Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження адаптивних методів складання розкладу навчального процесу в університеті
(тема)

Виконала: студентка 2 курсу, групи ІПЗмзд-18-1
Гайтан О.М.
(прізвище, ініціали)

спеціальності 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

освітньо-наукової програми
(тип програми)
Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник доцент Назаров О.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентці Гайтан Олені Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження адаптивних методів складання розкладу навчального процесу в університеті

затверджена наказом університету від "___" _____20 ___ р. № _____

2. Термін подання студентом роботи до екзаменаційної комісії
22 травня 2020 р.

3. Вихідні дані до роботи *структура та організація освітнього процесу, навчальні плани спеціальностей, навчальне навантаження кафедр, календарі доступності та пріоритети учасників навчального процесу, методи складання розкладу навчального процесу в університеті.Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, формалізація предметної галузі, огляд адаптивних складання розкладу навчального процесу вищого навчального закладу, розробка гібридного методу складання розкладу, розробка програмної системи для автоматизованого складаннярозкладу навчального процесу в університеті, дослідження гібридного методу*

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доцент Назаров О.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	15 березня 2020 р.	виконано
2.	Огляд існуючих методів	01 квітня 2020 р.	виконано
3.	Методи складання розкладу навчального процесу в університеті	15 квітня 2020 р.	виконано
4.	Підготовка пояснювальної записки	01 травня 2020 р.	виконано
5.	Спецчастина	10 травня 2020 р.	виконано
6.	Нормоконтроль, рецензування	15 травня 2020 р.	виконано
7.	Підготовка презентації та доповіді	15 травня 2020 р.	виконано
8.	Попередній захист	20 травня 2020 р.	виконано
9.	Занесення диплома в електронний архів	22 травня 2020 р.	виконано
10.	Допуск до захисту у зав. кафедри	22 травня 2020 р.	виконано

Дата видачі завдання _____ 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доцент Назаров О.С.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота магістра містить: 151 с., 43 рис., 7 табл., 33 джерела.

АДАПТИВНИЙ АЛГОРИТМ, АЛГОРИТМ ОПТИМІЗАЦІЇ МУРАШИНОЇ КОЛОНІЇ, БАГАТОКРИТЕРІЛЬНА КОМБІНАТОРНА ОПТИМІЗАЦІЯ, ГЕНЕТИЧНІ АЛГОРИТМИ, МЕТАЕВРИСТИЧНІ АЛГОРИТМИ, МЕТОД ДЕФОРМОВАНОГО БАГАТОГРАННИКА, ОБМЕЖЕННЯ, ПЛАТФОРМА .NET, РОЗКЛАД, ТАБУ-ПОШУК, C#, EXCEL, MICROSOFT SQL SERVER, VISUAL STUDIO, XML.

Об’єкт дослідження – розклад навчального процесу в університеті.

Метою роботи є дослідження та аналіз методів складання розкладу, розробка гібридного алгоритму, проектування та розробка програмної системи для автоматизованого складання розкладу навчального процесу в університеті.

Методи розробки базуються на принципах об’єктно-орієнтованого проектування та об’єктно-орієнтованого програмування, інструментах розробки застосувань мовою C# за допомогою платформи .NET та VisualStudio, методів створення WYSIWYG-редакторів з використанням Drag-and-drop технологій.

У результаті роботи розроблено гібридний адаптивний метод автоматичного складання розкладу та створено програмну реалізацію системи складання розкладу навчального процесу в університеті, яка на основі постійних даних, що зберігаються в sql-базі даних, а також оперативної семестрової інформації, імпортованої з xml-довідників, забезпечує можливість ручного, автоматичного та автоматизованого складання розкладу навчального процесу у вищому навчальному закладі з використанням розробленого методу. Передбачена можливість експорту розробленого розкладу в txt, xlsx та xml-формати.

ABSTRACT

ADAPTIVE ALGORITHMS, ANT COLONY OPTIMIZATION, MULTIOBJECTIVE COMBINATORIAL OPTIMIZATION, GENETIC ALGORITHMS, METAHEURISTIC ALGORITHMS, NELDER–MEAD METHOD, RESTRICTIONS, .NET PLATFORM, SCHEDULE, TIMETABLING, TABU SEARCH, C#, EXCEL, MICROSOFT SQL SERVER, VISUAL STUDIO, XML.

The object of the study is the university course timetabling.

The purpose of the work is research and analysis of the university course timetabling methods, the hybrid algorithm development, development and implementation the software for automated university course timetabling.

Development methods are based on the principles of object-oriented design and object-oriented programming, C# application development tools using .NET and Visual Studio, creation of WYSIWYG editors using Drag-and-drop technologies.

As a result of the work, the hybrid adaptive algorithm for automated university course timetabling has been developed as well as a program implementation of the system. The developed software for university course timetabling based on constant data stored in a sql database and operational semester information imported from xml files provides the possibility of manual, automatic and automated course timetabling using the described method and drap&drop technology. It is possible to edit manually the created schedule and export it to txt, xml and xlsx files.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АСО (англ. ant colony optimization) – алгоритм мурашиної колонії

АСУ ЗВО – автоматизована система управління закладу вищої освіти

БД – база даних

ГА – генетичний алгоритм

ЗВО – заклад вищої освіти

НП – навчальний план

ОПП – освітньо-професійна програма

СУБД – система управління бази даних

ВСТУП

Актуальність теми. Складання розкладу навчального процесу у закладах вищої освіти (ЗВО) є важливою складовою системи забезпечення навчального процесу, оскільки від якості розкладу залежить комфорт учасників навчального процесу та його якість і ефективність.

Завдання складання розкладу в університеті характеризується [1]: значним обсягом різноманітної оперативної інформації, отриманої з різних структурних підрозділів ЗВО; складністю ідентифікації та формалізації параметрів та обмежень розкладу, ступінь обліку яких залежать від досвіду і кваліфікації диспетчера; конфліктом інтересів основних учасників навчального процесу: студентів і викладачів, кафедр – власників аудиторій та обладнання, що накладає обмеження на свободу їх використання в розкладі; трудомісткістю адаптації універсальних алгоритмів складання розкладу до потреб конкретного навчального закладу.

Незважаючи на те, що ручна побудова розкладів займає багато часу, вона все ще широко поширена через значну вартість або відсутність відповідних комп'ютерних програм. Тому дослідження відповідних методів та розробка ефективних сучасних програм для автоматизованого складання розкладу є актуальною задачею.

Про актуальність розробки ефективних методів автоматизованого складання розкладу університету свідчить проведення серії міжнародних конференцій «Practice and Theory of Automated Timetabling (PATAT)» (Практика та теорія складання автоматизованого розкладу) [2]. PATAT проводиться щороку та виступає форумом для міжнародної спільноти дослідників, практиків та розробників з усіх аспектів формування комп'ютерного розкладу. У 2020 р. конференція PATAT-2020 відбудеться 25 – 28 серпня у м. Брюгге, Бельгія.

PATAT підтримує ряд змагань та викликів, зокрема International Timetabling Competition ITC 2019, під час якого розробникам найбільш ефективних методів

складання комп’ютерного розкладу присуджується грошова премія.

Дана робота присвячена дослідженню адаптивних методів складання розкладу та розробці програмної системи складання розкладу навчального процесу в університеті.

Об’єкт дослідження – організація навчального процесу в університеті, розклад навчального процесу.

Предмет дослідження – моделі та методи комп’ютерного складання розкладу навчального процесу.

Мета і задачі дослідження – розроблення та дослідження гібридного підходу до розв’язання задачі автоматизованого складання розкладу в університеті з метою підвищення якості та швидкості створення розкладу та програмна реалізація системи складання розкладу навчального процесу з використанням розробленого методу.

Методи дослідження. Було досліджено основні класичні та метаевристичні методи формування розкладу навчального процесу, які застосовуються в сучасних автоматизованих системах формування розкладу та АСУ ЗВО, для аналізу яких був використаний системний аналіз. Формалізація предметної області та розробка гібридного методу формування розкладу проведені за допомогою матричного аналізу, теорії графів, математичної логіки та теорії прийняття рішень. При розробці програмної системи використані технології об’єктно-орієнтованого проектування та програмування, теорія баз даних, Drag-and-drop технології.

Елементи наукової новизни. У роботі запропоновано гібридний підхід до розв’язання задачі автоматизованого складання розкладу вищого навчального закладу на основі методу мурашиних колоній, генетичного алгоритму та методу деформованого багатогранника. Метод мурашиної колонії є основою даного алгоритму, що формує початкову популяцію для генетичного алгоритму. Комбінація даного метода з генетичним алгоритмом та методом деформованого багатогранника спрямована на усунення таких недоліків даного метода як невизначеність часу збіжності алгоритму та сильна залежність ефективності роботи методу від початкових параметрів пошуку, які необхідно

підбиратиекспериментально. Метод деформованого багатогранника використовується для знаходження параметрів методу мурашиних колоній. Використання генетичного алгоритму дозволяє зменшити час роботи алгоритму та збільшити ймовірність попадання в глобальний оптимум.

Практичне значення одержаних результатів. Розроблений гібридний метод є основою програмної реалізації системи складання розкладу навчального процесу в університеті. Використання даної системи в університеті дозволить полегшити та підвищити ефективність роботи диспетчерів, які займаються складанням розкладу, та якість навчального процесу за рахунок максимального врахування побажань учасників навчального процесу при складанні розкладу.

