початкова популяція.

2. Функція сили особини.

3. Відбір найсильніших рішень.

4. Обмін характеристиками між двома особинами.

5. Мутація.

6. Нова ітерація зі створенням початкової популяції.

Процес починається з набору особин, який називається популяцією. Кожна особина - це рішення проблеми, яка була поставлена. Особливість характеризується набором параметрів, які називають генами. Гени об'єднані в один рядок і формують хромосому - вирішення завдання.

У генетичному алгоритмі набір генів особини представлений у вигляді бінарного рядка. Закодована комбінація генів називається хромосомою.

1.3.7.1. Функція сили

Функція сили визначає, наскільки сильна окрема особина. Сила визначається як здатність особини конкурувати з іншими особинами за заданою метрикою. Функція присвоює кожній особині рівень сили. Ймовірність того, що особина буде обрана для творення наступної популяції, ґрунтується на рівні сили особини.

1.3.7.2. Відбір

Ідея відбору полягає в тому, щоб відібрати найбільш сильних особин і передати їх гени наступному поколінню особин. N-пар батьків відбираються на підставі їх сили. Відбір не завжди ґрунтується на парах, іноді у кожному турі відбору беруть участь три або більше особин.

1.3.7.3. Схрещення

Схрещення - це основна частина генетичного алгоритму. Для кожної пари батьків випадково вибирається точка в бінарному рядку хромосоми, до якої особини обмінюються генами. Після цього модифіковані особини називаються потомством.

1.3.7.4. Точка схрещування

Потомство створюється через процес обміну генами батьків до випадково заданої позиції в рядку або між випадково заданими позиціями у рядку (рис. 1.5). Нижче можна побачити, як батьки обмінюються генами, коли точка схрещування = 3.

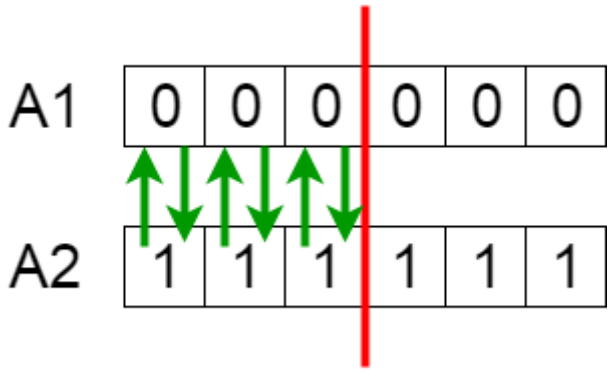


Рис. 1.5. Обмін генами при точці схрещування – 3.

Після обміну генами між батьками потомство додається в нову популяцію.

1.3.7.5. Мутація

Якщо не використовувати механізми, що урізноманітнюють популяцію за деякий час (певну кількість поколінь), усі особини набувають генів найсильніших батьків. Через це популяція вироджується, тому, щоб запобігти

цьому для підтримання різноманіття популяції використовується процес мутації генів особини (рис. 1.6).

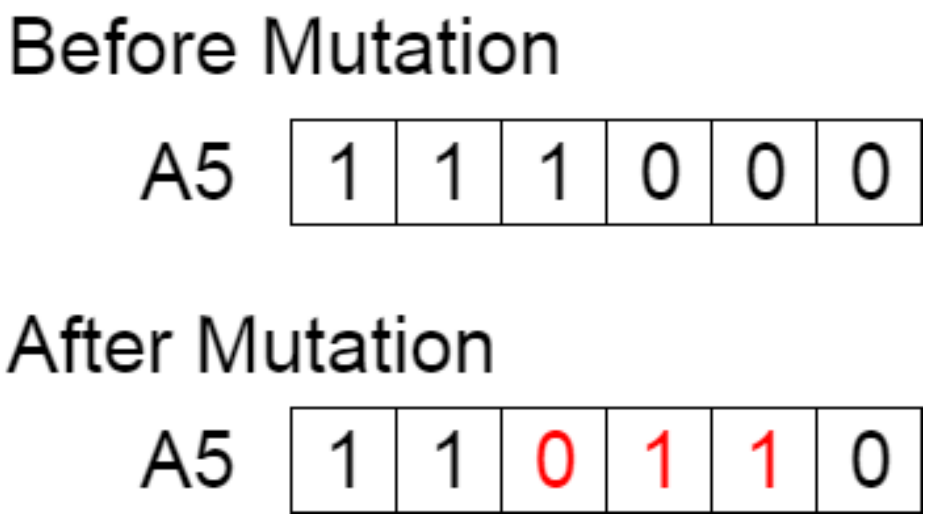


Рис. 1.6. Вигляд генів до та після мутації

1.3.7.6. Завершення алгоритму

Алгоритм завершує роботу, коли популяція зійшлася, тобто не виробляє потомство, яке значно відрізняється від попереднього покоління. Коли алгоритм зійшовся, на виході утворюється набір оптимальних рішень заданої проблеми.

Аналіз наведений в цьому розділі наочно демонструє що найчастіше задля розв’язання перебірних задач варто використовувати генетичні алгоритми як найзручніші, за умови не найгіршого часу розв’язання задачі, та задовільної якості розв’язання.

РОЗДІЛ 2
МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1. Вибір і обґрунтування рішення поставленої задачі

Генетичні алгоритми були успішно застосовані до комбінаторних задач і здатні обробляти величезні простори пошуку, як і ті, що виникають у реальних проблемах розробки розкладів руху потягів. Генетичні алгоритми здійснюють різноспрямований стохастичний пошук по всьому простору пошуку, який постійно посилюється в найбільш перспективних областях.

Проблема розкладу руху потягів є складним і трудомістким завданням у випадку реальних мереж. Величезний простір пошуку, який потрібно досліджувати під час вирішення реальних випадків проблеми розкладу потягів, робить генетичні алгоритми придатним для використання алгоритмом для ефективного вирішення цієї проблеми. Можливий розклад руху потягів повинен визначати для кожного потяга час відправлення та прибуття до кожної частини мережі таким чином, щоб брати до уваги пропускну здатність лінії та інші експлуатаційні обмеження. Традиційно розклади створювалися вручну та коригувалися відповідно до всіх обмежень, що зустрічалися. Однак нові рамки жорсткої конкуренції, приватизація та дерегуляція разом із збільшенням швидкодії комп'ютерів є причинами, які виправдовують потребу в автоматичних інструментах, здатних ефективно генерувати здійсненні та оптимізовані розклади.

Враховуючи залізничну лінію, яка може мати як одноколійні, так і двоколійні ділянки, задача розкладу руху потягів полягає в обчисленні розкладів руху пасажирських і вантажних потягів, які задовольняють наявним обмеженням та оптимізують багатокритеріальну цільову функцію. Залізнична лінія може бути зайнята іншими потягами, пріоритет яких вище, ніж у нових, а нові потяги, які будуть додані, можуть належати іншим операторам потягів.

Місця, які відвідає кожен потяг, також можуть відрізнятися один від одного. Розклад, наданий кожному новому потягу, має бути здійсненним, тобто він повинен задовольняти заданому набору обмежень. Серед обмежень, що виникають у цій задачі, можлива вимога періодичності часових можливостей. Періодичність призводить до класифікації проблеми розкладу потягів як (*i*) періодичний (або циклічний) розклад руху потягів і (*ii*) неперіодичний розклад руху потягів.

У періодичному розкладі кожна поїздка виконується періодично. Тобто кожен період розкладу однаковий. Перевагою періодичної залізничної системи є те, що розклад такої системи легко запам’ятовується пасажирами. Недоліком є те, що така система дорога в експлуатації з точки зору використання таких ресурсів, як рухомий склад і екіпажі. У літературі найбільш широко використовується математична модель Серафіні та Уковича [12] під назвою проблема планування періодичних подій. У проблемі планування періодичних подій набір повторюваних подій планується за періодичних обмежень часового вікна. Отже, події плануються на один цикл таким чином, щоб цикл можна було повторити. Модель проблеми планування періодичних подій була використана Начтігалом і Вогетом в [13], Одіжком в [16], Кроном і Пітерсом в [13], Лібхеном в [14].

Неперіодичний розклад руху потягів особливо актуальний у коридорах з інтенсивним рухом на далекі відстані, де пропускна здатність інфраструктури обмежена через велику щільність руху. Це дозволяє менеджеру інфраструктури оптимально розподіляти маршрути потягів, які запитують оператори потягів, і продовжити процес складання загального розкладу, можливо, з остаточними локальними уточненнями та незначними змінами, внесеними планувальником. Багато статей розглядають формулювання змішаної цілочисельної задачі (Mixed Integer Problem, MIP), в якій час прибуття та відправлення представлено безперервними змінними, а є двійкові змінні, що виражають порядок відправлення потягів з кожної станції.

Неперіодична проблема розкладу потягів розглядалася кількома авторами: Шпігелем [17], Явановичем і Харкером [18], Кай і Гох [5], Кері та Локвудом [19], Хіггінсом та ін. [20], Сільва де Олівейра [21], Кван і Містрі [22], Капрара та ін. [23], Інголотті та ін. [24].

2.2. Нотація проблеми, використана у даній роботі

Нотація, що використовуються для розв’язання задачі, представлена нижче:

Параметри:

- T: скінчений набір потягів t, що розглядаються у проблемі. $T = \{t_1, t_2, \dots, t_k\}$.
- $T_C \subset T$: підмножина потягів, які знаходяться в обігу та розклад яких не може бути змінений (T_C може бути порожнім) .
- $T_{new} \subseteq T$: підмножина нерегулярних потягів, які ще не мають розкладу руху і це необхідно додати до залізничної лінії з можливим розкладом. Таким чином $T = T_C \cup T_{new}$ та $T_C \cup T_{new} = \emptyset$.
- l_i: розташування (станція, зупинка, розв'язка). Розглянуті типи місць описуються таким чином:
  – Станція: місце для паркування, зупинки або проїзду потягів. Кожна станція пов'язана з унікальним ідентифікатором станції. На станції є дві або більше колії, де можна здійснювати переїзди або обгін.
  – Зупинка: місце для зупинки, проходження потягів, але не паркування. Кожна зупинка пов'язана з унікальним ідентифікатором зупинки.
  – Розв'язка: місце, де дві різні колії розгалужуються. Немає часу зупинки.
- N_i: кількість треків у місці l_i.

