Національний технічний університет України

«Київський політехнічний інститут»

ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____2017 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Застосування генетичного алгоритму оптимізації для формування розкладу занять

Виконала: студент 4 курсу, групи ДА-32
(шифр групи)

Натальчук Максим Олегович__ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доцент Чкалов О.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний к.е.н., доцент Рощина Надія Василівна _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль старший викладач Британ О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України

«Київський політехнічний інститут»

ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

_____
(прізвище, ім'я, по батькові)

1. Тема роботи Застосування генетичного алгоритму оптимізації для формування розкладу занять,
керівник роботи Чкалов О.В. к.т.н., доц.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» травня 2017 р. №1477-с
2. Термін подання студентом роботи _____
3. Вихідні дані до роботи
   1) Інформація про мову програмування написання програмного продукту;
   2) Інструкція по доступним операційним системам, на яких можна використати програмний продукт.
4. Зміст роботи
   1) Визначення мети та задачі проекту;
   2) Визначення факторів успіху дипломної роботи;
   3) Опис загальних термінів генетичного алгоритму;
   4) Описати математичну модель об'єкта оптимізації;
   5) Складання генетичного алгоритму.
   6) Написання коду програми, опис інтерфейсу та результати її роботи;
   7) Порівняння власної реалізації з існуючими рішеннями;
   8) Визначення переваг та недоліків власного алгоритму;

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1.
2.
3.
4.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., к.е.н., доцент		

7. Дата видачі завдання 10.03.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	10.03.2017	
2	Збір інформації	24.03.2017	
3	Опис об’єкту оптимізації	07.04.2017	
4	Розробка цільової функції	27.04.2017	
5	Розробка генетичного алгоритму оптимізвції	27.05.2017	
6	Розробка програмного продукту	30.05.2017	
7	Отримання дипломної роботи	09.06.2017	
	Отримання допуску до захисту та подача роботи в ДЕК	11.06.2017	

Студент _____ М.О. Натальчук
(підпис) (ініціали, прізвище)

Керівник роботи _____ О.В. Чкалов
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи студента Натальчука Максима Олеговича на тему «Застосування генетичного алгоритму оптимізації для формування розкладу занять»

В даній дипломній роботі розглянуто застосування генетичного алгоритму в задачі автоматизації складання розкладу занять.

Мною було сформульовано задачу оптимізації розкладу в рамках генетичних алгоритмів. На основі цієї задачі було реалізовано програмний продукт на мові програмування Java. Також було досліджено існуюче автоматизоване рішення задачі оптимізації розкладу з допомогою генетичного алгоритму, і розклад занять, складений вручну.

За допомогою програмного продукту, існуючого автоматизованого і ручного рішення було складено розклади занять для 4 груп з кафедри системного проектування. На їх основі було проведено порівняльну характеристику трьох рішень, доведено перевагу валсного рішення над іншими, визначено переваги та недоліки самостійно розробленого алгоритму.

Дипломна робота займає 76 сторінок, містить 20 таблиць, 15 малюнків, 8 посилань. Додаток з вихідним кодом програмного продукту займає 18 сторінок.

Ключові слова: генетичний алгоритм, оптимізація, програмний продукт, Java, розклад розроблений вручну, порівняльна характеристика, дослідження.

АННОТАЦИЯ

бакалаврской работы студента Натальчука Максима Олеговича на тему «Применение генетического алгоритма оптимизации для формирования расписания занятий»

Мной было сформулировано задачу оптимизации расписания в рамках генетического алгоритма. На основе этой задачи было реализовано програмный продукт на языке программирования Java. Также было исследовано существующее решение задачи оптимизации расписания с помощью генетического алгоритма, и расписание занятий, составленное вручную.

При помощи программного продукта, существующих автоматизированного и ручного решения, были составлены расписания занятий для 4 групп кафедры системного проэктирования. На их основе было проведено сравнительную характеристику решений, докано преимущество собственного решения над другими, определено преимущества и недостатки разработанного алгоритма.

Дипломная работа занимает 76 страниц, имеет 20 таблиц, 15 рисунков, 8 ссылок. Дополнение с исходным кодом программы занимает 18 страниц.

Ключевые слова: генетический алгоритм, оптимизация, программный продукт, Java, расписание вручную, сравнительная характеристика, исследования.

ANNOTATION

on Natalchuk Maksim Bachelor’s degree

Entitled «Application of genetic optimization algorithm to the timetable modeling»

In this thesis work deals with the application of genetic algorithm for the problem of automating the university timetable.

I formulated optimization problem decomposition in terms of genetic algorithms. Based on this task it was implemented software in the Java programming language. It was also examined existing automated solution with scheduling optimization problem using genetic algorithm and schedule drawn up manually.

With the help of the software product, the existing automated and manual solutions, timetables were compiled for the 4 groups of the system design department. Based on these timetables, the comparative characteristics of the solutions were made, the advantage of own solution over others was proved, the advantages and disadvantages of the developed algorithm were determined.

Thesis occupies 76 pages application and source code occupies 18 pages. Thesis has 20 tables, 15 images and 8 links.

Keywords: genetic algorithm, optimization, programming product, Java, schedule developed manually, comparative characteristics, research.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

Генетичний алгоритм — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання, шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.[1]

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.

ВСТУП

Складання розкладу занять являється одним із важливих завдань, що вирішуються при плануванні навчального процесу. В першу чергу, це пов'язано з тим, що без розкладу занять неможливе функціонування будь-якого навчального закладу. Іншими словами, розклад навчальних занять має бути складено своєчасно. Крім того, розклад повинен бути «якісно» складеним, тобто відповідати ряду вимог і критеріїв. В якості таких критеріїв можуть виступати такі, що відображають економічну ефективність використання наявних ресурсів освітньої системи, комфортність навчання студентів і роботи ППОС, обмеження за часом навчання і т.д.

Складання розкладу занять відноситься до завдань цілочисельного програмування, складність вирішення яких зростає експоненціально з ростом числа і можливих значень варійованих змінних (такі задачі відносяться до класу NP-важких задач). Крім того, для неї характерна наявність більшого обсягу різної за своїм складом вихідної інформації і великого числа важко формульованих вимог.

Зазначені складності перешкоджають автоматизації процедури складання розкладу, незважаючи на наявність широкого спектра методів цілочисельного програмування: методів повного перебору, методу гілок і кордонів, методу розмальовки графів, евристичних методів. Для автоматизації процедури складання розкладу занять часто пропонуються підходи, що базуються на так званих точних (класичних) методах і алгоритмах цілочисельного програмування.

Недоліком даних методів є великий об’єм і складність отриманої математичної моделі задачі складання розкладу, різке зростання витрат часу з ростом обсягів вихідної інформації на пошук рішення NP-складного характеру задачі складання розкладу в її класичній постановці. Крім того, в таких

випадках погано враховуються структурні особливості об'єктів розкладу (викладачі, групи, аудиторії, дисципліни, часові інтервали занять), між якими існують зв'язки, обумовлені специфікою організації навчального процесу. Так, наприклад, викладачі ведуть заняття в суворо визначених групах і в строго визначений час. Окрім того, заняття в групах проводяться зі строго визначених дисциплін згідно учбового плану. Відзначимо, що частина з вказаних об'єктів має ієрархічну структуру. Так, наприклад, навчальні групи можуть об’єднуватися в потоки, що включають в себе групи одного курсу однієї спеціальності. Такі потоки назвемо «малими». В свою чергу ці «малі» потоки можуть входити як складові частини в інші більші потоки з груп кількох спеціальностей і т. д. Тому такі підходи виявляються малоефективними при складанні розкладу занять для освітніх систем масового навчання.

Останнім часом все частіше для вирішення великорозмірних завдань цілочисельного програмування, в тому числі завдання складання розкладу, використовують різні евристичні методи. До числа таких методів відносяться так звані генетичні алгоритми, які і є предметом дослідження в даній дипломній роботі. З урахуванням усього вищесказаного можна сказати, що розробка генетичних алгоритмів оптимізації, які враховують структурні особливості розкладу занять, є актуальним завданням. У даній роботі для складання розкладу занять пропонується власне розроблений генетичний алгоритм складання розкладу занять, заснований на структуризації вихідної інформації, адаптованих до неї генетичних операціях і структурному поданні об'єктів генетичної оптимізації.

1 СИСТЕМНИЙ АНАЛІЗ ЗАВДАНЬ БАКАЛАВРСЬКИХ ДОСЛІДЖЕНЬ

1.1 Мета та огляд літератури

Мета дипломної роботи

Дослідження методів застосування генетичного алгоритму оптимізації для формування розкладу занять, розробка власного генетичного алгоритму для складання розкладу.

Огляд літератури

Основні джерела інформації, що необхідні для вирішення заданої задачі, це джерела з описом генетичних алгоритмів та їх застосуванням у прикладних задачах (наприклад, Рутковская Д., Пилиньский М., Рутковский Л. «Нейронные сети, генетические алгоритмы и нечеткие системы») [1].

Ю. В. Берегових та Н.А. Васильєв обґрунтували необхідність автоматизації процесу складання розкладу занять та чому саме генетичний алгоритм варто застосувати для вирішення цієї задачі. Ними було розроблено одне з рішень генетичного алгоритму, а саме описано в математичних термінах об’єкт оптимізації, сформульовано жорсткі та м’які умови, що накладаються на розклад, та в загальному сформульовано варіанти для етапів генетичного алгоритму.[2]

Младен Жанковіч запропонував власну реалізацію генетичного алгоритму для складання розкладу занять для одного курсу в університеті на мові програмування C++. Для схрещування, мутації та селекції він використав найпоширеніші їх типи, але запропонував власний варіант функції пристосованості, що враховує деякі жорсткі та м’які обмеження, що стосуються

розкладу занять. Дану реалізацію буде використано мною для дослідження та порівняльної характеристики власного алгоритму з уже існуючими рішеннями.[3]

Юрій Кабальников описав композиційний генетичний алгоритм оптимізації розкладу занять, у якому детально описав об’єкт оптимізації з урахуванням усіх можливих тонкощів у рамках математичних термінів. Також він запропонував власну реалізацію даного алгоритму на мові програмування Delphi.[4]

1.2 Задачі дипломної роботи

Задачами дипломної роботи є:

1. Провести аналіз існуючих рішень реалізації генетичного алгоритму і аналіз вимог до розкладу занять.
2. Проектування архітектури програми для складання розкладу занять.
3. Розробка дизайну цієї програми.
4. Реалізація цієї програми.
5. Тестування реалізованої програми.
6. Порівняльна характеристика власного методу та існуючих рішень та обґрунтування чому саме цей метод є найоптимальнішим у задачі формування розкладу занять.

Завдання, що мають бути виконані в ході написання дипломної роботи:

1. Вивчити структуру об’єкта оптимізації.
2. Сформулювати критерій якості розкладу занять.
3. Сформулювати оптимізаційну задачу в термінах і поняттях генетичного алгоритму.
4. Реалізувати генетичний алгоритм оптимізації для формування розкладу занять.

5. Дослідити власне розроблений алгоритм на основі порівнянь з існуючими рішеннями задачі та визначити переваги та недоліки власного алгоритму.

1.3 Висновки

У даному розділі було проведено системний аналіз завдань досліджень моєї роботи, визначено мету роботи та проведено огляд основних джерел літератури, яку було використано для написання дипломної роботи.

2 ЦІЛІ ДИПЛОМНОЇ РОБОТИ, КЛЮЧОВІ ФАКТОРИ УСПІХУ РЕЗУЛЬТАТІВ РОБОТИ

2.1 Цілі роботи

Основними цілями бакалаврської роботи є зниження затрат часу складання розкладу занять, надання зручного інтерфейсу для його складання, виключення людського фактору при складанні розкладу та покращення його якості.

2.2 Фактори успіху результатів

Основним фактором успіху бакалаврської роботи є таке виконання цілей проекту, при якому програмний продукт згенерує зручний та близький до ідеального розклад занять та матиме простий інтерфейс. Але важливими також є і чітко сформульована мета роботи та спроектовані етапи роботи, доступність теоретичних даних та опису використання подібних методів.

2.3 Висновок

У даному розділі було визначено основні цілі проекту, виконання яких і буде ключовим фактором успіху результатів даної дипломної роботи.

3 ЗАГАЛЬНИЙ ОПИС ГЕНЕТИЧНОГО АЛГОРИТМУ

3.1 Поняття генетичного алгоритму

Генетичний алгоритм (англ. *genetic algorithm*) — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання, шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.[1]

Цей метод заснований на аналогії з природними еволюційними процесами. [2] Його перевагою є те, що не висуваються додаткові вимоги до виду цільової функції, на кожній ітерації він працює з множиною рішень, що дозволяє в багатьох випадках більш детально в порівнянні з градієнтними методами багатовимірної нелінійної безумовної оптимізації аналізувати простір пошуку. [3]

Особливістю генетичного алгоритму є акцент на використання оператора “схрещування”, який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещування в живій природі.

При моделюванні цим методом, активно використовується понятійний апарат генетики. Тому слід дати визначення наступним поняттям:

- 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. У генетичних методах терміни «хромосома» та «особина» використовуються як синоніми;[3]

- популяція – досить велике співтовариство організмів, що схрещуються між собою. Популяції характеризуються набором ланцюжків генів кожного з об’єктів, сукупність яких визначає генофонд популяції .[5]

Генетичний алгоритм має такі переваги:

- відсутність необхідності в специфічних знаннях про вирішувану задачу. Проте у випадку, якщо існує додаткова інформація про досліджувану систему, об’єкт або процес є відомим, то вона може бути використана в процесі пошуку;
- концептуальна простота та прозорість реалізації;
- можливість розпаралелювання;
- простота кодування вхідної і вихідної інформації. Некритичність до виду параметрів досліджуваних систем (можливість використання експертної, емпіричної, довідкової та іншої інформації про об’єкт, поданої різними типами даних);
- можливість застосування до великого кола задач без внесення серйозних змін у внутрішню структуру методу;
- можливість адаптивності параметрів генетичного пошуку до особливостей вирішуваної задачі;
- менша ймовірність попадання і зациклення в локальному оптимумі, яка досягається за рахунок використання популяційного підходу;
- можливість застосування в методі інших пошукових процедур.

