Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Складання розкладу руху мереж громадського транспорту за допомогою еволюційних алгоритмів
(тема)

Виконав:
студент II курсу, групи КІТм-22-1
Олександр МЕЛЬНИЧЕНКО
(власне ім'я, прізвище)

Спеціальність 123 Комп’ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник проф. Олександр БЕЗСОНОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри (підпис) Олег РУДЕНКО (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

«_____»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мельниченко Олександру Володимировичу
(прізвище, ініціали)

1. Тема роботи (проекту) Складання розкладу руху мереж громадського транспорту за допомогою еволюційних алгоритмів

затверджена наказом університету від «_6_» __листопада___2023 р. №_ _____

2. Термін подання студентом роботи до екзаменаційної комісії _13_ _січня___2024р._____

3. Вихідні дані до роботи (проекту)
ДСТУ щодо обробки інформації, нормативно правові та законодавчі акти України, періодичні видання, науково-методичні розробки вітчизняних та зарубіжних авторів, літературні джерела, матеріали практики, результати спостережень поведінки споживачів

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз предметної області та постановка задачі
Аналіз підходів, методів та моделей вирішення завдань дослідження
Опис процесу складання розкладу та формулювання наукової задачі
Порівняльний аналіз існуючих методів вирішення завдань складання розкладу

Опис технологій і методологій

Результати роботи алгоритму та екранні форми веб-додатку

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням кафедри) _____
Модель організаційної структури підприємства та його підрозділу;

Модель формату хромосоми; діаграма компонентів системи; екранні форми додатку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявністю консультантів згідно до наказу, зазначеному у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та актуальності теми роботи	6.11 - 10.11.2023	виконано
2	Огляд сучасної літератури за напрямком дослідження	11.11 - 17.11.2023	виконано
3	Постановка задачі дослідження	18.11.2023	виконано
4	Аналіз підходів, методів та моделей вирішення завдань	19.11 - 23.11.2023	виконано
5	Порівняльний аналіз існуючих методів	24.11 - 27.11.2023	виконано
6	Еволюційні стратегії	27.11-30.11.2023	виконано
7	Опис задачі оптимізації в термінах алгоритму еволюційних стратегій	1.12-8.12.2023	виконано
8	Алгоритм еволюційних стратегій	9.12 - 15.12.2023	виконано
9	Результати роботи алгоритму	16.12 - 22.12.23	виконано
10	Оформлення пояснювальної записки	23.12 - 31.12.23	виконано
11	Оформлення презентації	1.01- 4.01.2024	виконано

Дата видачі завдання «03» _листопада_2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 73 с., 12 рис., 4 табл., 54 джерел, 1 додатки.

Об'єктом дослідження є розклад відділу технічної підтримки та адміністрування на IT-підприємстві та способи його формування, предметом дослідження – безпосередньо алгоритм еволюційних стратегій у контексті складання розкладу.

Метою дослідження є скорочення часу на формування розкладу змін та підвищення якості одержуваного розкладу шляхом розробки гнучкої автоматизованої системи складання розкладу змін відділу технічної підтримки.

Методи дослідження включають в себе:

1) методи теорії розкладів;
2) методи теорії ймовірності;
3) методи алгоритмізації і програмної реалізації математичних моделей;
4) метод аналізу та класифікації;
5) метод еволюційних стратегій.

Теоретична і практична значущість роботи полягає у застосуванні алгоритму еволюційних стратегій для вирішення спеціалізованого завдання отримання близьких до оптимальних розкладів робочих змін відділу технічної підтримки та адміністрування.

Створений програмний додаток дозволяє зменшити час, необхідний на вирішення задачі складання розкладу. Результати дослідження та проектування були впроваджені як частина інфраструктури в IT-компанії «СИСТЕМ КЛАУД КОНЕКШН». Також, вони можуть бути застосовані для створення розкладів у інших різноманітних галузях.

ЕВОЛЮЦІЙНІ СТРАТЕГІЇ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ПОПУЛЯЦІЯ, МУТАЦІЯ, КРОСОВЕР, ГЕН, ХРОМОСОМА.

ABSTRACT

Explanatory note to the diploma project: 73 pages, 12 figures, 4 tables, 1 annexes, 54 sources.

The object of the research is the organization of the technical support and administration department work at the IT enterprise, the subject of the study is the algorithm of evolution strategies in the context of the expansion of the schedule.

The purpose of the study is to reduce the time to create a shift schedule and improve the resulting schedule quality by developing a flexible automated shifts scheduling system for the technical support department.

Research methods include:

1) methods of the theory of decompositions;
2) methods of probability theory;
3) methods of algorithmization and program realization of mathematical models;
4) methods of analysis and classification;
5) the method of evolutionary strings.

The theoretical and practical significance of the work is to apply the algorithm of evolutionary strategies to solve a specialized task of obtaining close to the optimal schedules of the technical support and administration department working shifts.

The created software application with minimal expenses and low labor intensity solves the tasks set. The research and design results were implemented as part of the infrastructure of the IT company. Also, they can be used to create schedules in other diverse industries.

EVOLUTIONAL STRATEGIES, GENETIC ALGORITHM, POPULATION, MUTATION, CROSSVER, GEN, CHROMOSOME.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ СКЛАДАННЯ РОБОЧОГО РОЗКЛАДУ ТА ВИМОГ 10
1.1 Опис процесу складання розкладу та формулювання наукової задач 10
1.2 Коротка характеристика об’єкта управління «СІСТЕМ КЛАУД КОНЕКШН», проблематики та вимог 13
1.3 Аналіз підходів, методів та моделей вирішення завдань дослідження 18
2 ТЕОРЕТИЧНЕ ТА МЕТОДИЧНЕ ДОСЛІДЖЕННЯ ВИРІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ РОБОЧОГО РОЗКЛАДУ 26
2.1 Порівняльний аналіз існуючих методів вирішення завдань складання розкладу 26
2.2 Еволюційні стратегії 34
2.3 Опис задачі оптимізації в термінах алгоритму еволюційних стратегій 38
2.4 Алгоритм еволюційних стратегій 42
3 ТЕХНІЧНІ РІШЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМУ 48
3.1 Опис технологій і методологій 48
3.2 Результати роботи алгоритму та екранні форми веб-додатку 54
3.3 Аналіз результатів та експериментальне дослідження методу еволюційних стратегій 56
ВИСНОВКИ 61
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 63
ДОДАТКИ 68
Додаток А Лістинги основних операцій алгоритму еволюційних стратегій 69

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ:

ЕА – еволюційний алгоритм

ГА – генетичний алгоритм

ГП – генетичне програмування

ЕС – еволюційні стратегії

ПЗ – програмне забезпечення

ТР – теорія розкладів

CC – Customer Care

SA – System Administrator

EX – Expert

L1 – Level 1, перша лінія технічної підтримки; Help Desk

L2 – Level 2, друга лінія технічної підтримки; адмініструваання

L3 – Level 3, третя лінія технічної підтримки; експертна підтримка

SLA – Service Level Agreement

БД – база даних

СУБД – система управління базами даних

ВСТУП

Одне з найважливіших завдань, яке необхідно вирішити при плануванні робочих процесів у компаніях з позмінним графіком, - це складання розкладу. Перш за все, команда, яка працює за позмінним графіком, не може функціонувати без програми. Це означає, що робочі програми повинні бути підготовлені вчасно. Крім того, графік повинен бути складений "якісно". Ці критерії включають в себе відображення економічної ефективності використання наявних ресурсів компанії, комфорт співробітників, обмеження робочого часу тощо.

Розклад - це задача цілочисельного програмування, складність якої зростає експоненціально зі збільшенням кількості змінних, що підлягають зміні, та їх можливих значень (такі задачі класифікуються як NP-важкі задачі). Вона також характеризується великою кількістю вхідної інформації в різних конфігураціях і великою кількістю вимог, які важко сформулювати.

Останніми роками для розв'язування великих задач цілочисельного програмування, в тому числі задач розкладів, використовуються різні евристичні методи. Ці методи включають еволюційні стратегії.

Об'єктом дослідження роботи є складання робочого розкладу змін відділу технічної підтримки та адміністрування на IT-підприємстві.

Предметом дослідження в даній дипломній роботі є алгоритм еволюційних алгоритмів у контексті задачі складання розкладу.

Метою даного дослідження є розробка гнучкої автоматизованої системи планування мережі громадського транспорту, що дозволить скоротити час, який витрачається на створення розкладів руху, та покращити якість отриманих розкладів.

Задачами дипломної роботи є:

1. Вивчити структуру підприємства та особливості розподілу робочого навантаження;

2. Провести аналіз існуючих рішень реалізації еволюційних стратегій і аналіз вимог до робочого розкладу;

3. Сформулювати функцію пристосовності розкладу занять.

3. Сформулювати оптимізаційну задачу в термінах і поняттях еволюційних стратегій.

4. Реалізувати алгоритм еволюційних стратегій для формування розкладу робочих змін відділу технічної підтримки.

5. Дослідити розроблений алгоритм на основі порівнянь з існуючими рішеннями задачі та визначити переваги та недоліки власного алгоритму; також, обґрунтувати чому саме цей метод є найоптимальнішим у задачі формування розкладу занять.

Для вирішення поставлених в роботі завдань використані: методи системного аналізу, методи математичного моделювання, структури і бази даних, теорія реляційних відносин, еволюційні і генетичні алгоритми, методи узагальненого і об'єктно орієнтованого програмування.

Теоретична значимість роботи полягає в тому, що детально розглянуто та проаналізовано комплекс математичних, інформаційних та алгоритмічних моделей задачі складання розкладу занять.

Практичне значення даного дослідження полягає в тому, що в ньому запропоновано використання еволюційних алгоритмів для вирішення задачі отримання наближеного до оптимального розкладу робочих змін для відділів технічної підтримки та управління. Також це створення цільової функції для розкладів, яка має можливість враховувати як вимоги, так і вподобання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ СКЛАДАННЯ РОБОЧОГО РОЗКЛАДУ ТА ВИМОГ

1.1. Опис процесу складання розкладу та формулювання наукової задачі

У сучасному суспільстві діяльність людини планується в часі та просторі. Планування є синонімом організації та одним з найважливіших засобів забезпечення ефективного виконання будь-якої діяльності та будь-якого виду робіт (операцій). Чим краще складений графік, тим вища продуктивність праці, менші витрати ресурсів, зумовлені даною діяльністю, кращі отримані результати та умови їх отримання. Задача календарного планування є предметом наукових досліджень з середини минулого століття. Сфера її застосування охоплює різні сфери людської діяльності, такі як транспорт, державні служби, промисловість та освіта.

Розклади супроводжують нас всюди: розклад кіносеансів, поїздів, авіарейсів, занять в школі або на курсах, телепрограма, години прийому лікарів і так далі. При складанні списку справ ми орієнтуємося на необхідність кожного з його елементів, зіставляємо тимчасові рамки, ґрунтуючись на обмеження розкладів роботи або надання послуг іншими людьми або компаніями.

Щодня люди стикаються з необхідністю планувати свій день. Це може варіюватися, наприклад, від базових щоденних завдань (їжа, сон, гігієна) до більш складних завдань (наприклад, планування вечірки на день народження). Плануючи наступний день, ми плануємо і враховуємо тривалість дій, їх важливість і порядок, в якому вони повинні виконуватися одночасно.

Якщо з плануванням свого дня людина справляється майже інтуїтивно, то впоратися з більш складними задачами, що включаюсь в себе велику кількість елементів та обмежень, людині набагато важче.

Коли йдеться про робочі програми, варто пам'ятати, що правильно побудована програма може допомогти поліпшити сприйняття працівниками

своєї роботи і, таким чином, мати значний вплив на якість і прибутковість їхньої праці в довгостроковій перспективі. Процес створення такої програми має багато підводних каменів, обмежень та особливих умов, які необхідно враховувати для певних компаній.

У загальній постановці задача теорії розкладів є процес упорядкування деякого набору робіт (операцій) в часі в умовах ресурсних та інших обмежень [2].

Мета рішення таких задач – побудова допустимих розкладів, при якому всі обмеження дотримані, або, що є більш складним, - знаходження оптимального допустимого розкладу по тим або іншим критерієм оптимальності.

Задача планування є задачею цілочисельного програмування, і складність її розв'язання зростає експоненціально зі збільшенням кількості змінних та можливих значень. Крім того, задача планування характеризується великою кількістю інформації про ресурси в різних конфігураціях і великою кількістю вимог, які важко сформулювати. Ці труднощі перешкоджають автоматизації процедур складання розкладів, незважаючи на наявність різноманітних методів цілочисельного програмування [5].

На практиці від того, наскільки ефективно складено розклад, залежить:

1) якість виконання робочих обов'язків;

2) економічна ефективність роботи;

3) комфортність роботи співробітників і задоволеність клієнтів якістю наданої роботи;

4) а також дозволяє без збоїв проводити робочий процес і уникнути необхідності коригування розкладу після початку робочого періоду [3].

Автоматизація процедури складання графіку роботи дозволяє:

1) врахувати всі вимоги, що пред'являються до розкладу;

2) визначити критерії для оцінки ефективності та оптимізації розкладу;

3) зменшити тимчасові витрати на складання розкладу, це важливо при необхідності швидко отримати розклад після зміни зовнішніх умов;

4) дозволить врахувати побажання співробітників, щодо переважного часу роботи і особистих обставин і забезпечить можливість роботи за сумісництвом.

Задачі теорії розкладів належать до класу NP-повних задач, які є не тільки теоретично дуже важливими, але й поширеними на практиці. Для того, щоб знайти оптимальний розв'язок цього класу задач, необхідно повністю дослідити всі можливі альтернативи, що не завжди можливо через обмеженість ресурсів. У цьому випадку повне дослідження альтернатив призводить до логарифмічного збільшення ресурсів, що витрачаються на пошук рішення, зі збільшенням розміру проблеми. Виникає потреба в методах, що характеризуються поліноміальною залежністю часу обчислень від розміру задачі з близькою до оптимальної точністю [4].

На даний момент часу такого універсального алгоритму не існує. Слід зазначити, що алгоритми, призначені для вирішення широкого класу задач, часто менш ефективні, ніж алгоритми, що враховують конкретні умови більш вузького класу задач [3, 7].

Ручне вирішення завдань планування вимагає багато часу і кваліфікованих фахівців. Одна людина не може вивчити і проаналізувати всі варіанти розкладу, особливо великих завдань, з урахуванням всіх обмежень. Більше того, навіть найдосвідченіший працівник не завжди може скласти програму, яка задовольнить усіх учасників. Одним із виходів із цієї ситуації є автоматизація процесу складання розкладу. Чітко визначений і протестований алгоритм може прискорити процес і підвищити якість отриманого розкладу. Якщо цей алгоритм також реалізувати програмно, то витрати часу можна значно скоротити.

В даний час перед підприємствами стоїть наступний вибір:

1) продовжувати складати розклад вручну;

2) придбати відповідне ПЗ у сторонньої організації;

3) реалізувати систему автоматизації складання розкладу особисто.