На даний час було проведено тестування розробленої системи на реальних даних Національного університету «Полтавська політехніка імені Юрія Кондратюка», в Навчально-науковому інституті інформаційних технологій і механотроніки, на наступних даних: 1 факультет, 56 викладачів, 54 групи, 20 аудиторій. Результати тестування підтвердили ефективність запропонованого метода

Публікації. Результати атестаційної роботи викладені в 3 публікаціях, у тому числі – 2 статтях у журналах з переліку наукових фахових видань України, а також в матеріалах конференції:

– Hybrid approach to solving of the automated timetabling problem in higher educational institution («Системи управління, навігації та зв’язку», фахове видання з технічних наук (17.03.2020), категорія «Б», індексація в Index Copernicus);

– Автоматизація генерації розкладу навчального процесу університету (Вчені записки Таврійського національного університету імені В.І. Вернадського, серія «Технічні науки», фахове видання з технічних наук (17.03.2020), категорія «Б», індексація в IndexCopernicus);

– Підвищення ефективності методу мурашиних колоній в задачі комп’ютерної генерації розкладу навчального процесу ВНЗ (72-а наукова конференція професорів, викладачів, наукових працівників, аспірантів та студентів Національного університету «Полтавська політехніка імені Юрія

Кондратюка»).

Результати роботи доповідалися на 72-ї науковій конференції професорів, викладачів, наукових працівників, аспірантів та студентів Національного університету «Полтавська політехніка імені Юрія Кондратюка» (Полтава, 2020).

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ СКЛАДАННЯ КОМП'ЮТЕРНОГО РОЗКЛАДУ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі складання розкладу навчального процесу в університеті

Проблема складання розкладу навчального процесу в університеті полягає у розподілі набору занять згідно навчальних планів в межах заданої кількості аудиторій та періодів часу (пар). Основна відмінність розкладу університету від середньої школи полягає в тому, що навчальні курси в університеті можуть мати спільних студентів, тоді як шкільні класи – це множини учнів, що не перетинаються. Якщо два заняття мають спільних студентів або проводяться одним викладачем, вони конфліктують, і тому не можуть бути заплановані на той самий період. Крім того, необхідно передбачити можливість поділу академічних груп на підгрупи та об'єднання лекційних занять у потоки.

У проблемі складання розкладу університету важливу роль відіграє наявність аудиторій (їх розмір, тип і обладнання, належність до відповідної кафедри), тоді як у середній школі цим нехтують, оскільки в більшості випадків можна припустити, що кожен клас має свій кабінет і окремі вимоги висуваються лише до занять з іноземної мови, інформатики та фізкультури.

Проблема складання розкладу іспитів полягає в розподілі певної кількості іспитів протягом заданої кількості часу. Складання розкладу іспитів аналогічно складанню розкладу навчального процесу, ці два процеси відрізняються м'якими (необов'язковими) вимогами, що висуваються до розкладу.

Основними керівними документами щодо складання розкладів навчальних занять та контролю навчального процесу є:

– Закони України: від 05.09.2017р. № 2145-VIII «Про освіту»; від 01.07.2014 р. № 1556-VII «Про вищу освіту»;

– стандарти освітньої діяльності, стандарти вищої освіти, інші законодавчі акти України з питань освіти;

– положення про організацію навчального процесу вищого навчального закладу;

– графік-календар навчального процесу на навчальний рік;

– статут, правила внутрішнього розпорядку, накази та розпорядження керівництва університету щодо планування та організації навчальної й методичної роботи на навчальний рік;

– освітньо-професійні програми спеціальностей та освітніх програм;

– навчальні плани, робочі навчальні плани та програми навчальних дисциплін;

– робочі програми навчальних дисциплін.

Розклади навчальних занять складаються, як правило, на кожен семестр відповідним відділом в організаційній структурі університету та мають періодичність 2 тижні. Розклади екзаменів складаються на одну заліко-екзаменаційну сесію без повторень.

1.2 Аналіз останніх досліджень і публікацій

Проблемі автоматизованого формування розкладу навчального процесу протягом останніх десятиліть приділяється значна увага.

Публікації з даного питання можна поділити на декілька категорій:

– формалізація предметної області:

Так, в [3] розглянуто математичну постановку задачі складання розкладу занять у ЗВО в умовах розбіжності вимог і побажань викладачів і студентів, введено поняття віртуальних та узагальнених груп і підгруп, проаналізовано жорсткі та м’які обмеження в задачі складання розкладу.

Робота [4] присвячена стандартизації даних для складання розкладу у навчальних закладах. Автори описують формат представлення даних у частині опису навчального плану і розкладу занять на основі XML-технологій.

В роботі [5] розглядається питання формування цільової функції, за допомогою якої оцінюється якість складеного розкладу навчального процесу, з подальшою оптимізацією цільової функції на основі використання еволюційних технологій і матричного подання розкладу;

– аналіз існуючих та розробка нових алгоритмів, моделей та методів складання розкладу:

У роботі [6] наведено огляд методів складання розкладу ЗВО, проаналізовані їхні переваги та недоліки. Опис основних методів, що використовуються для розв’язання даної задачі, також представлено у [7].

Для автоматичного складання розкладу широко використовуються як класичні методи такі, як лінійне цілочисельне програмування, метод розфарбування графу, метод імітаційного моделювання, так і метаевристичні методи.

Так, застосування методу розфарбування графу для складання розкладу розглядається у роботах, починаючи з 1967 р. (Welsh) по сьогоднішній день. У роботі [8] розглянуті варіанти підходів до складання розкладу за допомогою даного методу, запропоновані різними авторами, та описане використання методу розфарбування графу для складання розкладу екзаменів.

Але оскільки дану задачу можна віднести до класу NP-важких задач багатокритеріальної комбінаторної оптимізації зі значною кількістю обмежень та складністю побудови математичної моделі, для яких не доведено існування ефективних методів та алгоритмів, які знаходять точне рішення за поліноміальний час, використання класичних методів обмежене. Тому на даний час для розв’язання задачі складання розкладу ЗВО дуже поширене застосування метаевристичних методів. Застосовують такі метаевристичні методи як метод імітації відпалу, генетичні алгоритми, метод мурашиних колоній тощо. Ці алгоритми пошуку можуть забезпечувати високоякісні рішення, але часто мають значну обчислювальну вартість.

У роботі [7] розроблено алгоритм побудови розкладу для дистанційного навчання на основі генетичного алгоритму, реалізований у якості підсистеми для

системи дистанційного навчання «Віртуальний Університет». Генетичний алгоритм для побудови розкладу також розглядається у роботах [9, 10] для побудови розкладу іспитів. При цьому особлива увага приділяється послідовності іспитів. Також генетичні алгоритми застосовують в автоматизованій системі розподілу навантаження викладачів і студентів в роботі [11].

У роботі [12] запропоновано новий варіант алгоритму імітації відпалу, який називається FastSA, для автоматичної побудови розкладу іспитів в ЗВО. В FastSA кожен іспит, обраний для планування, переміщається (і даний рух оцінюється) лише в тому випадку, якщо цей іспит мав якісь прийняті рухи на попередньому відрізку температури. Для побудови початкового рішення використовується евристика на основі ступеня насичення, поєднана зі статистикою на основі конфліктів, щоб уникнути зациклення.

У [13] розглядається побудова нейронної мережі для розв’язання даної задачі.

Робота [14] описує підхід до складання розкладу за допомогою максимінного методу мурашиного алгоритму (Best-WorstAntSystem) та максимінного методу мурашиних колоній (Best-WorstAntColonySystem), а також у комбінації з 2 типами локального пошуку. Доведено, що комбінація методу мурашиних колоній та локального пошуку значно підвищує ефективність алгоритму. Також результати експериментів показали перевагу методу максимінного мурашиних колоній над максимінним мурашиним алгоритмом.

Також в останній час з’являються роботи, присвячені гібридним методам складання розкладу. Так, у [15] автор запропонував метод складання розкладу, використовуючи комбінований підхід методу підживлення бактерій та генетичного алгоритму. Але обчислювальна вартість такого підходу порівняно висока;

– аналіз існуючих та розробка нових програмних засобів генерації комп’ютерного розкладу: технології, що використовуються, та їх програмна реалізація [16 – 19]:

Задача генерації розкладу відноситься до задач автоматизації діяльності

вищого навчального закладу, тому деякі навчальні заклади використовують засоби складання розкладу, які є складовою системи документообігу.

Основними системами для автоматичного складання розкладу, представленими на ринку програмного забезпечення України, є:

– програма Галактика [17];

– програма 1С: Автоматизоване складання розкладів. Університет [18];

– програма Ректор-ВНЗ [19].

Порівняльний аналіз функціональних можливостей наведених систем наведено в [11].

Розглянемо особливості деяких систем автоматизованого формування розкладу навчального процесу.

Комплексна система складання університетського розкладу UniTime [16] розроблена зусиллями групи викладачів, студентів і співробітників університетів Північної Америки та Європи.

UniTime підтримує складання розкладів курсів та іспитів, управління змінами в цих розкладах, спільне використання аудиторій з іншими заходами і розподіл студентів за окремими курсами (див. рис. 1.1).

Рисунок 1.1 – Структура системи UniTime

Це розподілена система, яка дозволяє декільком відповідальним особам координувати зусилля зі створення і зміни розкладу, що відповідає їхнім

організаційним потребам, мінімізуючи конфлікти в ході роботи.

Систему можна використовувати окремо для створення і ведення розкладу занять і / або іспитів або для взаємодії з існуючою АІС ЗВО. Дані для розкладу можуть вводитися безпосередньо в системі або імпортуватися із xml-файлів (див. рис. 1.2).