Генетичний алгоритм реалізується у декілька етапів (рис. 3.1) :

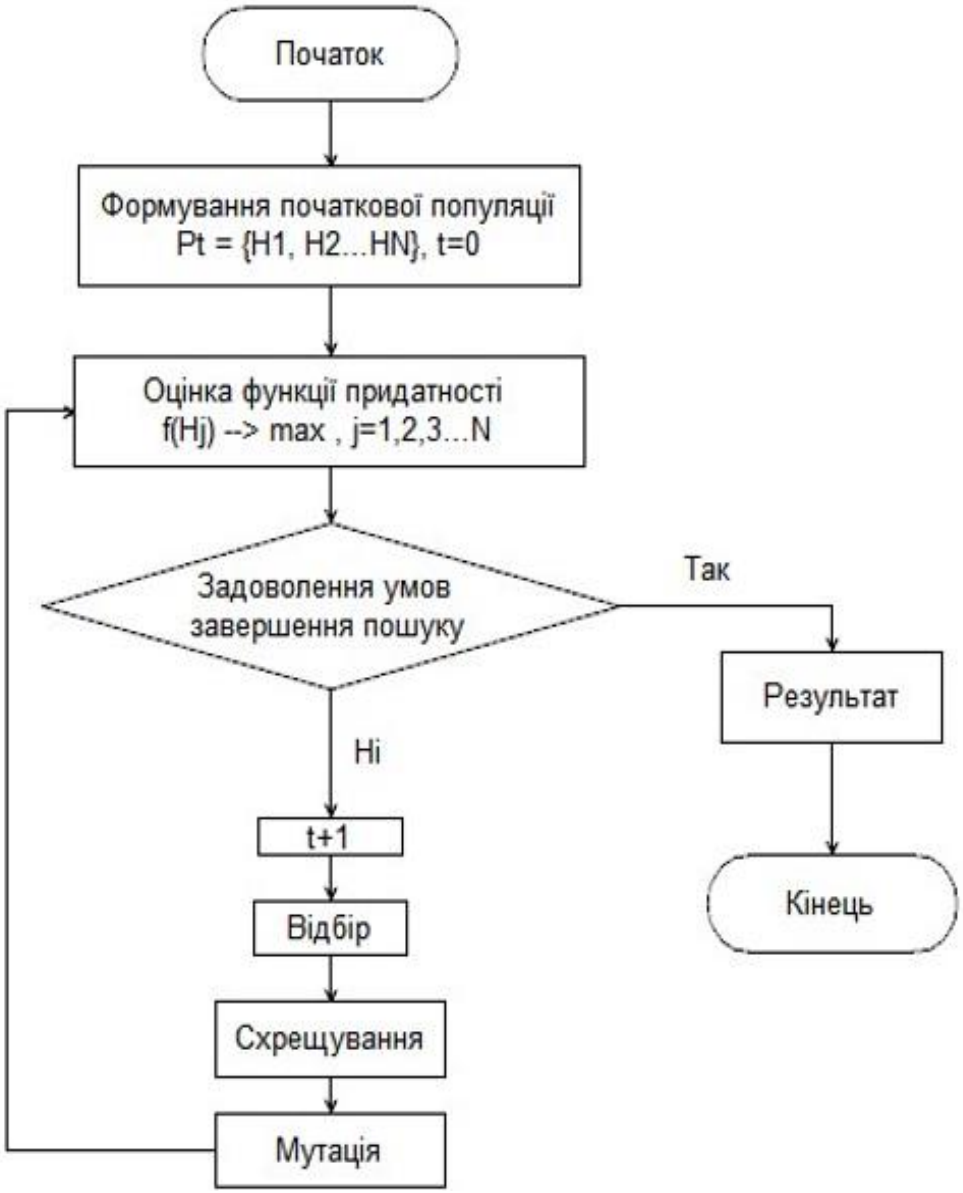

Рис. 3.1 – Схема роботи генетичного алгоритму [4]

3.2 Висновок

У даному розділі було в загальному описано термінологію генетичного алгоритму та етапи, з яких він складається. Для зрозумілості було побудовано блок-схему алгоритму. Також було визначено його основні переваги та недоліки загалом та конкретно у задачі оптимізації розкладу занять.

4 ЗАГАЛЬНИЙ ОПИС ОБ'ЄКТУ ОПТИМІЗАЦІЇ

4.1 Елементи розкладу занять

Розклад занять в університеті складається з таких об'єктів:

1. Множина груп студентів, елементами якої є, власне, групи.

$$G = \{g_1, .., g_m\}, \quad (4.1)$$

де g_i – це група;

m – кількість груп на курсі.

Кожен елемент цієї множини складається з двох полів:

- назва групи;
- множина предметів, що викладаються для цієї групи:

$$g_i = \{n_i, d_i\}, \quad (4.2)$$

де n_i – назва групи;

d_i – множина предметів.

$$d_i = \{d_i^1, .., d_i^k\}, \quad (4.3)$$

де:

$$d_i^j = \{nd_i^j, p_i^j, t_i^j\}, \quad (4.4)$$

d_i^j – предмет;

nd_i^j – назва предмету;

p_i^j – ПІБ викладача, що читає предмет;

t_i^j – тип заняття (лекція, практика чи лабораторне заняття).

2. Множина аудиторій, що містить дані про аудиторії, де можуть навчатись студенти.

$$A = \{a_1, .., a_l\}, \quad (4.5)$$

де l – кількість аудиторій.

Кожен елемент цієї множини містить таку інформацію:

$$a_i = \{na_i, da_i, c_i\}, \quad (4.6)$$

де na_i – номер аудиторії;

da_i – тип (для лекцій, практичних занять чи лабораторних робіт);

c_i – місткість (кількість студентів, що можуть там навчатись).

3. Двомірний масив, у вигляді якого буде представлено розклад занять.
Нульовий стовпець матриці – групи, для яких формується розклад.
Нульовий рядок – час проведення занять, що буде представлено у вигляді «Номер тижня - день тижня - номер пари».
Інші елементи – це заняття:

$$z_{ij} = \{n_i, nd_i^h, p_i^h, t_i^h, a_s\}, \quad (4.7)$$

де n_i – група;

nd_i^h - заняття з предмету;

t_i^h - тип заняття;

p_i^h - викладач;

a_s – аудиторія.

Даний масив матиме наступний вигляд:

Таблиця 4.1 – Представлення розкладу занять

Група \ час	1-Понеділок-1	1-Понеділок-2	…	2-П’ятниця-5
Група 1	…	…	…	…
Група 2	…	Предмет Викладач Аудиторія Тип	…	…
…	…	…	…	…
Група n	…	…	…	…

4.2 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]

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

1. Викладач не може бути присутнім на двох заняттях одночасно;
2. В одній аудиторії не можуть проводитися різні заняття одночасно;
3. Місткість аудиторії не може бути меншою, ніж кількість присутніх у ній студентів;
4. Деякі заняття вимагають особливих аудиторій (наприклад, комп'ютерних класів, лабораторій, спортзалів та інше);
5. Одна група не може вивчати декілька предметів на одному занятті одночасно;
6. Кількість занять на день не має перевищувати 4.

М'які обмеження, які можуть виконуватись для розкладу занять:

1. У розкладі не має бути вікон;
2. Викладачі можуть віддавати перевагу конкретному часу проведення занять;
3. Заняття мають проводитись в мінімальну кількість днів, щоб у студентів було якомога більше вихідних;
4. Усі лабораторні заняття мають проводитись в один день;
5. Складні заняття мають проводитись на першій та другій парах;
6. Заняття, на яких не відмічають присутніх, мають проводитись в один день;
7. Заняття з фізичної підготовки мають проводитись на 3-4 парах;
8. Заняття одного дня мають проводитися в одному навчальному корпусі;
9. У розкладі не має бути проміжних вихідних днів (наприклад, у понеділок та середу відбуваються заняття, а вівторок – вихідний день);
10. Кількість занять в останній день навчання в тижні має бути меншою ніж у інші дні;
11. В перший день навчання в тижні на початковій парі має бути відносно легкий предмет, щоб не знизилась успішність студентів.[5]

4.3 Висновок

У даному розділі було детально описано об’єкт оптимізації, а саме розклад занять. Було визначено з яких елементів ві буде складатись та як ці елементи буде поєднано для його утворення. Також було визначено обмеження, що накладаються на розклад, а саме ті, дотримання яких є обов’язковим для успішного складання розкладу, і ті, виконання яких є бажаним для того, щоб розклад занять був близьким до оптимального.

5 ОПИС ЗАДАЧІ ОПТИМІЗАЦІЇ В ТЕРМІНАХ ГЕНЕТИЧНОГО АЛГОРИТМУ

5.1 Вхідна інформація

Хромосома – це набір генів. У нашому випадку цим набором являються заняття в розкладі. Хромосоми представляють у вигляді масиву даних, кожний елемент якого – це одне заняття в розкладі (формула 4.7).

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

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

5.2 Формування початкової популяції

Спочатку випадковим чином створюється деяка початкова популяція. Навіть якщо популяція виявиться абсолютно не конкурентоздатною, генетичний алгоритм все одно достатньо швидко переведе її в придатну для життя популяцію. Таким чином, на цьому кроці можна не старатися зробити надто пристосованих осіб, достатньо, щоб вони відповідали формату осіб популяції, і на них можна було порахувати функцію пристосованості.

Етапи створення особини для популяції представлено на Рис. 5.1 (див. наступна сторінка):

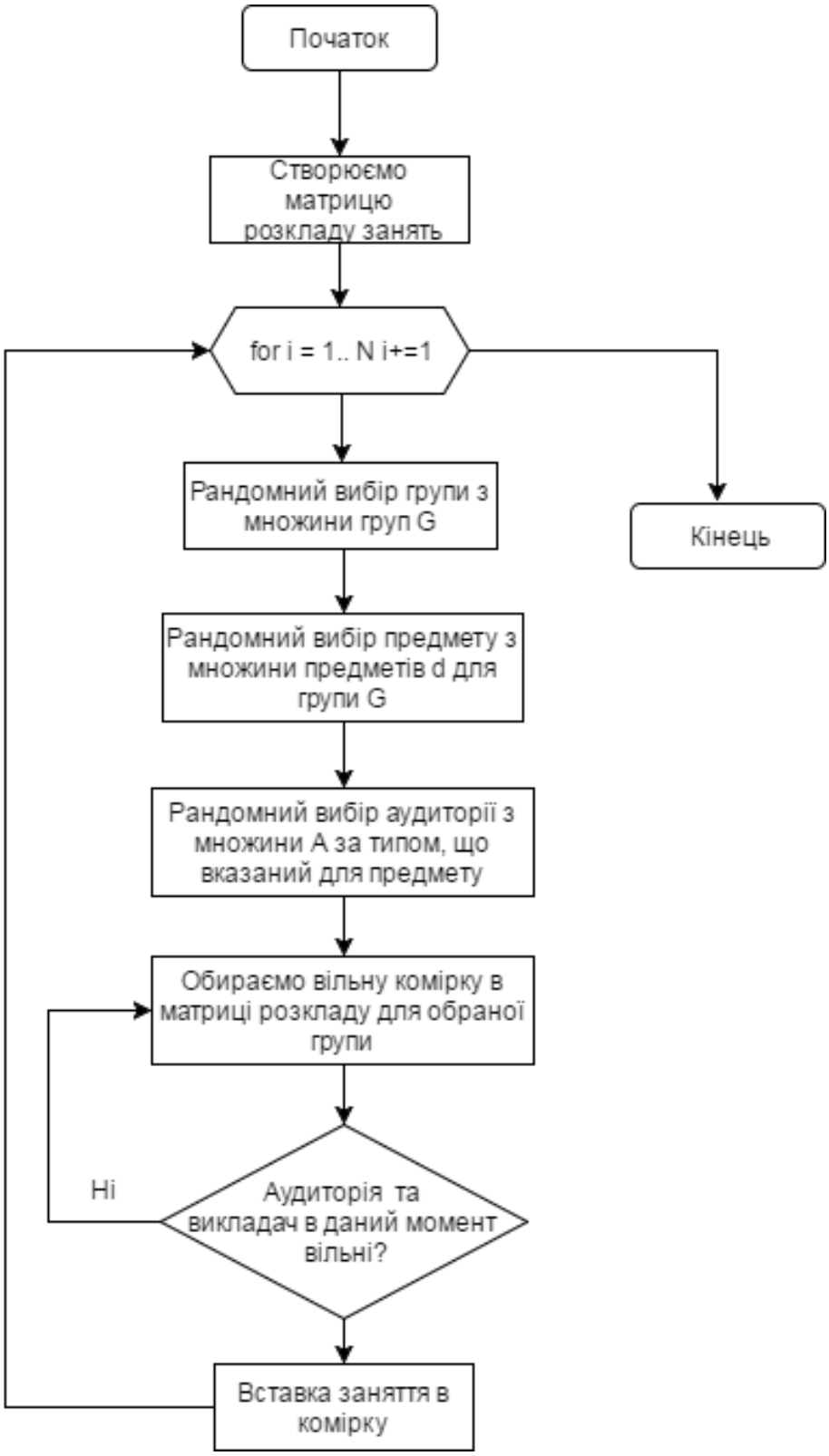

Рис. 5.1 – Алгоритм створення початкової популяції

5.3 Фітнес-функція

Рішення щодо хромосоми (особини) оцінюється за допомогою значення цільової функції, тобто перевіряється, чи придатна особина для подальшого використання. Цільову функцію часто називають фітнес-функцією чи функцією придатності. Значення цієї функції оцінюється для кожної особини популяції окремо і на його основі, приймається рішення використовувати цю хромосому чи перейти до етапу покращення особин популяції, за допомогою генетичних операторів схрещування та мутації.

Для задачі складання і оптимізації розкладу занять було розроблено цільову 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. Наприклад, у розкладі занять є 5 вікон. Тоді ступінь не виконуваності для обмеження №1 (у розкладі немає бути вікон) становить 5. Штраф для одного обмеження обчислюється так:

$$P_i = n_i w_i \qquad (5.1)$$

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.

$$F(H_j) = \frac{1}{1+\sum_{i=1}^{m} P_i} \to max \qquad (5.2)$$

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]

5.4 Селекція

Селекція - це вибір тих хромосом, які будуть брати участь в створенні нащадків для наступної популяції, тобто для чергового покоління. Такий вибір проводиться відповідно до принципу природного відбору, за яким найбільші шанси на участь в створенні нових особин мають хромосоми з найбільшими значеннями функції пристосованості. У нашому алгоритмі буде застосовано метод рулетки.

Заснований на принципі колеса рулетки метод селекції вважається для генетичних алгоритмів основним методом відбору особин для батьківської популяції з метою подальшого їх перетворення генетичними операторами, такими як схрещування і мутація. Незважаючи на випадковий характер процедури селекції, батьківські особини вибираються пропорційно значенням їх функцій пристосованості: кожній хромосомі зіставлений сектор колеса рулетки, величина якого встановлюється пропорційною значенню функції пристосованості даної хромосоми:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \qquad (5.3)$$

p_i – це величина сектора рулетки (імовірність, з якою особина буде відібрана для схрещування);

i – це номер особини, що досліджується;

N – це кількість особин, що міститься в початковій популяції;

f_i – це значення цільової функції для і-ї особини.