Як було зазначено вище, ручне складання розкладу має свої недоліки в силу людського фактору. Мозок людини не здатний з тією ж легкістю як комп'ютер обробляти велику кількість інформації. Подібна робота довіряється тільки досвідченим менеджерам, але і вони не завжди здатні скласти оптимальний варіант розкладу.

Укладення контракту зі сторонньою організацією на надання необхідного програмного забезпечення - це, зрештою, хороший варіант, але він також має свої недоліки. Першим основним недоліком є фінансовий аспект, оскільки інвестиції потрібні не лише для придбання системи, але й для її впровадження та підтримки.

Ще одним важливим недоліком даного підходу є той факт, що більшість систем не мають можливості врахування індивідуальних особливостей кожного підприємства, що, безсумнівно, дуже сильно впливає на якість одержуваного розкладу.

Саме тому все більше і більше організацій з часом відмовляються від ручного способу і від придбання системи у стороннього виробника і все частіше схиляються до варіанту власної реалізації.

Позитивною стороною цієї ситуації є те, що впроваджена система адаптована під конкретну компанію, що збільшує можливість створення більш оптимізованої програми. Крім того, вартість такої системи буде в кілька разів дешевшою, ніж витрати на придбання стороннього програмного забезпечення. Важливим аспектом такого підходу є те, що він вимагає ретельного дослідження та аналізу цільових областей та розробників, необхідних для впровадження, реалізації та підтримки системи.

1.2. Коротка характеристика об’єкта управління «систем клауд конекшн», проблематики та вимоги

Компанія була заснована у 2000 році. Починаючи з невеликої команди з управління серверами, компанія перетворилася на провідного постачальника IT-послуг у двох основних сферах бізнесу: управління серверами та технічна підтримка і розробка програмного забезпечення. Сьогодні компанія складається з двох підрозділів, відділів та понад 280 компетентних, мотивованих та цілеспрямованих співробітників, які діють в інтересах компанії та її клієнтів.

Компанія обслуговує клієнтів з 30 країн по всьому світу. На даний час працює з клієнтами з Німеччини, США, Великобританії або Канади, Фінляндії, Сінгапуру, Лівану, Малайзії, Норвегії, Бразилії, Індії, ОАЕ, Австралії, Іспанії, Італії, Данії, Нідерландів, Ліхтенштейну та інших.

Схему організаційної структури підприємства наведено на рис 1.1.

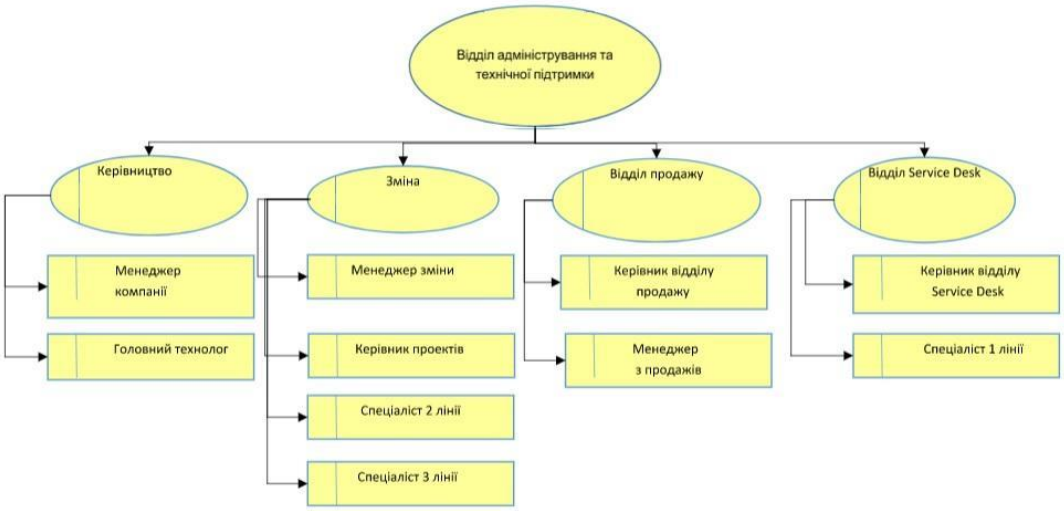


Рисунок. 1.1. Організаційна структура підприємства

Успіх компанії, що забезпечує технічну підтримку, цілком залежить від клієнтів. Тому графік роботи може стати одним з найважливіших конкурентних переваг.

Найбільш поширені 8-годинний або 12-годинний графіки роботи. Розглянуте підприємство використовує 12-годинний графік роботи на даний момент.

Класична структура технічної підтримки інформаційних бізнес-систем передбачає наявність трьох основних ліній роботи, послідовно вирішальних надходять сервісні запити.

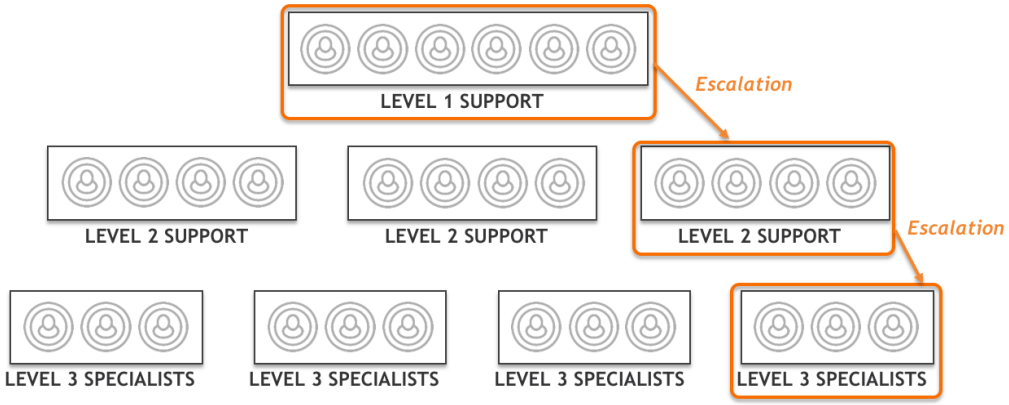


Рисунок. 1.2. Схема трирівневої підтримки

Існує 3 групи співробітників, для яких складається розклад (на даний момент, розклад складається менеджером вручну):

1) CC (Customer Care) або L1 (перша лінія підтримки)

Основне завдання першої лінії - підтримка контакту з клієнтом. Тобто в першу чергу це інструмент реалізації стратегії більш клієнт-орієнтованої підтримки, що особливо важливо на ринку з високою конкуренцією (клієнтський сервіс і його якість - це тема окремої розмови).

Перша лінія служить "єдиною точкою входу" для клієнтських питань і заявок. Без неї клієнтам по будь-якому питанню доводиться мати справу з "вузькими" спеціалістами. І, швидше за все, не задоволені цим взаємодією всі учасники. З одного боку, ключові співробітники постійно зайняті. З іншого - в стані перевантаження простими завданнями вони навряд чи коректно відповідають на запити, що не найкращим чином впливає на сприйняття

клієнтом рівня сервісу. Іншими словами, часу і грошей на зарплату співробітників витрачено багато, а результат - так собі.

Перша лінія бере на себе левову частку навантаження, пов'язану з простим спілкуванням, оптимізуючи витрати.

Перша лінія приймає заявку клієнта і класифікує її.

Коректність класифікації звернення залежить від наданої клієнтом інформації. Але сам він не знає, що важливо, а що ні. Завдання першої лінії - вивудити необхідні дані, задати додаткові питання, щоб уточнити класифікацію. Так знижується ймовірність вибору неправильного виконавця.

Фахівці першої лінії встановлюють якісні характеристики звернення: наприклад, виставляють пріоритети - це завдання досить складно вирішити автоматично з урахуванням всіх вхідних даних.

Найпростіші проблеми перша лінія може вирішити без передачі заявки фахівцям. Залежно від специфіки галузі і методів поділу підтримки на лінії, частка звернень, закритих на першій лінії, може досягати 80%. А при відсутності першої лінії не тільки найпростіші проблеми, а й нецільові звернення (з розряду "не туди потрапили") змушені вирішувати кваліфіковані співробітники - собівартість рішення таких проблем виходить досить високою.

2) SA (System Administration) або L2 (друга лінія підтримки)

Це більш поглиблений рівень технічної підтримки, який коштує дорожче, оскільки технічні спеціалісти повинні бути більш досвідченими, знайомими та обізнаними з конкретним продуктом або послугою. Як правило, це управлінський рівень або лінія підтримки, яка покладається на більш складні технічні та аналітичні методи (так званий "пошук і усунення несправностей") для вирішення проблем. Технічні експерти цієї лінії зазвичай відповідають не лише за допомогу працівникам служби підтримки у вирішенні базових технічних проблем, а й за аналіз проблеми, пошук та узагальнення досвіду у вирішенні більш складних проблем. [9] Якщо проблема є новою або якщо фахівець технічної підтримки на цьому рівні не може знайти потенційне рішення проблеми, проблему слід передати на третій рівень технічної

підтримки. Крім того, багато компаній можуть визначати зону відповідальності технічної підтримки, починаючи з другого рівня, наприклад, обмежуючи завдання певними сегментами послуг або обладнанням. [7] [10]

3) EX (Experts) або L3 (третій рівень підтримки)

Це найвищий рівень технічної підтримки в трирівневій моделі цієї діяльності, і фахівці цього рівня відповідають за вирішення найскладніших проблем. Як правило, фахівці цього рівня технічної підтримки відповідають не тільки за допомогу фахівцям попереднього рівня, а й за дослідження та розробку рішень нових, раніше невідомих проблем. [9] Крім того, проблеми можуть виходити за межі можливостей технічної підтримки. Наприклад, може виникнути проблема зі стороннім обладнанням, яке використовує компанія. У цьому випадку технічна підтримка третього рівня або спеціалізований відділ зв'язується з постачальником або ключовими розробниками для детального аналізу та розробки рішення. При розробці автоматизованої системи планування необхідно враховувати низку факторів.

По-перше, потрібно пам'ятати, що графік роботи може бути одним з мотивуючих або демотивуючих моментів для персоналу. Від задоволеності співробітників буде залежати і якість їх роботи.

По-друге, організація праці на підприємстві регулюється відповідними нормами законодавства, і її грамотна організація дозволить керівникам уникнути юридичних і економічних ризиків.

І, по-третє, клієнт-орієнтовані компанії повинні відштовхуватися від побажань своєї «цільової» аудиторії. Тому, щоб визначити потрібний графік роботи, потрібно врахувати частоту звернень в різний час доби, наявність періодів «затишшя», середню тривалість обробки поточних завдань, а також особисті дані співробітників (як мінімум, сімейний стан і вік).

При складанні схем не слід забувати і про самих співробітників (вікові категорії, сімейний стан, стать і т.д.).Базуючись на специфіці конкретного підприємства та враховуючи проблематику та задачі, складемо список первинних обов'язкових вимог до майбутньої системи:

1) система повинна бути кросплатформеною (дана умова є обов'язковою, щоб уникнути проблем з впровадженням і підтримкою системи на різних операційних системах);

2) повинна бути можливість визначати дані, необхідні для генерації розкладу;

3) система повинна дозволяти вносити зміни в отриманий розклад (можливість редагування збільшує оптимальність розкладу, а також дозволяє враховувати несподівано виникли умови);

4) розширюваність системи (можливість її доопрацювання в разі підвищення вимог до автоматизованої системі);

5) зручний інтерфейс користувача;

6) врахування побажань співробітників;

7) використання норм часу для розрахунку обсягу робочих годин.

Виконання всіх вищезазначених вимог у процесі розробки дозволяє створити універсальну систему, яка може бути використана будь-якою компанією для створення програми, близької до найкращого з можливих варіантів.

Оскільки правильно організований робочий процес (включаючи правильно складений розклад) є важливою складовою робочого місця, з середини минулого століття і до сьогоднішнього дня питання створення та автоматизації систем складання розкладів залишається актуальним з ряду причин (наприклад, системи навчання, що постійно розвиваються і змінюються, технологічний прогрес).

1.3. Аналіз підходів, методів та моделей вирішення завдань дослідження

Питання щодо способів створення оптимального, швидкого і якісного алгоритму складання розкладу є актуальним вже довгі роки, однак, суміжні з ним питання планування з'явилися ще раніше.

Як зазначено в джерелі [1], ще в 1784 році у Великобританії була здійснена успішна спроба складання розкладу відправлення маршрутних возів, що є одним з перших згадок про розклад в історії. Пізніше в 1840 році з'явився розклад паровозів, що включає в себе не тільки час відправлення, але також і прибуття.

Через 7 років в результаті договору між усіма паровозними компаніями Великобританії визначилося загальне для країни час за Гринвічем.

Однак стрімке прискорення життя людей та розвиток науки і техніки зробили процес складання розкладу та планування дуже складним через велику кількість залежностей та учасників. Тому на початку 20-го століття стало очевидним, що зростаючі проблеми в цій сфері потребують термінового вирішення.

Приблизно в 1903 році Генрі Лоуренс Гант, видатний менеджер, опублікував у своїй книзі метод побудови діаграми послідовності виконання робіт (календарний графік) [2]. Метод виявився настільки успішним, що Гант вирішив продовжити його розробку і присвятив йому решту свого життя (до 1919 року). Таким чином, Гант увійшов в історію як творець знаменитої і досі стандартизованої діаграми Ганта, названої на честь свого творця.

Саме з моменту створення діаграм Ганта бере свій початок теорія розкладів. Вона є одним з розділів так званого дослідження операцій (ДО). У джерелі [4] дається таке визначення ДО: дисципліна, що займається розробкою і застосуванням методів знаходження оптимальних рішень на основі математичного моделювання статистичного моделювання і різних евристичних

підходів в різних областях людської діяльності. З визначення ІС слід, що воно спрямоване на пошук і розробку кращих рішень, що є важливою частиною ТР.

Відповідно до джерела [5], теорія розкладів - це розділ дискретної математики, що займається проблемами впорядкування. Сам же термін "Теорія розкладів", як сукупність всіх теоретичних і практичних завдань, з'явився пізніше в 1956 році в роботі [6] американського математика Річарда Беллмана, що став відомим, завдяки своїм видатним заслугам в сфері математики і обчислювальної техніки. В одній з його книг [7] дається таке визначення ТР: "Теорія розкладів - це розділ дослідження операцій, в якому будуються і аналізуються математичні моделі календарного планування (тобто упорядкування в часі) різних цілеспрямованих дій з урахуванням цільової функції і різних обмежень ".

Говорячи про складність алгоритмів розв'язання задач ТР, варто відзначити статтю, випущену ще в 1965 році Едмондс [14], в якій він не тільки представляв алгоритм по знаходженню досконалого паросполучення в довільному графі, а й вказував на високу важливість відмінності завдань з точними поліноміальними алгоритмами від тих, для яких відомі лише експоненціальні алгоритми, підставою яких є методи неявного перебору. Важливим припущенням з'явилася неможливість побудови поліноміальних алгоритмів для більшості завдань, що важко вирішити.