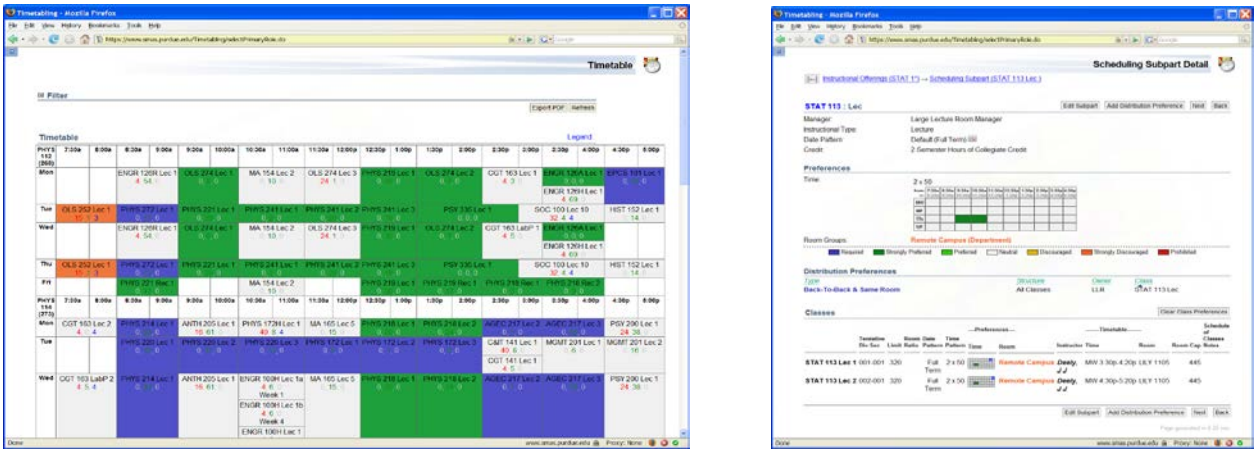


Рисунок 1.2 – Екранні форми системи UniTime

Алгоритм пошуку в UniTime ґрунтується на ітеративному алгоритмі прямого пошуку (iterativeforwardsearchalgorithm). Цей алгоритм схожий з локальними методами пошуку; однак, на відміну від класичних локальних методів пошуку, він працює з допустимими, але не обов’язково повними рішеннями. У цих рішеннях деякі заняття можуть залишатися не розміщеними. Однак усі жорсткі обмеження щодо призначених занять повинні виконуватися. Такі рішення простіші для візуалізації та більш значущі для користувачів, ніж повні, але нездійсненні рішення. Через ітеративний характер алгоритму алгоритм може запускати, зупиняти або продовжувати обробляти будь-яке допустиме рішення – повне або неповне.

Даний додаток поширюється безкоштовно за ліцензією з відкритим вихідним кодом для використання іншими коледжами та університетами, а також для сторонніх дослідників, які захочутьвнести вклад в дослідження в цій області. Недоліком цієї системи є орієнтація на англомовні країни.

Система Галактика: Розклад навчальних занять [17,20].

Система «Галактика: Розклад навчальних занять» призначена для автоматизації процесу складання розкладів навчальних занять в освітніх установах вищої та середньої професійної освіти. Дану систему можна використовувати як самостійний додаток і в комплексі з іншими системами, наприклад, з Галактикою ERP (рішення Галактика Управління Вузом). Дані для розкладу можуть вводитися безпосередньо в системі або імпортуватися із зовнішніх систем у xml та excel–файлах (див. рис. 1.3).

Рисунок 1.3 – Структура системи системи Галактика: Розклад навчальних занять

«Галактика» автоматизує процес складання розкладу, з огляду на множину обов'язкових, умовно обов'язкових і бажаних умов, зокрема, облаштування аудиторій під окремі дисципліни, приналежність до одного або різних навчальних корпусів, завантаженість викладачів, формування зведених груп студентів.

Важливим інструментом системи є можливість планування порядку вивчення академічних предметів. Ця опція включає графік вивчення дисциплін протягом тижня і технологічну карту з послідовністю їх проходження.

Система розкладу навчальних занять дозволяє: паралельно вести кілька розкладів по різних корпусах, днях тижня, групам; контролювати зв'язок вільних аудиторій, зокрема, спеціально обладнаних, з предметами, видами робіт, кафедрами, факультетами; налаштовувати пріоритети використання ресурсів;

враховувати віддаленість корпусів, їх пріоритетність; підтримувати різні групи студентів (потік, підгрупа, зведена група), контролювати їх переміщення при формуванні розкладу, щоб заняття не дублювалися і не перетиналися; використовувати чотири десятки показників ефективності розкладу.

Екранні форми системи Галактика представлені на рисунку 1.4.

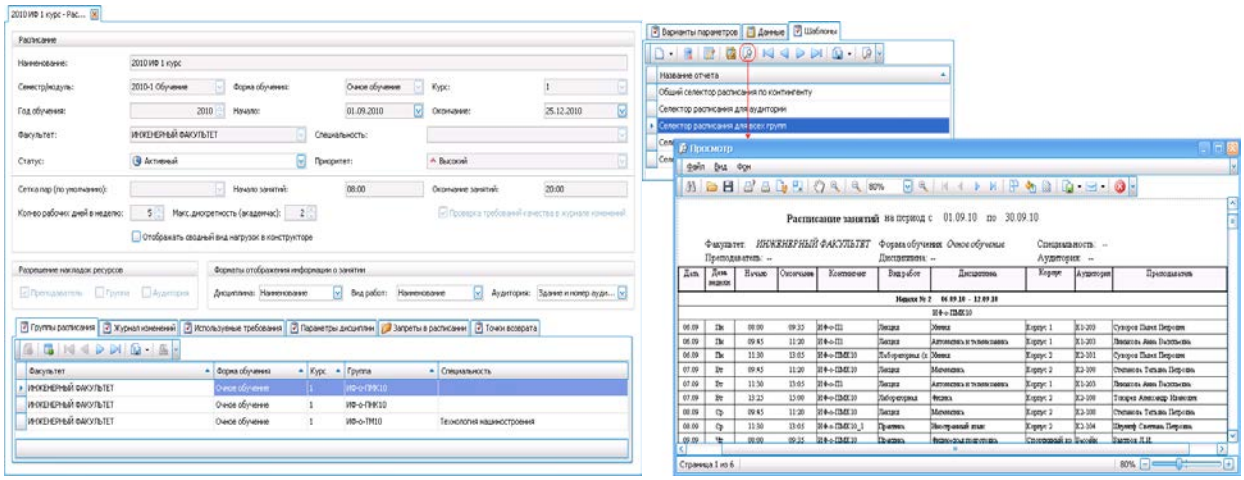


Рисунок 1.4 – Екранні форми системи Галактика: Розклад навчальних занять

На відміну від попереднього рішення головна мета системи Галактика – системність рішень, спроба уникнути клаптикової автоматизації, коли для кожної критичної області, в тому числі і для складання розкладу, застосовується окрема система.

Дана система орієнтована на російськомовну аудиторію і використовується переважно в російських ЗВО. Вартість ліцензії – індивідуальна для кожного ЗВО.

Програма «1С: Автоматизированное составление расписания. Университет» [18].

Продукт призначений для автоматизованого складання розкладів, управління ними під час навчального процесу і оперативного управління аудиторіями. З його допомогою складати розклад можна в автоматичному, ручному і змішаному режимах, а також в режимах вибірки по приміщеннях, по групах, по викладачах з урахуванням обмежень і умов. При цьому можна побудувати як допустимий розклад, так і оптимізований, в якому зменшено

кількість вікон або кількість використовуваних аудиторій.

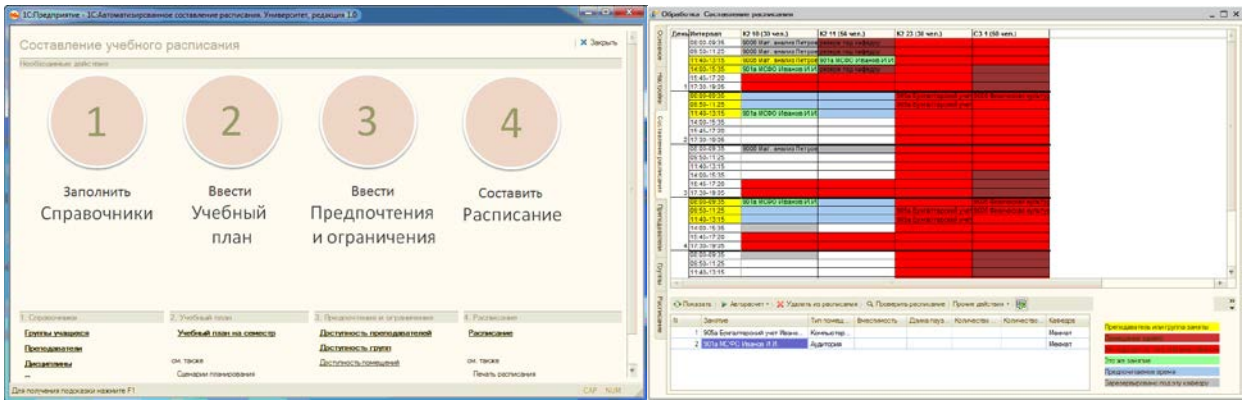


Рисунок 1.5 – Екранні форми системи

Система розкладу навчальних занять дозволяє: ручну модифікацію розкладу перетягуванням drap&drop; складання кількох розкладів і вибір кращого; консолідацію розкладів і оптимізацію за одним з критеріїв; враховувати побажання і можливості викладачів, груп, аудиторій; перевіряти на допустимість при складанні розкладу (за показниками: тип приміщення / тип заняття, ємність приміщення / кількість студентів в групі); вибирати довільну періодичність розкладу; вести облік паралельних занять, поділ на підгрупи і потоки лекцій; регулювати максимальну кількість занять в день для групи студентів або викладача при складанні розкладу; вести облік змін; оперативно резервувати приміщення.

Для автоматичної генерації розкладу навчального процесу дана програмна система використовує алгоритм розв’язання задачі, запропонований співробітниками лабораторії № 68 «Теорії розкладів і дискретної оптимізації» Інституту проблем управління ім. В.А Трапезникова РАН.