Тому, чим більше значення функції пристосованості, тим більший сектор на колесі рулетки. А отже, тим вищий шанс, що буде обрана саме ця хромосома. [4]

5.5 Схрещування

Схрещування – це один із видів оператора рекомбінації генетичного алгоритму. В науковій літературі зустрічається ще під назвою кроссовер чи кросинговер.[5]

Метою оператора схрещування є породження з наявної множини рішень нового, в якому кожна хромосома буде нащадком деяких двох елементів попередньої популяції, тобто нести в собі частково інформацію кожного батька. Допускається ситуація, коли обидва батька подані одним і тим же елементом популяції.[4]

Схрещування відбуватиметься наступним чином:

1) З батьківської популяції обираємо дві особини.

2) Від першого батька успадковується розташування предметів по часовим інтервалам. Від другого батька успадковується розташування аудиторій по часовим інтервалам.

Інші методи схрещування (такі як точкове схрещування) не будуть ефективними в задачі оптимізації розкладу занять, тому що більшість створених нащадків будуть відкидатись через порушення жорстких обмежень розкладу (проведення двох занять в одній аудиторії одночасно, тощо).

5.6. Мутація

Оператор мутації полягає в зміні генів у випадково вибраних позиціях. На відміну від оператора схрещування, який використовуються для поліпшення структури хромосом, метою оператора мутації є диверсифікація, тобто підвищення різноманітності пошуку і введення нових хромосом в популяцію для того, щоб більш повно досліджувати простір пошуку. Мутація ініціює різноманітність в популяції, дозволяючи проглядати більше точок в просторі пошуку і долати таким чином локальні екстремуми в ході пошуку.

Необхідно відзначити, що оператор мутації є основним пошуковим оператором і існують методи, що не використовують інших операторів окрім мутації.

Для вирішення нашої задачі будемо використовувати просту мутацію, яка використовується для бінарних, гомологічних, числових і векторних хромосом.[2]

Мною буде реалізовано наступні алгоритми мутації, які будуть протестовані в ході написання коду програми, і найкращий алгоритм буде застосовано для генетичного алгоритму:

1) Зміна аудиторії

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Обираємо випадково заняття.

Крок 3. Змінюємо його аудиторію на 1.

Крок 4. Якщо жорсткі обмеження не порушились, то завершуємо мутацію. Інакше зменшуємо значення аудиторії на 1.

2) Зміна пар

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Випадково обираємо два заняття з розкладу для групи.

Крок 3. Міняємо їх місцями.

3) Зміна аудиторій

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Випадково обираємо два заняття з розкладу для групи.

Крок 3. Міняємо їх аудиторії місцями.

4) Зміна часу проведення заняття.

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Випадково обираємо заняття з розкладу для групи.

Крок 3. Збільшуємо або зменшуємо випадково час його проведення.

5) Послідовний зсув занять

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Випадково обираємо день занять в розкладі.

Крок 3. Обираємо одне чи декілька занять у цей день.

Крок 3. Зсуваємо ці заняття на 1 пару вперед.

Крок 4. Якщо заняття зсунулись на інше заняття, які відбувається, то це заняття також зсовується на 1 пару вперед. Перехід на крок 5. Інакше завершуємо мутацію.

Крок 5. Якщо це заняття – 5 пара, то воно зсовуються на першу пару наступного дня.

Крок 6. Якщо перша пара наступного дня також занята, то ця пара переноситься на місце першої пари, що була обрана для зсуву в кроці 3. Завершуємо мутацію.

6) Випадкова зміна аудиторії

На рис. 5.2 зображено блок-схему даного алгоритму мутації.

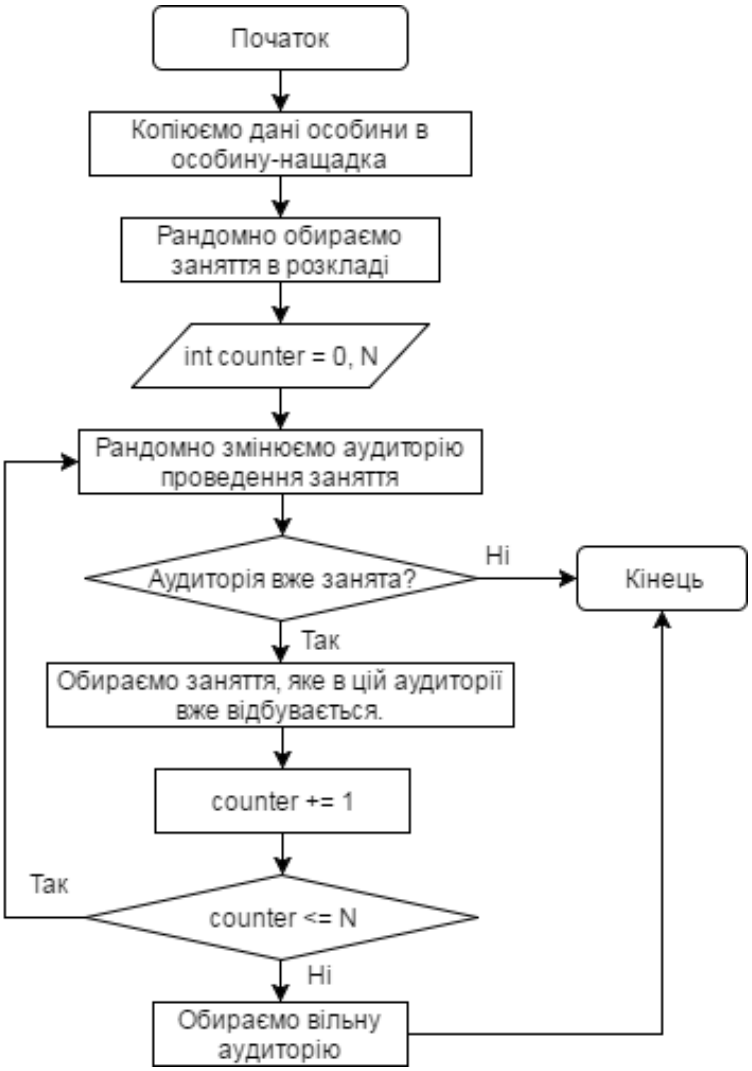

Рис. 5.2 – Алгоритм мутації шляхом випадкової зміни аудиторії заняття

7) Зміна часу для всіх груп

Крок 1. Копіюємо дані особини в особину-нащадка.

Крок 2. Випадково обираємо два стовпчики з матриці розкладу занять (два часи проведення занять).

Крок 3. Міняємо їх місцями.

5.7 Висновок

У даному розділі було детально описано самостійно розроблений генетичний алгоритм оптимізації розкладу занять. Було описано задачу оптимізації в його термінах, визначено основні його етапи (формування початкової популяції, селекція, схрещування, мутація, вибір найкращого варіанту). Можна зазначити, що підібрані хромосоми та особини займатимуть багато пам’яті та сам процес генерування розкладу програмно займатиме багато оперативної пам’яті, проте виграш буде у оптимальності згенерованого розкладу та більша гарантія того, що близький до оптимального варіант не буде відсіяно на одному з етапів роботи генетичного алгоритму.

6 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Для зручності реалізації програмного продукту, що складатиме розклад з допомогою генетичного алгоритму, складено спеціальні діаграми, що ілюструють функціонал певних частин та програми повністю.

6.1 ARIS Objective Diagram

Рис. 6.1 - Діаграма цілей дипломного проекту

6.2 DFD (IDEF0) діаграми

Діаграма на рис. 6.2 ілюструє в загальному те, що управляє програмою генерування розкладу занять (зверху), що подається на вхід (зліва), що використовується для його генерування (знизу) та який результат отримаємо на виході (справа).

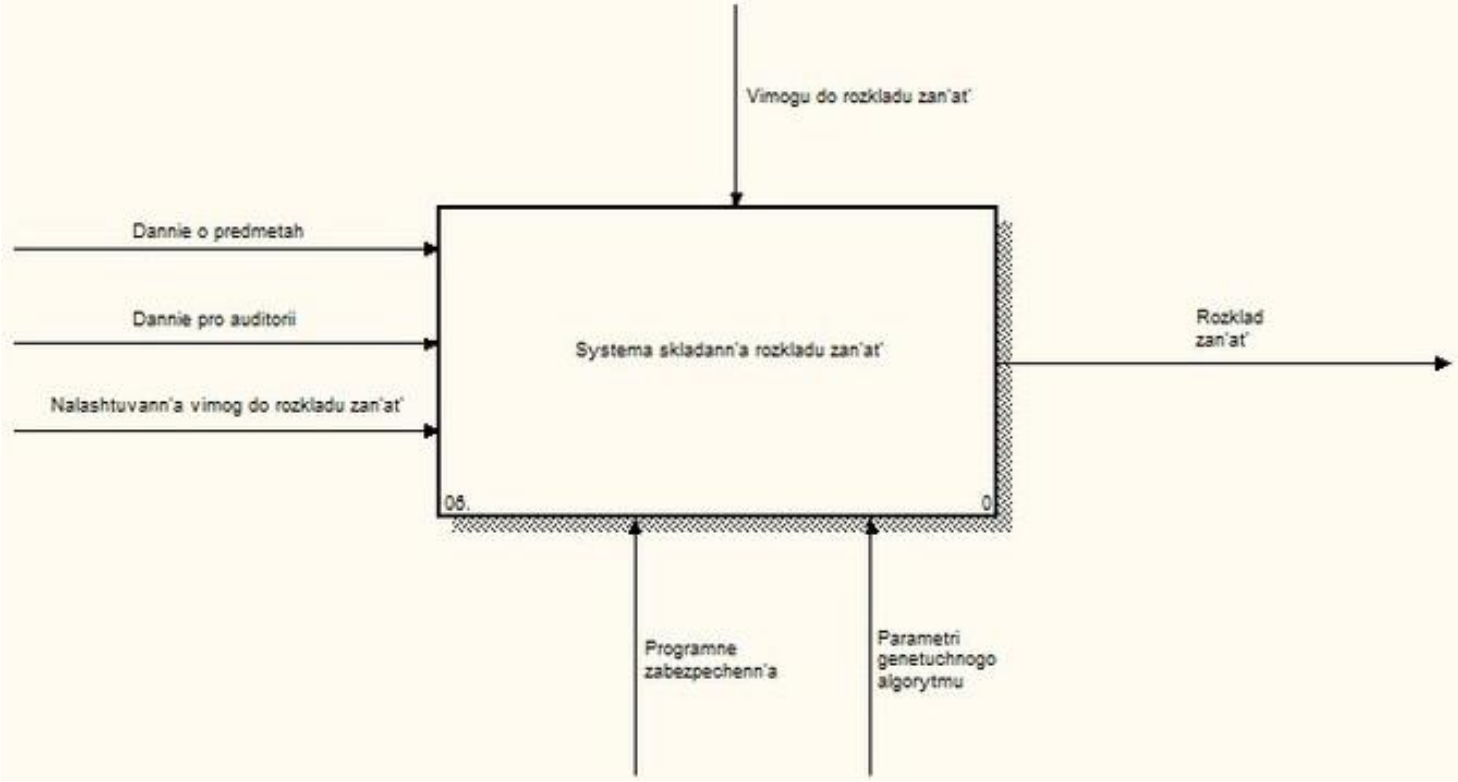

Рис. 6.2 - Загальна DFD діаграма.

На рисунку 6.3 ілюструється розширена DFD-діаграма, що уточнює функції, що має система формування розкладу занять. Це оператори генетичного алгоритму, з допомогою якого і формується розклад. Це формування популяції, формування функції пристосованості, з допомогою їх комбінування відбувається пошук оптимального розкладу, після чого найкращий варіант розкладу подається на вихід програми.

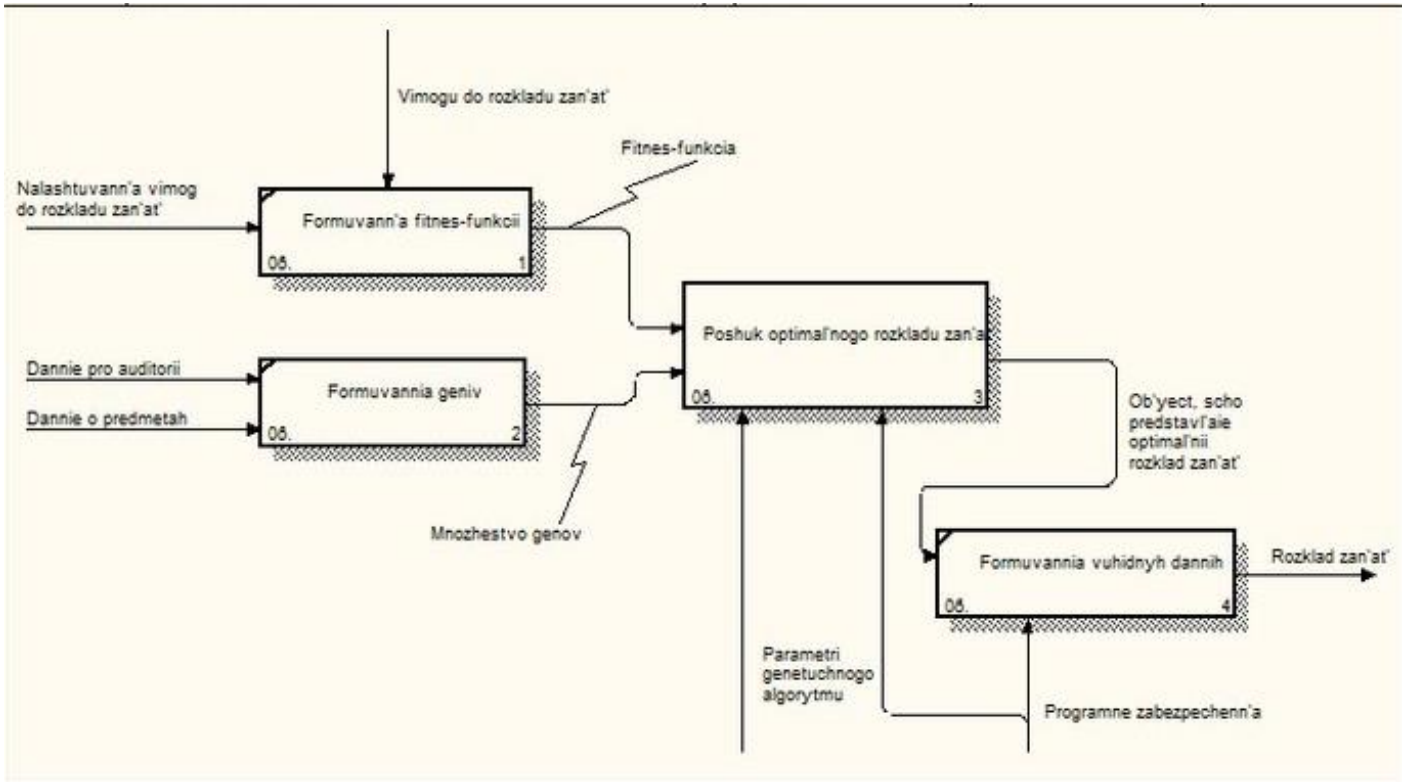

Рис. 6.3 - Загальна діаграма

Діаграма на рис. 6.4 ілюструє з яких етапів складається процес пошуку оптимального розкладу занять. Для початку формується початкова популяція особин, серед яких відбувається відбір найкращих особин для схрещування, яке є наступним етапом оптимізації розкладу. Після схрещування особини, що потребують покращення значення фітнес-функції, зазнають мутації (зміни значення якогось гена в допустимих межах). Після цього оптимальний розклад занять подається на наступний етап формування розкладу.