- NP_i: кількість треків з платформою (необхідно для комерційних зупинок) у місці l_i.
- $L = \{ l_0, l_1, \dots, l_m\}$: залізнична лінія, яка складається з упорядкованої послідовності місць, які можуть відвідувати потяги $t \in T$. Суміжні локації l_i та $l_i + 1$ зв'язані одинарною або подвійною колійною ділянкою.
- $J_t = \{ l_0^t, l_1^t, \dots, l_{n^t}^t, \}$: подорож потягом t. Описується впорядкованою послідовністю місць, які має відвідати потяг t, такий що $\forall t \in T, \exists J_i: J_t \subseteq L$. Подорож J_t показує порядок, який використовується потягом t для відвідування заданого набору місць. Таким чином, l_i^t та $l_{n^t}^t$ представляють i-е та останнє місце, яке відвідав потяг t відповідно.
- T_D: набір потягів, що рухаються в напрямку вниз. $t \in T_D \leftrightarrow (\forall l_i^t : 0 \leq i < n_t, \exists l_i \in \{L \{l_m\}\} : l_i^t = l_i \wedge l_{i+1}^t = l_{j+1})$.
- T_U: набір потягів, що рухаються в напрямку вгору. $t \in T_U \leftrightarrow (\forall l_i^t : 0 \leq i < n_t, \exists l_i \in \{L \{l_0\}\} : l_i^t = l_i \wedge l_{i+1}^t = l_{j-1})$.

Таким чином, $T = T_D \cup T_U$ та $T_D \cup T_U = \emptyset$.

- C_i^t: мінімальний час, необхідний потягу *t* для виконання комерційних операцій (наприклад, посадки або виходу пасажирів) на станції i (комерційна зупинка).
- $\Delta_{i\to(i+1)}^t$: час у дорозі потяга t з місця l_i^t до l_{i+1}^t.
- $\lfloor I_L^t, I_U^t \rfloor$: інтервал часу відправлення потяга $t \in T_{new}$ з початкової станції його шляху.
- $\lfloor F_L^t, F_U^t \rfloor$: інтервал часу прибуття потяга $t \in T_{new}$ до кінцевої станції його шляху.

Змінні:

- dep_i^t: час відправлення потяга $t \in T$ з місця i, де $i \in J_t\{l_{n^t}^t\}$.
- arr_i^t: час прибуття потяга $t \in T$ з місця i, де $i \in J_t\{l_0^t\}$.

Планувальники зазвичай використовують поточні карти як графічні інструменти, задля полегшення процесу планування. Карта пробігу — це часово-просторова діаграма, подібна до показаної на рис. 2.1, де можна спостерігати кілька переїздів потягів. Назви станцій представлені з лівого боку, а вертикальна лінія відображає кількість колій між станціями (односторонні або двосторонні). Горизонтальні пунктирні лінії позначають зупинки або перехрестя, а суцільні лінії – станції. У мережі залізниць планувальнику необхідно запланувати маршрути n_k потягів, що йдуть в одному напрямку, i_{mk} потягів, що прямують у протилежному напрямку для потягів даного типу. Потяги для розкладу можуть вимагати (або ні) певну частоту.

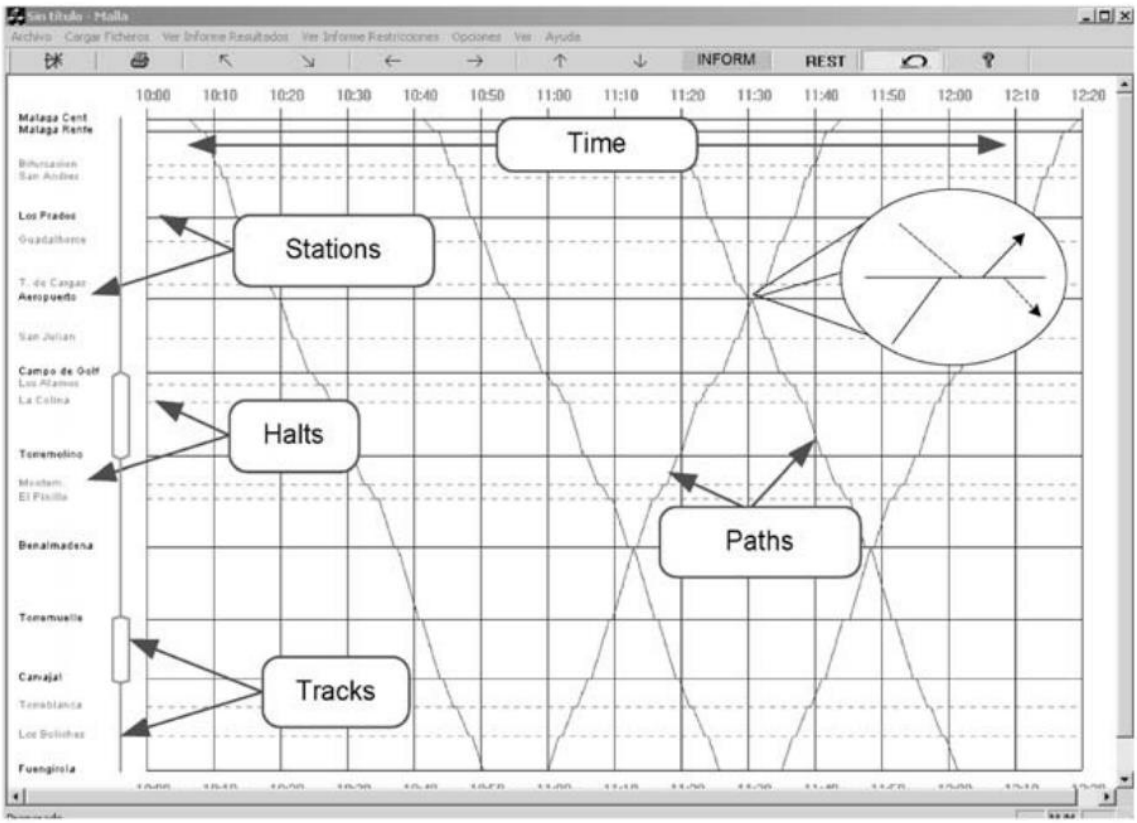


Рис. 2.1. Карта пробігу

2.3. Набір обмежень

Розклад має відповідати набору обмежень, які можна розділити на три основні групи:

- (i) вимоги користувача (параметри потягів, які плануються),
- (ii) правила дорожнього руху,
- (iii) топологія залізничної інфраструктури.

Обмеження, в цій роботі описані таким чином, щоб отриманий графік був здійсненним і практичним.

Обмеження користувача:

- Інтервал для початкової станцію: кожен потяг $t \in T_{new}$ повинен залишити свою початкову станцію l_0^t в момент часу dep_0^t такий, що,

$$I_L^t \leq dep_0^t \leq I_U^t \tag{2.1}$$

- Інтервал для часу прибуття: кожен потяг $t \in T_{new}$ повинен прибути у свою кінцеву станцію $l_{n^t}^t$ в момент часу $arr_{n_t}^t$ такий, що,

$$F_L^t \leq arr_{n_t}^t \leq F_U^t \tag{2.2}$$

- Максимальна затримка: для кожного потяга $t \in T_{new}$ вказано максимальну затримку Λ_t та мінімальний час у дорозі M_t; таким чином, верхня межа для часу подорожі $t \in T_{new}$ визначається таким виразом:

$$\frac{arr_{n_t}^t - dep_0^t - \mathrm{M}_t}{\mathrm{M}_t} \leq \Lambda_t \tag{2.3}$$

Обмеження руху:

- Час у дорозі: для кожного потяга та кожної ділянки колії час у дорозі визначається як $\Delta^t_{i\to(i+1)}$, що представляє час, який потяг має використати для переходу з місця l^t_i до розташування l^t_{i+1}. Отже, має виконуватися наступний вираз

$$arr^t_{i+1} = dep^t_i + \Delta^t_{i\to(i+1)} \tag{2.4}$$

- Перетин: згідно з наступним виразом, одноколійну ділянку ($i \to i+1$, напрямок вниз) не можуть займати два потяги, що їдуть у протилежних напрямках ($t \in T_D$ і $t' \in T_U$) (рис. 2.2).

$$dep^{t\prime}_{i+1} > arr^t_{i+1} \vee dep^t_i > arr^{t\prime}_i \tag{2.5}$$

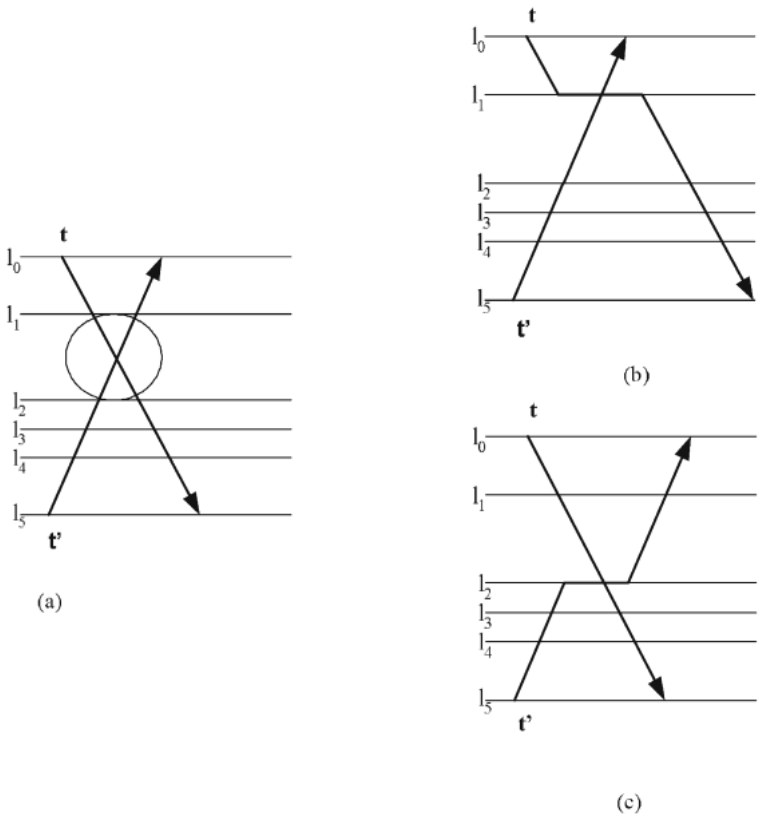


Рис. 2.2. (a) Конфлікт перетину. (b) Поїзд у напрямку вниз чекає. (c) Поїзд у напрямку вгору чекає

- Комерційна зупинка: кожен потяг $t \in T_{new}$повинен залишатися на станції l_i^t принаймні C_i^t одиниць часу:

$$dep_i^t \geq arr_i^t + C_i^t \tag{2.6}$$

- Обгін на ділянці колії: слід уникати обгону між будь-якими двома потягами, $\{t, t'\} \subseteq T$, що рухаються в одному напрямку на будь-яких двоколійних ділянках, $k \to k+1$, їх подорожей:

$$(arr_{k+1}^t > arr_{k+1}^{t'}) \leftrightarrow (dep_k^t > dep_k^{t'}) \tag{2.7}$$