Важливим фактором, що вплинув на визначення складності завдань ТР, є робота Кука [15], випущена в 1971 році, в якій він визначає клас NP, вводить поняття NP-повноти і вказує на те, що завдання здійсненності також є NP-повною. Відповідно до джерела [16], класом NP в теорії алгоритмів прийнято вважати безліч проблем можливості розв'язання, для перевірки рішення яких можна скористатися машиною Тюрінга за час, який не перевищує полінома від розміру вхідних даних, за умови наявності деякої додаткової інформації. А NP-повна задача, що належить класу NP, є завданням, що має відповідь 0 або 1, а знаходження алгоритму її рішення за поліноміальний час означає наявність можливості настільки ж швидкого вирішення будь-якої іншої задачі класу NP.

Пошуки доказів NP-складності завдань ТР велися аж до виходу статті Пападімітріоу і Янакакіса в 1991 році [18], в якій був нарешті представлений перший теоретичний підхід для вивчення обчислювальної складності в побудові наближених алгоритмів задач оптимізації, а також визначені класи цієї складності. Дані висновки збільшили інтерес дослідників до питання ТР в цілому, і до цього дня вчені всього світу знаходяться в пошуку відповідей.

Способи класифікації задач теорії розкладів

Розглянемо сучасні способи класифікації задач ТР, наведені в навчальному посібнику Лаврова А.А. і його колег [19]. Автори умовно виділяють 4 способи класифікації задач:

1) У розділі ТР:

- календарне планування або складання розкладу для приладів;
- мережеве планування або складання розкладу для проекту;
- побудова розкладів спортивних заходів;
- побудова розкладів руху і циклічні розкладу для транспортних засобів;
- доставка товарів у магазини;
- побудова тимчасових таблиць.

2) За типом шуканого рішення:

- завдання розподілу (знаходження оптимального вирішення питання розподілу наявних робіт по виконавцям);
- завдання узгодження (визначення необхідних параметрів, таких як час надходження, тривалість виконання і так далі);
- завдання упорядкування (побудова розкладу з точки зору знаходження оптимального порядку виконання робіт при вже наявному розподілі робіт за виконавцями, заздалегідь визначених параметрах робіт, наприклад, тривалість і так далі).

3) За способом визначення вхідної інформації:

- детерміновані завдання, вони ж "offline" (заздалегідь, тобто до початку виконання завдання, визначені всі необхідні вхідні параметри (дані));

- динамічні задачі, вони ж "online" (спочатку не відомі всі вхідні параметри, розклад будується в міру їх надходження, в будь-який момент має бути доступне побудоване розклад).

4) За типом цільової функції:

- завдання на побудову допустимого розкладу (складання розкладів, які відповідають поставлені умови, результат якого не завжди є оптимальним; оптимальність досягається додаванням функції штрафів);

- завдання з сумарними критеріями оптимізації (результат досягається шляхом мінімізації загального значення моментів завершення обслуговування робіт);

- завдання з мінімаксними критеріями оптимізації (результат досягається шляхом мінімізації максимального значення моментів завершення обслуговування робіт);

- багатокритеріальні задачі оптимізації (побудова оптимального розкладу з урахуванням наявності декількох цільових функцій, наприклад, мінімізація не одного, а декількох параметрів).

Існуючі способи подання розкладів

Грунтуючись на даних джерела [20], на сьогоднішній день існує 3 основних способи подання розкладів:

- графічне (прикладом можуть служити популярні донині діаграми Ганта (рисунку. 1.2));

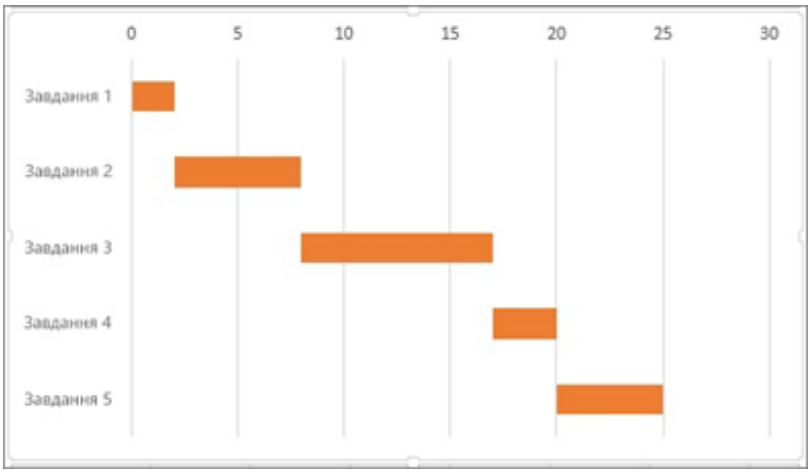


Рисунок. 1.3. Приклад діаграми Гантта

- табличне (рисунок. 1.3) (в підсумковій таблиці відображаються залежності між часом, виконавцями, роботами та інше);

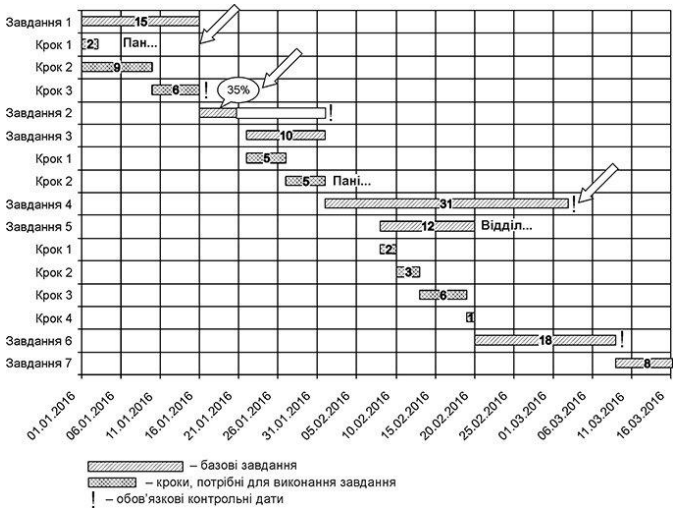


Рисунок. 1.4. Приклад табличного представлення завдань

- векторне (рисунок. 1.4) (основним завданням є відображення порядку виконання робіт).

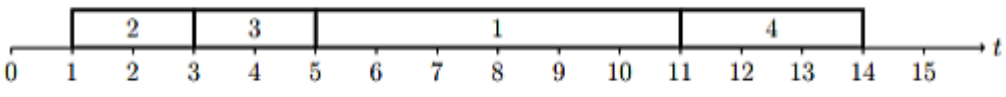


Рисунок. 1.5. Приклад векторного подання розкладів

Класифікація алгоритмів розв'язання задач ТР

У навчальному посібнику Лазарєва А.А, і Гафарова О.Р. [20] наводяться способи класифікації алгоритмів розв'язання задач ТР:

1) за трудомісткістю:

- поліноміальні (алгоритми, час виконання яких майже не залежить від розмірів вхідних даних; вважаються швидкими алгоритмами);

- псевдополіноміальние (алгоритми, які поводяться експоненціально при великих значеннях вхідних даних);

- експоненціальні (складність таких алгоритмів експоненціально залежить від довжини вхідних параметрів).

2) за точністю:

- точні (результатом роботи точного алгоритму завжди є оптимальне рішення; як правило, в цих алгоритмах застосовують різні способи скорочення перебору);

- наближені – часто застосовується назва "евристичні" – що є швидкими поліноміальними алгоритмами без гарантованої точності в рішенні (зазвичай це швидкі поліноміальні алгоритми, засновані на будь-якої здогаду про властивості оптимальних рішень);

- наближені з гарантованою оцінкою якості, оптимальність отриманих результатів роботи яких визначається за допомогою відхилення отриманого значення абсолютної або відносної похибки від оптимального (їх відмінність від алгоритмів з попереднього пункту полягає в тому, що можна оцінити абсолютну або відносну похибку, тобто відхилення отриманого значення цільової функції від оптимального);

- наближені з регульованою точністю (час роботи таких алгоритмів залежить від обраної користувачем необхідної точності; наприклад, нас може влаштувати відхилення від оптимуму, але лише на 1%).

Висновки до розділу 1

В ході написання даного розділу роботи була приведена загальна постановка задачі складання розкладу. Відзначено, що завдання складання розкладу відноситься до завдань цілочисельного програмування і для завдань даного класу характерна наявність великої кількості вимог, що важко формалізувати.

Також, були виконані наступні пункти:

1) проведений аналіз актуальності розроблюваної системи;

2) виділено ряд основних вимог до розроблюваної системі;

3) розглянута історія теорії розкладів;

4) розглянуті існуючі способи класифікації задач теорії розкладу;

6) визначено способи подання розкладів;

7) проведена класифікація алгоритмів розв'язання задач теорії розкладів.

На основі сформульованих і зібраних разом загальних вимог стало очевидно, що не існує єдиного і повного рішення для системи планування робочого часу. Тому було вирішено розробити власну систему планування часу.

Належним чином організований робочий процес (у тому числі правильно створений табель обліку робочого часу) є невід'ємною складовою робочого місця. Тому питання створення та автоматизації систем обліку робочого часу має велике значення з різних причин (наприклад, постійний розвиток і зміна систем навчання, технологічний прогрес тощо).

2 ТЕОРЕТИЧНЕ ТА МЕТОДИЧНЕ ДОСЛІДЖЕННЯ ВИРІШЕННЯ ЗАДАЧІ ФОРМУВАННЯ РОБОЧОГО РОЗКЛАДУ

2.1. Порівняльний аналіз існуючих методів вирішення завдань складання розкладу

Складання розкладу є невід'ємною частиною будь-якої виробничої діяльності. Тому дуже важливо скласти розклад, що дозволяє ефективно розподілити наявні ресурси. Для вирішення даної проблеми була сформульована теорія розкладів, що дозволяє класифікувати поставлені завдання і алгоритми для їх вирішення.

"Часовий" характер завдань теорії розкладів виділяє їх в особливий клас, що істотно відрізняється від економічних задач об'ємного планування. Якщо в останніх потрібно відповісти на питання, що і в якому обсязі виробляти на кожному з можливих видів ресурсів, то в задачах теорії розкладів необхідно визначити, коли почати і коли завершити, а також в якій послідовності виконувати ці роботи. Ця різниця в суті завдань визначає також і відмінність у використовуваних математичних методах і оцінці ефективності їх вирішення. У той час як для задач об'ємного планування досить успішно використовується апарат математичного програмування, що дозволяє отримувати оптимальні і достатні ефективні рішення практичних завдань великого розміру, то для задач теорії розкладів використання методів математичного програмування не дозволяє отримати ефективні рішення задач, пов'язаних з реальними практичними додатками.

Для вирішення завдань теорії розкладів в даний час найбільшого поширення набули такі підходи і методи:

1) метод випадкового пошуку

2) методи математичного цілочисельного і булевого програмування;

3) метод теорії графів;

4) методи глобального випадкового пошуку і локальних варіацій, перестановки стратегії;

5) генетичні алгоритми і еволюційні стратегії;

6) евристичні методи.

Розглянемо кожен з перерахованих вище методів більш докладно.

Випадковий і вичерпний пошук

Ефективність алгоритмів випадкового пошуку зазвичай залежить від співвідношення кількості знайдених розв'язків до кількості так званих "хороших розв'язків". У більшості випадків це співвідношення невелике, оскільки більшість задач мають дуже жорсткі обмеження на якість розв'язків. Тому випадковий пошук хорошої програми схожий на пошук голки в стозі сіна. Класичні методи евристичного пошуку працюють дуже ефективно для задач невеликого розміру, але зі збільшенням простору пошуку час роботи таких алгоритмів значно зростає. Класичний евристичний пошук часто порівнюють з роботою людей-експертів. З цієї ж причини можна очікувати, що локальні мінімуми будуть знайдені відносно швидко. В роботах [11, 12, 13] показано застосування високоефективних алгоритмів вичерпного пошуку до задач розкладів, але ці алгоритми стають непридатними при збільшенні розміру задачі. Математичною основою методів випадкового пошуку служить ітераційний процес:

$$x_{i+1} = x_i + \Delta x_{i+1}, \tag{2.1}$$

Ненаправлений випадковий пошук (метод Монте-Карло) полягає в багаторазовому випадковому виборі допустимих варіантів рішень і запам'ятовуванні найкращого з них.

Всі алгоритми спрямованого випадкового пошуку без самонавчання роблять крок від поточного значення А оптимізуються параметрів. Відомі такі алгоритми спрямованого випадкового пошуку без самонавчання [14]: алгоритм з парної пробою, алгоритм з поверненням після невдалої спроби здійснити кроці, алгоритм з перерахунком при невдалому кроці, алгоритм з лінійної

екстраполяцією, алгоритм найкращою проби, алгоритм статистичного градієнта.

Алгоритми випадкового спрямованого пошуку з самонавчанням полягають в перебудові імовірнісних характеристик пошуку, тобто в певному цілеспрямованому впливі на випадковий вектор. Він вже перестає бути рівно імовірнісним і в результаті самонавчання набуває певну перевагу в напрямках найкращих кроків. Це досягається введенням вектора пам'яті *P*. Алгоритм рекурентно коригує значення компонентів цього вектора на кожній ітерації в залежності від того, наскільки вдалим/невдалим (змінилося значення цільової функції) був зроблений крок. Опис і аналіз різних способів корекції вектора *P* наведені в [14].

Дня алгоритмів випадкового пошуку можна виділити наступні особливості:

1) слабка адаптація (або її відсутність) до поведінки оптимізується функції для алгоритмів ненаправленого і спрямованого випадкового пошуку без самонавчання;

2) низька швидкість збіжності, зменшується зі зростанням кількості оптимізуються параметрів.

Огляд відомих на даний момент методів пошуку, а також результати деяких нових алгоритмів наведені в роботах [15, 16, 17].

Методи цілочисельного програмування

Завдання цілочисельного програмування зводиться до виділення змінних, значення яких необхідно знайти, складання математичної моделі задачі у вигляді обмежень, які описують задачу і накладають певні обмеження на шукані змінні, і складання цільової функції.

Застосування алгоритму:

1) виділення змінних;

2) складання математичної моделі (виділення обмежень для змінних);

3) складання цільової функції;

4) знаходження максимуму (мінімуму) цільової функції за допомогою математичних методів.

Основні недоліки:

- експоненціальне збільшення витрат часу на пошук кращого (прийнятного) рішення з ростом розмірності розв'язуваної задачі;

- відсутність гарантії отримання прийнятного рішення;

- в силу великої розмірності математичної моделі складно оцінити вплив різних чинників процес вирішення завдання і його результат;

- складність обліку переваг.

Метод теорії графів

У цьому випадку будується неорієнтовані граф, в якому кожна вершина являє собою заплановане навчальним планом заняття. Якщо між якимись двома вершинами можливі конфлікти, то вони з'єднуються ребром. Це еквівалентно забороні одночасного проведення занять. Тоді задача зводиться до розфарбуванні графа в задану кількість квітів.