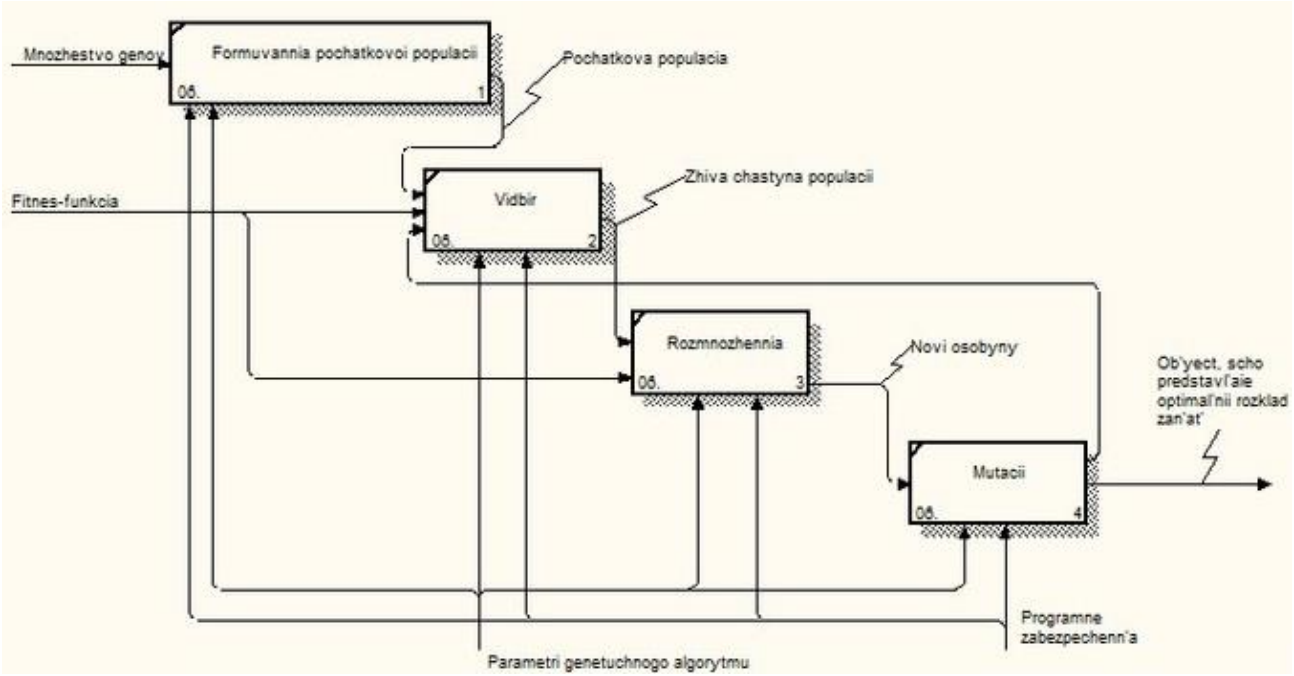

Рис. 6.4 - Декомпозиція блоку “Пошук оптимального розкладу занять”.

6.3 Загальна діаграма прецендентів

На рис. 6.5 представлена діаграма прецедентів, яка ілюструє які функції з програмою для формування розкладу занять можуть виконувати дійові особи. Дійовою особою є тільки користувач, який вводить вхідну інформацію щодо майбутнього розкладу і генерує його. Генерація включає в себе налаштування вимог до розкладу та зберігання готового розкладу в файл з розширенням .xls.

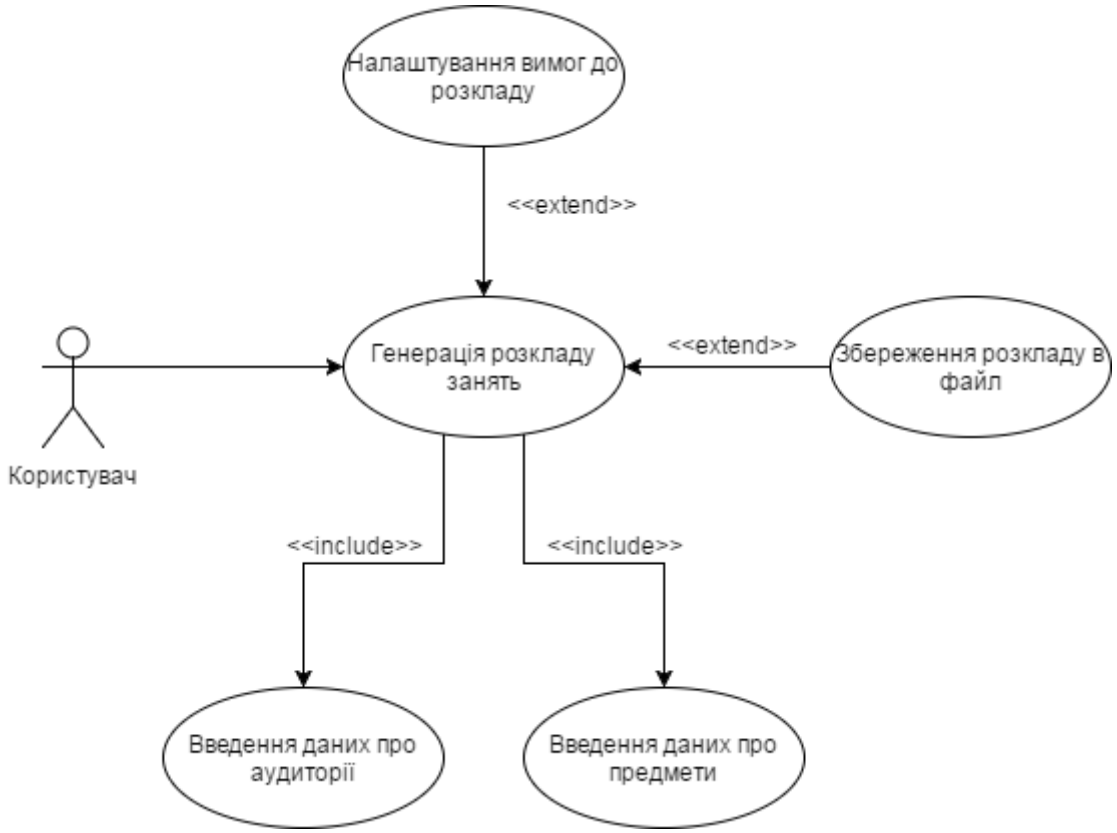

Рис. 6.5 - Діаграма прецедентів

6.4 Характеристика та результати роботи програмного продукту

Для формування розкладу занять мною було розроблено програмний продукт, що формує розклад за допомогою самостійно розробленого генетичного алгоритму. Програму реалізовано на кросплатформовій мові програмування Java та протестовано на вхідних даних для 4 груп кафедри системного проектування. Програма зберігає згенерований розклад у файлі .xls. Результати представлені в таблицях 6.6 – 6.9.

Таблиця 6.6 – Розклад занять для групи ДА-51

DA-51	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50
Monday	Philosophy, lec 105 Zuev	Num. methods, prac 105 Bulakh		Physical training ----
Tuesday		Comp. circ., lec 304 Stikanov	Num. met., lab 206 Golubova	Phys. training 103 ----
Wednesday		Comp. arc., lec 309 Artuhov	ICT, lec 310 Kapshuk	Com. circ., lab 303 Kirusha
Thursday	TOP, lec 305 Stus	Comp. arc., lab 203 Britov	TOP, prac 304 Stus	ICT, lab 208 Kirusha
Friday	Num. met., lec 301 Petrenko	Comp.circ., prac 304 Giorgizova	English, prac 304 Dukhanina	

Таблиця 6.7 – Розклад занять для групи ДА-52

DA-52	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday	Philosophy, lec 105 Zuev		Computer architecture, lab 203 Britov		
Tuesday		Computer circuitry, lec 304 Stikanov	Information coding theory, lab 203 Kirusha	Computer circuitry, lab 201 Kirusha	Physical training ----
Wednesday		Computer architecture, lec 309 Artuhov	Information coding theory, lec 310 Kapshuk	Physical training ----	
Thursday	Theory of probability, lec 305 Stus	Computer circuitry, prac 304 Giorgizova	English, prac 205 Dukhanina	Theory of probability, prac 206 Stus	
Friday	Numerical Methods, lec 301 Petrenko	Numerical Methods, lab 206 Golubova	Numerical Methods, prac 105 Golubova		

Таблиця 6.8 – Розклад занять для групи ДА-61

DA-61	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday		Discrete math, prac 103 Stus	Discrete maths, lec 306 Stus		Algorithms and prog., lab 104 Besnosyk
Tuesday	Algorithms and prog., lec 302 Romanov	Algorithm analysis, lec 307 Romanov		Algorithm analysis, lab 306 Besnosyk	
Wednesday	Physics, lec 203 Kalita	Math analysis, lec 306 Bokhonov	Physics, prac 203 Kalita	Math analysis, prac 310 Bokhonov	Physical training 305 ----
Thursday	Linear Algebra, lec 103 Minarko	History, lec 308 Drabko	English, prac 103 Dukhanina	Linear Algebra, prac 102 Minarko	
Friday		Physics, lab 306 Lyashenko	Physical training 305 ----		

Таблиця 6.9 – Розклад занять для групи ДА-62

DA-62	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50
Monday		Physical training 305 ----	Discrete mathematics, lec 306 Stus	Discrete mathematics, prac 103 Stus
Tuesday	Algorithms and programming, lec 302 Romanov	Algorithm analysis, lec 307 Romanov	Algorithms and programming, lab 104 Besnosyk	
Wednesday	Physics, lec 203 Kalita	Mathematical analysis, lec 306 Bokhonov	Mathematical analysis, prac 310 Bokhonov	Physics, prac 203 Kalita
Thursday	Linear Algebra, lec 103 Minarchenko	History, lec 308 Drabko	Linear Algebra, prac 104 Minarchenko	
Friday	Physical training 305 ----	English, prac 103 Dukhanina	Physics, lab 306 Lyashenko	

6.5 Висновок

У даному розділі було описано реалізацію програмного продукту, з допомогою якого буде генеруватись розклад занять. Було продемонстровано спеціальні діаграми функціонування програми, проілюстровано результати роботи програмного продукту.

7 ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ

В даній дипломній роботі було реалізовано програмний продукт, що використовує самостійно розроблений генетичний алгоритм оптимізації для формування розкладу занять на мові програмування Java. Для порівняння алгоритмів було взято готову реалізацію іншого генетичного алгоритму на мові програмування C++, а також розклад занять, складений вручну, з однаковим набором вхідних даних для 4 навчальних груп.

7.1 Опис алгоритму для першого готового рішення, реалізованого на мові програмування C++

7.1.1 Опис вхідної інформації

Розклад занять задається з допомогою наступних об’єктів:

1) Множина аудиторій

$$A = \{a_1, \dots, a_n\}, \quad (7.1)$$

$$a_i = \{id, n, c\}, \quad (7.2)$$

де a_i – це аудиторія;

id – індекс аудиторії;

n – номер аудиторії;

c – місткість аудиторії (у кількості студентів).

2) Множина часових інтервалів

$$T = \{id, d_i, int_i\}, \quad (7.3)$$

де id – ідентифікатор інтервалу;

d_i – день проведення заняття;

int_i – часовий інтервал проведення заняття;

3) Множина викладачів:

$$P = \{id, name\}, \quad (7.4)$$

де id – індекс викладача;

name – прізвище викладача.

4) Множина предметів:

$$D = \{id, code_i, n_i, p\}, \quad (7.5)$$

де id – ідентифікатор предмету;

$code_i$ – код предмету;

n_i – назва предмету та його тип;

p – множина викладачів, що можуть цей предмет викладати.

5) Множина груп:

$$G = \{id, name, size, d\}, \quad (7.6)$$

де id – індекс групи;

$name$ – назва групи;

$size$ – кількість студентів, що навчаються в групі;

d – предмети, що викладаються для цієї групи;

7.1.2 Функція пристосованості

Функція пристосованості залежить від коефіцієнту, що враховує кількість вікон для особини розкладу занять v. Чим менше вікон у розкладі, тим більший коефіцієнт:

$$F_H = \frac{V}{k} \rightarrow max, \quad (7.7)$$

де k – кількість вікон у розкладі H.