- Затримка для несподіваної зупинки: коли потяг t зупиняється на станції j, щоб уникнути конфліктів з іншими потягами (обгін/переїзд), а комерційна зупинка не планувалась ($C_j^t = 0$) на цій станції, час у дорозі потяга t відповідає до попередньої ($l_{j-1}^t \longrightarrow l_j^t$) і наступної ($l_j^t \longrightarrow l_{j+1}^t$) ділянок колії j необхідно збільшити на Γ_t одиниць часу. Це збільшення означає зниження швидкості потяга через гальмування та прискорення на станції.

$$\begin{aligned} dep_j^t - arr_j^t > 0 \wedge C_j^t = 0 \longrightarrow \Delta_{j-1\to j} &= \Delta_{j-1\to j} + \Gamma_t \wedge \Delta_{j\to j+1} \\ &= \Delta_{j\to j+1} + \Gamma_t \end{aligned} \tag{2.8}$$

- Час прийому: різниця між часом прибуття будь-яких двох потягів $\{t, t'\} \subseteq T \wedge \{t, t'\} \subseteq T_{new}$ на тій самій станції l визначається виразом нижче, де R_t - час прийому, зазначений для потяга. що приходить до l першим (рис. 2.3).

$$arr_l^{t'} \geq arr_l^t \to arr_l^{t'} - arr_l^t \geq R_t \tag{2.9}$$

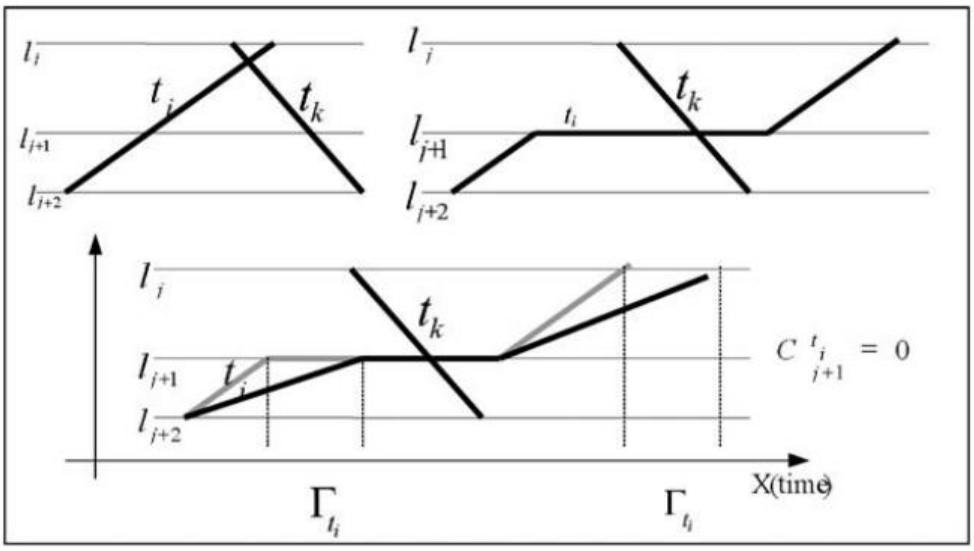


Рис. 2.3. Неочікувана зупинка

- Час експедиції: різниця між часом відправлення та прибуття будь-яких двох потягів $\{t, t'\} \subseteq T \wedge \{t, t'\} \subseteq T_{new}$ на тій самій станції l визначається виразом нижче, де E_t — час експедиції, визначений для t (рис. 2.4, 2.5).

$$|dep_l^{t'} - arr_l^t| \geq E_t \tag{2.10}$$

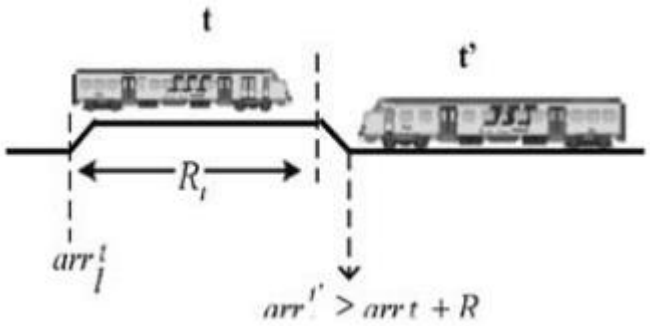


Рис. 2.4. Час прийому між поїздом t і поїздом t'

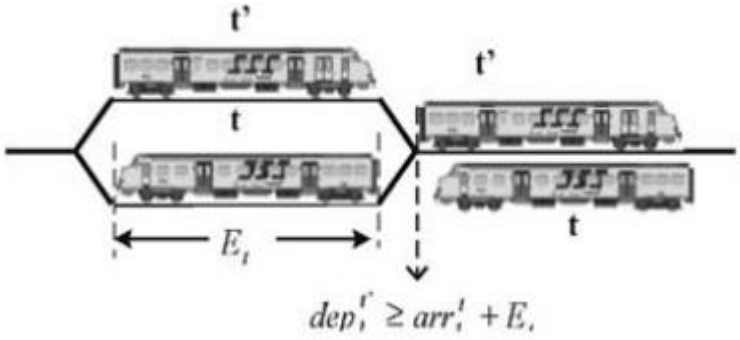


Рис. 2.5. Час експедиції між поїздом t і поїздом t'

- Одночасне відправлення: різниця між часом відправлення з однієї станції двох потягів, що прямують у протилежних напрямках, має бути не менше S, коли обидва потяги зупиняються на станції. Це обмеження формулюється так:

$$\forall t, t' \in T_{new} : dep_i^t > 0 \wedge dep_i^{t'} - arr_i^{t'} > 0 \rightarrow \left| dep_i^t - dep_i^{t'} \right| \geq S \quad (2.11)$$

Обмеження інфраструктури:

- Кінцева пропускна спроможність станцій: потяг $t \in T_{new}$ може прибути до місця l_i^t тоді й тільки тоді, коли він має принаймні одну доступну колію (з платформою, якщо $C_i^t > 0$). Щоб сформулювати це обмеження, розглянемо:

$$\forall x \in T_{new}: T_x = \{t \in T: t \neq x, J_t \cap J_x \neq \emptyset\}$$

та

$$Meet(x,t,l) = \begin{cases} 1 & if\ [arr_l^x, dep_l^x] \cap [arr_l^t, dep_l^t] \neq \emptyset \wedge C_i^t = 0 \\ 0 & else \end{cases}$$

$$Meet_P(x,t,l) = \begin{cases} 1 & if\ [arr_l^x, dep_l^x] \cap [arr_l^t, dep_l^t] \neq \emptyset \wedge C_i^t > 0 \\ 0 & else \end{cases}$$

Отже, обмеження кінцевої потужності станцій формулюється так:

$$\forall x \in T_{new}, \forall l \in J_x : \left(\left(\sum_{t \in T_x} Meet(x,t,l) + \sum_{t \in T_x} Meet_P(x,t,l) < N_l \right) \wedge \left(C_l^x > 0 \rightarrow \sum_{t \in T_x} Meet_P(x,t,l) < N_{P_l} \right) \right) \quad (2.12)$$

- Час закриття: Нехай $[H_l^1, H_l^2]$ - час закриття для операцій технічного обслуговування станції l. Час закриття накладає обмеження на регулярні операції - потяги можуть проходити, але не можуть зупинятися на станції (див. наступний вираз). І навіть можуть заборонити регулярні рейси, потяги не можуть ні проїхати, ні зупинитися (тобто: кількість колій на станції зменшується до однієї (див. (2.12)).

$$dep_l^t < H_l^1 \vee arr_l^t < H_l^2 \quad (2.13)$$

- Час руху: якщо два потяги, $\{t, t'\} \subseteq T$, що рухаються в одному напрямку, виїжджають з того самого місця l_k до місця l_{k+1}, вони повинні мати різницю в часі відправлення щонайменше φ_k^d і різницю в часі їх прибуття не менше φ_k^a. Якщо тип блокування в ділянці колії автоматичний, тоді $\varphi_k^d = \varphi_k^a$. Розглянемо наступний вираз

$$\left| dep_k^t - dep_k^{t'} \right| \geq \varphi_k^d \quad (2.14)$$

$$\left|arr_{k+1}^{t} - arr_{k+1}^{t'}\right| \geq \varphi_k^a \qquad (2.15)$$

Запропонований метод повинен отримати найкраще доступне рішення, щоб задовольнити всі вищезазначені обмеження. Як ми зазначали раніше, мережу могли зайняти інші потяги, розклад яких не змінювався. Тобто $\forall t \in T_C$, змінні arr_i^t і dep_i^t, були попередньо створені з заданими значеннями. Це означає $\forall t \in T_C, \forall i \in J_t, arr_i^t \in CONSTANT, dep_i^t \in CONSTANT$ і процес генерує обмеження так, що час прибуття та відправлення потягів, що знаходяться в обігу, є постійними, і він не створює обмежень, які включають лише змінні, що відповідають потягам, що знаходяться в обігу. Далі процес перевіряє, що кожен новий потяг задовольняє кожному обмеженню, беручи до уваги нові потяги, що залишилися, а також усі потяги, які вже в обігу. Іншими словами, якщо обмеження порушується, і воно пов'язує нові потяги з потягами в обігу, слід змінити лише розклади, які відповідають новим потягам.

2.4. Цільова функція

Щоб оцінити якість кожного рішення, ми отримуємо оптимальне рішення (оптимальний час проходження) для кожного конкретного потягу $t \in T_{new}$. Оптимальне рішення потяга t обчислюється шляхом планування нового потяга t (перевірка всіх проблемних обмежень) у мережі, яка зайнята лише потягами в обігу (T_C). Це оптимальне рішення для потяга $t(\Gamma_{opt}^t)$ є найнижчим часом, необхідним t для завершення його подорожі. Інші потяги, які плануються в T_{new}, ігноруються.