Застосування алгоритму:

1) виділення безлічі занять в навчальному плані;

2) уявлення кожного заняття у вигляді вершини графа;

3) з'єднання вершин графа ребрами в разі неможливості одночасного проведення занять;

4) рішення задачі розмальовки графа в задану кількість квітів.

Основні недоліки:

- мала ефективність при застосуванні точних методів для розмальовки графів великої розмірності;

- відсутність можливості обліку переваг.

переваги:

- проста в розумінні математична модель;

- застосування графа разом з наближеними методами може давати хороші результати.

Методи глобального випадкового пошуку і локальних варіацій, перестановки стратегії

На кожній ітерації алгоритму метод глобального випадкового пошуку генерує розклад завдань на основі застосування заданого вектора випадкових чисел із заданим законом розподілу. Серед згенерованих таким чином розкладів відбираються тільки ті, які надійно виконують всю систему обмежень задачі. Серед цих розкладів вибирається рішення з найкращим значенням критерію ефективності.

У методі локального випадкового пошуку рішення, отримане другим алгоритмом, піддається різним локальним модифікаціям (наприклад, заміна двох сусідніх робіт під час виконання, використання різних типів ресурсів для виконання даної роботи тощо) для отримання нового допустимого рішення з кращими показниками ефективності.

Генетичні алгоритми та еволюційні стратегії

Генетичні алгоритми є частиною більш загальної групи методів, які називаються еволюційними обчисленнями, що поєднують різні варіації з використанням еволюційних принципів для досягнення певної мети.

Також в ній виділяють наступні напрямки:

1) Еволюційні стратегії - метод оптимізації, заснований на ідеях адаптації та еволюції. Ступінь мутації в даному випадку змінюється з часом - це призводить до, так званої, самоадаптації.

2) Генетичне програмування - застосування еволюційного підходу до популяції програм.

3) Еволюційний програмування - було вперше запропоновано Л.Дж. Фогелем в 1960 році для моделювання еволюції як процесу навчання з метою створення штучного інтелекту. Він використовував кінцеві автомати, що пророкують символи в цифрових послідовностей, які, еволюціонуючи, ставали більш пристосованими до вирішення поставленого завдання.

У генетичному алгоритмі можна виділити такі компоненти, як хромосома (рішення задачі), що складається із сукупності генів (параметрів рішення), початкова популяція хромосом, оператори генерації нового покоління: кросовер (процес обміну генами особинами в випадкової точки розриву) і мутація (інверсія випадкового гена), фітнес-функція (цільова функція, що дозволяє зробити оцінку пристосованості отриманого рішення).

Генетичний алгоритм є ітераційним процесом, що продовжується до моменту досягнення точки зупину. В якості точки зупинки можна вибрати кількість поколінь (ітерацій). У кожному поколінні алгоритму виконується селекція (відбір), кросовер і мутація. Загальна схема роботи генетичного алгоритму представлена на рисунку. 2.1.

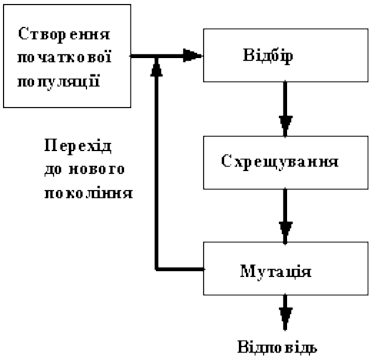


Рисунок. 2.1. Загальна схема роботи генетичного алгоритму

Основними перевага ГА є те, що вони дуже прості в реалізації, а також можуть бути застосовані практично до будь-якої проблеми оптимізації. Через загальний бінарній стиль кодування практично будь-який тип даних може зберігатися в індивідуальному середовищі, а потім бути оптимізованим за допомогою евристичної системи ГА.

Важливою перевагою генетичного алгоритму є можливість застосування його на розкладах великої розмірності.

Генетичні алгоритми оперують сукупністю розкладів. Це відрізняє ГА від більшості інших алгоритмів оптимізації, які оперують лише з одним розкладом, покращуючи його. [22, 23]

До недоліків можна віднести складність роботи з даними складної структури (з урахуванням багатьох властивостей об'єктів) та їх взаємозв'язків, а також можливість передчасного завершення роботи алгоритму до досягнення оптимальної програми (якщо точки зупинки обрані через недостатню кількість ітерацій або через недостатнє хромосомне різноманіття в популяції).

Також, є і деякі недоліки у використанні двійкового кодування. Наприклад, якщо реальні числа використовуються в якості атрибутів, вони стають дискретизованими, і через нелінійне поводження стандартного двійкового кодування простір пошуку стає порушеним і міцним.

Якщо ми розглянемо цільову функцію:

$$f(x) = -\frac{(x-16)^2}{16} + 16, \qquad (2.1)$$

яка повинна бути максимізована, і двійкове кодування із l=5 біт використовується для атрибута єдиного дійсного числа x, тоді виникає досить різка дискретизація ($\Delta=1$), і сусідні відносини між можливими рішеннями порушуються. Якщо використовується відстань Хеммінга і вимірюється відстань оптимального значення атрибута $opt=16$ до всіх інших можливих значень, ми бачимо, що реальне відстань і відстань Хеммінга не відповідають одне одному. Особливо крок від $x=15$ до $opt=16$, хоча і малий в матеріальному просторі пошуку, є максимальним в двійковій-кодованому просторі пошуку. Таким чином, унімодальне простір пошуку може стати полімодальний тільки через використовуваного стилю кодування. З іншого боку, простір пошуку складається таким чином, що раніше різні значення атрибутів стають ближче в довічним кодованому просторі пошуку.

Один із способів подолання цієї проблеми - використовувати інший стиль кодування, наприклад, Gray Coding.

Інша можливість - опустити кодування і перейти від орієнтованих на генотип операторів ГА до операторів, орієнтованим на фенотип. Це робиться у випадку з еволюційними стратегіями.

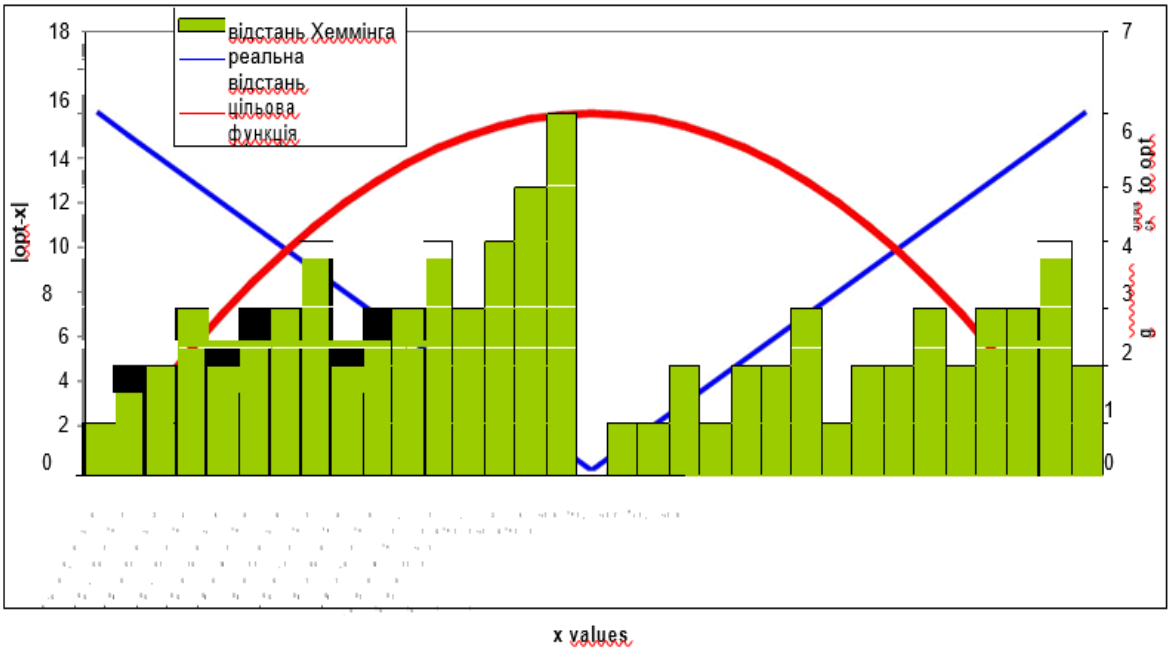


Рисунок. 2.2. Спотворення простору пошуку, викликане кодуванням ГА

Еволюційні стратегії - ще один з методів евристичного пошуку, працює за принципами біологічної адаптації і еволюції. Різниця між генетичними алгоритмами та еволюційними стратегіями полягає в тому, що перші використовують ланцюжки генів як елементи популяції, тоді як другі використовують дійсні числа як розв'язок задачі. Іншими словами, на відміну від стандартних генетичних алгоритмів, еволюційні стратегії виконують ту саму операцію кросинговеру, використовуючи вектор дійсних чисел. Мутація виконується шляхом додавання нормально розподілених значень до кожного компонента вектора. Ці два аспекти дуже близькі один до одного і можуть бути нерозрізненими.

Еволюційні стратегії застосовуються в чисельній оптимізації, дозволяючи ефективно вирішувати завдання складної нелінійної оптимізації з обмеженнями.

Схожість між ГА та ЕС очевидна, а відмінності не видаються суттєвими. Тому необхідно обговорити, які характеристики цих алгоритмів слід враховувати при виборі процедури оптимізації. З експериментальних досліджень можна зробити висновок, що поведінка алгоритмів значною мірою

залежить від модальності цільової функції та зв'язності допустимої множини. Зокрема, алгоритми, що реалізують ЕС, є більш привабливими для унімодальних функцій і зв'язних допустимих множин, тоді як ГА працюють краще для мультимодальних функцій і незв'язних допустимих множин. Це можна пояснити представленням об'єктних змінних, з якими працює алгоритм: Оскільки ЕС оперує на рівні фенотипу, тобто перетворює самі об'єктні змінні, йому дуже важко заповнити "прогалини" в допустимому просторі, а також важко увійти в індуктивний простір іншого локального мінімуму. Зрозуміло, що це не так. Це пов'язано з тим, що невелика мутація в бітовій послідовності означає дуже різкий стрибок в області толерантності на фенотипічному рівні. Тому в простому випадку не очікується, що ГА буде дуже ефективним.

Вибір методу рішення наукового завдання

Реальні практичні задачі теорії розкладів належать до класу NP-повних задач. На практиці важко передбачити потреби і алгоритми, які точно розв'язують NP-повні задачі, вимагають, в гіршому випадку, експоненціального часу для отримання точного розв'язку. Тому алгоритм можна застосовувати на практиці лише для отримання точних розв'язків для задач практично невеликого розміру.

Враховуючи великий розмір практичних задач, жоден з описаних вище підходів не гарантує точного розв'язку за розумний час. Це призвело до розробки методів наближеного розв'язання, які дозволяють отримати прийнятні рішення за відносно короткий час і за відносно низьку вартість. Наближені методи розв'язання включають глобальний випадковий пошук, локальну варіацію, перестановку стратегій, генетичні алгоритми та еволюційні стратегії, а також евристики.

У генетичних алгоритмах та еволюційних стратегіях, на відміну від методів випадкового пошуку, генерується і використовується не одне, а декілька допустимих рішень одночасно. Ітеративно, на основі комбінації найкращих характеристик цих кількох рішень, створюються нові, більш

ефективні варіанти розкладу, які замінюють отримані на попередній ітерації алгоритму і використовуються на наступному етапі розв'язання задачі.

Евристичного алгоритму, який би однаково добре працював для будь-яких вхідних значень для вирішення задачі складання розкладу, поки що не знайдено. Для підвищення ефективності алгоритм може бути модифікований з урахуванням особливостей складання розкладу та вимог навчального закладу.

Враховуючи конкретну задачу складання розкладу для відділу технічної підтримки кількістю 25 чоловік (що є малою розмірністю) та зваживши ресурси, необхідні для реалізації алгоритму, з точністю та якістю отримуваного розкладу – в якості методу складання розкладу пропонується використовувати алгоритм еволюційних стратегій, який реалізує всі переваги класичного генетичного алгоритму і позбавлений деяких його недоліків.

2.2. Еволюційні стратегії

Еволюційні стратегії були розроблені в шістдесятих роках Рекенбергом і Швефелем в Берліні [SH 75]. Вони були задумані як евристика пошуку для задач оптимізації в галузі інженерії, і вони були спеціалізовані для оптимізації дійсних чисел в якості атрибутів рішення. На відміну від ГА і ГП, ЕС були розроблені з самого початку як практичний метод оптимізації. Основна відмінність від ГА полягає в тому, що метод ЕС підкреслює фенотип особини і не вводить жодного додаткового кодування. Через це методи кросовера і мутації повинні змінювати реальне значення атрибутів, а не абстрактне кодоване подання.

Цей підхід на основі фенотипів дозволяє оптимізувати оператор для подальшої оптимізації. Зокрема, для атрибутів з дійсними значеннями запропоновано оператор мутації, який може автоматично приймати розмір і напрямок мутацій відповідно до локальної топології простору пошуку. Параметри, що визначають атрибути мутації, називаються параметрами стратегії, а атрибути рішення - параметрами рішення; ЕС оптимізує і ті, і інші.

Терміни ГА та ЕС

Ген – реально існуюча, незалежна одиниця спадковості, що комбінується й розщеплюється при схрещуваннях

Хромосома – структурний елемент клітинного ядра біологічних організмів, який є носієм генів у клітинному ядрі особини.

Популяція – досить велике співтовариство організмів, що схрещуються між собою.

Жорсткі обмеження – це обмеження, які повинні неодмінно задовольнятися; такі які фізично не можуть бути порушені.

М’які обмеження – це обмеження, які можна порушувати, але це порушення повинно бути зведене до мінімуму. Їхнє виконання не є таким же обов’язковим, як жорстких.

Селекція – це вибір тих хромосом, які будуть брати участь в створенні нащадків для наступної популяції, тобто для чергового покоління.

Схрещування – генетичний оператор, що відповідає за передачу ознак від батьків до потомків.

Мутація – генетичний оператор, що відповідає за зміну генів особини в допустимих значеннях для покращення його цільової функції.

Фітнес-функція – це функція оцінки генетичного алгоритму, що визначає міру пристосованості отриманого рішення.

Особина в ЕС

Оскільки ЕС спеціалізується на дійсних числах як атрибути рішення, особина ЕС полягає в більшості випадків вектора *n* дійсних чисел, параметри рішення, які повинні бути оптимізовані:

$$a_{i,d} = (x_1, x_2, \dots, x_n), \tag{2.2}$$

де x_i – дійсні числа.

У простій реалізації ЕС параметри стратегії зберігаються в додатковому векторі дійсних чисел *n*:

$$a_{i,s} = (\sigma_1, \sigma_2, \dots, \sigma_n), \tag{2.3}$$

де σ_i – позитивні дійсні числа.

У цьому простому випадку ми використовуємо один параметр стратегії σ_i для кожного параметра рішення x_i. σ_i призначає стандартне відхилення мутації фенотипу для кожного параметра рішення, що дорівнює середньому розміру етапу мутації.