7.1.3 Формування початкової популяції

Випадковим чином формується задане число особин. Формування кожної особини відбувається наступним чином: по черзі для кожного гена першої та другий хромосоми, умовно позначає цикл (блок) занять, приписується деяке значення, для першої хромосоми цим значенням буде номер аудиторії з числа допустимих (відповідних) для даного циклу (блоку) заняття, для другої хромосоми - номер пари з підмножини допустимих для даного циклу (блоку) пар. Аналогічним чином формуються наступні особини популяції.[2]

7.1.4 Селекція особин

На даному етапі відбувається відбір (селекція) найбільш пристосованих особин (варіантів розкладу), що мають найкращі значення функції придатності порівняно з іншими особинами. Пропонується використати метод, званий «елітним відбором» або «елітної стратегією», який при вирішенні даного завдання полягає в наступному: з попередньої популяції обирається тільки певна кількість окремих особин (варіантів розкладів), що мають найкращі значення вагової функції , що відбиває виконання бажаних вимог. Виявлені таким способом «елітні »особини без будь-яких змін переходять в наступне покоління. Залишок «вільних місць» в новій популяції заповнюється особинами, отриманими в результаті схрещування (кросовера) і мутації особин.[4]

7.1.5 Схрещування

Схрещування (кросинговер) особин проходить за наступною схемою: випадковим чином з числа найбільш пристосованих вибираються дві особини. Далі для кожної пари відібраних особин випадковим чином розігруються позиції гена (локус) L1 і L2 та проводиться обмін ділянками генетичного коду між відповідними хромосомами батьківських особин. При запропонованій

схемі схрещування двох особин не відбувається зміна загальної кількості генів, умовно позначають цикли занять. Також в кожній з хромосомі обраних особин не змінюється порядок слідування генів всередині кожної хромосоми. З огляду на той факт, що при виборі номера аудиторії і номера пари строго враховувався вид заняття, після процедури схрещування не вимагається перевірки особин на «правильність» вибору аудиторії і номера пари для проведення занять. Це дозволяє уникнути зайвих перевірок на коректність особин, отриманих в результаті процедури схрещування.[4]

7.1.6 Мутація

До деяких осіб застосовується оператор мутації. У пропонованому алгоритмі цей оператор змінює значення декількох генів особини на інші допустимі для даного гена значення, наприклад, замінює номер аудиторії для деякого заняття на інший номер аудиторії з допустимого для даного типу занять підмножини аудиторій. Отримані в результаті такої мутації варіанти розкладів залишаються допустимими по м’яким умовам.[3]

7.1.7 Перевірка умови зупинки алгоритму

В результаті застосування операторів відбору, схрещування і мутації формується популяція нащадків, яка замінює батьківську популяцію, після чого виконується перевірка умови зупинки алгоритму. Пропонується в якості такого умови використовувати наступне: певний відсоток (60-80%) особин має однакові значення функції пристосованості. При виконанні даної в алгоритмі умови зупинки здійснюється перехід до наступного етапу, інакше відбувається перехід до етапу 2 і процес пошуку оптимального рішення триває.[1]

7.1.8 Результат роботи алгоритму

Даний алгоритм було реалізовано на мові програмування C++. Розклад було складено для 4 груп кафедри системного проектування. Результати представлено в таблиці 7.1– 7.4.

Таблиця 7.1 – Розклад занять для групи ДА-51

DA-51	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday	Numerical Methods, prac 310 Golubova	Computer architecture, lec 310 Artuhov	Information coding theory, lab 101 Sergeev-Gorchynsky		Physical training 203 ----
Tuesday	Information coding theory, lec 105 Kapshuk	Computer circuitry, lec 102 Stikanov			Computer architecture, lab 312 Kirusha
Wednesday	Theory of probability, lec 302 Stus	Computer circuitry, prac 102 Giorgizova		Computer circuitry, lab 102 Kirusha	English, prac 309 Dukhanina
Thursday	Philosophy, lec 310 Zuev		Numerical Methods, lab 308 Golubova		
Friday			Theory of probability, prac 203a Stus		Numerical Methods, lec 301 Petrenko

Таблиця 7.2 – Розклад занять для групи ДА-52

DA-52	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday	Philosophy, lec 103 Zuev				Numerical Methods, prac 304 Bulakh
Tuesday	Information coding theory, lab 304 Sergeev-Gorchynsky	Computer circuitry, lec 307 Stikanov	Computer circuitry, prac 208 Giorgizova		Numerical Methods, lab 309 Golubova
Wednesday			Numerical Methods, lec 206 Petrenko	Theory of probability, prac 302 Stus	
Thursday		Computer architecture, lab DA-52 Britov	Theory of probability, lec 309 Stus	Computer circuitry, lab 305 Kirusha	Computer architecture, lec 310 Artuhov
Friday	Physical training 103 ----	Information coding theory, lec 208 Kapshuk	English, prac 306 Gaidenko		

Таблиця 7.3 – Розклад занять для групи ДА-61

DA-61	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday			Physics, prac 203 Kalita		
Tuesday	English, prac 302 Gaidenko	History, lec 101 Drabko		Algorithm analysis, lec 301 Romanov	Algorithm analysis, lab 304 Romanov
Wednesday	Mathematical analysis, prac 305 Bokhonov	Physics, lab 309 Lyashenko		Physical training 310 ----	Algorithms and prog., lab 208 Besnosyk
Thursday	Linear Algebra, prac 104 Minarchenko	Discrete mathematics, lec 308 Stus	Physics, lec 105 Kalita	Mathematical analysis, lec 304 Bokhonov	Discrete mathematics, prac 105 Stus
Friday	Linear Algebra, lec 105 Minarchenko			Algorithms and programming, lec 102 Romanov	

Таблиця 7.4 – Розклад занять для групи ДА-62

DA-62	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday	Discrete mathematics, lec 310 Stus	Physics, lec 304 Kalita	Discrete mathematics, prac 105 Stus		Math analysis, lec 104 Minarko
Tuesday	Linear Algebra, lec 104 Minarchenko	Algorithm analysis, lec 104 Romanov	History, lec 307 Drabko		
Wednesday		Algorithms and programming, lec 203a Romanov	Mathematical analysis, prac 104 Bokhonov	Physics, lab 310 Lyashenko	Linear Algebra, prac 304 Minarko
Thursday	Linear Algebra, prac 104 Minarchenko	Discrete mathematics, lec 308 Stus	Physics, lec 105 Kalita	Math analysis, lec 304 Bokhonov	Discrete math, prac 105 Stus
Friday	Algorithm analysis, lab 104 Besnosyk			Physical training 203a ----	

7.2 Ручне формування розкладу занять

Для порівняльної характеристики для того ж набору вхідних даних сформовано розклад занять вручну. Результати представлено в таблицях 7.5 – 7.8.

Таблиця 7.5 – Розклад для групи ДА-51

DA-51	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50	16:10-17:45
Monday		Computer architecture, lec 310 Artuhov	Information coding theory, lab 101 Sergeev-Gorchynsky	Physical training ----	
Tuesday	Information coding theory, lec 208 Kapshuk	Computer circuitry, lec 307 Stikanov	Computer architecture, lab 305 Kirusha	Computer circuitry, lab 305 Kirusha	
Wednesday	Numerical Methods, lec 301 Petrenko	Computer circuitry, prac 309 Giorgizova	English, prac 309 Dukhanina		
Thursday		Philosophy, lec 310 Zuev	Numerical Methods, lab 308 Golubova	Numerical Methods, prac 310 Golubova	
Friday	Theory of probability, lec 302 Stus	Theory of probability, prac 203a Stus	Physical training ----		

Таблиця 7.6 – Розклад для групи ДА-52

DA-52	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50
Monday		Computer architecture, lec 310 Artuhov	Physical training -----	Information coding theory, lab 101 Sergeev-Gorchynsky
Tuesday	Information coding theory, lec 208 Kapshuk	Computer circuitry, lec 307 Stikanov	English, prac 306 Gaidenko	
Wednesday	Numerical Methods, lec 301 Petrenko	Computer architecture, lab 203 Britov	Computer circuitry, prac 310 Giorgizova	
Thursday	Numerical Methods, lab 309 Golubova	Philosophy, lec 310 Zuev	Computer circuitry, lab 305 Kirusha	Numerical Methods, prac 304 Bulakh
Friday	Theory of probability, lec 302 Stus	Physical training ----	Theory of probability, prac 203a Stus	

Таблиця 7.7 – Розклад занять для групи ДА-61

DA-61	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50
Monday	Physics, lec 312 Kalita	Physics, prac 310 Kalita	Algorithms and programming, lab 208 Besnosyk	Physical training ----
Tuesday		Algorithm analysis, lec 301 Romanov	Algorithms and programming, lec 301 Romanov	Algorithm analysis, lab 304 Romanov
Wednesday		History, lec 101 Drabko	English, prac 302 Gaidenko	
Thursday	Discrete mathematics, lec 308 Stus	Discrete mathematics, prac 105 Stus	Mathematical analysis, lec 304 Bokhonov	Mathematical analysis, prac 305 Bokhonov
Friday	Linear Algebra, lec 105 Minarchenko	Linear Algebra, prac 105 Minarchenko	Physics, lab 309 Lyashenko	Physical training ----

Таблиця 7.8 – Розклад занять для групи ДА-62:

DA-62	8:30-10:05	10:25-12:00	12:20-13:55	14:15-15:50
Monday	Physics, lec 312 Kalita	Physical training ----	Physics, prac 310 Kalita	Algorithms and programming, lab 208 Besnosyk
Tuesday		Algorithm analysis, lec 301 Romanov	Algorithms and programming, lec 301 Romanov	Algorithm analysis, lab 104 Besnosyk
Wednesday	English, prac 302 Gaidenko	History, lec 101 Drabko		
Thursday	Discrete mathematics, lec 308 Stus	Mathematical analysis, prac 304 Bokhonov	Mathematical analysis, lec 304 Bokhonov	Discrete mathematics, prac 105 Stus
Friday	Linear Algebra, lec 105 Minarchenko	Physics, lab 310 Lyashenko	Linear Algebra, prac 105 Minarchenko	Physical training -----

7.3 Порівняльна характеристика існуючих рішень

Порівняння було проведено за наступними параметрами: кількість вікон у розкладі, мінімальна кількість днів, у які проводяться заняття, проведення лабораторних занять в один день, наявність проміжних вихідних днів, кількість пар в останній день навчання в тижні, час, витрачений на складання розкладу та порівняння фітнес-функцій.

Результати порівнянь були зведені в таблицю 7.9:

Таблиця 7.9 – Порівняння існуючих рішень

Критерій	Власне рішення	Готове рішення	Ручний розрахунок
Кількість вікон у розкладі	4	15	0
Кількість днів, у які проводяться заняття	5	5	5
Проведення лабораторних занять в один день	Не виконується	Не виконується	Не виконується
Наявність проміжних вихідних днів у розкладі	Немає	Немає	Немає
В останній день тижня навчання кількість пар менша ніж у інші дні	Виконується	Не виконується	Виконується
Тривалість складання розкладу	15 секунд	30 секунд	1 година
Значення фітнес-функції	1.0	1.0	Не використовується

7.4 Висновок

У даному розділі було проведено дослідження існуючих рішень задачі оптимізації розкладу занять та проведено порівняльну характеристику цих рішень з власне розробленим. В результаті, за більшістю пунктів порівняння рішення не відрізняються. За м’якими обмеженнями найкращий результат показує розклад, складений вручну. Проте рішення, розроблене мною, формує розклад набагато швидше аніж розрахунок вручну. Мій результат програє ручному лише по одному пункту, але виграє по швидкості формування. Також моє рішення по всім пунктам або не відрізняється, або краще за готове автоматизоване рішення. На основі цих порівнянь можна зробити висновок, що мій варіант формування розкладу занять є найоптимальнішим.

8 ПЕРЕВАГИ ТА НЕДОЛІКИ РОЗРОБЛЕНОГО ГЕНЕТИЧНОГО АЛГОРИТМУ ОПТИМІЗАЦІЇ

8.1 Переваги алгоритму

До переваг самостійно розробленого алгоритму можна віднести:

- враховано всі м’які умови, побажання студентів та викладачів щодо майбутнього розкладу занять;
- висока швидкість роботи алгоритму;
- зручне подання вхідної інформації;
- відсутність необхідності у специфічних знаннях про вирішувану задачу;
- концептуальна простота та прозорість реалізації;
- можливість застосування до великого кола задач без внесення серйозних змін у внутрішню структуру методу.

8.2 Недоліки

До недоліків розробленого мною алгоритму відносяться такі:

- розклад можна сформувати для однотижневого навчання;
- є великий шанс на одному з етапів втратити рішення, що може бути близьким до оптимального;
- великий набір даних (розклад), через що сам процес його формування з допомогою генетичного алгоритму займатиме багато оперативної пам’яті.

8.3 Висновок

У даному розділі було розглянуто основні переваги та недоліки, що стосуються генетичного алгоритму оптимізації розкладу занять, розробленого самостійно.

9 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для складання розкладу занять в університеті. Програма була створена на мові програмування Java. Розробка проводилась у середовищі Intellij Idea.

Програмний продукт призначено для використання на ПК під управлінням ОС Windows XP, 7, 8, 10.

9.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на ПК під управлінням ОС Windows XP, 7, 8, 10;
- забезпечувати стабільну роботу, а при некоректних вхідних даних чи у разі виникненні помилок в процесі роботи програми надавати інформацію про їх походження;
- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

9.2 Обґрунтування функцій програмного продукту

9.2.1 Формування варіантів функцій

Головна функція F0 – розробка програмного продукту, який складає розклад занять за певними вхідними даними. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – мова програмування;
F2 – середовище розробки;
F3 – спосіб зберігання даних;

Кожна з основних функцій може мати декілька варіантів реалізації:

Функція F1:

a) C#;
b) C++;

Функція F2:

a) Microsoft Visual Studio 2015;
b) Xamarin Studio;

Функція F3:

a) Не зберігати дані на пристрої;
b) Зберігати дані у вигляді файлу .xls;

9.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи. На її основі побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 9.1).

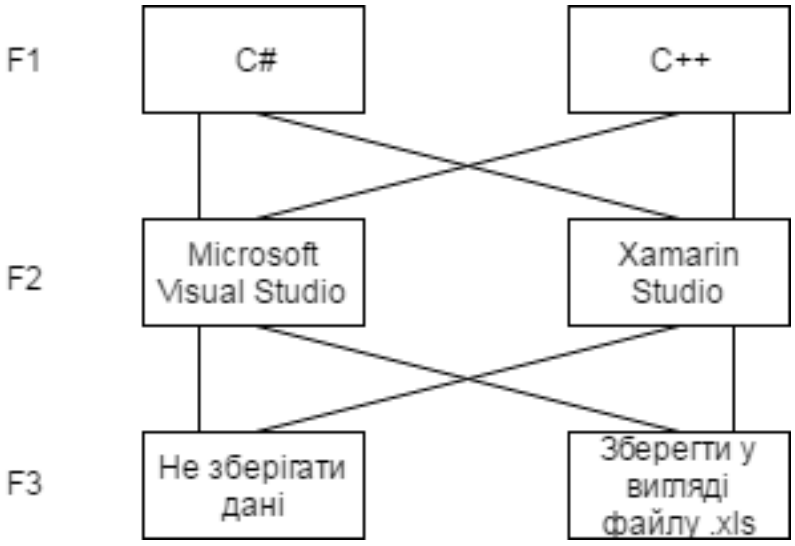

Рисунок 9.1 – Морфологічна карта системи

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 9.1 – Позитивно-негативна матриця варіантів основних функцій

Ф-ї	**Вар-ти**	**Переваги**	**Недоліки**
F1	a	Написана більша частина додатків, документації	Програми займають багато пам’яті.
	b	Зручна у використанні	Не кросплатформова.
F2	a	Підтримує багато різних мов	Необхідно підлаштовувати під вибрану мову
	b	Підлаштована спеціально для розробки під C#	Потребує багато ресурсів ЕОМ
F3	a	Потребує менше пам’яті	Потребує більше часу на обробку запитів
	b	Потребує менше часу на обробку запитів	Потребує більше пам’яті

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, як такі, що вони не відповідають поставленим перед програмним продуктом технічним вимогам.