Після того, як буде обчислено оптимальний час для кожного нового потяга, який має бути запланований, критерієм для вимірювання якості кожного рішення буде середня затримка нових потягів щодо їхнього оптимуму (δ). Тобто:

$$\delta_t = \frac{\left(arr_{n_t}^t - dep_0^t\right) - \Gamma_{opt}^t}{\Gamma_{opt}^t};\ \delta_U = \frac{\sum_{t \in T_U \cap T_{new}} \delta_t}{|T_U \cap T_{new}|};\ \delta_D = \frac{\sum_{t \in T_D \cap T_{new}} \delta_t}{|T_D \cap T_{new}|};$$

$$\delta = \frac{\delta_U + \delta_D}{T_{new}}$$

Припускаючи, що TTABLE є одним рішенням проблеми і, розкладом для всіх нових потягів, цільова функція цієї проблеми формулюється так:

$$f(TTABLE) = MIN(\delta) \tag{2.16}$$

Якщо немає потягів в обігу в $L(T = T_{new})$, оптимальний час нового потяга *t* буде:

$$M_t = \Gamma_{opt}^t = \sum_{i=0}^{n_t - 1} \Delta_{i \to (i+1)}^t + \sum_{i=0}^{n_t - 1} C_i^t \tag{2.17}$$

2.5. Розв’язання задачі: генетичний алгоритм

Запропонований у цій роботі метод розв’язання для проблеми розкладу потягів базується на підході «робочий цех» (Job-Shop). Коротше кажучи, загальна проблема планування робочих місць, яка виникає в багатьох компаніях, передбачає виконання набору замовлень, що складається з робочих місць, що задовольняють певні обмеження часу та ресурсів. Відношення пріоритету завдань кожного порядку означають тимчасові та ресурсні обмеження, які виникають, коли один ресурс не може одночасно використовуватися двома роботами, тобто виробнича машина не може одночасно виконувати дві роботи. Основною метою є створення графіка (розкладу роботи), в якому кожна робота задовольняє обмеження часу та ресурсів із найменшими обчислювальними зусиллями та оптимізує показник продуктивності.

У випадку проблеми розкладу потягів кожен потяг можна розкласти на набір упорядкованих ділянок колії (T-ts), які повинні перевірити набір обмежень часу та ресурсів: відношення пріоритету ділянок колії для кожного потяга як тимчасові обмеження та обмеження ресурсів, коли одностороння ділянка колії не може бути одночасно зайнята двома потягами. Мета полягає в тому, щоб побудувати розклад (розклад руху потягів), у якому кожна секція залізничної колії задовольняє обмеження часу та ресурсів і оптимізує показник продуктивності з найменшими обчислювальними зусиллями.

У цьому контексті, якщо не враховувати обмеження ресурсів, то проблема зводиться до поширення початкового часу відправлення, dep_0^t, від l_0^t до $l_{n^t}^t$, використовуючи час у дорозі $\Delta_{i\to(i+1)}^t$, який відповідає кожному потяга t і кожній ділянці колії $l_i^t \to l_{i+1}^t$. Однак у цій задачі ресурси мають обмежену ємність. Одна лінія містить набір одноколійних ділянок, які не можуть бути зайняті більш ніж одним потягом у певний момент часу, так само, як машини в цеху можуть обробляти лише одне завдання в певний час. Проблема розкладу потягів — це оптимізаційна комбінаторна задача, простір пошуку якої зростає експоненціально, коли кількість конфліктів зростає. Це може бути пов’язано зі збільшенням кількості потягів, або зі зменшенням пропускної спроможності станцій (кількість колій з платформою за наявності комерційної зупинки) тощо. Відомо, що ця проблема NP-складна [2] Таким чином, евристичні методи є поширеними підходами, здатними отримувати «хороші» рішення з розумним часом обчислень. Ці евристики повинні мати можливість досліджувати великі простори пошуку, що виникають у цьому типі задач, і досягати гарного рішення за короткий час обчислень.

Тому підхід генетичного алгоритму підходить для боротьби з проблемою розкладу потягів. Припускаючи щойно встановлений паралелізм між обома проблемами, генетичний алгоритм, розроблений для вирішення проблеми розкладу потягів, припускає кожен потяг як порядок t, який розбивається на конкретні завдання t_{jk} і де кожне завдання означає, що потяг

t використовує одну колію для переходу з місця j до розташування k. Ми представляємо роботу парою (t, s_i^t), де t — потяг, а s_i^t — i-ий відрізок колії його шляху.

2.5.1. Базова схема генетичного алгоритму

На рис. 2.6 показана загальна схема базового генетичного алгоритму. По-перше, початкова сукупність (P на рис. 2.6), розмір якої дорівнює POP_SIZE, генерується та оцінюється за схемою планування, яка описана як на рис. 2.8. Наступні кроки повторюються, доки не буде досягнуто кінцевої умови end_cond (час виконання, кількість можливих рішень або кількість поколінь). Деякі особини, які складають популяцію P на рис. 2.6, модифікуються шляхом застосування процедур Selection(), Crossover() і Mutation(). Таким чином, в кожному поколінні виходить нова популяція P. Кожна ітерація відповідає новому поколінню індивідів (рис. 2.6).

```
Function Genetic_Algorithm(POP_SIZE, end_cond) As Timetabling

begin
    P=Generate_Initial_Population(POP_SIZE)
    while NOT (end_cond) do
    begin
        P=Selection(P)
        P=Crossover(P)
        P=Mutation(P)
        BEST_L=Evaluate_Population(P)
    end
    return BEST_L
end
```

Рис. 2.6. Загальний вигляд генетичного алгоритму

2.5.2. Визначення індивідів

Щоб застосувати генетичний алгоритм до певної проблеми, потрібне внутрішнє представлення простору розв'язків. Вибір цього компонента є одним із критичних аспектів успіху/невдачі генетичного алгоритму для досліджуваної проблеми. У літературі є різні типи представлень для розв'язання різних задач планування. У цій роботі було використано список діяльності як представлення рішення. Це подання рішення широко використовується в плануванні проекту. Рішення кодується як список можливих перед пріоритетом пар (t, s_i^t), тобто якщо (t, s_x^t), і (t, s_y^t), є j-им і k-им геном хромосоми однієї особи і $x < y$, тоді $j<k$.

Відповідний розклад потягів формується із застосуванням модифікованої версії схеми генерування послідовного розкладу, що використовується в проектному плануванні [25]. У списку пар усі потяги об'єднуються, щоб отримати можливе рішення. В рамках цієї кваліфікаційної роботи нові потяги плануються в порядку, встановленому переліком. Кожна особина в популяції представлена масивом з такою кількістю позицій, скільки пар (t, s_i^t) існує в розглянутій задачі розкладу залізниці. На рис. 2.7 показано представлення списку дій для задачі з N пар (t, s_i^t).

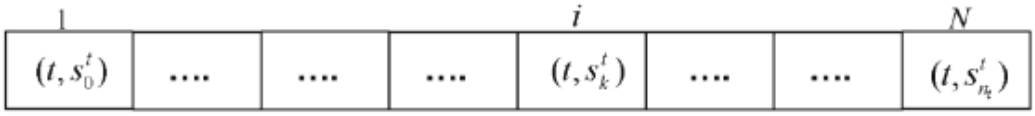


Рис. 2.7. Представлення списку діяльності

Відповідно до цього списку, (t, s_k^t) є i-ю парою, яка планується. Враховуючи, що $s_k^t = l_k^t \rightarrow l_{k+1}^t$, час відправлення потяга t з l_k^t буде найбільш можливим раннім часом від $arr_k^t + C_k^t$. Зауважте, що при застосуванні процесу декодування із заданої послідовності можна вивести один і тільки один

розклад, але різні послідовності можуть бути перетворені в той самий розклад (рис. 2.8).

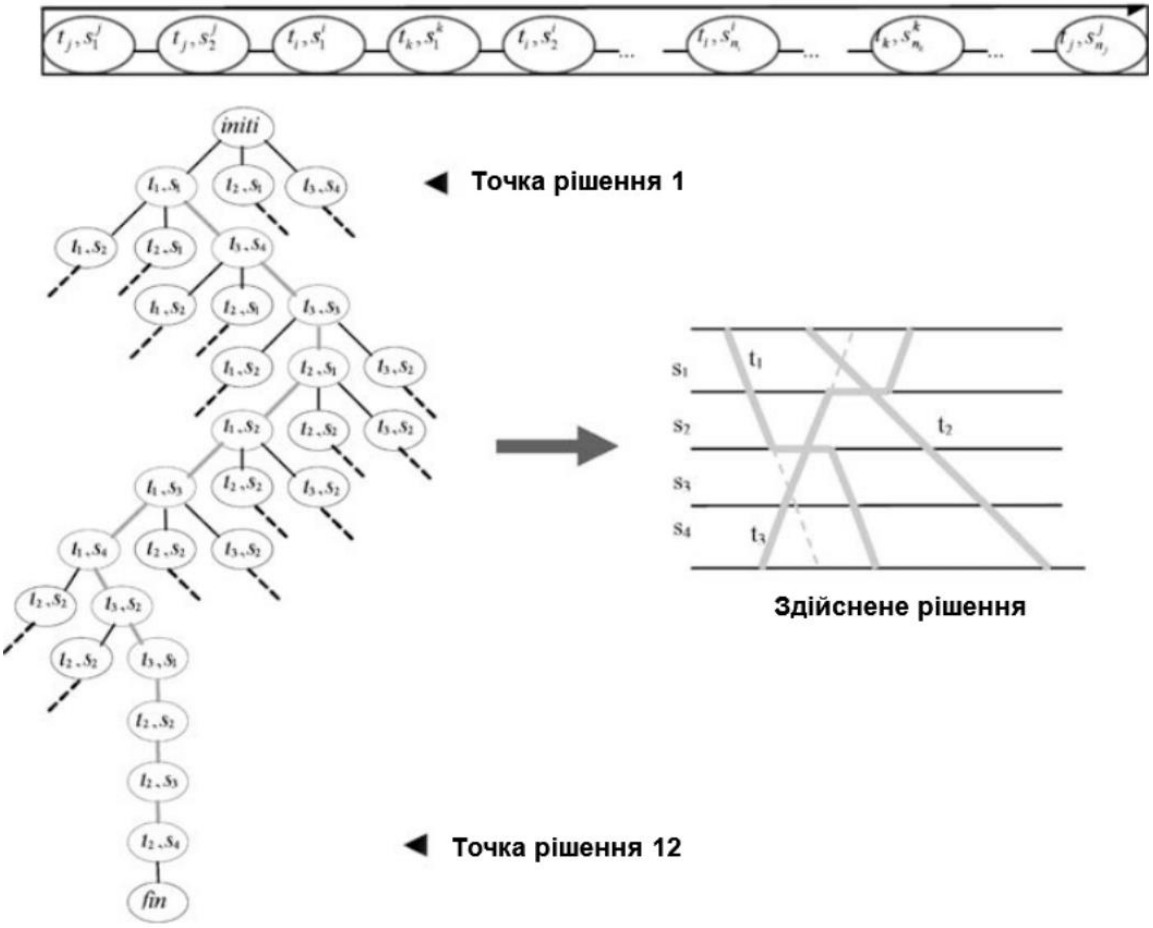


Рис. 2.8. Кожен список діяльності представляє один шлях дерева пошуку

Як показано на рис. 2.8, список діяльності представляє один шлях дерева пошуку. Різний порядок списку діяльності зазвичай означатиме інше рішення.