Якщо кодування опущено, цільову функцію можна оцінити без подальших обчислень:

$$\Phi(a_i) = F\left(a_{i,d}\right) \tag{2.4}$$

Селекція в ЕС

У більшості ЕС процедура детермінованого відбору використовується для вибору можливих батьків для наступного покоління. Тільки λ найкращих особин обрані з популяції *A(s)* розмірністю μ і тільки ці λ кращих особин використовуються для батьківської популяції *B(S)*. Цей підхід називається (μ,λ) стратегія.

Якщо λ кращих індивідуумів розглядається як еліта, яка переходить в наступне покоління без змін, стратегія називається $(\mu+\lambda)$ стратегія.

Кросовер в ЕС

ЕС використовує два різних методи кросовера для стратегій і параметрів рішення, щоб змішувати двох обраних батьків (*e1* та *e2*) з *B(s)*:

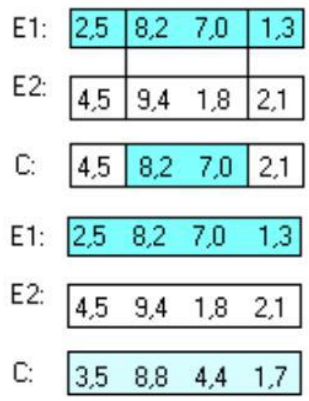


Рисунок. 2.3. Оператор кросовера в ЕС

Дискретний кросовер використовується для параметрів прийняття рішення: якщо ми використовуємо дискретний кросовер, нащадок с або успадковує атрибут x_i від одного або другого з батьків. Фактичний батько, який поставляє атрибут, може бути обраний довільно.

Межсегментний кросовер використовується для параметрів стратегії: проміжний кросовер привласнює нащадку *c* значення σ_i, що є середнім для відповідного σ_i обох батьків.

Мутація в ЕС

Для мутації параметрів стратегій і рішення можуть використовуватися два різних методи:

Спочатку параметри стратегії мутують відповідно до:

$$\sigma'_j = \sigma'_j \times \exp(\tau_1 \times N(0,1) + \tau_1 \times N_j(0,1), \quad (2.5)$$

N(0,1) - рівномірно розподілене випадкове число в діапазоні [0,1], яке генерується для кожної мутації на одну особину. $N_j(0,1)$ також рівномірно розподілене випадкове число в діапазоні [0,1], це число генерується для кожного σ_i. τ_1 і τ_2 є параметрами екзогенної стратегії.

Новий розмір кроку мутації σ'_i використовується для зміни параметрів рішення x_i:

$$x'_j = x_j \times +\sigma'_j \times N'_j(0,1), \quad (2.6)$$

$N'_j(0,1)$ є гаусовим розподіленим випадковим числом, породженим для кожного x_i.

Нащадки утворюють наступне покоління *A(s+1)* після проходження цих процесів зміни. Процес поколінь може повторюватися знову і знову до тих пір, поки не буде знайдено задовільне рішення.

2.3. Опис задачі оптимізації в термінах алгоритму еволюційних стратегій

В рамках дипломної роботи була поставлена наступна задача: необхідно скласти розклад робочих змін для співробітників відділу технічної підтримки з режимом роботи 2/2 (2 робочих дня, 2 вихідних дня) і 12-годинним графіком роботи. Необхідно знайти оптимальний розклад для заданої кількості співробітників.

Для вирішення поставленого завдання основні поняття генетичного алгоритму були переформульовані наступним чином:

1) Під популяцією розуміється безліч розкладів.

2) Особина - конкретний розклад. Всі розклади є допустимими, але не кожен може бути оптимальним. Оптимальним, як правило, він стає під час виконання алгоритму.

3) Під хромосомами розуміються ресурси використовуваного розкладу, тобто робочі зміни/співробітники.

4) Гени являють собою конкретного співробітника і час.

Елементи розкладу

Робочий розклад колективу, що працює позмінно, складається з таких об’єктів:

1. Множина груп робітників, елементами якої є, власне, робітник.

$$G = \{g_1, \dots, g_m\}, \tag{2.7}$$

де – g_i це робітник;

m – кількість робітників у робочому колективі.

Кожен елемент цієї множини складається з поля:

- ім’я працівника;
- стать працівника;
- посада, яку займає працівник.

$$g_i = \{n_1\}, \tag{2.8}$$

де n_1 – ім’я працівника.

2. Двомірний масив, у вигляді якого буде представлено робочий розклад. Нульовий стовпець матриці – працівники, для яких формується розклад. Нульовий рядок – дата робочої зміни, що буде представлено у вигляді «число – день тижня».

Таблиця 2.1 – Представлення розкладу занять

		Місяць/Рік							
Ім'я робітника	Посада	Число/ День	2 Fri	3 Sat	4 Sun	5 Mon	6 Tue	7 Wed	8 Thu
	Група 1	Тайм-слот (час)							
	Група 2								
	Група 3								
	...								

Обмеження

Проблема процесу складання розкладу полягає в тому, що розклад повинен відповідати ряду обмежень. Усі обмеження поділяють на так звані м’які та жорсткі.

Жорсткі – це обмеження, які повинні неодмінно задовольнятися; такі які фізично не можуть бути порушені (наприклад один і той самий викладач не може бути присутній в один і той самий час у двох місцях одночасно).

М’які – це обмеження, які можна порушувати, але це порушення повинно бути зведене до мінімуму. Їхнє виконання не є таким же обов’язковим, як жорстких.[4]

Жорсткі показники формалізуються у вигляді обмежень, а "бажані" - у вигляді критеріїв оптимальності. Тоді задача планування розкладу ставиться як багатокритеріальна задача (облік "бажаних" показників) з обмеженнями (облік "жорстких" показників), які в даній ідеалізації є лінійними.

Наведемо жорсткі обмеження, що мають бути враховані при складанні розкладу:

1. Мінімальна кількість людей на робочій зміні:
    a. вночі – 2;
    b. вдень – 3.
2. Співробітник повинен мати не менше 1 дня відпочинку.
3. Співробітник не повинен працювати:
    a. більше 3 днів поспіль
    b. менше 1 дня поспіль
4. Робочих часів у місяці повинно бути:
    a. не менше 166;
    b. не більше 192;
5. Не можна ставити денну та нічну зміни поспіль;
6. На денній зміні повинно бути більше людей, ніж на вечірній;
7. У будні людей на змінах повинно бути більше, аніж на вихідних;
8. На зміні повинні бути хоча б працівники з груп L1 та (L2 або L3);
9. На зміні обов’язково повинні бути присутні співробітники (тобто не можна залишити тайм-слот порожнім).

Наступним етапом формалізації є аналіз бажаних властивостей розкладу (м'яких обмежень):

1. На зміні повинен бути працівник з кожної групи (L1/L2/L3);
2. Жінки не повинні працювати на ночних змінах;
3. Співробітник має працювати рівно 168 годин у місяці;
4. Якщо співробітник одружений/одружена, має сім'ю – не повинно бути нічних змін;
5. Режим роботи 2/2 (2 дні робота/2 дні відпочинок);
6. Денні зміни чергуються із нічними (тобто [2 дні денна]-[2 дні вихідні]-[2 дні нічна]);
7. День народження співробітника – вихідний день.

Фітнес-функція як критерій якості розкладу

Для завершення постановки задачі необхідно розробити критерій, за допомогою якого можна оцінити якість складеного розкладу. Кожне з перерахованих м'яких обмежень формує свій критерій якості розкладу, який оцінює ступінь порушення відповідного обмеження. Наявність безлічі критеріїв дозволяє віднести завдання складання розкладу занять до класу багатокритеріальних задач. Критерії якості при цьому суперечать один одному: бажання отримати розклад з найкращим значення одного з критеріїв призводить до погіршення значень інших критеріїв (одного або декількох).

Для задачі складання і оптимізації розкладу занять було розроблено цільову фітнес-функцію, що враховуватиме м’які обмеження на розклад занять. За невиконання м’якого обмеження на розклад накладається штраф. Кожне обмеження має свій штрафний коефіцієнт, а також у кожного обмеження існує свій ступінь невиконуваності для розкладу. Мною обрано за ступінь не виконуваності кількість порушень і-го обмеження у розкладі.

Штраф для одного обмеження обчислюється так:

$$P_i = n_i w_i, \tag{2.9}$$

де n_i – це ступінь невиконуваності і-го обмеження; w_i – штрафний коефіцієнт для і-го обмеження; P_i – штраф за невиконання і-го обмеження

$$F(H_j) = \frac{1}{1+\sum_{i=1}^{m} P_i} \to max, \tag{2.10}$$

H_j – це досліджувана особина; m – кількість м’яких обмежень для розкладу занять; P_i – штраф за невиконання і-го обмеження.

Отже, чим менший штраф за невиконання м’яких обмежень, тим більшим буде значення цільової функції, а отже тим кращою буде особина.

Якщо цільова функція << 1, то це означає, що не виконалось якесь із м’яких обмежень, тому досліджувана особина є непридатною і підлягає покращенню своїх характеристик.

Цільова функція застосовується до кожної особини популяції і спрямована на знаходження максимального значення.[4]

Ідеальним вважається випадок, коли кожен з приватних критеріїв приймає значення одиниці.

2.4. Алгоритм еволюційних стратегій

Загальний вигляд алгоритму еволюційних стратегій такий:

1. Створення початкової популяції
   1.1. Ініціалізація хромосоми кожної особини
   1.2. Оцінювання початкової популяції
2. Етап еволюції - побудова нового покоління
   2.1. Відбір кандидатів на схрещування (селекція)
   2.2 Схрещування, тобто породження кожною парою відібраних кандидатів нових індивідів
   2.3. Мутація
   2.4. Фільтрація хромосом, що представляють собою неможливі рішення
   2.5. Оцінювання нової популяції.
3. Перевірка критерію зупину алгоритму

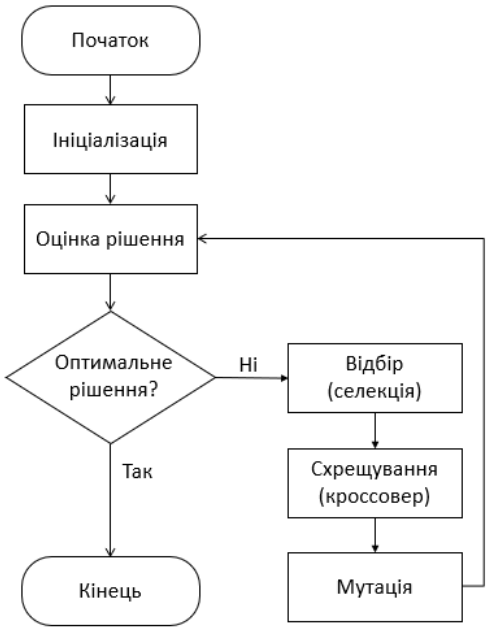


Рисунок. 2.4. Схема роботи алгоритму еволюційних стратегій

Вхідна інформація

Ген – це найменша одиниця розкладу. Кожен ген представляє собою окремого співробітника та час роботи.

Хромосома – це набір генів. У нашому випадку цим набором являються зміни в розкладі. Хромосоми представляють у вигляді масиву даних, кожний елемент якого – це одна зміна в розкладі.

Особина – це набір хромосом, ним буде являтись сам робочий розклад, що буде представлено у вигляді двомірного масиву хромосом.

Популяція – це сукупність особин, що схрещуються між собою. В нашому випадку це буде набір сформованих розкладів.[3]

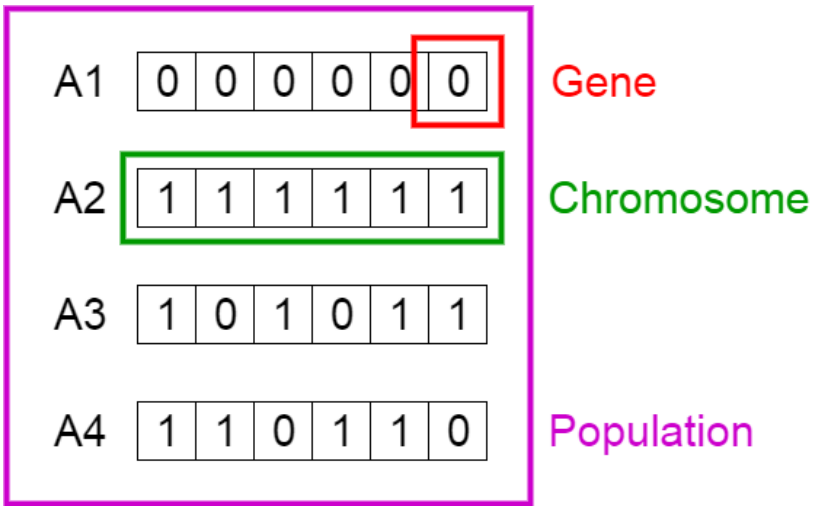


Рисунок. 2.5. Загальне представлення елементів генетичного алгоритму

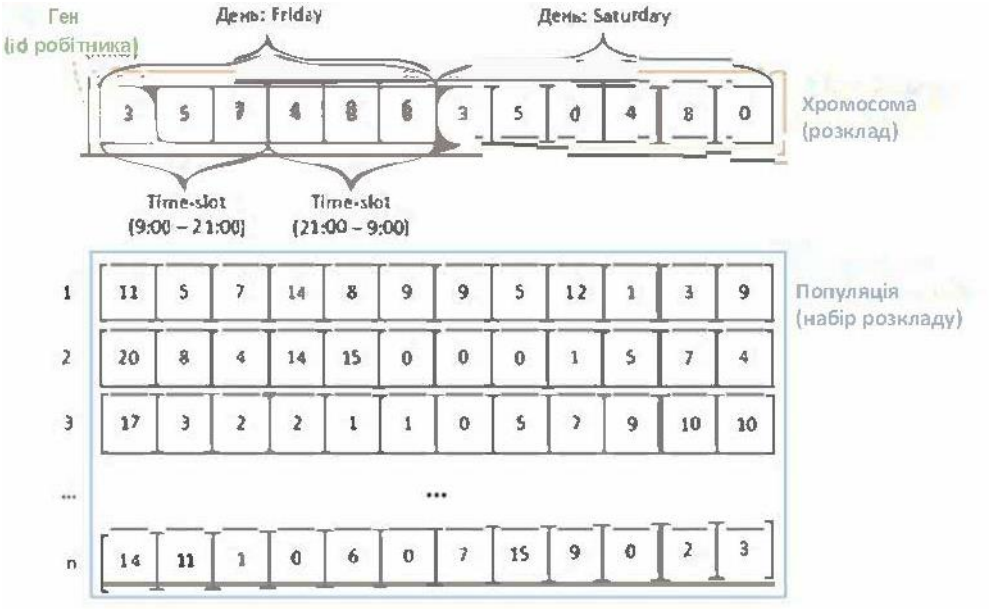


Рисунок. 2.6. Формат хромосоми для задачі складання розкладу

Етапи алгоритму

Розглянемо етапи алгоритму еволюційних стратегій більш докладно.