Функція F1.

Варіант b) можна відкинути, тому що це не кросплатформова мова програмування, не призначена для розробки складних додаткыв з графычним ынтерфейсом.

Функція F2.

Варіант b) можна відкинути, тому що Xamarin Studio має менше функцій.

Функція F3

Варіанти a) та b) вважаємо гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізаціїї ПП:

1. F1a – F2a – F3a

2.F1a – F2 a – F3b

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

9.3 Обґрунтування системи параметрів ПП

Для характеристики розроблюваної програми можна використати такі параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті, потрібної для програми;

– *X3* – час обробки запитів користувача;

– *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у табл. 9.3.1.

1) *X1*: параметр, що відображає швидкодію операцій залежно від обраної мови програмування.
2) *X2*: параметр, що відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.
3) *X3*: параметр, що відображає час, який витрачається на дії.
4) *X4*: параметр, який показує розмір програмного коду який необхідно створити безпосередньо розробнику.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту як показано у табл. 9.2.

Таблиця 9.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Гб	16	8	4
Час обробки запитів користувача	X3	с	1000	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	3000	2500	2000

За даними таблиці 9.2 будуються графічні характеристики параметрів – рис. 9.3 – рис. 9.6.

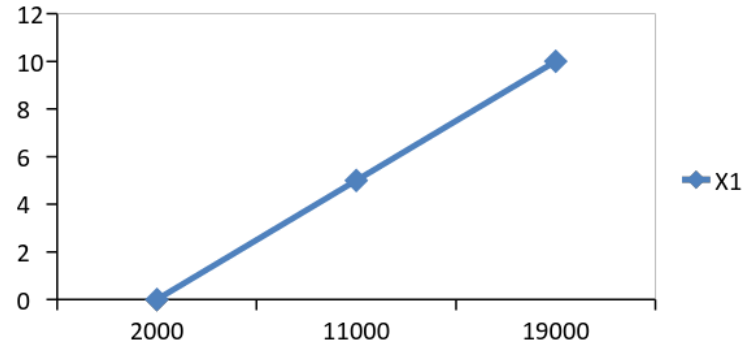

Рисунок 9.3 – X1, швидкодія мови програмування

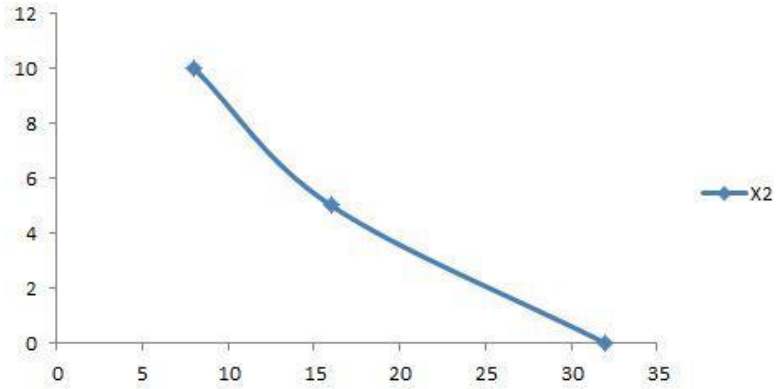

Рисунок 9.4 – X2, об’єм пам’яті для збереження даних

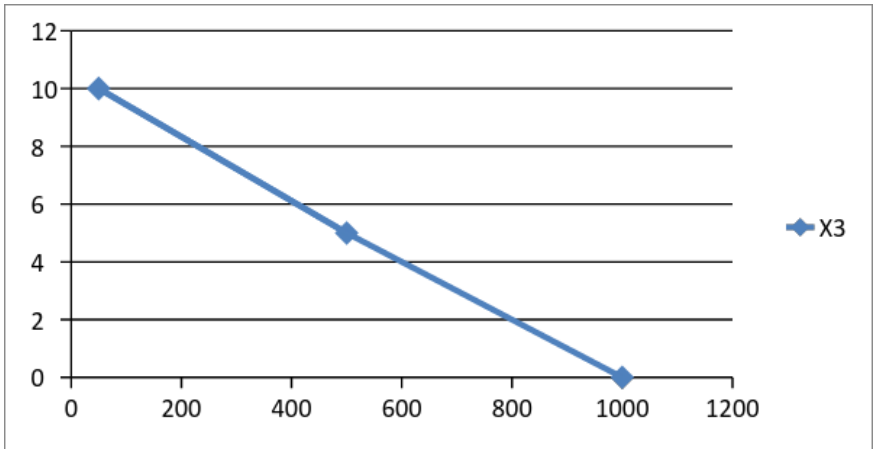

Рисунок 9.5 – X3, час виконання запитів користувача

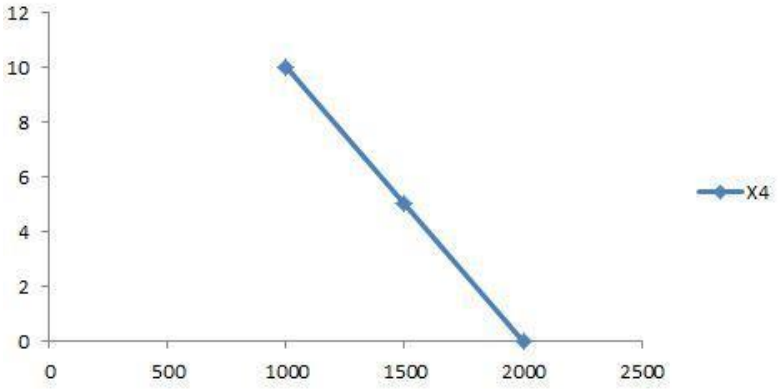

Рисунок 9.6 – X4, потенційний об’єм програмного коду

9.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень. Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

1) визначення рівня значимості параметра шляхом присвоєння різних рангів;
2) перевірку придатності експертних оцінок для подальшого використання;
3) визначення оцінки попарного пріоритету параметрів;
4) обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 9.7

Таблиця 9.7 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$Ri=j=1NrijRij=Nn(n+1)2=105, \qquad (9.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T=1nRij=26,25 \qquad (9.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta i=Ri-T \qquad (9.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S=i=1N\Delta i2=420,75 \qquad (9.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = 12SN2(n3 - n) = 12\cdot420,7572(53-5) = 1,03 > Wk = 0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 9.8.

Таблиця 9.8 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$Kві=bii=1nbi,де\ bi=i=1Naij \qquad (9.5)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$Kві=bi'i=1nbi',де\ bi'=i=1Naijbj \qquad (9.6)$$

Як видно з таблиці 12.3.1, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 9.9 – Розрахунок вагомості параметрів

Параметрихі	Параметрихj				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	bi	Кві	bi1	Кві1	bi2	Кві2
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

9.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо. Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так:

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (9.7)$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Результати розрахунків зведено в табл. 9.9.

Таблиця 9.9 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	b	15000	7.5	0,208	1.56
F2	a	24	2.5	0,158	0.395
F3	a	500	4	0,361	1.444
	b	300	8	0,361	2.888

За даними з таблиці 9.9 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (9.8)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.56 + 0.395 + 1.444 = 3.40$$

$$K_{K2} = 1.56 + 0.395 + 2.888 = 4.84$$

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт рівня якості має найбільше значення.

9.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Але варіант II реалізації програмного забезпечення включає ще одне завдання:
3. Написання алгоритму збереження інформації у вигляді компонента, зручного для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б, завдання 3 до групи Г. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Завдання 3 відноситься за складністю до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \qquad (9.9)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 9.6.1, загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Для третого завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$$Т_р = 8 \text{ людино-днів;}$$

$$К_п = 0,6; \; К_{ст} = 1;$$

$$Т_З = 8 \bullet 0,6 \bullet 1 = 4,8.$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122.4 + 19.44) \cdot 8 = 1134.72 \text{ людино-годин;}$$

$$T_{II} = (122.4 + 19.44 + 4.8) \cdot 8 = 1173.12 \text{ людино-годин;}$$

Найбільш високу трудомісткість має варіант II.

В розробці бере участь один спеціаліст з мобільної розробки з окладом 15 765 грн. Визначимо зарплату за годину за формулою:

$$C_ч = M/(Tm \cdot t) \text{грн.}, \qquad (9.10)$$

де M – місячний оклад працівників;
Tm – кількість робочих днів тиждень;
t – кількість робочих годин в день.

$$C_ч = 15\ 765/(21 \cdot 8) = 93.84 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$C_{ЗП} = Cч \cdot Ti \cdot K_Д, \qquad (9.11)$$

де $C_ч$ – величина погодинної оплати праці програміста;
Ti – трудомісткість відповідного завдання;
$K_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробника за варіантами становить:

I. $C_{ЗП} = 93.84 \cdot 1134.72 \cdot 1.2 = 127778.55$ грн.

II. $C_{ЗП} = 93.84 \cdot 1173.12 \cdot 1.2 = 132102.70$ грн.

Відрахування на єдиний соціальний внесок становить 22% на 01.05.2017. Отримуємо такі значення:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 56372.89 \cdot 0.22 = 28111.28$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 58208.60 \cdot 0.22 = 29062.59$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного аналітика з окладом 15 765 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 15\,765 \cdot 0{,}2 = 37836 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 37836 * (1 + 0.2) = 45403.2 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 45403.2 \cdot 0{,}22 = 9988.70 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 18 888 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 18\,888 = 5430.3 \text{ грн.}, \quad (9.12)$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 18\,888 \cdot 0.05 = 1086.06 \text{ грн.}, \quad (9.13)$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = \qquad (9.14)$$

$$=(365 – 105 – 11 – 16) \cdot 8 \cdot 0.9 = 1677.6 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0.156 \cdot 1.94 = 507.71 \text{ грн.}, \quad (9.15)$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 18\,888 \cdot 0.67 = 12654.96 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \qquad (9.16)$$

$$С_{ЕКС} = 45403.2 + 9988.70 + 5430.3 + 1086.06 + 507.71 + 12654.96 = 75070.93 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 75070.93 / 1706.4 = 43.99 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М\text{-}Г} \cdot T \qquad (9.17)$$

I. $C_М = 43.99 \cdot 1134.72 = 49916.33$ грн.;

II. $C_М = 43.99 \cdot 1173.12 = 51605.55$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67 \qquad (9.18)$$

I. $C_Н = 127778.55 \cdot 0.67 = 85611.63$ грн.;

II. $C_Н = 132102.70 \cdot 0.67 = 88508.81$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \qquad (9.19)$$

I. $C_{ПП} = 127778.55 + 28111.28 + 49916.33 + 85611.63 = 291417.79$ грн.;

II. $C_{ПП} = 132102.70 + 29062.59 + 51605.55 + 88508.81 = 301279.65$ грн.;

9.7. Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \qquad (9.20)$$

$К_{ТЕР1} = 3.4 / 291417.79 = 1.17 * 10 \text{^-}5;$

$К_{ТЕР2} = 4.84 / 301279.65 = 1.61 * 10\text{^-}5;$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР2} = 1.61*10\text{^-}5$.

9.8 Висновок

У даному розділі було проведено функціональний аналіз розробленого програмного продукту з економічної точки зору. Було запропоновано два шляхи реалізації продукту, визначено основні економічні показники та ефективність роботи цих варіантів, на основі чого було визначено, що розробка програмного продукту на мові програмування Java в середовищі Intellij idea буде найкращим.

ВИСНОВКИ

Процес складання розкладу - це надзвичайно складний процес, який потребує від осіб, які ним займаються, неабияких вмінь та навичок. Його математично можна формалізувати як складну оптимізаційну задачу, для вирішення якої сьогодні існує ряд методів.

Було детально розглянуто еволюційний метод розв'язання оптимізаційної задачі, а саме генетичний алгоритм, і можна сказати, що цей алгоритм заснований на аналогії з природними еволюційними процесами. Його перевагою є те, що не висуваються додаткові вимоги до виду цільової функції, алгоритм на кожній ітерації працює з множиною рішень, що дозволяє в багатьох випадках більш детально в порівнянні з градієнтними методами багатовимірної нелінійної безумовної оптимізації аналізувати простір пошуку.

Дослідивши, як застосовується генетичний алгоритм для проблеми складання розкладу в університетах, можна сказати, що цей метод має ряд переваг у порівнянні із іншими:

- відсутність необхідності у специфічних знаннях про вирішувану задачу;
- концептуальна простота та прозорість реалізації;
- можливість розпаралелювання;
- простота кодування вхідної і вихідної інформації;
- можливість застосування до великого кола задач без внесення серйозних змін у внутрішню структуру методу.

В ході виконання переддипломної практичної роботи мною було досліджено застосування генетичного алгоритму для задачі створення і оптимізації розкладу занять в університеті.

На даному етапі мною було виконано наступні завдання:

1. Складено загальний опис досліджуваного об’єкту (розкладу занять).
2. Було сформульовано задачу оптимізації розкладу занять в термінах генетичного алгоритму, а саме:
    1. Складено опис вхідної інформації:
        - генами є предмети, викладачі, аудиторії та групи;
        - хромосомами є заняття;
        - особиною буде розклад занять;
        - популяцією буде список згенерованих розкладів занять.
    2. Було сформульовано критерій якості розкладу (цільова функція, що залежить обернено пропорційно від об’єму порушень м’яких обмежень, що накладаються на розклад, значення якої потрібно максимізувати для отримання оптимального розкладу).
    3. Сформовано алгоритм створення початкової популяції.
    4. Визначено методи селекції для відбору батьківської популяції (турнірний відбір, метод рулетки та метод рангів), серед яких у процесі написання коду шляхом тестування буде вибрано найкращий метод.
    5. Було визначено спосіб схрещування батьківських особин.
    6. Було представлено різні способи мутації особин (зміна значень аудиторій, зміна пар, зміна аудиторій місцями, зміна часу, зсув занять, випадкова зміна аудиторії, зміна часових інтервалів для усіх груп одночасно), найкращий з яких буде обрано в ході проектування архітектури програми шляхом тестування.
3. На основі сформульованого генетичного алгоритму було реалізовано програмний продукт, що формує розклад занять з допомогою даного алгоритму, на мові програмування Java.
4. Розроблене рішення було досліджено і порівняно з існуючим автоматизованим рішенням та розкладом занять. Складеним вручу. Внаслідок цього було зроблено висновок, що розроблений мною алгоритм є найкращим варіантом застосування для формування розкладу занять.