Система орієнтована на російськомовну аудиторію і використовується переважно в російських ЗВО. Крім того, необхідно відмітити, що компанія-постачальник ERP-систем "1С" підпадає під дію економічних санкцій згідно Указу Президента України №133/2017. Вартість ліцензії – 70000 грн.

Програма "Ректор-ВУЗ" [19].

Розклад занять можна складати в автоматичному, ручному або змішаному режимі. При складанні розкладу в автоматичному режимі програма враховує всі сформульовані вимоги до розкладу. При складанні розкладу в ручному режимі програма підказує можливі варіанти розстановки занять обраного викладача, можливі варіанти заповнення порожніх клітин в розкладі групи, стежить за кількістю місць в аудиторіях. Розклад занять груп і викладачів можна зберегти у форматах Microsoft Word, Excel або HTML.

Дана система використовується переважно в російських ЗВО, але має переклад на українську мову.

Серед розглянутих систем програма "Ректор-ВУЗ" має найменший функціонал, оскільки не забезпечує врахування паралельних занять, поділу на підгрупи при складанні розкладу, оперативного резервування приміщень, врахування побажань викладачів та студентських груп тощо. Вартість ліцензії – 12000 грн. на рік (оренда на 3 комп'ютера).

Порівняння функціональних можливостей наведених систем [11] представлено в таблиці 1.1.

Таблиця 1.1 – Функціональні можливості програм для автоматизованого складання розкладів

Функціональні можливості	Галактика – Розклад навчальних занять	1С: Автоматизированное составление расписания. Университет	Ректор-ВНЗ
Формати довідників	excel, xml	excel, xml, html	xml, html
Задання критеріїв оптимізації розкладу	+	+	–
Пріоритети учасників навчального процесу	+	+	–
Пріоритети використання ресурсів	–	–	–
Обмеження кількості занять в день	+	+	+
Поділ на підгрупи	+	+	–
Оперативне резервування аудиторій	+	+	–
Врахування змінності	+	+	+
Дотримання інтервалу між певними заняттями	+	–	–

Кінець таблиці 1.1

Функціональні можливості	Галактика – Розклад навчальних занять	1С: Автоматизированноесоставлениерасписания. Университет	Ректор-ВНЗ
Можливість складання довільних розкладів	+	+	+
Паралельне складання декількох розкладів	+	–	–
Розділ прав доступу	+	–	+

На основі аналізу особливостей, переваг та недоліків представлених систем, була поставлена задача розроблення гібридного алгоритму створення розкладу навчального процесу та реалізації системи складання розкладу навчального процесу в університеті з використанням розробленого методу.

1.3 Постановка задачі

1. Розробити гібридний алгоритм створення розкладу навчального процесу в університеті з метою підвищення якості та швидкості формування розкладу.

2. Спроектувати та програмно реалізувати систему генерації розкладу навчального процесу в університеті, яка повинна забезпечувати виконання таких функцій:

– імпорт даних із xml-файлу;

– редактор вхідних xml-файлів;

– зберігання, завантаження, оновлення та видалення даних в довідниках в базі даних системи;

– ручну, автоматичну та автоматизовану генерацію розкладу навчального процесу;

– підтримку Drag-and-drop технологій при ручному формуванні розкладу;

– можливість поділу занять на підгрупи та об’єднання лекційних занять в потоки, врахування змінності;

– візуалізацію навантаження учасників навчального процесу (викладачів, студентів, аудиторного фонду) у вигляді діаграм та графіків;

– вибірку даних в розрізі факультетів, кафедр, навчальних груп, викладачів, навчальних корпусів та навчальних аудиторій;

– експорт даних в xml та xlsx файли.

2 МАТЕМАТИЧНА МОДЕЛЬ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Формалізація постановки задачі складання розкладу

Відповідно до навчальних планів у поточному семестрі необхідно прочитати *d* навчальних дисциплін S_1, S_2,… S_d: для кожного *i*, $i = \overline{1...d}$ дисципліна S_i складається з lec_i лекцій, $pract_i$практичних занять і семінарів й lab_iлабораторних робіт, $cons_i$ консультацій. Кожне заняття *s* проводиться визначеним викладачем pr_j. Є *gr* навчальних планів G_1, G_2,…, G_{gr}, які являють собою набір навчальних дисциплін, що читаються певній групі студентів Gr_k. Це означає, що дисципліни в G_j, $k = \overline{1...gr}$ повинні бути рознесені у часі.

ts_max – кількість навчальних періодів (пар), r_max_k – максимальна кількість занять, яку можна запланувати в період t_k (кількість аудиторій, доступних у період t_k, $k = \overline{1...ts\_max}$).

Необхідно розподілити заняття по всім курсам у межах певної кількості аудиторій і періодів часу, тобто

знайти таке s_{tr}^{ijk} ($i = \overline{1...d}$, $j = \overline{1...pr}$, $k = \overline{1...gr}$, $t = \overline{1...ts}$, $r = \overline{1...r\_max}$) , що

– розподіл усіх занять:

$$\forall i = \overline{1...d} \ \sum \{s_{tr}^{ijk} \mid j \in Pr, k \in Gr, \ t \in Ts, r \in R\} = lec_i + pract_i + lab_i + cons_i; \quad (2.1)$$

– використання обмеженого аудиторного фонду:

$$\forall t = \overline{1...ts\_max} \ \sum \{s_{tr}^{ijk} \mid i \in D, \ j \in Pr, k \in Gr, \ t \in Ts, r \in R\} \le r\_max_t; \quad (2.2)$$

– запобігання накладок по аудиторії:

$$\forall t = \overline{1...ts\_max} \ \forall r = \overline{1...r\_max} \ \sum \{s_{tr}^{ijk} \mid i \in D, \ j \in Pr, k \in Gr, \ t \in Ts\} \le 1; \quad (2.3)$$

– запобігання накладок по групі:

$$\forall t = \overline{1...ts\_max}\ \forall k = \overline{1...gr}\ \sum\{s_{tr}^{ijk} \mid i \in D,\ i \in G_k, j \in Pr, t \in Ts, r \in R\} \le 1; \quad (2.4)$$

– запобігання накладок по викладачу:

$$\forall t = \overline{1...ts\_max}\ \ \forall j = \overline{1...pr}\ \ \sum\{s_{tr}^{ijk} \mid i \in D,\ k \in Gr,\ t \in Ts, r \in R\} \le 1; \quad (2.5)$$

$$\forall i = \overline{1...d}\ \ \forall k = \overline{1...ts}\quad s_{ijk} \in \{0,1\}\quad s_{tr}^{ijk} = \begin{cases} 1, \text{якщо } d_i \text{ призначено на пару } t, \\ 0, \text{в іншому випадку.} \end{cases} \quad (2.6)$$

2.2 Змінні математичної моделі предметної області

Для побудови математичної моделі розкладу занять в університеті введемо змінні і визначимо обмеження.

Одиницею навчального процесу, і відповідно розкладу, є навчальне заняття. Навчальне заняття характеризується трійкою <Дисципліна, група, викладач>, інші параметри є необов'язковими (див. рис. 2.1).

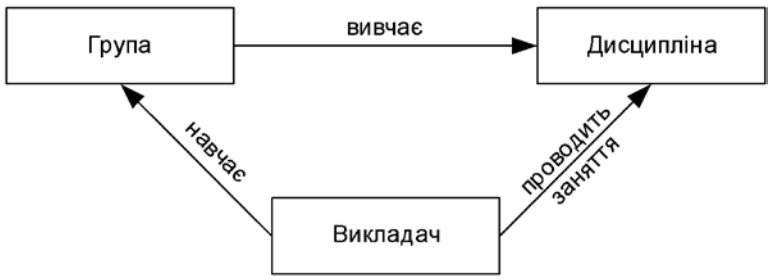


Рисунок 2.1 – Тріада <Дисципліна, група, викладач>

Математично заняття можна представити у вигляді множини *S*:

$$S = \{s_i \mid s_i = (s_i^{gr}, s_i^{pr}, s_i^{d}, s_i^{subgr}, s_i^{dep}, s_i^{study\_type}, s_i^{n})\}, i = \overline{1, s\_max}, \quad (2.7)$$

де s_max – кількість навчальних занять, які необхідно розмістити в розкладі;

s_i^{gr} – навчальна група;

s_i^{pr} – викладач, який проводить дані заняття;

s_i^{d} – дисципліна;

s_i^{subgr} – підгрупа;

s_i^{dep} – кафедра, яка читає дану дисципліну;

$s_i^{study\_type}$ – тип заняття (лекція, практичне, лабораторне заняття);

s_i^{n} – кількість занять в блоці.

При складанні розкладу кожному заняттю ставлять у відповідність таймслот s_i^t та аудиторію s_i^r.

Множина таймслотів (навчальних пар) *Ts*:

$$Ts = \{t_k \mid t_k = (t_k^w, t_k^d, t_k^p \mid k = \overline{1, d\_max \cdot t\_max})\} \quad (2.8)$$

де t_k^w – тиждень;

$t_k^d = \overline{1, d\_max}$ – день тижня;

d_max– кількість навчальних днів протягом тижня;

$t_k^p = \overline{1, p\_max}$ – номер пари;

t_max– максимальна кількість занять в день.

Окремі ресурси можуть бути недоступними протягом деяких таймслотів. Для врахування у розкладі недоступності ресурсів для кожного ресурсу вводиться календар доступності, який задається двовимірною матрицею, стовпці якої відповідають дню, а рядки – часу, окремо для кожного тижня.

Отже, попередньо недоступність ресурсів можна виразити додаванням

набору обмежень такої форми:

$$calendar_{ij} = \begin{cases} 1, \textit{ якщо ресурс доступний в день } i \textit{ та час } j; \\ 0, \textit{ якщо ресурс недоступний в день } i \textit{ та час } j. \end{cases} \quad (2.9)$$

Розклад занять може мати різну періодичність, наприклад, для шкільного розкладу типовий період повторення – один тиждень, для університету – два тижні. Часова сітка розкладу описується множиною таймслотів.