2.5.3. Розрахунок придатності

Застосовуючи генетичний алгоритм, нам потрібно визначити функцію оцінки, яка визначає ймовірність виживання особини до наступного покоління.

У цьому розділі середнє відхилення щодо оптимального рішення для потяга повертається як значення пристосованості. Іншими словами, індивід χ матиме значення придатності, отримане з цільової функції.

2.5.4. Початкова популяція

Генетичний алгоритм починається з створення початкової популяції, тобто набору можливих рішень POP_SIZE. Цей набір можливих рішень можна отримати за допомогою різних методів планування. Для розробки генетичного алгоритму початкова популяція повинна включати різноманітні середні та хороші можливі рішення, щоб підвищити якість найкращого рішення в еволюційному процесі. У цій роботі значення POP_SIZE дорівнює 50. Початкова сукупність була отримана за допомогою ітераційної евристики, заснованої на методах випадкової вибірки, яка повторюється POP_SIZE разів (рис. 2.9). N – загальна кількість ділянок колії для всіх потягів $t \in T_{NS}$. Показано, як створюється список діяльності шляхом вибору потяга (t) і набору ділянок колії $s \in J_t$ для кожної ітерації, поки всі (N) ділянок колії не будуть заплановані для кожного нового потяга. Рішення буде отримано після завершення N ітерацій. Кожне рішення дає порядок планування, $L = \left(t_x, s_0^{t_x}\right), \dots, \left(t_z, s_j^{t_z}\right), \dots, \left(t_y, s_{n_{t_y}}^{t_y}\right),$ який представляє нову особину початкової популяції.

Розроблений метод планування передбачає пошук шляху в дереві, як показано на рис. 2.8. У кожній точці прийняття рішення вибирається потяг із незапланованими ділянками колії. Цей процес отримує можливий розклад зі значенням функції придатності.

Основним рішенням у процедурі Select_Train() є: який потяг має бути запланований у кожній точці прийняття рішення. Незважаючи на те, що можна прийняти випадкове рішення (RANDOM) щодо вибору потяга для випадкового розкладу, ми застосували процедуру випадкової упередженої вибірки на основі відмови, яка робить вибір потяга залежним від його

відхилення щодо оптимального рішення (оптимальний час руху потяга). Цей підхід керує процесом планування, щоб отримати кращий результат.

Параметризована випадкова упереджена вибірка на основі відмови вибирає потяги T_{NS} за допомогою випадкового пристрою. Використання випадкового пристрою можна розглядати як відображення $\psi: i \in T_{NS} \rightarrow [0..1]$, де ймовірність $\psi(i)$ бути обраною (як $\sum_{i \in T_{NS}} \psi(i) = 1$) присвоюється кожному $t \in T_{NS}$. Значення відмови (ρ_i) для кожного потяга $i \in T_{NS}$ порівнює значення пріоритету потяга $i - \upsilon(i) -$ з найгіршим значенням пріоритету $\upsilon(i)$ потягів T_{NS} і розраховується наступним чином:

$$\rho_i: \max_{j \in T_{NS}} (\upsilon_j) - (\upsilon_i) \tag{2.18}$$

Отже, параметризоване відображення ймовірності $\psi(i)$ розраховується як:

$$\psi(i): \frac{(\rho_i + \varepsilon)^\alpha}{\sum_{j \in T_{NS}} (\rho_i + \varepsilon)^\alpha} \tag{2.19}$$

Ця параметризована випадкова упереджена вибірка на основі відмови, широко та успішно використовується в плануванні проектів (Шірмер і Різенберг [26], Тормос і Лова [27]). Пріоритетність кожного потяга розраховується відповідно до його поточної затримки щодо розкладу руху. Потяги з більшими затримками мають більше шансів бути обраними. У даній роботі реалізувано процедуру Generate_Initial_Population() (рис. 2.9) за допомогою методу випадкової упередженої вибірки на основі відмови для вибору наступного потяга, який буде заплановано, з $\alpha = 1$ і $\varepsilon = 0{,}5$.

```
Function Generate_Initial_Population(POP_SIZE) As Population
begin
    ref=Get_Low_Bound_Opt_Sol()
    i=0
    P=""
    While (i<POP_SIZE)
    begin
        L= "" //L is a new list of chromosomes
        j=0
        While (j<N)
        begin
            t=Select_Train() //using the RBRS method
            s=Select_Track_Section(t)
            d=Get_Departure(t,s)
            Set_Timetable((t,s),d)
            L=L+(t,s) //(t,s) is inserted in L
            j=j+1
        end
        Set_Fitness_To_Individual(L, ref)
        P=P+L
        i=i+1
    end
    return P
end
```

Рис. 2.9. Порядок отримання початкової популяції

2.5.5. Перехрещування (кросовер)

Одним з унікальних і важливих аспектів методів, що стосуються генетичних алгоритмів, є важлива роль, яку відіграє рекомбінація (традиційно у формі оператора кросовера). Кроссовер поєднує ознаки двох батьківських хромосом, щоб утворити двох нащадків, які успадковують їх характеристики. Особи популяції спаровуються випадковим чином, і кожна пара проходить операцію схрещування з імовірністю P_{cross}, що призводить до появи двох дочірніх індивідів. Батьківську популяцію замінює популяція нащадків. Кросовер є одним з найважливіших генетичних операторів і повинен бути

правильно розроблений. Кроссовер повинен поєднувати рішення для виробництва нових індивідів. Кросовер повинен зберігати та поєднувати «хороші будівельні блоки», щоб створити кращих індивідів [5]. Враховуючи двох індивідів, відібраних для схрещування, матір *M* і батька *F*, виходять двоє нащадків, дочка *D* і син *S*.

Ми реалізували добре відомий одноточковий кросовер з P_{cross} = 0,8. Спочатку ми малюємо випадкову точку перетину *k*, де *k* від *1* до *N* (кількість ділянок потяга в задачі). Перші *k* позицій у *D* безпосередньо беруться з *M*, у тому самому порядку. Решта діяльності в *D* взято з їх відносним порядком у послідовності батька. Таким чином, створене рішення, дочка, є першочерговим рішенням. Очевидно, покоління *S* подібне до дочірнього, але *S* успадковує перші позиції безпосередньо від *F*, а решту Train-Track Section від *M*. Псевдокод для цієї техніки кросовера показано на рис. 2.10.

```
Function Random_Crossover_Point()
begin
    //Draw a random integer k, with 1<= k<= N
    //k is the random crossover-point
    //Generation of the daughter
    for i=1 to k do
        Di = Mi
    for i=k+1 to N do
    begin
        I = lowest index  1<= I<= N and Fi not in {D1, ..., D(i-1)}
        Di = Fi
    end
    //Generation of the son
    ........
end
```

Рис. 2.10. Процедура кросоверу

2.5.6. Мутації

Після застосування оператора схрещування популяція нащадків замінює батьківську популяцію, та оператор мутації застосовується до популяції нащадків. Мутація змінює один або кілька генів (положень) вибраної хромосоми (розчину), щоб повторно ввести втрачений генетичний матеріал і ввести додаткову мінливість у популяції.

Оператор мутації, який ми реалізували, працює так: для кожної пари (t, s_i^t) у послідовності випадковим чином вибирається нова позиція. Для того, щоб генерувати лише можливі рішення з пріоритетом, ця нова позиція повинна бути вищою, ніж її попередник, і нижчою, ніж її наступник. Хромосома вставляється в нову позицію з ймовірністю P_{mut}. У нашій реалізації $P_{mut} = 0.05$.

2.5.7. Відбір

Відбір – це штучна версія природного явища, яке називається виживання найсильнішого. У природі конкуренція між особинами за дефіцитні ресурси та за пару призводить до того, що найпридатніші особини переважають над слабкішими.

На основі їх відносної якості або рангу особи отримують певну кількість копій.

Пристосована особина отримує більшу кількість потомства і, отже, має більшу ймовірність вижити в наступному поколінні. Існує кілька способів реалізації механізму відбору.

Ми впровадили турнірний відбір. Цей механізм відбору передбачає, що дві особини випадковим чином вибираються з популяції і змагаються за виживання. Найкращий з них (з найкращим значенням придатності) з’явиться в наступній популяції. Ця процедура повторюється POP_SIZE разів, поки особини POP_SIZE не будуть обрані для появи в наступній популяції.

2.5.8. Процес декодифікації

У цьому підрозділі буде детально розглянуто процедуру Evaluate_Population() на рис. 2.6. Ця процедура отримує популяцію *P*, з якої вона повинна отримати рішення POP_SIZE. Кожне рішення буде оцінено відповідно до цільової функції.

Для кожної пари $p = (t, s_i^t) \in L$ час відправлення обчислюється за допомогою функції Get_Departure(), яка повертає $d = arr_i^t + C_i^t$, якщо $i > 0$, інакше $d = m$ таке, що m — початковий час відправлення, заданий користувачем.

Враховуючи, що s_i^t починається на станції l_i^tі закінчується на станції l_{i+1}^t, процедура Set_Timetable() призначає можливий час відправлення та прибуття потяга *t* у кожному місці між l_i^t та l_{i+1}^t відповідно до часу в дорозі $\left(\Delta_{i\to(i+1)}^t\right)$ визначено для цього потягу від l_i^t до l_{i+1}^t. Наступний крок полягає у перевірці, чи задовольняються всі обмеження, визначені раньше, графіком, наведеним для t у s_i^t. Якщо будь-яке обмеження не задовольняється, час відправлення в l_i^t збільшується, доки це обмеження не буде задоволено. Це збільшення часу відправлення в l_i^t викликає технічну зупинку потяга *t* на цій станції. Відхід назад може статися, якщо станція закрита на технічні операції або якщо станція не має достатньої кількості колій.