ПЕРЕЛІК ПОСИЛАНЬ

1. Проблемы составления расписания в ВУЗе. – Режим доступу: http://www.pulsar.ru/wiki/detail.php?title=Проблемы+составления+расписания+в+вузе . – Дата доступу: 12.05.2017.
2. Расписание занятий как организационный документ, определяющий режим работы ОУ. – Режим доступу: http://festival.1september.ru/articles/519295/ . – Дата доступу: 15.05.2017.
3. Композиционный генетический алгоритм для составления расписания занятий. – Режим доступу: http://cyberleninka.ru/article/n/kompozitsionnyy-geneticheskiy-algoritm-sostavleniya-raspisaniya-uchebnyh-zanyatiy
4. СОСТАВЛЕНИЕ РАСПИСАНИЯ УЧЕБНЫХ ЗАНЯТИЙ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА – Режим доступу: http://www.vestnik.vsu.ru/pdf/analiz/2013/02/2013-02-17 . – Дата доступу: 26.05.2017.
5. Редактор створення діаграм. – Режим доступу: https://drive.draw.io/ . – Дата доступу: 15.05.2017.
6. 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.
7. Бойко О.М. Еволюційна технологія розв’язування задачі складання розкладів навчальних занять/ Бойко О.М. // Штучний інтелект. – 2006. – №.3. – С. 341-348.
8. Bremermann HJ, Roghson J., Salaff S. Global properties of evolution processes. Natural automata and useful simulations. / Bremermann HJ, Roghson J., Salaff S. – London: Macmillan, 1966. – pp 3-42.

ДОДАТОК 1. ЛІСТИНГ ПРОГРАМИ

Файл Classroom.java

```
public class Classroom {
    private final int classId;
    private final int groupId;
    private final int moduleId;
    private int professorId;
    private int timeslotId;
    private int roomId;

    public Classroom(int classId, int groupId, int moduleId){
        this.classId = classId;
        this.moduleId = moduleId;
        this.groupId = groupId;
    }

    public void addProfessor(int professorId){
        this.professorId = professorId;
    }

    public void addTimeslot(int timeslotId){
        this.timeslotId = timeslotId;
    }

    public void setRoomId(int roomId){
        this.roomId = roomId;
    }

    public int getClassId(){
        return this.classId;
    }

    public int getGroupId(){
        return this.groupId;
    }

    public int getModuleId(){
        return this.moduleId;
    }

    public int getProfessorId(){
        return this.professorId;
    }

    public int getTimeslotId(){
        return this.timeslotId;
    }

    public int getRoomId(){
        return this.roomId;
    }
}
```

Файл GeneticAlgorithm.java

```
public class GeneticAlgorithm {

   private int populationSize;
   private double mutationRate;
   private double crossoverRate;
   private int elitismCount;
   protected int tournamentSize;

   public GeneticAlgorithm(int populationSize, double mutationRate, double
crossoverRate, int elitismCount,
         int tournamentSize) {

      this.populationSize = populationSize;
      this.mutationRate = mutationRate;
      this.crossoverRate = crossoverRate;
      this.elitismCount = elitismCount;
      this.tournamentSize = tournamentSize;
   }

   public Population initPopulation(Timetable timetable) {
      // Initialize population
      Population population = new Population(this.populationSize, timetable);
      return population;
   }

   public boolean isTerminationConditionMet(int generationsCount, int
maxGenerations) {
      return (generationsCount > maxGenerations);
   }

   public boolean isTerminationConditionMet(Population population) {
      return population.getFittest(0).getFitness() == 1.0;
   }

   public double calcFitness(Individual individual, Timetable timetable) {

      // Create new timetable object to use -- cloned from an existing timetable
      Timetable threadTimetable = new Timetable(timetable);
      threadTimetable.createClasses(individual);

      // Calculate fitness
      int clashes = threadTimetable.calcClashes();
      double fitness = 1 / (double) (clashes + 1);

      individual.setFitness(fitness);

      return fitness;
   }

   public void evalPopulation(Population population, Timetable timetable) {
      double populationFitness = 0;

      // Loop over population evaluating individuals and summing population
      // fitness
      for (Individual individual : population.getIndividuals()) {
         populationFitness += this.calcFitness(individual, timetable);
      }

      population.setPopulationFitness(populationFitness);
   }

   public Individual selectParent(Population population) {
```

```
      // Create tournament
      Population tournament = new Population(this.tournamentSize);

      // Add random individuals to the tournament
      population.shuffle();
      for (int i = 0; i < this.tournamentSize; i++) {
         Individual tournamentIndividual = population.getIndividual(i);
         tournament.setIndividual(i, tournamentIndividual);
      }

      // Return the best
      return tournament.getFittest(0);
   }

   public Population mutatePopulation(Population population, Timetable
timetable) {
      // Initialize new population
      Population newPopulation = new Population(this.populationSize);

      // Loop over current population by fitness
      for (int populationIndex = 0; populationIndex < population.size();
populationIndex++) {
         Individual individual = population.getFittest(populationIndex);

         // Create random individual to swap genes with
         Individual randomIndividual = new Individual(timetable);

         // Loop over individual's genes
         for (int geneIndex = 0; geneIndex < individual.getChromosomeLength();
geneIndex++) {
            // Skip mutation if this is an elite individual
            if (populationIndex > this.elitismCount) {
               // Does this gene need mutation?
               if (this.mutationRate > Math.random()) {
                  // Swap for new gene
                  individual.setGene(geneIndex,
randomIndividual.getGene(geneIndex));
               }
            }
         }

         // Add individual to population
         newPopulation.setIndividual(populationIndex, individual);
      }

      // Return mutated population
      return newPopulation;
   }

   public Population crossoverPopulation(Population population) {
      // Create new population
      Population newPopulation = new Population(population.size());

      // Loop over current population by fitness
      for (int populationIndex = 0; populationIndex < population.size();
populationIndex++) {
         Individual parent1 = population.getFittest(populationIndex);

         // Apply crossover to this individual?
         if (this.crossoverRate > Math.random() && populationIndex >=
this.elitismCount) {
            // Initialize offspring
```

```
            Individual offspring = new
Individual(parent1.getChromosomeLength());

            // Find second parent
            Individual parent2 = selectParent(population);

            // Loop over genome
            for (int geneIndex = 0; geneIndex < parent1.getChromosomeLength();
geneIndex++) {
               // Use half of parent1's genes and half of parent2's genes
               if (0.5 > Math.random()) {
                  offspring.setGene(geneIndex, parent1.getGene(geneIndex));
               } else {
                  offspring.setGene(geneIndex, parent2.getGene(geneIndex));
               }
            }

            // Add offspring to new population
            newPopulation.setIndividual(populationIndex, offspring);
         } else {
            // Add individual to new population without applying crossover
            newPopulation.setIndividual(populationIndex, parent1);
         }
      }

      return newPopulation;
   }}
```

Файл Group.java

```
public class Group {
    private final int groupId;
    private final String groupName;
    private final int groupSize;
    private final int moduleIds[];

    public Group(int groupId, String groupName, int groupSize, int moduleIds[]){
        this.groupId = groupId;
        this.groupName = groupName;
        this.groupSize = groupSize;
        this.moduleIds = moduleIds;
    }

    public int getGroupId(){
        return this.groupId;
    }

    public String getGroupName(){
        return this.groupName;
    }

    public int getGroupSize(){
        return this.groupSize;
    }

    public int[] getModuleIds(){
        return this.moduleIds;
    }
}
```

Файл Individual.java

```
public class Individual {
   private int[] chromosome;
   private double fitness = -1;

   public Individual(Timetable timetable) {
      int numClasses = timetable.getNumClasses();

      // 1 gene for room, 1 for time, 1 for professor
      int chromosomeLength = numClasses * 3;
      // Create random individual
      int newChromosome[] = new int[chromosomeLength];
      int chromosomeIndex = 0;
      // Loop through groups
      for (Group group : timetable.getGroupsAsArray()) {
         // Loop through modules
         for (int moduleId : group.getModuleIds()) {
            // Add random time
            int timeslotId = timetable.getRandomTimeslot().getTimeslotId();
            newChromosome[chromosomeIndex] = timeslotId;
            chromosomeIndex++;

            // Add random room
            int roomId = timetable.getRandomRoom().getRoomId();
            newChromosome[chromosomeIndex] = roomId;
            chromosomeIndex++;

            // Add random professor
            Module module = timetable.getModule(moduleId);
            newChromosome[chromosomeIndex] = module.getRandomProfessorId();
            chromosomeIndex++;
         }
      }

      this.chromosome = newChromosome;
   }

   public Individual(int chromosomeLength) {
      // Create random individual
      int[] individual;
      individual = new int[chromosomeLength];
care of that for us."
       */
      for (int gene = 0; gene < chromosomeLength; gene++) {
         individual[gene] = gene;
      }

      this.chromosome = individual;
   }

   public Individual(int[] chromosome) {
      // Create individual chromosome
      this.chromosome = chromosome;
   }

   public int[] getChromosome() {
      return this.chromosome;
```

```
    }

    public int getChromosomeLength() {
        return this.chromosome.length;
    }

    public void setGene(int offset, int gene) {
        this.chromosome[offset] = gene;
    }

    public int getGene(int offset) {
        return this.chromosome[offset];
    }

    public void setFitness(double fitness) {
        this.fitness = fitness;
    }

    public double getFitness() {
        return this.fitness;
    }

    public String toString() {
        String output = "";
        for (int gene = 0; gene < this.chromosome.length; gene++) {
            output += this.chromosome[gene] + ",";
        }
        return output;
    }

    public boolean containsGene(int gene) {
        for (int i = 0; i < this.chromosome.length; i++) {
            if (this.chromosome[i] == gene) {
                return true;
            }
        }
        return false;
    }
}
```

Файл Module.java

```
public class Module {
    private final int moduleId;
    private final String moduleCode;
    private final String module;
    private final int professorIds[];

    public Module(int moduleId, String moduleCode, String module, int
professorIds[]){
        this.moduleId = moduleId;
        this.moduleCode = moduleCode;
        this.module = module;
        this.professorIds = professorIds;
    }

    public int getModuleId(){
        return this.moduleId;
    }

    public String getModuleCode(){
        return this.moduleCode;
    }

    public String getModuleName(){
        return this.module;
    }

    public int getRandomProfessorId(){
        int professorId = professorIds[(int) (professorIds.length *
Math.random())];
        return professorId;
    }
}
```

Файл Population.java

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Random;

public class Population {
   private Individual population[];
   private double populationFitness = -1;

   public Population(int populationSize) {
      // Initial population
      this.population = new Individual[populationSize];
   }

   public Population(int populationSize, Timetable timetable) {
      // Initial population
      this.population = new Individual[populationSize];

      // Loop over population size
      for (int individualCount = 0; individualCount < populationSize;
individualCount++) {
         // Create individual
```

```
            Individual individual = new Individual(timetable);
            // Add individual to population
            this.population[individualCount] = individual;
        }
    }


    public Population(int populationSize, int chromosomeLength) {
        // Initial population
        this.population = new Individual[populationSize];

        // Loop over population size
        for (int individualCount = 0; individualCount < populationSize;
individualCount++) {
            // Create individual
            Individual individual = new Individual(chromosomeLength);
            // Add individual to population
            this.population[individualCount] = individual;
        }
    }

    public Individual[] getIndividuals() {
        return this.population;
    }

    public Individual getFittest(int offset) {
        // Order population by fitness
        Arrays.sort(this.population, new Comparator<Individual>() {
            @Override
            public int compare(Individual o1, Individual o2) {
                if (o1.getFitness() > o2.getFitness()) {
                    return -1;
                } else if (o1.getFitness() < o2.getFitness()) {
                    return 1;
                }
                return 0;
            }
        });

        // Return the fittest individual
        return this.population[offset];
    }

    public void setPopulationFitness(double fitness) {
        this.populationFitness = fitness;
    }

    public double getPopulationFitness() {
        return this.populationFitness;
    }

    public int size() {
        return this.population.length;
    }

    public Individual setIndividual(int offset, Individual individual) {
        return population[offset] = individual;
    }

    public Individual getIndividual(int offset) {
        return population[offset];
    }
```

```
    public void shuffle() {
        Random rnd = new Random();
        for (int i = population.length - 1; i > 0; i--) {
            int index = rnd.nextInt(i + 1);
            Individual a = population[index];
            population[index] = population[i];
            population[i] = a;
        }
    }

}
```

Файл Professor.java

```
public class Professor {
    private final int professorId;
    private final String professorName;

    public Professor(int professorId, String professorName){
        this.professorId = professorId;
        this.professorName = professorName;
    }

    public int getProfessorId(){
        return this.professorId;
    }

    public String getProfessorName(){
        return this.professorName;
    }
}
```

Файл Room.java

```
public class Room {
    private final int roomId;
    private final String roomNumber;
    private final int capacity;
    public Room(int roomId, String roomNumber, int capacity) {
        this.roomId = roomId;
        this.roomNumber = roomNumber;
        this.capacity = capacity;
    }

    public int getRoomId() {
        return this.roomId;
    }

    public String getRoomNumber() {
        return this.roomNumber;
    }

    public int getRoomCapacity() {
        return this.capacity;
```

```
    }
}
```

Файл Timeslot.java

```
public class Timeslot {
    private final int timeslotId;
    private final String timeslot;

    public Timeslot(int timeslotId, String timeslot){
        this.timeslotId = timeslotId;
        this.timeslot = timeslot;
    }

    public int getTimeslotId(){
        return this.timeslotId;
    }

    public String getTimeslot(){
        return this.timeslot;
    }
}
```

Файл Timetable.java