Наявні в задачі ресурси задаються у вигляді наступних множин.

Множина навчальних аудиторій ***R*:**

$$R = \{r_i \mid r_i = (r_i^n, r_i^b, r_i^{dep}, r_i^{study\_type}, r_i^v)\}, i = \overline{1, r\_max}, \quad (2.10)$$

де r_i^n – номер аудиторії;

r_i^b – навчальний корпус, в якому розташована аудиторія;

r_i^{dep} – кафедра, відповідальна за аудиторію;

$r_i^{study\_type}$ – тип аудиторії;

r_i^v – обсяг аудиторії (кількість робочих місць).

Тип аудиторії залежить від особливостей навчального закладу та визначає можливість її використання для проведення визначеного типу занять, наприклад, «лекційна», «комп'ютерний клас», «лабораторія».

Календар доступності навчальних аудиторій:

$$r_m^{w_i, d_j, t_k} = \begin{cases} 1, \textit{ якщо аудиторія зайнята на } i - \textit{тому тижні в день } d_j \textit{ на парі } t_k \\ 0, \textit{ якщо аудиторія вільна на } i - \textit{тому тижні в день } d_j \textit{ на парі } t_k \end{cases} \quad (2.11)$$

За замовчуванням $\forall d\ \forall t\ \forall i\ r_i^{d,t} = 0,$ тобто усі аудиторії вільні від занять.

Кожен студент може знаходитися на занятті у складі навчальної групи,

підгрупи, узагальненої групи (потоку) або віртуальної групи.

Множина навчальних груп *Gr*:

$$Gr = \{g_i | g_i = (g_i^n, g_i^v, g_i^r)\}, i = \overline{1...gr} \quad (2.12)$$

де gr – кількість груп в університеті;

g_i^n – номер групи;

g_i^r – розмір групи (кількість студентів).

За номером навчальної групи однозначно визначаються:

$$\forall i = \overline{1...gr}\ \exists!\ (g_i^{spec}, g_i^{kurs}, g_i^{dep}), \quad (2.13)$$

де g_i^{spec} – спеціальність, за якою навчається дана група,

g_i^{kurs} – курс;

g_i^{dep} – кафедра, до якої належить група.

Календар доступності групи:

$$g_l^{w_i, d_j, t_k} = \begin{cases} 1, \text{ якщо групі } l \text{ можна призначати заняття} \\ \text{на } i - \text{тому тижні в день } d_j \text{ на парі } t_k \\ 0, \text{ якщо групі } l \text{ не можна призначати заняття} \\ \text{на } i - \text{тому тижні в день } d_j \text{ на парі } t_k \end{cases} \quad (2.14)$$

Лекції з однієї дисципліни проводяться для всіх груп потоку одночасно одним викладачем, отже необхідно забезпечити поняття узагальненої групи (потоку).

Оскільки студенти можуть вибирати дисципліни для вивчення за бажанням (з переліку, представленого у блоці за вибором навчального плану), необхідно формувати тимчасові зведені (віртуальні) групи з різних академічних груп для

вивчення конкретної дисципліни. У різних підгрупах однієї навчальної групи заняття можуть проводитися одночасно, якщо вони ведуться різними викладачами.

Множина узагальнених груп (потоків):

$$Str = \{Str_i \subseteq Gr \mid i = \overline{1, str\_max}\}, \tag{2.15}$$

де Str – множина узагальнюючих занять (потоків);

Str_i – потік (множина занять, об'єднаних у потік);

S – загальна множина навчальних занять;

str_max– загальна кількість потоків.

Потоки можуть перетинатися, тобто містити спільних студентів, тобто

$$\exists\, i \in [1, str\_max],\ j \in [1, str\_max] : Str_i \cap Str_j \neq \varnothing. \tag{2.16}$$

де Str_i, Str_j – потоки;

str_max – загальна кількість потоків.

Заняття з фізкультури або дисциплін гуманітарного блоку, таких як філософія, соціологія, релігієзнавство тощо, може проводитися одночасно для всього курсу різними викладачами відповідної кафедри, отже, необхідно ввести поняття узагальненого викладача, який складається з реальних викладачів. Викладач може бути взагалі не прив'язаний до реальних викладачів, якщо заняття проводиться сторонньою особою, наприклад, представником бізнесу або профільних виробничих компаній, при цьому вводиться поняття віртуального викладача.

Множина викладачів:

$$Pr = \{Pr_i \subseteq Pr\_real \mid i = \overline{1,\ pr\_max}\}, \tag{2.17}$$

де Pr – множина узагальнених викладачів;

Pr_i – узагальнений викладач;

Pr_real – множина реальних викладачів, що входять до узагальненого викладача;

pr_max– загальна кількість узагальнених викладачів.

Узагальнені викладачі можуть перетинатися, тобто містити спільних реальних викладачів, тобто

$$i \in [1, pr\_max],\ j \in [1, pr\_max]:\ Pr_i \cap Pr_j \neq \varnothing. \tag{2.18}$$

Віртуальний викладач не містить реальних викладачів, тобто

$$\exists i \in [1, pr\_max]: Pr_i = \varnothing. \tag{2.19}$$

Отже, трійка <Дисципліна, група, викладач> приймає наступну форму (див. рис. 2.2).

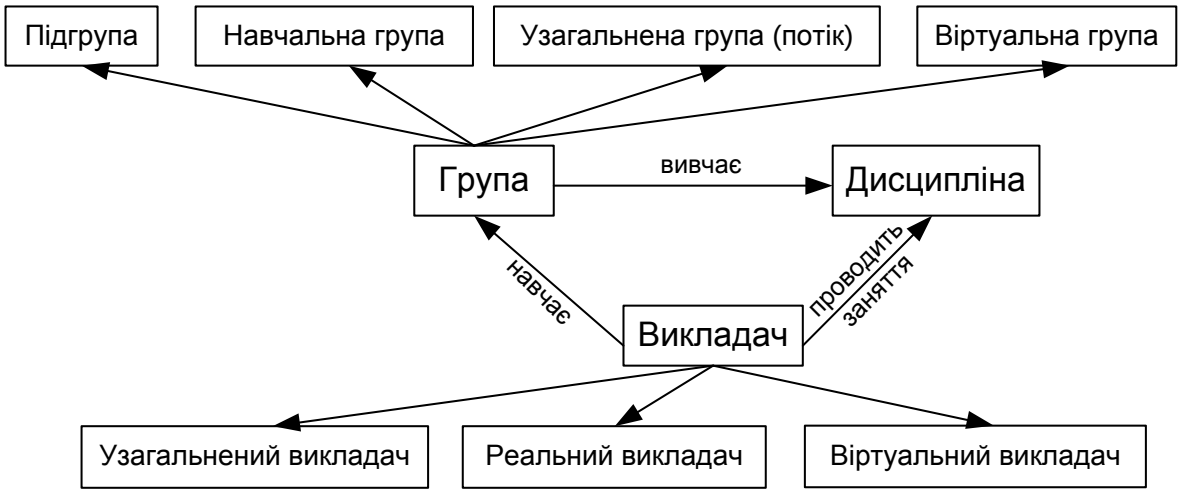


Рисунок 2.2 – Відношення між учасниками навчального процесу

При складанні розкладу для зведених груп або узагальненого викладача необхідно контролювати зайнятість усіх академічних груп, з яких складається зведена група, та усіх викладачів, з яких складається узагальнений викладач.

2.3 Представлення занять в розкладі

У роботі алгоритму використаємо матричне представлення розкладу. Оптимальним є представлення розкладу у вигляді паралелограма (див. рис. 2.3), у якого в якості осей виступають аудиторія, день та час, на перетині яких знаходиться заняття [21]. При цьому вказується лише номер заняття, за яким однозначно можна визначити викладача, групу та дисципліну.

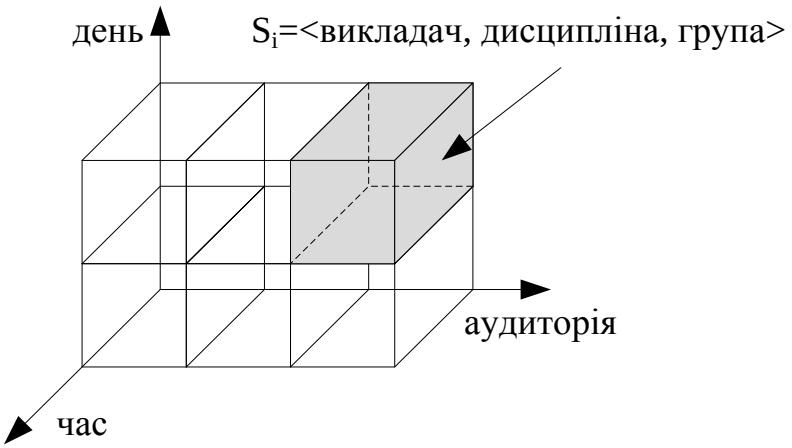


Рисунок 2.3 – Представлення занять в розкладі

Таке представлення забезпечить відсутність накладок по аудиторному фонду та можливість швидкої вибірки та візуалізації даних, наприклад, за допомогою OLAP-технологій.

Математично даний паралелограм опишемо у вигляді тривимірної матриці candidate_list {room, day, time}.

$$candidate\_list\{room, day, time\} = \begin{cases} i, \text{ якщо } S_i \text{ проводиться в } day, time \text{ в аудиторії } room \\ 0 \text{ в іншому випадку} \end{cases} \quad (2.20)$$

Взаємозв’язки між окремими заняттями у загальному розкладі навчального

закладу представляються у вигляді *матриці об'єднань* та *матриці конфліктів*, які складаються згідно навчальних планів.