Після того, як на цій ділянці колії знайдено можливий розклад для потяга *t*, ту ж процедуру повторюють з наступною хромосомою $(t', s_k^{t'})$

Пріоритет потягів у кожній ділянці колії, тобто, який потяг слід відкласти, якщо виникає конфлікт, визначається порядком, у якому кожен ген нумерується в списку активностей. Коли виникає конфлікт між двома потягами на одній ділянці колії, пріоритет має той потяг, розклад якого на цій ділянці колії було призначено першим. Оскільки різні особи визначають різні пріоритети між потягами, то можуть бути отримані різні рішення (рис. 2.11).

```
Procedure Evaluate_Population(P)
begin
    i=0
    While(i<|P|) do
    begin
        L=Get_Individual(i,P)
        k=0
        while(k<|L|)
        begin
            p=Get_Chromosome(L,k)
            d=Get_Departure(p)
            Set_Timetable(p,d)
            k=k+1
        end
        i=i+1
        Set_Fitness_Individual(L)
    end
end
```

Рис. 2.11. Процес декодифікації

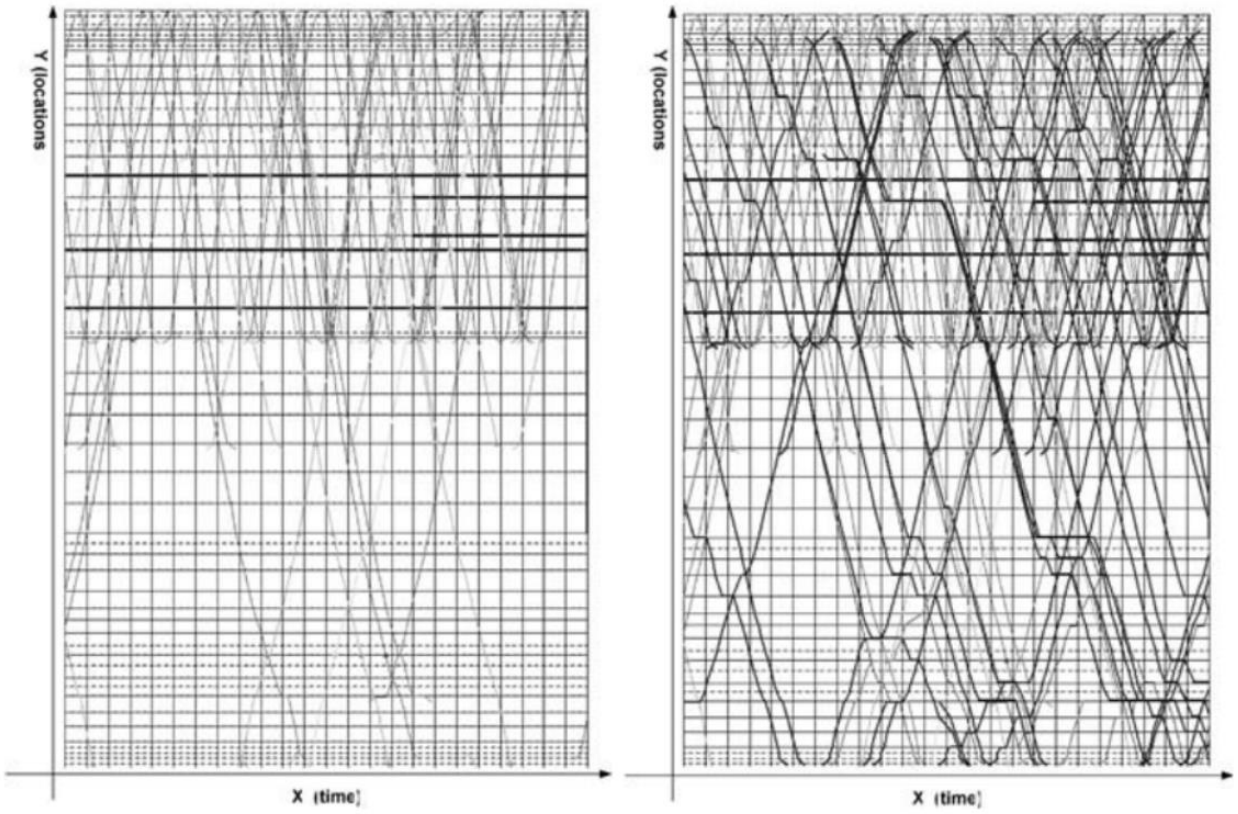
Налаштування параметрів запропонованого генетичного алгоритму є результатом попередніх обчислювальних експериментів.

2.6. Розв'язування реальних задач із застосуванням генетичного алгоритму

Застосування описаного генетичного алгоритму проілюстровано за допомогою наступної проблеми.

Існує лінія між двома містами, яка охоплює 54 місця (станції, зупинки, роз'їзні частини тощо); вона має довжину 369,4 кілометрів і має 30/23 дво-/односторонніх ділянок колій. Кожна горизонтальна лінія на рис. 2.12 показує положення кожного місця розташування.

Розклад, показаний на рис. 2.12a, відповідає руху на лінії між двома містами, і у часовому вікні [12:00, 24:00] в обігу перебуває 103 потяги до додавання нових потягів. Кожній точці похилих ліній відповідає положення одного потяга (вісь Y) у певний момент часу (вісь X).



(a) **Потяги в обігу** (b) **Рішення**

Рис. 2.12. Графічне представлення прикладу проблеми та її рішення

Потрібно додати 59 нових потягів до лінії у часовому вікні [12:00, 24:00], задовольняючи обмеження, враховуючи потяги в обігу та мінімізуючи середню затримку нових потягів. Зазвичай планувальник працює лише за допомогою графічного інструменту та розкладає потяги в лексикографічному порядку (один потяг за іншим). Отримане рішення є можливим рішенням, але не обов’язково хорошим рішенням через декілька комбінацій, які необхідно розглядати одночасно. На рис. 2.12b показано рішення, отримане за допомогою підходу генетичного алгоритму за 300 секунд. Значення цільової функції для цього рішення становить 7,2%.

РОЗДІЛ 3

РОЗРОБКА МЕТОДИКИ ЗАСТОСУВАННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДО РОЗВ'ЯЗАННЯ ЗАДАЧІ СКЛАДЕННЯ РОЗКЛАДУ РУХУ ТРАНСПОРТУ

3.1. Опис маршрутів

Продуктивність розробленого генетичного алгоритму була перевірена з використанням набору реальних проблем, наданих Укрзалізницею. Опис вхідних даних наведено в таблиці 3.1: довжина залізничної колії, кількість одно/двоколійних ділянок, кількість місць і станцій, кількість потягів і ділянок колії відповідно до всіх цих потягів, враховуючи потяги в обігу та нові потяги відповідно.

Таблиця 3.1

Опис маршрутів

Марш-рути	Опис інфраструктури					Потяги в обігу		Нові потяги	
	км	1 сторон-ні	2-сторон-ні	Місце-знахо-дження	Станції	Потяги	T-ts	Потяги	T-ts
1	96	16	0	13	13	47	1397	16	180
2	129	21	0	22	15	27	302	30	296
3	256	38	0	39	28	81	1169	16	159
4	401	37	1	39	24	0	0	35	499

Для перевірки ефективності застосування генетичного алгоритму, окрім розрахунків зроблених за його допомогою, також були застосовані такі алгоритми, як метод випадкового вибору кожного потягу який планується на кожній ітерації, а також процедура випадкової упередженої вибірки на основі відмови. Результати цих методів були порівняні між собою.

3.2. Результати роботи алгоритмів

У таблиці 3.2 узагальнено результати для кожного методу розв’язування щодо кількості створених рішень та середнього відхилення щодо оптимального рішення відповідно до виразу (2.16). Різна кількість рішень, що генеруються залежно від використаного методу, пояснюється тим, що при використанні підходів випадкового вибору кожного потягу, який планується на кожній ітерації, і процедури випадкової упередженої вибірки на основі відмови застосовується процедура скорочення. Тобто, коли частковий розклад видає значення цільової функції, гірше за найкраще значення, отримане на той момент, поточна ітерація переривається і починається побудова нової. Однак із підходом генетичного алгоритму скорочення неможливе, оскільки кожна ітерація повинна бути завершена, для отримання значення відповідності для особин.

Таблиця 3.2

Результати підходів випадкового вибору кожного потягу, який планується на кожній ітерації, і процедури випадкової упередженої вибірки на основі відмови та генетичного алгоритму

Маршрути	Випадковий вибір кожного потягу, який планується на кожній ітерації		Процедура випадкової упередженої вибірки на основі відмови		Генетичний алгоритм	
	# з рішень	Середнє відхилення щодо оптимального рішення	# з рішень	Середнє відхилення щодо оптимального рішення	# з рішень	Середнє відхилення щодо оптимального рішення
1	267	18	263	15.4	255	15.1
2	611	10.1	608	10.0	313	9.6
3	424	14.7	521	14.1	382	12.4
4	405	19.2	397	17.9	285	16.0

Результати таблиці 3.2 показують, що запропонований генетичний алгоритм перевершує методи випадкового вибору кожного потягу, який планується на кожній ітерації, і процедури випадкової упередженої вибірки на основі відмови у всіх випадках. Ці результати демонструють ефективність генетичного алгоритму для вирішення проблем розкладу залізниць у порівнянні з іншими алгоритмами.

3.3. Методика застосування генетичного алгоритму для складання розкладу залізничного транспорту

Під час розв’язання задачі складання РРТ треба виконати наступні операції, які фактично становлять методику застосування цього алгоритму. До їх складу входять наступні операції:

1. Описати поточний стан залізничної мережі з урахуванням ділянок, які тимчасово виведені з експлуатації або тих, які з якихось причин не можна використовувати. Результатом опису має бути орієнтований зважений граф з відстанями між відповідними станціями, представлений у вигляді матриці суміжності цих станцій. Цей етап необхідно повторювати кожен раз, коли виникає потреба змінити просторову конфігурацію мережі через технічні або адміністративні перешкоди, що заважають руху залізничного транспорту певними ділянками.

2. Визначити кількість потягів, поточну кількість маршрутів та обмеження, що накладаються на розклад, такі як: час потенційного запізнення на кожній ділянці (межа запізнення), час відправлення, час на відстій після завершення руху маршрутом тощо. Ці обмеження також треба переглядати кожен раз, коли у зв’язку зі зміною просторових, часових та матеріальних ресурсів мережі треба скорегувати розклад руху.

3. Визначити хромосоми (особини популяції) генетичного алгоритму. Залежно від того, які саме обставини враховуються при визначенні

розкладу (тільки час відбуття та прибуття, номер та різновид потягу, номер бригади, що обслуговує потяг тощо) хромосома може містити різний набір значень. У найпростішому випадку це може бути або пара «номер потягу» - «час руху окремою ділянкою маршруту», або «номер потягу» - «номер маршруту» - «станція відбуття» - «час відбуття» - «номер станції прибуття» - «час прибуття». Хромосома має містити достатню кількість даних для опису руху кожного потягу, який курсує на маршруті. Кількість потягів, розклад яких включається до складу хромосоми, може бути або замалим, або занадто великим, тому необхідно врахувати це під час формування популяції рішень. Цю операцію необхідно повторювати кожен раз, коли кількість потягів у мережі зменшується або збільшується, а також після змін просторової конфігурації мережі та адміністративних правил руху.