```
import java.lang.*;
import java.lang.Class;
import java.util.HashMap;

public class Timetable {
   private final HashMap<Integer, Room> rooms;
   private final HashMap<Integer, Professor> professors;
   private final HashMap<Integer, Module> modules;
   private final HashMap<Integer, Group> groups;
   private final HashMap<Integer, Timeslot> timeslots;
   private Classroom classes[];

   private int numClasses = 0;
   public Timetable() {
      this.rooms = new HashMap<Integer, Room>();
      this.professors = new HashMap<Integer, Professor>();
      this.modules = new HashMap<Integer, Module>();
      this.groups = new HashMap<Integer, Group>();
      this.timeslots = new HashMap<Integer, Timeslot>();
   }

   public Timetable(Timetable cloneable) {
      this.rooms = cloneable.getRooms();
      this.professors = cloneable.getProfessors();
      this.modules = cloneable.getModules();
      this.groups = cloneable.getGroups();
      this.timeslots = cloneable.getTimeslots();
   }

   private HashMap<Integer, Group> getGroups() {
      return this.groups;
```

```
    }

    private HashMap<Integer, Timeslot> getTimeslots() {
        return this.timeslots;
    }

    private HashMap<Integer, Module> getModules() {
        return this.modules;
    }

    private HashMap<Integer, Professor> getProfessors() {
        return this.professors;
    }

    public void addRoom(int roomId, String roomName, int capacity) {
        this.rooms.put(roomId, new Room(roomId, roomName, capacity));
    }

    public void addProfessor(int professorId, String professorName) {
        this.professors.put(professorId, new Professor(professorId,
professorName));
    }

    public void addModule(int moduleId, String moduleCode, String module, int
professorIds[]) {
        this.modules.put(moduleId, new Module(moduleId, moduleCode, module,
professorIds));
    }

    public void addGroup(int groupId, String groupName, int groupSize, int
moduleIds[]) {
        this.groups.put(groupId, new Group(groupId, groupName, groupSize,
moduleIds));
        this.numClasses = 0;
    }

    public void addTimeslot(int timeslotId, String timeslot) {
        this.timeslots.put(timeslotId, new Timeslot(timeslotId, timeslot));
    }

    public void createClasses(Individual individual) {
        // Init classes
        Classroom classes[] = new Classroom[this.getNumClasses()];

        // Get individual's chromosome
        int chromosome[] = individual.getChromosome();
        int chromosomePos = 0;
        int classIndex = 0;

        for (Group group : this.getGroupsAsArray()) {
            int moduleIds[] = group.getModuleIds();
            for (int moduleId : moduleIds) {
                classes[classIndex] = new Classroom(classIndex, group.getGroupId(),
moduleId);

                // Add timeslot
                classes[classIndex].addTimeslot(chromosome[chromosomePos]);
                chromosomePos++;

                // Add room
                classes[classIndex].setRoomId(chromosome[chromosomePos]);
                chromosomePos++;
```

```
                // Add professor
                classes[classIndex].addProfessor(chromosome[chromosomePos]);
                chromosomePos++;

                classIndex++;
            }
        }

        this.classes = classes;
    }

    public Room getRoom(int roomId) {
        if (!this.rooms.containsKey(roomId)) {
            System.out.println("Rooms doesn't contain key " + roomId);
        }
        return (Room) this.rooms.get(roomId);
    }

    public HashMap<Integer, Room> getRooms() {
        return this.rooms;
    }

    public Room getRandomRoom() {
        Object[] roomsArray = this.rooms.values().toArray();
        Room room = (Room) roomsArray[(int) (roomsArray.length * Math.random())];
        return room;
    }

    public Professor getProfessor(int professorId) {
        return (Professor) this.professors.get(professorId);
    }

    public Module getModule(int moduleId) {
        return (Module) this.modules.get(moduleId);
    }

    public int[] getGroupModules(int groupId) {
        Group group = (Group) this.groups.get(groupId);
        return group.getModuleIds();
    }

    public Group getGroup(int groupId) {
        return (Group) this.groups.get(groupId);
    }

    public Group[] getGroupsAsArray() {
        return (Group[]) this.groups.values().toArray(new
Group[this.groups.size()]);
    }

    public Timeslot getTimeslot(int timeslotId) {
        return (Timeslot) this.timeslots.get(timeslotId);
    }

    public Timeslot getRandomTimeslot() {
        Object[] timeslotArray = this.timeslots.values().toArray();
        Timeslot timeslot = (Timeslot) timeslotArray[(int) (timeslotArray.length *
Math.random())];
        return timeslot;
    }
```

```
   public Classroom[] getClasses() {
      return this.classes;
   }

   public int getNumClasses() {
      if (this.numClasses > 0) {
         return this.numClasses;
      }

      int numClasses = 0;
      Group groups[] = (Group[]) this.groups.values().toArray(new
Group[this.groups.size()]);
      for (Group group : groups) {
         numClasses += group.getModuleIds().length;
      }
      this.numClasses = numClasses;

      return this.numClasses;
   }

   public int calcClashes() {
      int clashes = 0;

      for (Classroom classA : this.classes) {
         // Check room capacity
         int roomCapacity = this.getRoom(classA.getRoomId()).getRoomCapacity();
         int groupSize = this.getGroup(classA.getGroupId()).getGroupSize();

         if (roomCapacity < groupSize) {
            clashes++;
         }

         // Check if room is taken
         for (Classroom classB : this.classes) {
            if (classA.getRoomId() == classB.getRoomId() &&
classA.getTimeslotId() == classB.getTimeslotId()
                  && classA.getClassId() != classB.getClassId()) {
               clashes++;
               break;
            }
         }

         // Check if professor is available
         for (Classroom classB : this.classes) {
            if (classA.getProfessorId() == classB.getProfessorId() &&
classA.getTimeslotId() == classB.getTimeslotId()
                  && classA.getClassId() != classB.getClassId()) {
               clashes++;
               break;
            }
         }
      }

      return clashes;
   }
}
```

Файл TimetableGA.java

```
public class TimetableGA {

    public static void main(String[] args) {
       // Get a Timetable object with all the available information.
        Timetable timetable = initializeTimetable();

        // Initialize GA
        GeneticAlgorithm ga = new GeneticAlgorithm(100, 0.01, 0.9, 2,
            5);

        // Initialize population
        Population population = ga.initPopulation(timetable);

        // Evaluate population
        ga.evalPopulation(population, timetable);

        // Keep track of current generation
        int generation = 1;

        // Start evolution loop
        while (ga.isTerminationConditionMet(generation, 1000) == false
            && ga.isTerminationConditionMet(population) == false) {
            // Print fitness
            System.out.println("G" + generation + " Best fitness: " +
population.getFittest(0).getFitness());

            // Apply crossover
            population = ga.crossoverPopulation(population);

            // Apply mutation
            population = ga.mutatePopulation(population, timetable);

            // Evaluate population
            ga.evalPopulation(population, timetable);

            // Increment the current generation
            generation++;
        }

        // Print fitness
        timetable.createClasses(population.getFittest(0));
        System.out.println();
        System.out.println("Solution found in " + generation + " generations");
        System.out.println("Final solution fitness: " +
population.getFittest(0).getFitness());
        System.out.println("Clashes: " + timetable.calcClashes());

        // Print classrooms
        System.out.println();
        Classroom[] classrooms = timetable.getClasses();
        int classIndex = 1;
        for (Classroom bestClassroom : classrooms) {
        //    System.out.println("Classroom " + classIndex + ":");

System.out.println(timetable.getModule(bestClassroom.getModuleId()).getModuleNam
e());

System.out.println(timetable.getGroup(bestClassroom.getGroupId()).getGroupName()
);

System.out.println(timetable.getRoom(bestClassroom.getRoomId()).getRoomNumber())
;
```

```
System.out.println(timetable.getProfessor(bestClassroom.getProfessorId()).getPro
fessorName());

System.out.println(timetable.getTimeslot(bestClassroom.getTimeslotId()).getTimes
lot());
            System.out.println("-----");
            classIndex++;
        }
    }
   private static Timetable initializeTimetable() {
      // Create timetable
      Timetable timetable = new Timetable();

      // Set up rooms
      timetable.addRoom(1, "101", 30);
      timetable.addRoom(2, "102", 30);
      timetable.addRoom(4, "103", 30);
      timetable.addRoom(5, "104", 30);
      timetable.addRoom(6, "105", 30);
      timetable.addRoom(7, "203", 30);
      timetable.addRoom(8, "203a", 30);
      timetable.addRoom(9, "206", 70);
      timetable.addRoom(10, "301", 30);
      timetable.addRoom(11, "302", 30);
      timetable.addRoom(12, "304", 70);
      timetable.addRoom(13, "305", 70);
      timetable.addRoom(14, "306", 70);
      timetable.addRoom(15, "307", 70);
      timetable.addRoom(16, "308", 30);
      timetable.addRoom(17, "309", 30);
      timetable.addRoom(18, "310", 60);
      timetable.addRoom(19, "312", 70);
      timetable.addRoom(20, "208", 150);
      timetable.addRoom(21, "303", 60);

      // Set up timeslots
      timetable.addTimeslot(1, "Mon 8:30 - 10:05");
        timetable.addTimeslot(2, "Mon 10:25 - 12:00");
        timetable.addTimeslot(3, "Mon 12:20 - 13:55");
        timetable.addTimeslot(4, "Mon 14:15 - 15:50");
        timetable.addTimeslot(5, "Mon 16:10 - 17:45");
        timetable.addTimeslot(6, "Tue 8:30 - 10:05");
        timetable.addTimeslot(7, "Tue 10:25 - 12:00");
        timetable.addTimeslot(8, "Tue 12:20 - 13:55");
        timetable.addTimeslot(9, "Tue 14:15 - 15:50");
        timetable.addTimeslot(10, "Tue 16:10 - 17:45");
        timetable.addTimeslot(11, "Wed 8:30 - 10:05");
        timetable.addTimeslot(12, "Wed 10:25 - 12:00");
        timetable.addTimeslot(13, "Wed 12:20 - 13:55");
        timetable.addTimeslot(14, "Wed 14:15 - 15:50");
        timetable.addTimeslot(15, "Wed 16:10 - 17:45");
        timetable.addTimeslot(16, "Thu 8:30 - 10:05");
        timetable.addTimeslot(17, "Thu 10:25 - 12:00");
        timetable.addTimeslot(18, "Thu 12:20 - 13:55");
        timetable.addTimeslot(19, "Thu 14:15 - 15:50");
        timetable.addTimeslot(20, "Thu 16:10 - 17:45");
        timetable.addTimeslot(21, "Fri 8:30 - 10:05");
        timetable.addTimeslot(22, "Fri 10:25 - 12:00");
        timetable.addTimeslot(23, "Fri 12:20 - 13:55");
        timetable.addTimeslot(24, "Fri 14:15 - 15:50");
        timetable.addTimeslot(25, "Fri 16:10 - 17:45");
```

```
        // Set up professors
        timetable.addProfessor(1, "Zuev");
        timetable.addProfessor(2, "Romanov");
        timetable.addProfessor(3, "Besnosyk");
        timetable.addProfessor(4, "Kalita");
        timetable.addProfessor(5, "----");
        timetable.addProfessor(6, "Petrenko");
        timetable.addProfessor(7, "Sergeev-Gorchynsky");
        timetable.addProfessor(8, "Kapshuk");
        timetable.addProfessor(9, "Golubova");
        timetable.addProfessor(10, "Dukhanina");
        timetable.addProfessor(11, "Stus");
        timetable.addProfessor(12, "Kirusha");
        timetable.addProfessor(13, "Giorgizova");
        timetable.addProfessor(14, "Minarchenko");
        timetable.addProfessor(15, "Bokhonov");
        timetable.addProfessor(16, "Stikanov");
        timetable.addProfessor(17, "Drabko");
        timetable.addProfessor(18, "Lyashenko");
        timetable.addProfessor(19, "Gaidenko");
        timetable.addProfessor(20, "Bulakh");
        timetable.addProfessor(21, "Artuhov");
        timetable.addProfessor(22, "Britov");


        // Set up modules and define the professors that teach them
        timetable.addModule(1, "pl1", "Philosophy, lec", new int[] { 1 });
        timetable.addModule(2, "en1", "English, prac", new int[] { 10, 19 });
        timetable.addModule(3, "ma1", "Mathematical analysis, lec", new int[] {
14, 15 });
        timetable.addModule(4, "ph1", "Physics, lec", new int[] { 4 });
        timetable.addModule(5, "hi1", "History, lec", new int[] { 17 });
        timetable.addModule(6, "nm1", "Numerical Methods, lec", new int[] { 6 });
        timetable.addModule(7, "ma2", "Mathematical analysis, prac", new int[] {
14, 15 });
        timetable.addModule(8, "ph2", "Physics, prac", new int[] { 4 });
        timetable.addModule(9, "nm2", "Numerical Methods, prac", new int[] { 20, 9
});
        timetable.addModule(10, "pt1", "Physical training", new int[] { 5 });
        timetable.addModule(11, "ph3", "Physics, lab", new int[] { 18 });
        timetable.addModule(12, "nm3", "Numerical Methods, lab", new int[] { 20, 9
});
        timetable.addModule(13, "la1", "Linear Algebra, lec", new int[] { 14 });
        timetable.addModule(14, "la2", "Linear Algebra, prac", new int[] { 14 });
        timetable.addModule(15, "it1", "Information coding theory, lab", new int[]
{ 12, 7 });
        timetable.addModule(16, "it2", "Information coding theory, lec", new int[]
{ 8 });
        timetable.addModule(17, "tp1", "Theory of probability, lec", new int[] {
11 });
        timetable.addModule(18, "tp2", "Theory of probability, prac", new int[] {
11 });
        timetable.addModule(19, "ca1", "Computer architecture, lab", new int[] {
12, 22 });
        timetable.addModule(20, "ca2", "Computer architecture, lec", new int[] {
21 });
        timetable.addModule(21, "cc1", "Computer circuitry, lec", new int[] { 16
});
        timetable.addModule(22, "cc2", "Computer circuitry, prac", new int[] { 13
});
        timetable.addModule(23, "cc3", "Computer circuitry, lab", new int[] { 12,
```

```
16 });
      timetable.addModule(24, "aa1", "Algorithm analysis, lec", new int[] { 2
});
      timetable.addModule(25, "aa2", "Algorithm analysis, lab", new int[] { 2, 3
});
      timetable.addModule(26, "ap1", "Algorithms and programming, lec", new
int[] { 2 });
      timetable.addModule(27, "ap2", "Algorithms and programming, lab", new
int[] { 2, 3 });
      timetable.addModule(28, "dm1", "Discrete mathematics, lec", new int[] { 11
});
      timetable.addModule(29, "dm2", "Discrete mathematics, prac", new int[] {
11 });

      // Set up student groups and the modules they take.
      timetable.addGroup(1, "DA-51", 30, new int[] { 1, 2, 6, 9, 10, 12, 15, 16,
17, 18,
            19, 20, 21, 22, 23 });
      timetable.addGroup(2, "DA-52", 28, new int[] { 1, 2, 6, 9, 10, 12, 15, 16,
17, 18,
            19, 20, 21, 22, 23 });
      timetable.addGroup(3, "DA-61", 29, new int[] { 2, 3, 4, 5, 7, 8, 10, 11,
13, 14,
            24, 25, 26, 27, 28, 29 });
      timetable.addGroup(4, "DA-62", 30, new int[] { 2, 3, 4, 5, 7, 8, 10, 11,
13, 14,
            24, 25, 26, 27, 28, 29 });
      return timetable;
   }
}
```