Матриця об'єднань $M\_union_{s\_max \times s\_max}$ – це бінарна матриця така, що:

$$M\_union_{ij} = \begin{cases} 1, \textit{ якщо заняття } S_i \textit{ та } S_j \textit{ повинні проводитися} \\ \quad \textit{одночасно в одній аудиторії;} \\ 0 \quad \textit{в іншому випадку.} \end{cases} \tag{2.21}$$

Матриця об'єднань являє собою формалізацію поняття навчального потоку.

Матриця об'єднань автоматично формується наступним чином:

$$\forall (s_i,\ s_j) \mid s_i \in S, s_j \in S\ ((s_i^d = s_j^d) \wedge (s_i^{kurs} = s_j^{kurs}) \wedge (s_i^{spec} = s_j^{spec}) \wedge s_i^{lec} \wedge s_j^{lec} \wedge (s_i^{mod} = s_j^{mod})) \rightarrow M\_union_{ij} = 1. \tag{2.22}$$

В ручному режимі можна об'єднувати в потоки будь-які заняття.

Матриця конфліктів $M\_conflict_{s\_max \times s\_max}$ – це бінарна матриця така, що:

$$M\_conflict_{ij} = \begin{cases} 1, \textit{ якщо заняття } S_i \textit{ та } S_j \textit{ мають спільних студентів} \\ \quad \textit{або проводяться одним викладачем та не є потоком;} \\ 0, \textit{в іншому випадку.} \end{cases} \tag{2.23}$$

Матриця конфліктів являє собою формалізацію вимоги заборони одночасної присутності однієї групи / підгрупи або викладача на декількох заняттях. Заняття S_i та S_j, для яких $M\_conflict_{ij}$=1, не можна призначати одночасно.

Матриця конфліктів автоматично формується наступним чином:

$$\forall (s_i,\ s_j) \mid s_i \in S, s_j \in S\ ((i \neq j) \wedge (M\_union_{ij} = 0) \wedge \wedge ((s_i^{pr} = s_j^{pr}) \vee ((s_i^{gr} = s_j^{gr}) \wedge (s_i^{subgr} = s_j^{subgr})) \vee \vee ((s_i^{gr} = s_j^{gr}) \wedge (\neg s_i^{subgr} \vee \neg s_j^{subgr})))) \rightarrow M\_conflict_{ij} = 1. \tag{2.24}$$

2.4 Обмеження до складання розкладу

Якість складання розкладу навчального процесу визначається кількістю порушень обмежень, що висуваються до такого розкладу.

Корне (Corne, 1995 р.) та Льюіс (Lewis, 2008 р.) виділили п’ять основних класів, на які можна поділити обмеження до складання розкладу навчального закладу:

– унарні обмеження: стосуються лише однієї події, наприклад, «лабораторні заняття з програмування проводяться у комп’ютерному класі» або «заняття може проводитися лише у визначений період»;

– бінарні обмеження: включають дві події, наприклад, «лекція повинна передувати практичному заняттю»;

– ємнісні обмеження: ці обмеження стосуються призначення аудиторій: призначена аудиторія повинна бути достатньо великою щоб вмістити усіх студентів, записаних на курс, та мати достатню кількість робочих місць;

– обмеження щодо розподілу подій у часі: політика складання розкладу часу повинна бути направлена на зменшення навантаження студентів та викладачів; зокрема це означає, наприклад, таке обмеження щодо розподілу подій: «група не повинна мати більше чотирьох пар поспіль»;

– агентні обмеження: зазвичай задаються викладачами (професорами), які не можуть проводити заняття протягом окремих пар з певних причин або зайнятістю відповідних аудиторій.

Усі вимоги, що висуваються до розкладу навчального процесу, поділяються на жорсткі та м’які.

Жорсткі умови повинні виконуватись завжди, інакше розклад є хибним і збитковим. Нежорсткі умови можуть і не виконуватись, але їх виконання напряму впливає на ефективність розкладу з психологічної точки зору.

Жорсткі вимоги:

– будь-який студент в складі академічної, зведеної (потік) або віртуальної

групи (вивчення дисципліни на вибір студента) вивчає тільки одну дисципліну одночасно;

– будь-який викладач в заданий таймслот веде тільки одну дисципліну, тільки одній академічній, зведеній або віртуальній групі одночасно;

– у будь-якій аудиторії ведеться лише одне заняття для однієї академічної, зведеної або віртуальної групи одночасно;

– доступність учасників навчального процесу (викладача, групи, аудиторії) в заданий таймслот;

– відповідність типу аудиторії заняттю та достатня кількість місць в аудиторії відповідно до кількості студентів.

М’які вимоги:

– послідовність проведення занять курсу (цикл занять не може починатися з практичного заняття або закінчуватися лекцією, до лекції кожна група повинна підходити з однаковою кількістю практичних або лабораторних занять);

– послідовність проведення занять (заборона проведення лекцій або-будь-яких занять після фізкультури);

– ефективне використання аудиторій (рівномірне завантаження аудиторного фонду, відповідність кількості робочих місць кількості студентів);

– компактність та рівномірність навантаження студентів та викладачів відсутність «вікон» у розкладі студентів та викладачів;

– мінімізація переходів між корпусами;

– врахування побажань викладачів та студентів щодо бажаного/ небажаного часу проведення занять.

Представлені обмеження можуть бути згруповані таким чином (див. рис. 2.4).

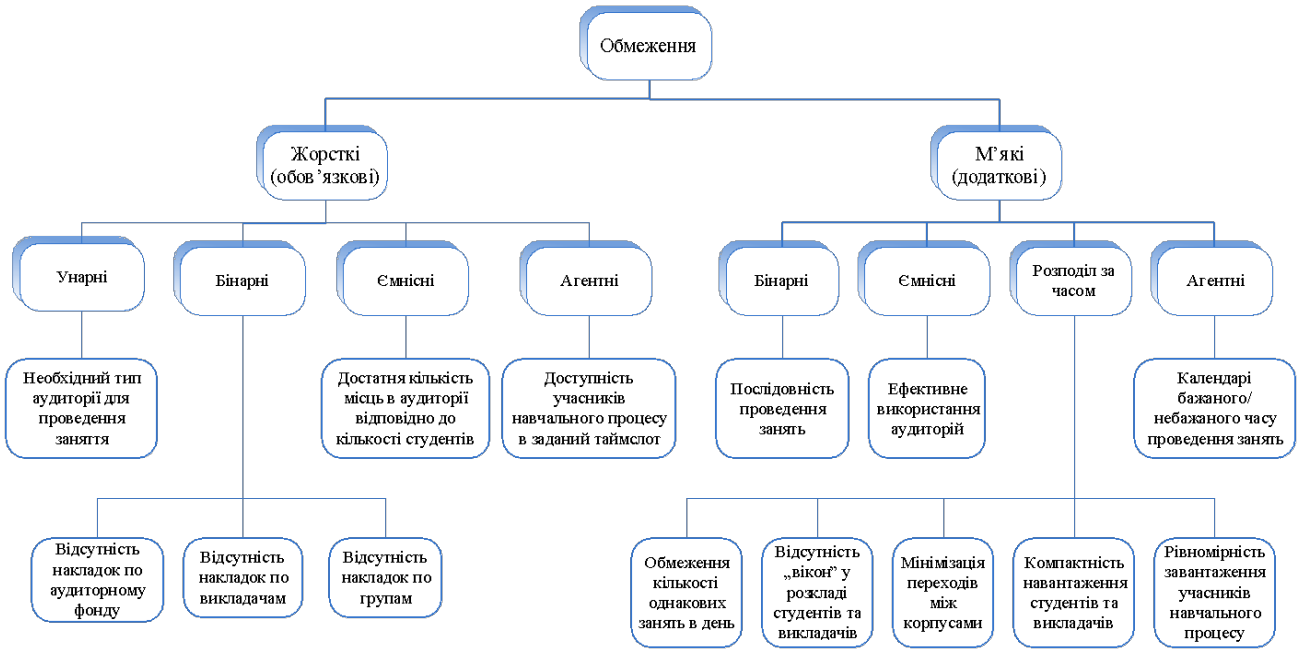


Рисунок 2.4 – Класифікація обмежень до розкладу

Розглянемо математичне представлення деяких обмежень.

Відсутність аудиторних накладок: призначення двох та більше занять в одну аудиторію одночасно:

$$\forall(r_i,\ t_j)\,|\,r_i \in R, t_j \in T\ (\exists!\, s_{r_i k}\,|\,s_{r_i k} \in S_{t_j}) \vee (\neg\exists\ s_{r_i k}\,|\,s_{r_i k} \in S_{t_j}) \qquad (2.25)$$

або

$$\forall(r_i,\ t_j)\,|\,r_i \in R, t_j \in T \sum(s_{r_i k}\,|\,s_{r_i k} \in S_{t_j})) \le 1. \qquad (2.26)$$

Для кожної упорядкованої пари елементів {аудиторія та таймслот}, існує лише одне заняття з множини S, що означає проведення цього заняття в цій аудиторії на даній парі, або заняття відсутнє, що означає, що аудиторія вільна.

Відсутність накладок для викладача: призначення двох та більше занять, які проводить один викладач, одночасно:

$$\forall(pr_i,\ t_j)\,|\,pr_i \in Pr, t_j \in T\ (\exists!\, s_{pr_i k}\,|\,s_{pr_i k} \in S_{t_j}) \vee (\neg\exists\ s_{pr_i k}\,|\,s_{pr_i k} \in S_{t_j}) \qquad (2.27)$$

або

$$\forall(pr_i,\ t_j)\,|\,pr_i \in Pr, t_j \in T \sum(s_{pr_i k}\,|\,s_{pr_i k} \in S_{t_j})) \le 1. \quad (2.28)$$

Для кожної упорядкованої пари елементів {викладач та таймслот}, існує лише одне заняття з множини *S*, що означає проведення цього заняття даним викладачем на даній парі, або заняття відсутнє, що означає, що викладач вільний.