4. Формування початкової популяції рішень. Кількість рішень у популяції, яку варто обирати, має бути більшою за потужність графа (більшою за кількість станцій). Якщо обрати замало або забагато особин в популяції, виникає проблема зростання часу знаходження локального оптимуму.

Таблиця 3.3

Експериментальний час розв’язання задачі PPT генетичним алгоритмом

Кількість особин в популяції	Ліміт часу	Факт знаходження рішень	Номер популяції локального оптимуму
50	300 с	Ні	4552
150	300 с	Так	619
250	300 с	Так	490
500	300 с	Ні	973

В ході виконання цієї кваліфікаційної роботи були обрані чотири варіанти кількості особин в популяції та отримані наступні рішення (таблиця 3.3).

Часові обмеження обов’язково треба накладати тому, що будь-який алгоритм оптимізації не може бути застосований абсолютно до всіх можливих вхідних даних і може знайти локальний мінімум замість глобального. Тому, щоб не витрачати зайвий час, треба обмежити час роботи програми. Етап формування початкової популяції треба повторювати кожен раз, коли змінюються хромосоми рішень.

5. Запуск процесу пошуку рішень протягом якого мають виконуватись операції відбору, схрещування та мутації. Задля досягнення найкращих результатів треба використовувати всі три операції. Під час виконання кваліфікаційної роботи був виконаний пошук рішень з наступними параметрами кожної з операцій (табл. 3.4).

Таблиця 3.4

Визначення кількості особин у турнірному відборі

Кількість особин в популяції	Кількість особин в кожному турі турніру	Номер покоління локального мінімуму
150	2	631
150	3	627
150	5	619
250	2	502
250	3	496
250	5	490

Як наведено у таблиці 3.4, кількість особин у популяції не суттєво впливає на час успішного пошуку локального мінімуму для цих даних. Подальше збільшення кількості особин в кожному турі турніру не призвело до

покращення результату, тому було обрано 5 особин. Під час схрещування були випробувані різні варіанти комбінації генів. Але для цих даних суттєвої різниці під час рекомбінації помічено не було, тому був обраний класичний метод обміну 50 на 50.

Були випробувані кілька варіантів мутації, результати яких наведені у наступній таблиці 3.5.

Таблиця 3.5

Визначення кількості особин у турнірному відборі

Кількість особин в популяції	Відсоток мутованих генів	Номер покоління знаходження рішення
150	≤5%	619
150	> 5%, ≤15%	644
150	>15%, ≤30%	760
250	≤ 5%	490
250	> 5%, ≤15%	517
250	>15%, ≤30%	689

В результаті експерименту було показано, що занадто сильна мутація суттєво сповільнює швидкість пошуку рішення. Тому для виконання пошуку рекомендується використовувати мутацію не більш як 5% генів.

Цей пункт методики необхідно застосовувати кожен раз для знаходження нового розкладу під час будь-яких змін.

3.4. Висновок

Задля проведення експерименту було створено програмний застосунок мовою Python в середовищі PyCharm. В результаті експерименту було визначено, що для наданих даних найкраща кількість особин в популяції

вірогідно знаходиться у межах 150 та 250 особин. При такій кількості особин та часових обмеженнях у 300 секунд вдалося збудувати розклад, який відповідає встановленим критеріям щодо часових та просторових обмежень руху транспорту в залізничній мережі. Було визначено кількість особин рішень під час туру турнірного відбору на кожній ітерації пошуку оптимального рішення, визначений відсоток мутації генів у хромосомах рішень та обраний метод схрещування.

На підставі обробки експериментальних даних була запропонована методика застосування генетичного алгоритму до розв’язання задач PPT. Було показано, що генетичний алгоритм є прийнятнішим варіантом розв’язання цієї задачі у порівнянні з випадковим вибором кожного потягу, який планується на кожній ітерації, та процедурою випадкової упередженої вибірки на основі відмови.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи були досліджені математичні методи та засоби складення розкладів руху транспорту. Були проаналізовані різні методики розв’язання повних перебірних задач. З метою знаходження локальних мінімумів за бюджет часу.

Було створено програмний застосунок мовою Python в середовищі PyCharm, що реалізує генетичний алгоритм розв'язання задачі.

У експериментальній частині були проаналізовані методи випадкового вибору кожного потягу, який планується на кожній ітерації, процедуру випадкової упередженої вибірки на основі відмови та генетичний алгоритм.

Було з’ясовано, що для наданих вхідних даних найкращі результати щодо знаходження локального мінімуму показує генетичний алгоритм з 250 особин популяції, 5-ма учасниками кожного туру відбору, кількістю мутацій не більше 5% та з застосуванням методу схрещування.

З отриманих результатів можна зробити висновок, що цей метод фактично не залежить від характеру вхідних даних і може бути застосований для побудови розкладу руху в будь-якій транспортній мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузнецов Н.А., Пащенко Ф.Ф., Рябых Н.Г., Захарова Е.М., Минашина И.К. Алгоритмы оптимизации в задачах планирования на рельсовом транспорте. Информационные процессы. 2014. Том 14, № 4, ст. 307–318. URL: http://www.jip.ru/2014/307-318-2014.pdf.
2. P. Liu, B. Han. Optimizing the train timetable with consideration of different kinds of headway time. Journal of Algorithms & Computational Technology. 2017. Vol. 11(2), p. 148–162. URL: https://journals.sagepub.com/doi/pdf/10.1177/1748301816689685.
3. R. Sanat, T. Dutt, A. Chandrababu, A. Abhilasha, G. N. Srinivasa Prasanna. Optimizing Schedule of Trains in Context of a Large Railway Network. 2018 21st International Conference on Intelligent Transportation Systems (ITSC) Maui, Hawaii, USA, November 4-7, 2018. URL:
4. Z. Huang, H. Niu. Study on the Train Operation Optimization of Passenger Dedicated Lines Based on Satisfaction. Discrete Dynamics in Nature and Society, vol. 2012, Article ID 451201, 11 pages, 2012. URL: https://doi.org/10.1155/2012/451201.
5. X.Cai, C.J.Goh. A fast heuristic for the train scheduling problem. Computers Ops Res. Vol. 21. No. 5, pp. 499-510, 1994.
6. P. Zhao, N. Tomii, N. Fukumura, T. Sakaguchi. An algorithm for train-set scheduling based on probabilistic local search. 2002 WIT Press, Ashurst Lodge, Southampton, SO40 7AA, UK. URL: https://www.witpress.com/Secure/elibrary/papers/CR02/CR02080FU.pdf.
7. Niu, and M. Zhang. An Optimization to Schedule Train Operations with Phase-Regular Framework for Intercity Rail Lines. Hindawi Publishing Corporation Discrete Dynamics in Nature and Society, Volume 2012, Article ID 549374, 13 pages. URL: https://www.hindawi.com/journals/ddns/2012/549374/

8. H. Niu, X. Zhou. Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. Transportation Research Part C: Emerging Technologies, Volume 36, 2013, Pages 212-230, ISSN 0968-090X. URL: https://www.sciencedirect.com/science/article/abs/pii/S0968090X13001800
9. M. Chen, H. Niu. Optimizing Schedules of Rail Train Circulations by Tabu Search Algorithm. Mathematical Problems in Engineering. 2013. 1-7. 10.1155/2013/102346. URL: https://www.researchgate.net/publication/270679332_Optimizing_Schedules_of_Rail_Train_Circulations_by_Tabu_Search_Algorithm/link/618d281407be5f31b7681d2c/download.
10. Introduction to Genetic Algorithms. URL: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3.
11. Генетический алгоритм. URL: https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC.
12. P. Serafini, W. Ukovich. A mathematical model for periodic scheduling problems. SIAM J. on Discrete Mathematics, 2:550–581, 1989.
13. L. Kroon, L. Peeters. A variable time model for cycling railway timetabling. Transportation Science, 37(2):198–212, 2003.
14. C. Liebchen. Periodic Timetable Optimization in Public Transport. dissertation.de - Verlag im Internet GmbH 2006, 2006.
15. K. Nachtigall and S. Voget. A genetic algorithm approach to periodic railway synchronization. Computers and Operations Research, 23:453–463, 1996.
16. M. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. Transportation Research Part B, 30(6):455–464, December 1996.

17. B. Szpigel. Optimal train scheduling on a single track railway. In M. In Roos, editor, Proceedings of IFORS Conference on Operational Research'72, pages 343–352, 1973.
18. D. Jovanovic and P. T. Harker. Tactical scheduling of rail operations: The scan i system. Transportation Science, 25(1):46–64, 1991.
19. M. Carey and D. Lockwood. A model, algorithms and strategy for train pathing. The Journal of the Operational Research Society, 46(8):988–1005, 1995.
20. A. Higgins, E. Kozan, and L. Ferreira. Heuristic techniques for single line train scheduling. Journal of Heuristics, 3(1):43–62, 1997.
21. E. Silva de Oliveira. Solving Single-Track Railway Scheduling Problem Using Constraint Programming. PhD thesis, The University of Leeds, School of Computing, September 2001.
22. R. K. S. Kwan and P. Mistry. A co-evolutionary algorithm for train timetabling. In IEEE Press, editor, Congress on Evolutionary Computation, pages 2142–2148, 2003.
23. A. Caprara, M. Monaci, P. Toth, and P. Guida. A lagrangian heuristic algorithm for a real -world train timetabling problem. Discrete Applied Mathematics, 154:738–753, 2006.
24. L. Ingolotti, A. Lova, F. Barber, P. Tormos, M.A. Salido, and M. Abril. New heuristics to solve the csop railway timetabling problem. Advances in Applied Artificial Intelligence. LNAI, Subseries of Lecture Notes in Computer Science, 2006.
25. J. Kelley. The critical-path method: Resources planning and scheduling. Industrial Scheduling, 1963.
26. A. Schirmer and S Riesenberg. Parameterized heuristics for project schedulingbiased random sampling methods. Technical Report 456, Institute fr Betriebswirtschaftslehre der UNIVERSITT KIEL, 1997.

27. P. Tormos and A. Lova. A competitive heuristic solution technique for resourceconstrained project scheduling. Annals of Operations Research, 102(1-4):65–81, 2001.