1) Формування початкової популяції

Спочатку генерується випадкова початкова популяція. Навіть якщо ця популяція абсолютно неконкурентоспроможна, генетичні алгоритми можуть досить швидко перетворити її на життєздатну популяцію. Тому на цьому етапі немає необхідності занадто збільшувати пристосованість особин, а лише в тій мірі, в якій індивідуальна форма популяції є придатною і для неї можна обчислити функцію пристосованості.

На цьому етапі довільно створюється початкова популяція з заданої кількості N особин. Кожна особина в популяції представляє незалежну версію програми (розв'язку задачі).

2) Селекція особин для нової популяції

Виконується фільтрація, і особини з невідповідними значеннями фітнес-функції ідентифікуються та видаляються. Невідповідні особини видаляються до тих пір, поки розмір популяції не стане таким же, як на першому етапі.

3) Розмноження особин довільними значеннями функції придатності

4) Мутації над отриманим потомством

Оператор мутації застосовується до особин, отриманих після третього етапу, щоб урізноманітнити популяцію і розширити простір пошуку оптимального рішення.

5) Відбір особин

На даному етапі виконується відбір особин, які мають більш підходящі значення функції придатності порівняно з іншими особинами.

6) Фітнес-функція

Хромосомна (індивідуальна) детермінація оцінюється значенням цільової функції. Цільову функцію часто називають функцією пристосованості або фітнес-функцією. Значення цієї функції оцінюється індивідуально для кожної особини в популяції і на цій основі приймається рішення, чи використовувати цю хромосому, чи перейти до етапу поліпшення особин в популяції за допомогою генетичних операторів гібридизації або мутації.

7) Тестування умови зупинки алгоритму

Покоління, отримані після кроків 2-5, замінюють початкову батьківську популяцію. Наступним кроком є перевірка умови зупинки алгоритму. Якщо приріст функції пристосованості "найкращої" особини є незначним протягом декількох поколінь, алгоритм припиняє роботу. Якщо умови зупинки виконуються, алгоритм переходить до наступного етапу, інакше - до етапу відбору, де процес пошуку оптимального рішення продовжується.

8) Вибір оптимальної особини.

Після виконання всіх попередніх етапів вибирається найкраща особина, яка є рішенням задачі. Краща особина - особина з мінімальним значенням функції придатності.

Висновки до розділу 2

В процесі написання цього розділу були виконані наступні пункти:

1) Проаналізовано алгоритми розв'язання задач теорії розкладів;

2) Обрано та проаналізовано один з алгоритмів складання розкладу через детальне обговорення алгоритму та створення блок-схеми;

Крім того, було детально описано об'єкт оптимізації - розклад занять. Визначено, з яких елементів він складається і як ці елементи об'єднуються для формування структури. Також були визначені обмеження, що накладаються на розклад. Тобто обмеження, які є необхідними для успішного складання розкладу та бажаними для наближення розкладу до оптимального.

Також детально описано алгоритм еволюційної стратегії, яка використовується для оптимізації розкладу. Описано постановку задачі оптимізації та визначено її основні етапи (формування початкової популяції, відбір, спарювання, мутації та вибір найкращого варіанту).

3 ТЕХНІЧНІ РІШЕННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМУ

У якості практичної задачі в даній дипломної роботи було прийнято рішення розробити веб-додаток для побудови розкладу «Shedule Generator». При побудові розкладу враховуються всі жорсткі ораніченія, а також максимальну кількість м'яких.

3.1. Опис технологій і методологій

В ході інтеграції модуля застосовувалися такі підходи і технології:

1) Frontend - Клієнтська частина (інтерфейс користувача), який використовує технології HTML 5, CSS, JavaScript, AJAX);

2) Backend - Серверна частина (безпосередньо алгоритм еволюційної стратегії та логіка роботи програми), який використовує технології Java 8, Struts-2 framework, Servlets.

3) Для зберігання призначений для користувача даних, допоміжної службової інформації використовувався сервер mysql-5.7 (https://dev.mysql.com/). Додаток було розгорнуто на локальному корпоративному веб-сервері Apache Tomcat-8 (http://tomcat.apache.org/).

Архітектура додатку

Додаток було побудовано на остові архітектури Struts 2, що є шаблоном дизайну MVC.

Struts 2 – це платформа з відкритим кодом для побудови веб-додатків, яка інтегрується з технологіями J2EE, такими як Java Servlets, JavaServer Pages та JavaBeans. Struts2 спирається на шаблон проектування Model-View-Controller (MVC).

Шаблон MVC широко визнаний, як один із найкращих та зрілих шаблонів проектування, які використовуються в даний час. При використанні шаблону MVC, обробка розбивається на три різні частини, а саме на контролер (Controller), модель (Model) і компоненти уявлення (View).

1) Model

Компонент моделі представляє "модель" бізнес-логіки та даних додатку Struts2. Наприклад, додаток Struts2, який керує даними клієнтів, може отримати вигоду від створення компонента моделі "Клієнт", який надає доступ до інформації про клієнтів для програми.

У багатьох випадках компонент моделі надає інтерфейс до бази даних або внутрішньої системи. Наприклад, якщо додатку Struts2 потрібно отримати доступ до даних про співробітників, що зберігаються в системі управління персоналом, слід розробити модель "Співробітник", яка слугуватиме інтерфейсом між додатком Struts2 і системою управління персоналом.

Компонентом моделі зазвичай є стандартний клас Java. Модель не вимагає спеціального форматування, тому можна використовувати код, написаний для інших проектів.

2) View

Компонент подання - це частина програми, яка відповідає за відображення інформації або отримання інформації від користувача. У Struts 2 компонент подання еквівалентний веб-сторінці. Компоненти подання використовуються для відображення інформації, що зберігається в моделях. Наприклад, описана вище модель "Клієнт" потребує компонента представлення для відображення даних для цієї моделі. Як правило, кожна веб-сторінка в додатку Struts2 має принаймні один компонент представлення.

Компоненти представлення створюються за допомогою Java Server Pages (JSP), і Struts2 надає розробникам набір "спеціальних тегів JSP" (іноді званих тегами Struts), які розширюють звичайну функціональність JSP і спрощують розробку компонентів представлення Struts2 надає розробникам набір "спеціальних тегів JSP" (іноді званих тегами Struts), які розширюють звичайну функціональність JSP і спрощують розробку компонентів представлення.

Struts2 підтримує широкий спектр компонентів подання: можна використовувати не лише JSP-сторінки, але й шаблони Velocity та Freemarker.

Struts2 також підтримує бібліотеку для побудови інтерфейсів форм на основі шаблонів Tiles.

3) Controller

Компонент контролера координує роботу веб-додатку. Контролери - це модулі, які отримують дані від користувача, оновлюють базу даних за допомогою компонентів моделі, а також виявляють і обробляють помилки у внутрішній системі. Контролер отримує дані від користувача, вирішує, які компоненти моделі потрібно оновити, обирає компонент представлення та надсилає повідомлення про те, що потрібно вказати оновлені результати.

Однією з головних переваг компонента контролера є те, що він дозволяє розробникам перенести код, який обробляє помилки, зі сторінки JSP в сам додаток. Коли виникає і обробляється помилка, компонент контролера викликає спеціальний компонент представлення, який відповідає за відображення помилки, а не компонент представлення, який відображає "нормальні" дані. Це значно спрощує логіку на сторінці і полегшує її розробку та підтримку.

Компоненти контролерів Struts 2 є класами Java і повинні бути побудовані за певними правилами, такими як успадкування властивостей від суперкласу і дотримання правил фреймворку в Struts, їх часто називають "Action-класами". Дії є важливим елементом побудови додатків за допомогою Struts 2. Вони можуть зберігати логіку програми, викликати сервіси та отримувати доступ до баз даних. В результаті виконання дії повертається рядок коду результату, який визначає виконання запиту користувача.

На діаграмі нижче показано, як можна реалізувати патерн проектування MVC за допомогою Struts2:

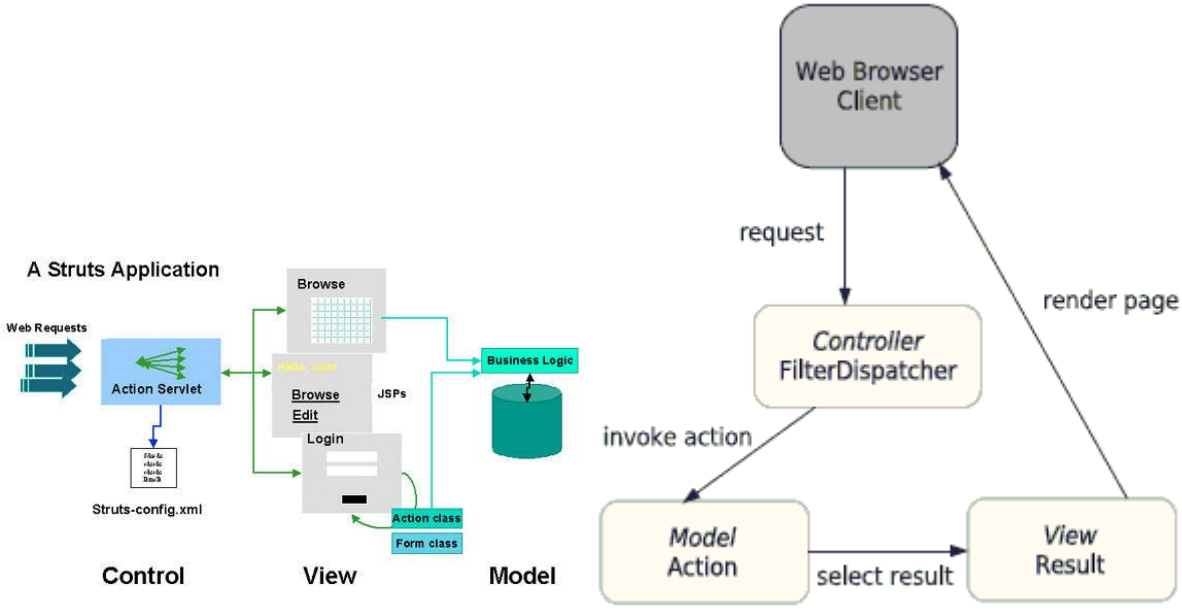


Рисунок. 3.1. Схема роботи Struts2

Загальна архітектура додатка наведена нижче:

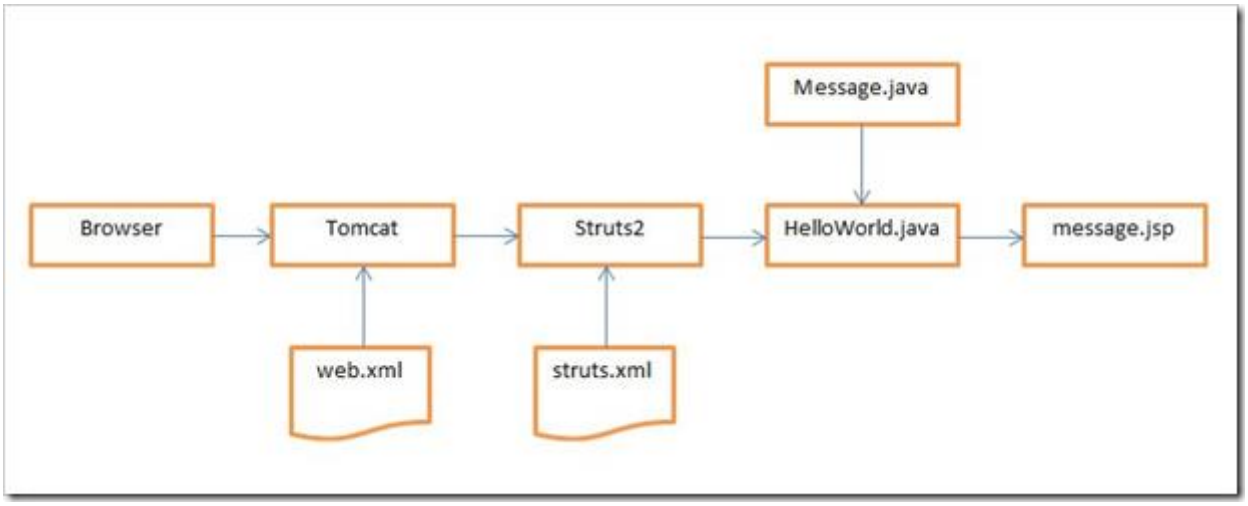
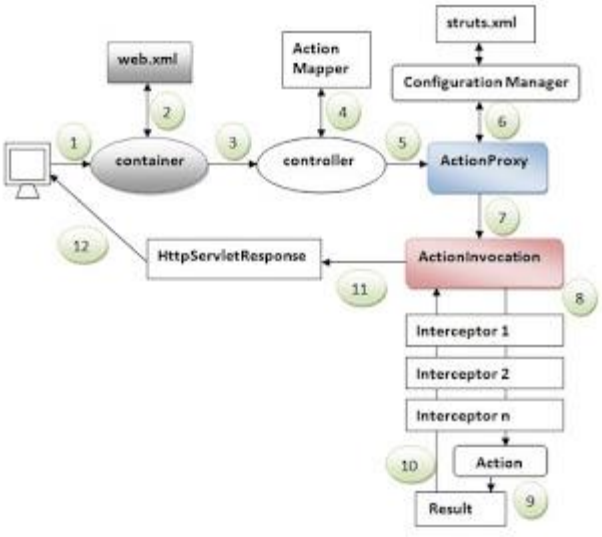


Рисунок. 3.2. Архітектура веб-додатку



Крок 1. Користувач надсилає запит до контейнера.

Крок 2. Контейнер викликає файл web.xml і отримає назву контролера.

Крок 3. Контролер викликає ActionMapper і отримує дію.

Крок 4: На основі дій він викликає Action Proxy.

Крок 5: Action Proxy викликає диспетчера налаштування, який у файлі struts.xml отримує інформацію про дію.

Крок 6. Action Proxy обробляє запит до виклику дії.

Крок 7: Action Invocations викликає всіх перехоплювачів (Interceptors) та викликає Action.

Крок 8: На основі результатів дій (Response) буде генерувати за допомогою Action Invocation.

Крок 9: Через HttpServletResponse буде надіслано результат кінцевому користувачеві.

Діаграма класів наведена у Рисунку.3.3.