Відсутність накладок для групи / підгрупи: призначення двох та більше занять, які проводяться в одній групі/підгрупі, одночасно: під час пари проходить одне заняття в усій групі, в одній з підгруп даної групи або в обох підгрупах даної групи одночасно, або таке заняття відсутнє.

Мовою предикатів дане обмеження записується так:

$$\forall gr_i\ \forall podgr_j\ \forall t_k\,|\,\mathrm{g}r_i \in Gr,\ podgr_j \subseteq gr_i, t_k \in T \sum(s_{gr_i t_k} + s_{gr_i\, podgr_j t_k}) \le 1. \quad (2.29)$$

Відповідність типу аудиторії типу заняття:

$$\forall i\ |\ s_{tr}^i \in S, r \subseteq R_{pos}^i, \quad (2.30)$$

де s_{tr}^i – поточне заняття;

r – поточна аудиторія;

R_{pos}^i – множина допустимих аудиторій для даного заняття.

Множина допустимих аудиторій R_{pos}^i для кожного виду занять визначається за типом аудиторії (комп'ютерний клас, лекційна, лабораторія тощо) та приналежністю до визначеної кафедри або задається вручну.

Достатня кількість місць в аудиторії відповідно до кількості студентів:

$$\forall i\ |\ s_{tr}^i \in S, r_{capacity} \ge s_{capacity}, \quad (2.31)$$

де s_{tr}^{i} – поточне заняття;

r – поточна аудиторія;

$r_{capacity}$ – ємність аудиторії r;

$s_{capacity}$ – кількість студентів, записаних на дане заняття.

Рівномірність навантаження студентів та викладачів задається дисперсією кількості занять протягом навчального тижня, наприклад:

$$\overline{N}_{gr} = \frac{1}{d}\sum_{i=1}^{d} N_{gr}^{d_i}, \tag{2.32}$$

$$D = \frac{1}{d}\sum_{i=1}^{d} (\overline{N}_{gr} - N_{gr}^{d_i})^2 \to min, \tag{2.33}$$

де $\overline{N}_{gr}$ – середнє завантаження групи протягом навчального тижня;

$N_{gr}^{d_i}$ – завантаження групи протягом дня d_i.

Мінімізація «вікон» у розкладі студентів та викладачів:

$$W_{gr}^{d_i} = \begin{cases} t\_max_{gr}^{d_i} - t\_min_{gr}^{d_i} - N_{gr}^{d_i} + 1, \text{ якщо } t\_max_{gr}^{d_i} \neq 0, \\ 0 \text{ в іншому випадку}, \end{cases} \tag{2.34}$$

$$W = \sum_{i=1}^{d} W_{gr}^{d_i} \to min, \tag{2.35}$$

де $t\_max_{gr}^{d_i}$ – максимальний номер пари у групи;

$t\_min_{gr}^{d_i}$ – мінімальний номер пари у групи;

$N_{gr}^{d_i}$ – завантаження групи протягом дня d_i.

Розташування корпусів задається у вигляді пари координат (x, y). Тоді відстань між двома корпусами можна оцінити як евклідову відстань у двовимірному просторі:

$$d=\sqrt{(x_2 - x_1)^2 + (x_2 - x_1)^2} \to min, \quad (2.36)$$

де x_1, y_1, x_2, y_2 – координати першого та другого корпусу відповідно.

Для викладачів задаються наступні обмеження:

– якщо відстань дорівнює нулю (одна і та ж будівля), то немає штрафу;

– якщо відстань вище нуля, але не більше 50 метрів, то перехід можливий, але накладається незначний штраф;

– якщо відстань становить від 50 до 200 метрів, перехід небажаний та накладається значний штраф.

Для студентів необхідно забезпечити часову можливість переходу між аудиторіями за 10 хвилин протягом звичайної перерви та за 30 хвилин – протягом великої перерви. Відстань між корпусами не більше 200 метрів будемо вважати допустимою для переходу. Якщо відстань більше ніж 200 метрів, аудиторії розташовані дуже далеко і перехід між ними протягом звичайної перерви неможливий.

2.5 Цільова функція

Завдання складання розкладу відноситься до NP-повних задач багатокритеріальної комбінаторної оптимізації. Кожний складений розклад характеризується цільовою функцією за критеріями оптимальності, яку необхідно мінімізувати на множині допустимих розкладів. Цільова функція в задачах теорії розкладів розраховується на основі набору штрафів (штрафних функцій), які виникають при невиконанні заданих обмежень в розкладі.

Розглянемо вектор змінних параметрів X, на основі якого формується множина допустимих розкладів:

$$X = (\{S_i, R_j, T_k\}), \quad (2.37)$$

$$| X |= s\_max \times r\_max \times t\_max \quad (2.38)$$

В якості змінних параметрів приймемо номер заняття S_i, аудиторію R_j та таймслотT_k, призначені даному заняттю.

Обмеження, які визначають допустимість розкладу, – жорсткі обмеження до розкладу. За умовою

$$F_i(X) = 0, \quad (2.39)$$

де F_i – функція штрафу за порушення і-того обмеження (цільова функція).

Невиконання жорстких обмежень дозволяє виключити з розгляду варіанти розкладів, що є свідомо неконкурентно спроможними.

Критеріальні обмеження – м'які обмеження. За умовою

$$F_i(X) \to min. \quad (2.40)$$

Можна виділити наступні типи критеріїв оптимальності:

а) мінімаксні критерії – в задачі з такими критеріями цільова функція являє собою функцію максимуму від значень штрафів вимог. Приклади мінімаксних критеріїв:

1) Wmax→ min – критерій мінімізації максимального «вікна» в день;
2) $T_{max} \to min$ – критерій мінімізації максимального часу переходу між корпусами / аудиторіями.

б) сумарні критерії – в задачі з такими критеріями цільова функція являє собою суму значень штрафів вимог/ обмежень. Приклади сумарних критеріїв:

1) $\sum_{i \in N_{gr}} W_i \to$ min – критерій мінімізації сумарної кількості «вікон» розкладу всіх навчальних груп;
2) $\sum_{i \in N_r} C_i \to$ min – критерій мінімізації сумарної кількості перевищення ємності аудиторій.

Підрахувавши значення штрафів за всіма критеріями, отримуємо вектор

критеріїв оптимальності:

$$F(X) = \left(F_1(X),\ F_2(X), ...,\ F_{|F|}(X)\right). \quad (2.41)$$

Основні труднощі, що виникають при визначенні цільової функції, полягають у складності визначення вагових коефіцієнтів, які відображають ступінь важливості критерію у процентному співвідношенні. Якщо задані ваги вимог/обмежень ω_i, використовуються зважені критерії оптимальності, їх значення обчислюється шляхом добутку початкового значення на коефіцієнт ω_i. При ручному складанні розкладу зазвичай урахування ваг вимог/обмежень забезпечується досвідом роботи диспетчера, в той час як при автоматизованому складанні розкладу необхідно передбачити задання ваг та можливість врахування окремих критеріїв, оскільки важливість критеріїв залежить від ЗВО та може змінюватися. Наприклад, «час переходу між корпусами» неважливий для університетів, будівлі яких розташовані компактно, та критично важливий для університетів, що мають відокремлені корпуси в іншій частині міста.

Введемо відносні вагові коефіцієнти важливості критеріїв $\omega_i \geq 0$ $(\sum_{i=1}^{|F|} \omega_i = 1)$. За замовчуванням $\omega_i = 1/|F|$, інші значення визначаються диспетчером за допомогою методу аналізу ієрархій та попарних порівнянь. Нормалізація критеріїв виконується з урахуванням *min* та *max* значень.

Таким чином, наявність множини критеріїв дозволяє віднести завдання складання розкладу занять до класу багатокритеріальних задач, коли необхідно побудувати оптимальне рішення з точки зору декількох цільових установок і при цьому критерії якості суперечать один одному: бажання отримати розклад з найкращим значення одного з критеріїв призводить до погіршення значень інших критеріїв.

Багатокритеріальні оптимізаційні задачі зводяться до однокритеріальних. В іншому випадку знаходяться Парето-оптимальні рішення або оптимум ієрархічної цільової функції.

Розглянемо можливі варіанти зведення багатокритеріальної задачі складання розкладу до однокритеріальної:

– принцип оптимальності Парето дозволяє звести множину усіх допустимих розкладів до множини Парето-оптимальних рішень:

$$P_f(X) = \{x^* \in X \mid \text{немає такого } x \in X, \text{ що } f(x) \geq f(x^*)\}. \qquad (2.42)$$

При цьому для остаточного вибору одного рішення з Парето-оптимальної множини необхідні додаткова інформація.

– метод головного критерію: в якості цільової функції обирається значення за одним із критеріїв і розв'язується однокритеріальна задача, в якій інші критерії беруть участь лише у вигляді додаткових обмежень, які дозволяють врахувати вимоги, які задаються іншими критеріями.

$$F(X) = F_i(X). \qquad (2.43)$$

Даний метод є слабоефективним, якщо критерії вступають між собою в протиріччя та є рівнозначними.

Методи згортки векторного критерію оптимальності в скалярний:

– метод лінійної адитивної згортки критеріїв реалізує принцип справедливої компенсації абсолютних значень окремих критеріїв. При цьому цільова функція визначається як зважена сума окремих критеріїв.

Види лінійної адитивної згортки:

– метод лінійної адитивної згортки з нормуючими множниками:

$$F(X) = \sum_{i=1}^{|F|} \lambda_i F_i(X); \qquad (2.44)$$

– метод лінійної адитивної згортки з ваговими коефіцієнтами:

$$F(X) = \sum_{i=1}^{|F|} \lambda_i \beta_i Z_i(X). \quad (2.45)$$

Вибір виду згортки визначається характером взаємозв'язків її критеріїв (рівнозначні, домінуючі тощо) та обмеженнями на область значень згортки.