Додаток А

Відомість матеріалів кваліфікаційної роботи

		Позначення	Найменування	Кількість	Примітка
	1				
	2		Документація	71	
	3				
	4	**ІТКІ. КР 15.01.ДА.ПЗ**	Пояснювальна записка	71	
	5		Презентація	20	
	6		Диск CD-R з презентацією	1	

Зм	Лист	№ докум.	Підпис	Дата	**ІТКІ.КР 15.01.ДА.ПЗ**			
Розроб.		Бережний			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
Керівник		Коротенко					1	1
Рецензент		Галушко				НТУ «ДП» 8; 126м-20-1		
Н.контр.		Коротенко						
Зав. каф.		Гнатушенко						

Додаток Б

Вихідний код

```
from pprint import pprint
import numpy as np
import matplotlib.pyplot as plt

STATIONS = 10
TRAINS = 7
SPEED = 50
PAUSE_TIME = 5
WAITING_AFTER = 15
ROUTES = [[1, 2, 3, 4, 5, 7],
          [7, 6, 5, 4, 3, 2,1]]

def get_distances_graph(N=STATIONS):
    matrix = np.zeros((N,N), int)
    for i in range(0, N):
        for j in range(0, N):
            if j == (i+1) and i<N-1:
                matrix[i][j] = 1
#            elif j == (i-1) and i>0:
#                matrix[i][j] = 1
    return matrix

def get_max_colors(graph):
    return graph[graph.sum(axis=1).argmax()].sum()+1

def generate_population(population_size, individual_size, max_colors):
    for
    return np.random.randint(max_colors+1, size=(population_size,
individual_size))

def fines(individual, graph):
    fine = 0
    for i in range(len(individual)-1):
        for j in range(i+1, len(individual)):
            if individual[i] == individual[j] and graph[i][j] == 1:
                fine += 1
    return fine

def tournament_selection(population, graph, k=2):
    new_population = [] #np.array((0,0), int)
    for t in range(k):
        np.random.shuffle(population)
        for i in range(len(population)//k):
            x = np.array([fines(ind, graph) for ind in population[i*k:
k*(i+1)] ])
            new_population.append(population[i*k: k*(i+1)][x.argmin()])
    return np.array(new_population[:len(population)+1])

def mutation(population, max_colors, probability=0.3):
    mutations = np.random.rand(len(population))
    count = 0
    for i in range(len(mutations)):
        if mutations[i] > probability:
            count += 1
```

```
            index = np.random.randint(len(population[i]))
            population[i][index] = np.random.randint(max_colors)
    return population, count

def crossing_over(population, graph, k=5):
    t = []
    for x in (k):
        i1 = population.pop(np.random.randint(len(population)-1))
        i2 = population.pop(np.random.randint(len(population)-1))
        for j in range(len(i1)//2, len(i1)):
            i1[j], i2[j] = i2[j], i1[j]
        t.append(i1)
        t.append(t2)
    population.extend(t)
    return population

def draw(population, graph, max_colors):
    x = np.array([fines(ind, graph) for ind in population])
    uniques, counts = np.unique(x, return_counts=True)
    fitnes = [[len(graph)-t[0], t[1]] for t in zip(uniques, counts)]
    fitnes.sort()
    to_show = [0 for i in range(len(graph)+1)]
    for obj in fitnes:
        to_show[obj[0]] = obj[1]

def evoluate(graphtype, budget=20):
    def avg_fitnes(population, graph):
        x = np.array([fines(ind, graph) for ind in population])
        uniques, counts = np.unique(x, return_counts=True)
        fitnes = [(len(graph)-t[0])*t[1] for t in zip(uniques,counts)]
        avg = 1
        for i in fitnes:
            avg += i
        return avg/len(population)

    solution = None
    fine = None
    graph = get_distances_graph(STATIONS)
    max_colors = get_max_colors(graph)
    population_size = 50 + np.random.randint(200)
    population = generate_population(population_size, STATIONS,
max_colors)
    generation = 0
    start_avg_fitnes = avg_fitnes(population, graph)
#    draw(population, graph, max_colors)
    while generation < budget:
        population = tournament_selection(population, graph, k=4)
        population, count = mutation(population, max_colors)
        generation += 1
        draw(population, graph, max_colors)
        x = np.array([fines(ind, graph) for ind in population])
        if x.min() == 0:
            solution = population[x.argmin()]
            fine = 0
            break
    if solution is None:
        x = np.array([fines(ind, graph) for ind in population])
        solution = population[x.argmin()]
        fine = x.min()
    result_avg_fitnes = avg_fitnes(population, graph)
```

```
    return solution, fine, generation, population_size, start_avg_fitnes,
result_avg_fitnes, max_colors

if __name__ == '__main__':
    np.random.seed(100)

    for i in range(10):
        solution, fine, generation, population_size, start_avg_fitnes,
result_avg_fitnes, max_colors = evoluate('bipartite')
        uniques = np.unique(solution)
        print(i, max_colors, fine, generation, population_size,
start_avg_fitnes, result_avg_fitnes)
```

ДОДАТОК В

ВІДГУК

на кваліфікаційну роботу рівня магістра

«Підвищення ефективності складання розкладу руху поїздів Укрзалізниці методами штучного інтелекту»

студента групи 126м-20-1 Бережного Андрія Анатолійовича

1 Мета даної кваліфікаційної роботи – дослідити можливості залучення методів штучного інтелекту до вирішення практичної задачі – складання розкладу руху поїздів, оскільки звісно, що штучний інтелект – це область інформатики, яка займається розробкою інтелектуальних комп'ютерних систем з можливостями, які спеціалісти традиційно пов'язують з людським розумом, тобто розумінням мови, навчанням, здатністю розмірковувати, вирішувати складні проблеми тощо.

2 Обрана тема актуальна тому, що на даний час поширення реалізацій застосування штучного інтелекта є досить значним.

3 Тема кваліфікаційної роботи відповідного рівня безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології» - створення інформаційних технологій різного призначення і застосування.

4 Явища і процеси, що досліджуються в даній кваліфікаційній роботі і обрані для моделювання, оцінювання та дослідження – віднесені в освітньо-кваліфікаційній характеристиці магістрів до класу дослідних та евристичних, рішення яких заснована на знаково-понятійних уміннях.

5 Робота складається з трьох розділів. Перший розділ присвячений аналізу теми дослідження та постановці задачі. У другому розділі наведено проектну складову вирішення завдання. Третій розділ присвячено детальному опрацюванню побудови, налаштування та безпеки фрагмента информаційної технології застосування елементів штучного інтелекту до вирішення практичної задачі. Оригінальність отриманих в роботі наукових результатів і їх наукова новизна полягають у наступному:

– досліджені математичні методи та засоби, що призначені для розв’язання задач складання розкладів руху залізничного транспорту у відповідних мережах;

– обґрунтувано вибір способу розв’язання повних перебірних задач з метою пошуку локального мінімуму за умови бюджетування часу проведення розв’язання;

– обрано метод розв’язання та визначити найкращі параметри його застосування з урахуванням наявних даних.

7 Практичні результати кваліфікаційної роботи отримані із застосуванням відповідних інформаційних технологій, а також програмних продуктів MS Word і MS PowerPoint на інформаційно-технологічній платформі Windows.

8 Оформлення графічних матеріалів до кваліфікаційної роботи рівня магістр виконано на сучасному рівні і відповідає вимогам, що пред'являються до рівня виконання робіт даної кваліфікації.

9 Ступінь самостійності виконання кваліфікаційної роботи достатньо висока.

10 Деякі дискусійні положення та недоліки, які мають місце в роботі:

а) недостатньо чітко визначено прив’язку даного підходу до вирішення задач конкретного підприємства, тобто Укрзалізниці;

б) досить просторо описано компоненти процесу вирішення поставленого завдання.

Незважаючи на вищевказані зауваження, кваліфікаційна робота в цілому заслуговує оцінки «добре» та присвоєння здобувачу відповідної кваліфікації.

Керівник кваліфікаційної роботи,
проф. кафедри ІТКІ, д.т.н. Г.М. Коротенко

ДОДАТОК Г
РЕЦЕНЗІЯ

на кваліфікаційну роботу рівня магістра
«Підвищення ефективності складання розкладу руху поїздів Укрзалізниці методами штучного інтелекту»
студента групи 126м-20-1 Бережного Андрія Анатолійовича

Розглянута робота присвячена дослідженню впровадження компонентів штучного інтелекту до створення інформаційних технологій підтримки етапів рішення складних практичних задач, пов’язаних з складанням розкладу руху поїздів. Це особливо важливо, оскільки останнім часом експерти міжнародної організації зі створення світових стандартів ISO/IEC також підтримали розробку проекту міжнародного документа «Інформаційні технології – Штучний інтелект (ШІ) – Огляд обчислювальних підходів для систем Ш І».

Завдання і зміст кваліфікаційної роботи відповідає головній цілі - перевірці знань і ступеня підготовленості студента за фахом 126 «Інформаційні системи та технології» галузі знань 12 «Інформаційні технології».

Зміст пояснювальної записки кваліфікаційної роботи відповідає необхідним критеріям та затвердженій темі.

Актуальність обраної теми обумовлена тим, що дослідження ефективності застосування інформаційних технологій і відповідних комп’ютерних засобів продовжують розвиватися на базі розширення їхнього спектру.

Повнота і глибина вирішення задач, поставлених в завданні на кваліфікаційну роботу є достатньою.

Оформлення пояснювальної записки кваліфікаційної роботи виконано в повній відповідності з діючими стандартами і нормативними вимогами.

Список літератури, наведений в роботі, налічує більше ніж 20 джерел, що свідчить про вміння автора працювати з літературою та іншими інформаційними матеріалами.

Наукова новизна результатів дипломної роботи визначається тим, що вперше виконано застосування елементів штучного інтелекту для вирішення задачі складання розкладу руху поїздів Укрзалізниці.

Практичне значення результатів роботи полягає в розробці моделей застосування щтучного інтелекту для вирішення практичних задач у галузі залізничного транспорта.

До числа загальних зауважень і недоліків роботи слід віднести:

1) відсутність порівняння результатів дослідження з іншими відомими роботами;

2) недостатнє розкриття компонентів роботи, пов’язаних з виконанням відповідного аналізу комплексу охоплених розрахунками елементами залізничних мереж.

Однак, зазначені зауваження не здійснюють істотного впливу на підсумкові результати кваліфікаційної роботи і не знижують її безумовну практичну та наукову цінність.

Таким чином, слід зробити висновок, що кваліфікаційна робота в цілому

заслуговує оцінки «_____», а її виконавець присвоєння відповідної кваліфікації.

Рецензент, професор кафедри безпеки
інформації та телекомунікацій, к.ф.-м.н. **доц. О.М. Галушко**