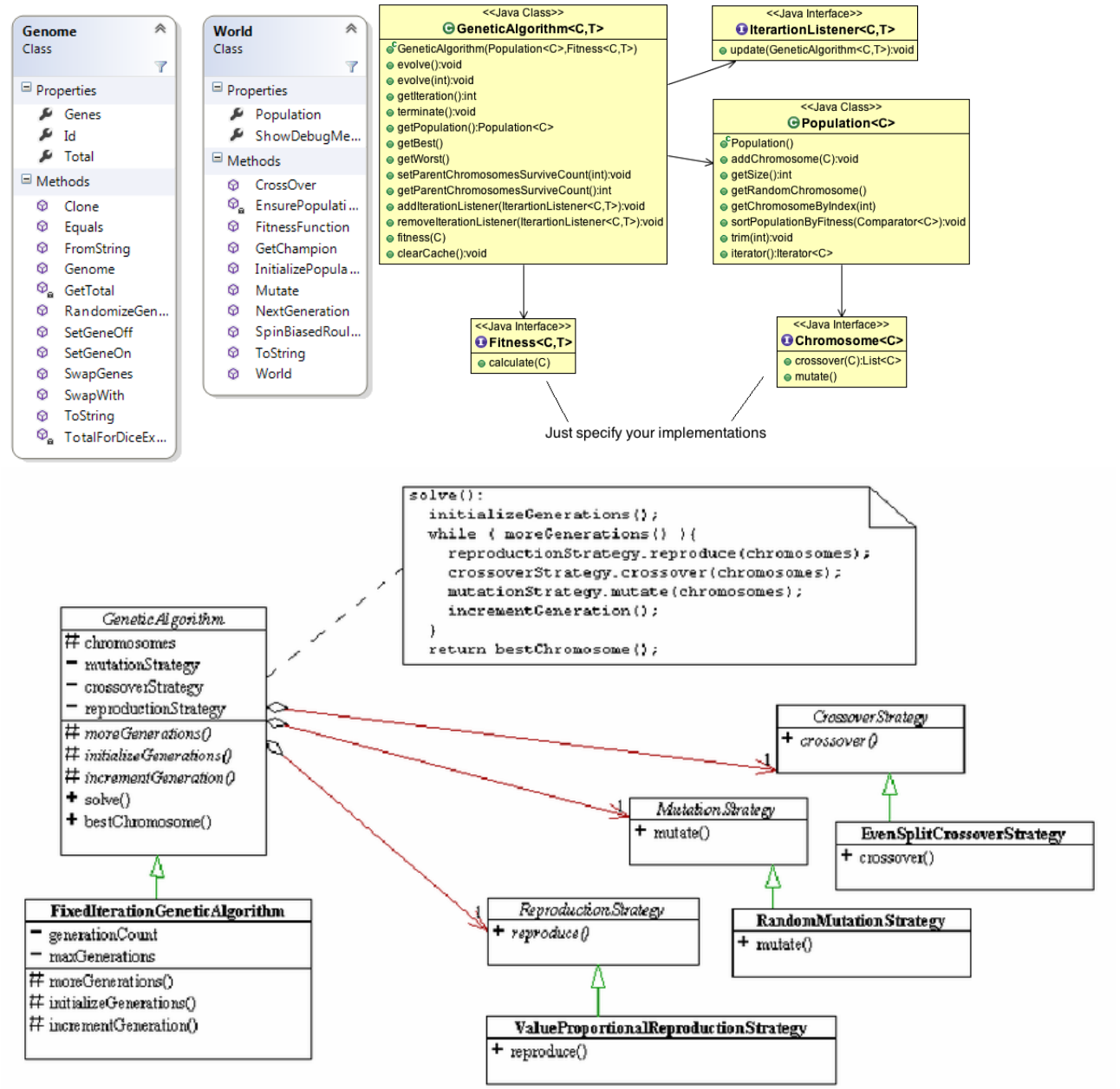


Рисунок. 3.3. Діаграма класів

Вибір системи управління базою даних є одним з найважливіших етапів розробки автоматизованої системи розкладу занять. Обраний програмний продукт повинен відповідати як поточним, так і майбутнім потребам організації, а також враховувати вартість розробки та налаштування необхідного програмного забезпечення і навчання персоналу.

MySQL - вільна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, що має на даний момент права на

торговельну марку. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією.

Основні переваги MySQL:

- багатопоточность, підтримка декількох одночасних запитів;
- оптимізація зв'язків з приєднанням багатьох даних за один прохід;
- записи фіксованої і змінної довжини;
- ODBC драйвер;
- гнучка система привілеїв і паролів;
- гнучка підтримка форматів чисел, рядків змінної довжини і міток часу;
- інтерфейс з мовами C і Perl, PHP;
- швидка робота, масштабованість;
- безкоштовна в більшості випадків;
- хороша підтримка з боку провайдерів послуг хостингу.

Керуючись тим, що одним з головних вимог при створенні автоматизованої системи розкладу є інтеграція в єдиний інформаційний простір, а також єдину базу даних підприємства, і в зв'язку виявленими перевагами поширеною СУБД, було прийнято рішення про використання в основі розроблюваної системи СУБД MySQL.

Побудована структура БД виглядає наступним чином:

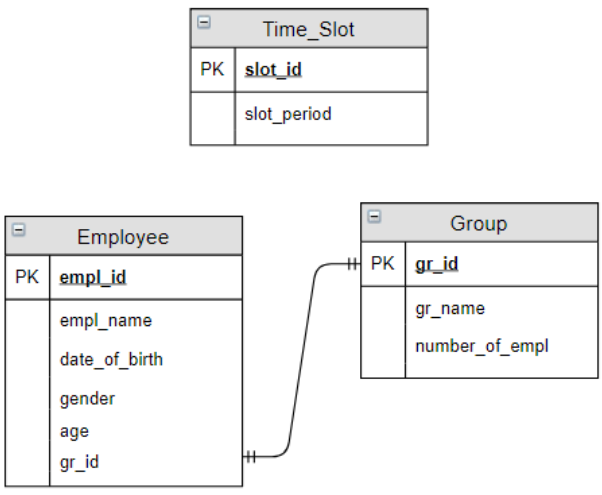


Рисунок. 3.4. Схема бази даних

3.2. Результати роботи алгоритму та екранні форми веб-додатку

Алгоритми були реалізовані за допомогою мови програмування Java 8. Програмні засоби, отримані в процесі розробки, були інтегровані на сторінки веб-сайту.

При практичній реалізації системи особливу увагу було приділено завданню написання "ядра" системи, тобто процедур, які формують спосіб розв'язання задачі та обмеження. Оскільки завданням не було написання повнофункціонального комерційного продукту, інтерфейсна частина була написана з метою тестування ядра та визначення меж роботи алгоритмів.

Алгоритми генерації рішень та обмежень були написані з використанням об'єктно-орієнтованих методів. Це зроблено для того, щоб їх можна було легко реалізувати в майбутніх модифікаціях системи без порушення цілісності взаємодії між різними алгоритмами. Об'єктний текст методу розв'язання задачі наведено у Додатку А.

Початкові дані для задачі вводяться в таблицю бази даних за допомогою анкети.

Таблиця 3.2 – Вхідні дані алгоритму генерації розкладу

Параметр	Атрибути/значення
Часові слоти	2
	Денна зміна: 9:00 - 21:00 Нічна зміна: 21:00 - 9:00
Кількість груп	3
	Група: CC; кількість співробітників в групі: 7 Група: SA; кількість співробітників в групі: 7 Група: EX; кількість співробітників в групі: 7
Повне ім'я співробітника	Іван Іванов
Стать	Чоловіча
Вік	25
Робочі години за місяць	168

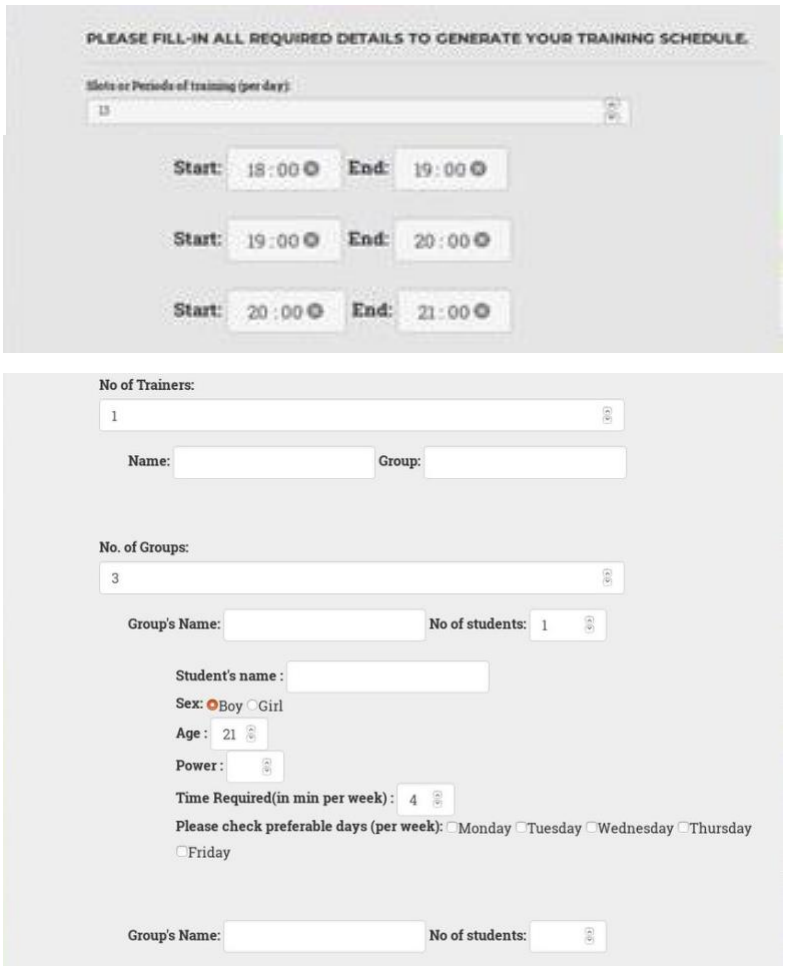


Рисунок. 3.5. Форма для заповнення параметрів алгоритму

Фінальний варіант розкладу робочих змін виводиться у вигляді таблиці, приклад якої відображений на рисунку. X.

Name	Group	31 Wed	1 Thu	2 Fri	3 Sat	4 Sun	5 Mon	6 Tue	7 Wed	8 Thu	9 Fri	10 Sat	11 Sun	12 Mon	13 Tue	14 Wed	15 Thu	16 Fri	17 Sat	18 Sun	19 Mon	20 Tue	21 Wed	22 Thu	23 Fri	24 Sat	25 Sun	26 Mon	27 Tue	28 Wed	29 Thu	30 Fri	1 Sat	9:00	21:00	sum	numb
Yakovlev, Alexey	EX	9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			84	96	180	15
Kolomyjchenko, Oleg	EX			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00	96	96	192	16
Brezhnev, Dmitry	SA	21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			96	84	180	15
Gutorov, Vladimir	SA			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00	96	96	192	16
Bessinnyj, Alexey	CC			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00	96	96	192	16
Rozhkovsky, Eugene	CC			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00	96	96	192	16
Petlya, Maksim	CC	21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			96	84	180	15
Rybalko, Serge	CC	9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			9:00	9:00			21:00	21:00			84	96	180	15

Рисунок. 3.6. Розклад, що було згенеровано алгоритмом

Наступний рисунок демонструє як виглядає безпосередньо сама хромосома, у якій закодовано розклад:

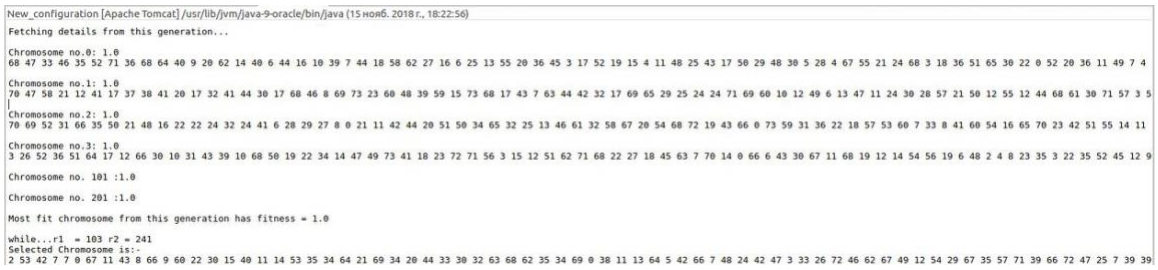


Рисунок. 3.7. Хромосома у якій закодовано розклад.

З метою покращення якості табелювання та для зручності працівників, отриманий табель надсилається кожному працівнику на електронну пошту. Працівники мають можливість оцінити отриманий табель (тобто, чи задовольняє їх отриманий табель чи ні).

Якщо більше 40% працівників не задоволені табелем, автоматично створюється новий табель, який надсилається працівникам.

3.3. Аналіз результатів та експериментальне дослідження методу еволюційних стратегій

Для перевірки працездатності програмного інструменту та якості результатів, отриманих за допомогою алгоритму, були підготовлені тестові набори даних різного рівня складності. Результати тестування показали, що програмний інструмент здатний отримати оптимальний (безконфліктний) розклад.

Основними оцінюваними параметрами є швидкість роботи алгоритму (таблиця 3.2) та кількість ітерацій, необхідних для отримання якісного розв'язку (рисунок 3.5).

На рисунку 3.7 показано результати тестування. По горизонтальній осі відкладено кількість ітерацій, необхідних для отримання бажаного розв'язку, а по вертикальній осі - значення фітнес-функції. Для тестування було обрано наступні алгоритми:

- Алгоритм еволюційної стратегії;
- Метод випадкового пошуку;
- Метод ручного складання розкладу людьми-експертами.

Отримані результати підтверджують ефективність еволюційної стратегії у вирішенні поставленої задачі. Застосовуючи алгоритм еволюційної стратегії за описаною методикою, можна зробити висновок, що такий підхід дозволяє скоротити час збіжності алгоритму і знайти необхідні результати за меншу кількість ітерацій.

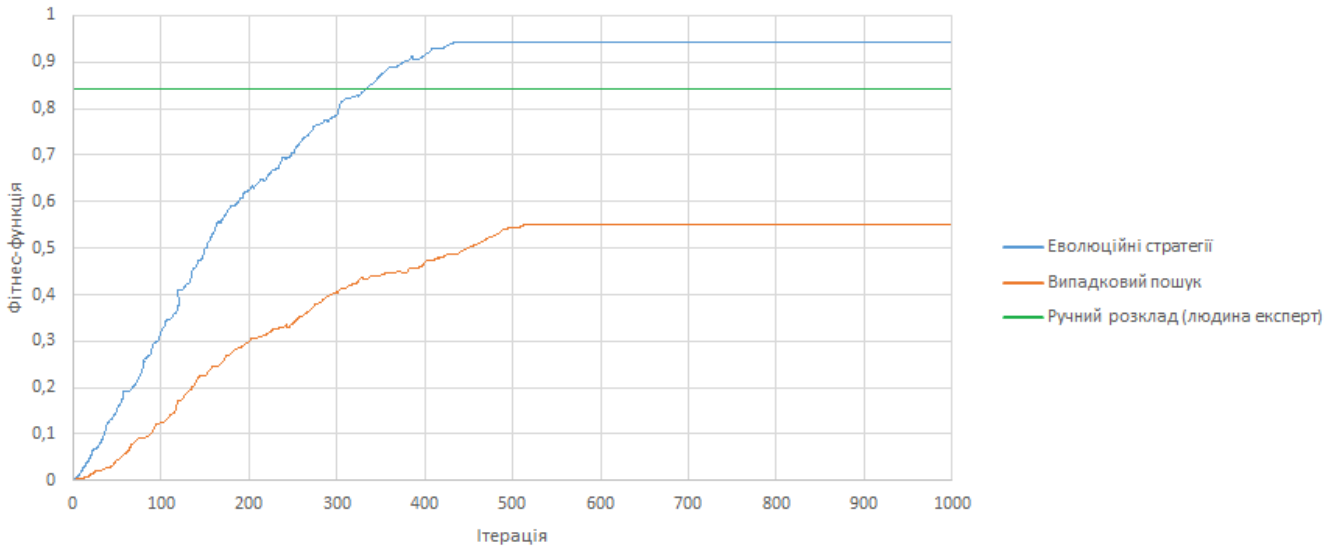


Рисунок. 3.8. Порівняння роботи різних алгоритмів

Створення "елітної" популяції дозволило отримати прийнятні оптимізаційні рішення за коротший час. Ця методологія зменшує кількість ітерацій алгоритму та час, необхідний для знаходження оптимального результату.

Час виконання кожної методології було протестовано для різних обсягів вхідних даних (працівників). Результати вимірювань представлені в таблиці нижче.

Таблиця 3.3 – Порівняльний аналіз часу роботи різних методів

	Кількість працівників		
	20	30	100
Еволюційні стратегії	0,171 с.	0,234 с.	0,559 с.
Випадковий пошук	1,438 с.	2,751 с.	7,856 с.
Ручний метод (людина-експерт)	1 г. 90 хв.	8 г.	1-2 тижні

Оскільки, у зв'язку з тим, що формування початкової популяції та використовувані оператори еволюційного відбору базуються на імовірнісному підході, повною мірою оцінити результати роботи досліджуваних алгоритмів шляхом тестування неможливо, було запропоновано зробити це шляхом опитування п'яти експертів у галузі календарного планування з метою оцінки якості результатів, отриманих алгоритмами. Експертне опитування проводилося в декілька етапів.

Спочатку експертам було запропоновано визначити ключові показники, необхідні для оцінки якості роботи системи планування. На наступному етапі експертам було запропоновано протестувати три варіанти системи на основі ручного методу планування, методу випадкового пошуку та алгоритму еволюційної стратегії.

Результати опитування наведені в таблиці 3.3. Для оцінки було запропоновано наступна шкала оцінювання:

- «Незадовільно» - 0-0,2 бали;
- «Задовільно» - 0,3-0,5 балів;
- «Добре» - 0,6-0,8 балів;
- «Відмінно» - 0,9-1 бал.

Таблиця 3.4 – Результат оцінки алгоритмів експертами

Критерій/експерт		1 експерт	2 експерт	3 експерт	4 експерт	5 експерт	Середня оцінка
Час пошуку рішення	Еволюційні стратегії	1	0,9	1	1	1	0,98
	Випадковий пошук	0,8	0,7	0,7	0,6	0,8	0,72
	Ручний метод (людина-експерт)	0,4	0,3	0,5	0,3	0.2	0,34
Якість отриманих результатів	Еволюційні стратегії	0,9	0,9	1	0,9	1	0,94
	Випадковий пошук	0,6	0,5	0,6	0,4	0,7	0,56
	Ручний метод (людина-експерт)	0,9	0,6	0,8	0,7	0,8	0,76

Дані, отримані під час анкетування, підтвердили результати попередніх тестів. Результати тестування та опитування показують, що алгоритми еволюційних стратегій можуть бути використані для вирішення поставленої задачі, розв'язання задач з великою кількістю подій, а також для створення повноцінних програмних додатків.

Висновки до розділу 3

В ході дослідження було розроблено математичну модель програми відділу технічної підтримки та адміністрування за умови 12-годинного робочого дня без урахування святкових днів, обрано метод розв'язання задачі та розроблено модель зберігання вихідних даних задачі. Модель зберігання вихідних даних, алгоритм математичної формалізації моделі та метод розв'язання були реалізовані у вигляді програмних модулів.

Швидкість роботи алгоритмів протестовано на гетерогенних наборах початкових даних.

Дослідження продемонструвало, що еволюційні стратегії можуть бути використані для розв'язання задачі планування. Оскільки в основі алгоритму лежить принцип еволюційного відбору, результати, отримані при запуску алгоритму на одних і тих же вхідних даних, можуть відрізнятися, але розв'язок є несуперечливим і оптимізованим з урахуванням обмежень, заданих користувачем. Під час дослідження застосовності алгоритмів еволюційних стратегій для планування завдань у відділах технічної підтримки та системного адміністрування алгоритми еволюційних стратегій були адаптовані для скорочення часу, необхідного для пошуку відповідного рішення, та отримання оптимізованого (безконфліктного) рішення.

ВИСНОВКИ

Складання розкладу - трудомісткий процес, що вимагає спеціальних навичок і знань. Математично цю проблему можна сформулювати як складну оптимізаційну задачу, для якої існує багато методів.

Еволюційні методи розв'язання оптимізаційної задачі вивчаються детально.

Можна сказати, що в основі алгоритму лежить аналогія з природними еволюційними процесами. Перевагами алгоритму є відсутність додаткових вимог щодо виду цільової функції та робота з множиною розв'язків на кожній ітерації, що в багатьох випадках дозволяє більш детально проаналізувати простір пошуку порівняно з градієнтними методами багатовимірної нелінійної безумовної оптимізації.

Проаналізувавши, як еволюційні стратегії застосовуються до задач розкладів, можна стверджувати, що цей метод має ряд переваг над іншими методами:

- Концептуальна простота та прозорість реалізації;

- Простота кодування вхідної/вихідної інформації.

- Застосовність до широкого кола задач без суттєвих змін у внутрішній структурі методу.

Під час виконання дипломної роботи було розглянуто застосування алгоритму еволюційної стратегії до задачі календарного планування та оптимізації.

На цьому етапі були виконані наступні завдання:

1. Створено загальний опис об'єкта дослідження (графік змінності)

2. Сформульовано задачу оптимізації розкладу занять в термінах генетичних алгоритмів:

1) Визначено вхідну інформацію:

- Гени - це працівники;

- Хромосоми - робочі зміни; 2;

- Індивіди - графіки робочих змін; 2;

- Популяція - список згенерованих робочих змін.

2) Сформульовано критерій якості розкладу (цільова функція пристосованості, значення якої обернено пропорційно залежить від кількості порушень м'яких обмежень, накладених на розклад, і значення якої необхідно максимізувати для отримання оптимального розкладу).

3. На основі сформульованого алгоритму на мові програмування Java реалізовано програмний продукт, який генерує розклади класів за цим алгоритмом.

4. Проведено аналіз та порівняння розробленого рішення з методами випадкового пошуку та ручного складання розкладу.

Швидкість роботи алгоритму була протестована на різнорідних наборах вхідних даних і зроблено висновок, що реалізований алгоритм еволюційної стратегії є найкращим вибором для складання розкладу занять.

Додаток був успішно впроваджений в ІТ-компанії, а саме у відділі технічної підтримки та адміністрування. Однак варто зазначити, що розроблений продукт також може бути адаптований до різних інших галузей (наприклад, управління проектами, навчальний процес в університетах, спорт, транспорт і т.д.). Єдина відмінність полягає в способі кодування рішень (хромосоми) та критеріях їх оцінювання. Всі інші оператори стандартні: селекція, кросинговер, мутація.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бевз С. В. Розробка автоматизованої системи формування розкладу магістратури / Бевз С. В., Войтко В. В., Бурбело С. М., Шоботенко А. М. // Наукові праці ВНТУ. – 2009. – №1. – С. 1-10.

2. M. Fisher, H. L. Greenberg, W. L. Berry. "An Investigation of the Pickup and Delivery Problem," Operations Research, Vol. 40, No. 3, 1992.

3. R. E. Bellman, S. E. Dreyfus. "Applied Dynamic Programming." Princeton University Press, 1962.

4. Бойко О.М. Еволюційна технологія розв’язування задачі складання розкладів навчальних занять/ Бойко О.М. // Штучний інтелект. – 2006. – №.3. – С. 341-348.

5. B. Chen, S. Yang, Y. Li. "An automated scheduling system for academic courses using genetic algorithm." Expert Systems with Applications, Volume 36, Issue 2, Part 1, 2009.

6. P. Li, H. F. Li. "A hybrid genetic algorithm for the curriculum-based course timetabling problem." Expert Systems with Applications, Volume 36, Issue 4, Part 1, 2009.

7. М. Пінедо. "Розкладання: теорія, алгоритми та системи." Спрінгер, 2016.

8. Р. Седжвік, К. Вейн. "Алгоритми, частина I." Princeton University Press, 2011.

9. Т. Кормен, Ч. Лейзерсон, Р. Рівест, К. Штайн. "Введення в алгоритми." MIT Press, 2009.

10. М. Пінедо. "Розкладання: теорія, алгоритми та системи." Springer, 2016.

11. J. Smith. "Class Schedule as an Organizational Document: Defining the Operation Mode of an Educational Institution."

12. Редактор створення діаграм. – Режим доступу: https://drive.draw.io/ . – Дата доступу: 15.05.2017.

13. D. E. Goldberg. "Genetic Algorithms in Search, Optimization, and Machine Learning." Addison-Wesley, 1989.

14. M. Pinedo. "Scheduling: Theory, Algorithms, and Systems." Springer, 2016.

15. P. Brucker. "Scheduling Algorithms." Springer, 2007.

16. Arnold D. V. Weighted multirecombination evolution strategies // Theoretical Computer Science. — 2006. — Т. 361, № 1. — С. 18—37.

17. Bäck T.: “Parallel Optimisation of Evolutionary Algorithms”, Y. Davidor, H.-P. Schwefel and

18. Banzhaf W.: “Genetic Programming for Pedestrians” MERL Technical Report, 93-03,

19. Bellman R. Mathematical aspects of scheduling theory // Journal of the Society of Industrial and Applaid Mathematics. 1956. Vol. 4. P. 168–205.

20. Cantu-Paz E.: “A survey of parallel genetic algorithms", Calculateurs Paralleles, Reseaux et

21. Cantu-Paz E.: “Topologies, Migration Rates, and Multi-Population Parallel Genetic Algorithms". To Appear in: GECCO-99, Genetic and Evolutionary Computation Conference, July 13--17, Orlando FL, 1999.

22. Chang T.-J., Meade N., Beasley J.E. and Sharaiha Y.M.: “Heuristics for cardinality constrained portfolio optimisation". working paper available from the third author at The Management School, Imperial College, London SW7 2AZ, 1998.

23. Chong F. S.: “A Java based distributed approach to genetic programming on the internet“ Master‘s Thesis, Computer Science, University of Birmingham, 1998.

24. Conference on on Evolutionary Computation, pp 418-427, Springer, Berlin, 1994.

25. Conway R.W., Maxwell W.L., Miller L.W. Theory of Scheduling. Addison-Wesley, Reading, MA. 1967. 87 12)

26. Cook S. A. The complexity of theorem proving procedures // Proceedings of the 3rd Annual ACM Symposium on the theory of Computing — 1971. — P. 151–158.

27. Darwin, C.R.: “The Origin of Species“ London (John Murray) 1859.

28. Davis, L.: “Genetic algorithms and financial applications“, in: Deoeck (ed., 1994), pp. 133-147, 1994.

29. Drake A.E. and Marks R.E.: “Genetic algorithms in economics and finance: forecastingstock market prices and foreign exchange”, Australian Graduate School of Management Working Paper 98-004, February, 1998

30. Edmonds J. Paths, trees, and flowers // Canadian Journal of Mathematics. — 1965. — Vol. 7 — P. 449–467.

31. Elhara O. A., Auger A., Hansen N. A median success rule for nonelitist evolution strategies: Study of feasibility // Genetic and Evolutionary Computation Conference. — 2013.

32. Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Optimization and approximation in determenistic scheduling: a servey // Annals of Discrete Mathematics — 1979. — Vol. 5 — P. 287–326.

33. Hansen N. An analysis of mutative σ-self-adaptation on linear fitness functions // Evolutionary Computation. — 2006. — Т. 14, № 3. — С. 255—275.

34. Igel C., Hansen N., Roth S. Covariance Matrix Adaptation for Multi-objective Optimization // Evolutionary Computation. — 2007. — Т. 1, № 15. — С. 1—28

35. Learning probability distributions in continuous evolutionary algorithms — A comparative review / S. Kern [и др.] // Natural Computing. — 2004. — Т. 3, № 1. — С. 77—112

36. Mitsubishi Electric Research Labs, Cambridge, MA, 1993

37. Papadimitriou C., Yannakakis M. Optimization, approximation and complexity classes // Journal of Computer and System Science. — 1991. — Vol 43. — P. 425–440.

38. R. Männer, editors: Parallel Problem solving from Nature - PPSN III, International

39. Rechenberg I. Evolutionsstrategie ‘94. — Frommann-Holzboog Verlag, 1994.

40. Schwefel H.-P.: “Evolutionsstrategie und numerische Optimierung“, Dissertation, Technische Universität Berlin, 1975

41. Systems Repartis, vol 10, number 2, pp. 141-171, 1997.

42. M. Fischer, H. L. Greenberg, W. L. Berry. "An Investigation of the Pickup and Delivery Problem." Operations Research, Vol. 40, No. 3, 1992.

43. P. Brucker. "Scheduling Algorithms." Springer, 2007.

44. H. A. Taha. "Operations Research: An Introduction." Prentice Hall, 2016.

45. Клас NP [Електроний ресурс] // Вікіпедія: вільна енциклопедія. Дата оновлення: 03.04.2016. Режим доступу: http://ru.wikipedia.org/?oldid=77544025 (дата звертання: 14.02.2017). 17) Garey M. and Johnson D. Computers and Intractability: A Guide to the theory of NP-completeness. — San Francisco, CA: W.H. Freemann and Company, 1979.

46. M. L. Pinedo. "Planning and Scheduling in Manufacturing and Services." Springer, 2009.

47. M. Pinedo. "Scheduling: Theory, Algorithms, and Systems." Springer, 2016.

48. M. Fischer, H. L. Greenberg, W. L. Berry. "An Investigation of the Pickup and Delivery Problem." Operations Research, Vol. 40, No. 3, 1992.

49. M. L. Pinedo. "Scheduling: Theory, Algorithms, and Systems." Springer, 2016.

50. M. L. Pinedo. "Scheduling: Theory, Algorithms, and Systems." Springer, 2016.

51. "Scheduling Theory" on Scholarpedia. https://www.scholarpedia.org/article/Scheduling_theory

52. Introduction to Scheduling Theory" / Daniel J. Rosenkrantz, Richard E. Stearns, Philip M. Lewis.

53. Scheduling Theory Wikipedia: The Free Encyclopedia. Дата обновления: 23.12.2016. Режим доступу: https://en.wikipedia.org/?oldid=82645503 (дата звернення: 13.10.2023).

54. Gantt H.L. ASME Transactions, 1903, 24, P. 1322–1336. 3) Matkovskiy I. Henry Gantt // Famous People [Електронний ресурс]. Режим доступу: https://www.biography.com/inventor/henry-l-gantt (дата обращения: 13.02.2017).