У даному випадку виберемо метод лінійної адитивної згортки з ваговими коефіцієнтами як достатньо об’єктивний і такий, який дозволяє врахувати абсолютні значення всіх окремих критеріїв.

– метод мультиплікативної згортки критеріїв можна розглядати як принцип справедливої компенсації відносних змін значень окремих критеріїв:

$$F(X) = \prod_{i=1}^{|F|} \lambda_i F_i(X). \quad (2.46)$$

При цьому згортка базується на принципі: “низька оцінка за одним критерієм призводить до низького значення функції цільової функції”, що не підходить у даному випадку, оскільки неможливо забезпечити виконання вимог за усіма критеріями.

В роботі [5] розглянуті аспекти формування та направленої оптимізації цільової функції:

$$F(r) = \alpha_S \sum_{j=1}^{l} x_j \chi\{Z_j^{\nu}\} + \sum_{j=1}^{k} y_j \sum_{i=1}^{m} \chi\{L_i \in T_j\} \sum_{l=1}^{n_i} d_{il}^{j} \chi\{Z_{il}^{T_j}\} \to \max, \quad (2.47)$$

$$r \in \Omega(P, S, L, A), \quad (2.48)$$

де r – розклад;

Ω – область обмежень;

P – множина навчальних дисциплін;

L – множина викладачів;

A – множина аудиторій;

α_S, α_L – вагові коефіцієнти (пріоритети викладачів і студентів);

x_j – пріоритети вимог студентів і викладачів;

Z_j^{ν} – вимоги навчальних груп;

L_i – викладачі;

T_j – групи викладачів;

$Z_{il}^{T_j}$ – переваги викладачів;

d_{il}^{j} – пріоритети переваг;

l – кількість вимог студентів;

k – кількість груп викладачів, що визначаються їхніми посадами, науковими ступенями та вченими званнями;

m – кількість викладачів;

n_i – кількість викладачів в i-й групі.

В роботі [21] пропонується такий розрахунок цільової функції:

$$F = max\left(\sum_{k=1}^{m_t} w_k \sum_{j=1}^{m_x} \frac{v_{kj}\pi_{kj}}{\sum_{i=1}^{m_x} \pi_{ki}} + \sum_{k=1}^{m_h} n_k h_k \right), \tag{2.48}$$

де $v_{kj} \in \{0,1\}$, $j = \overline{1, m_x}$ – ознака врахування додаткової умови для викладача;

π_{kj} – пріоритет реалізованої умови;

w_k – ваговий коефіцієнт рейтингу викладача;

π_{ki} – пріоритети комплексу умов для відповідного викладача.

Отже, в якості цільової функції будемо використовувати адитивну згортку окремих критеріїв (за кількістю м’яких обмежень) з ваговими коефіцієнтами:

$$F = \omega_i F_i(X) \to \min. \tag{2.49}$$

При порушенні жорстких вимог складений розклад вважається неприпустимим.

3 МЕТОДИ РОЗВ'ЯЗКУ ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДУ НАВЧАЛЬНИХ ЗАНЯТЬ

3.1 Дослідження методів розв'язку задачі складання розкладу

Завдання побудови розкладів в університеті є досить складним. В загальному випадку воно відноситься до класу оптимізаційних NP-повних задач з обмеженнями. Отже, розв'язувати дану задачу методом повного перебору недоцільно і необхідно застосовувати наближені методи.

Для автоматичного складання широко використовуються як класичні методи такі, як лінійне цілочисельне програмування [22], метод розфарбування графу [8], метод імітаційного моделювання [7, 23], так і метаевристичні методи, у тому числі метод імітації відпалу, метод мурашиних колоній та еволюційні алгоритми.

Класифікація основних методів, що застосовуються при автоматичному складанні розкладів наведена на рисунку 3.1.

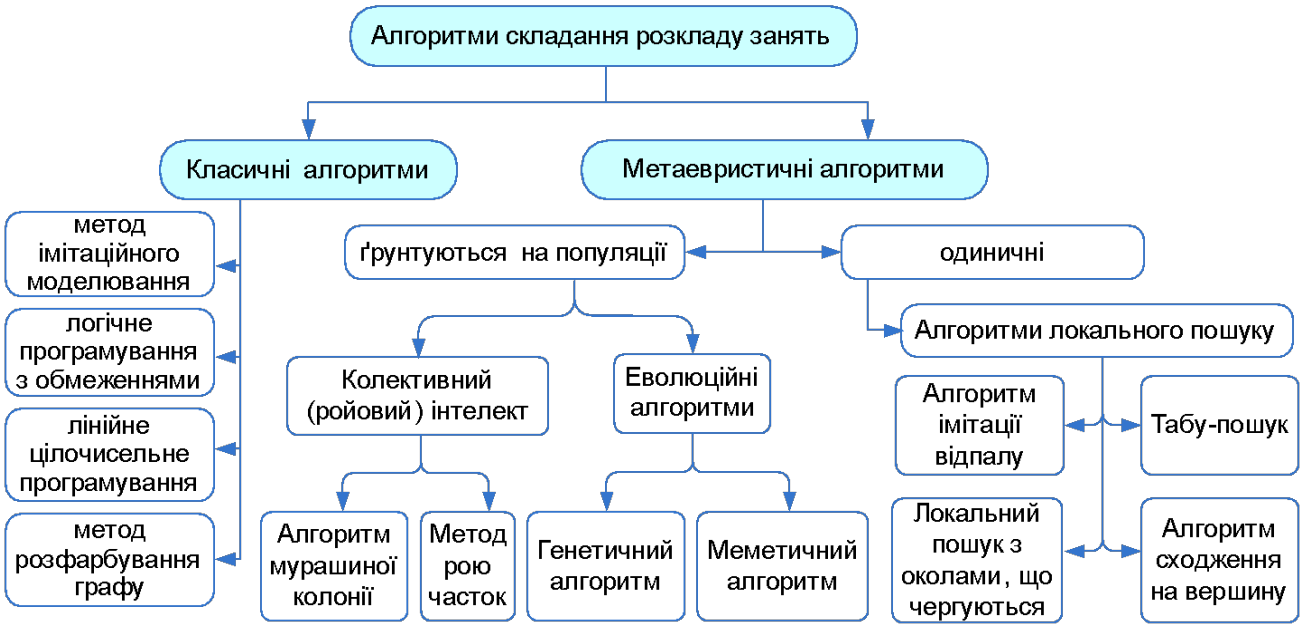


Рисунок 3.1 – Класифікація основних методів, що застосовуються при автоматичному складанні розкладів

Класичні методи в основному використовують ітераційну техніку неперервної оптимізації, тому в процесі складання розкладу можливе зациклення в локальному оптимумі. Тому для пошуку глобального оптимуму доцільно використовувати еволюційні алгоритми або алгоритми, що використовують колективний (ройовий) інтелект.

Метаевристичні методи починають з одного або декількох початкових розв'язків і використовують стратегії пошуку, за допомогою яких намагаються уникнути локальних оптимумів. Ці алгоритми пошуку можуть забезпечувати високоякісні рішення, але часто мають значну обчислювальну вартість.

Розглянемо наведені методи.

При складанні розкладу можна використовувати метод імітаційного моделювання [7, 23], який виконує імітацію дій диспетчера.

Алгоритм складання розкладу методом імітаційного моделювання:

а) на основі аналізу «вузьких місць» вибирається навчальне заняття, яке ще не розміщене в розкладі. В першу чергу складається розклад для найбільш дефіцитних ресурсів, наприклад, для викладачів, які мають обмежені можливості щодо часу занять або для найбільших потоків, які потребують великі аудиторії;

б) для обраного навчального заняття визначаються допустимі варіанти розміщення в розкладі, які задовольняють всім жорстким обмеженням. Кожний варіант оцінюється за допомогою евристичної цільової функції і найкращий варіант фіксується в розкладі;

в) при виникненні конфліктів заняття, що конфліктують, видаляються з розкладу і повертаються до списку неврахованих занять.

Реалізація алгоритму, що базується на принципах імітаційного моделювання, дозволяє врахувати специфіку конкретного ЗВО, однак при цьому обмежується можливість використання розробленої системи в інших ЗВО, крім того, при незначних внутрішніх змінах у ЗВО доводиться вносити істотні зміни в алгоритм. Крім того, необхідно відмітити значну залежність алгоритму від дій користувача, що робить роботу алгоритму при великій розмірності задачі неефективною.

Задачу формування розкладу можна розв'язати за допомогою методу розфарбування графа. Розфарбуванням графа називається процес призначення певного кольору визначеним елементам графа (вершинам або ребрам). При реалізації даного метода будується граф, в якому кожна вершина являє собою навчальне заняття. Якщо між двома вершинами можлива конфліктна ситуація (наприклад, обидва заняття проводяться одночасно та містять спільних студентів), вони поєднуються ребром. Наявність ребра відповідає забороні одночасного проведення цих занять. Тоді задачу формування розкладу можна представити як задачу розфарбування графа в мінімальну кількість кольорів, кожен з яких відповідає одному таймслоту, відповідно множині обмежень.

Застосування методу розфарбування графа для складання розкладу занять в університеті малоефективне, але може використовуватися в комбінації з іншими алгоритмами.

Для складання розкладу можна застосовувати логічне програмування з обмеженнями.

Зазвичай системилогічного програмування з обмеженнями являють собою інтерпретатор мови Пролог з вбудованим механізмом логічного виводу. Сутність методу полягає в тому, що користувач визначає деяку множину змінних $x_1,\ldots,x_n$ та їхню область значень $X_1,\ldots,X_n$, а також описує додаткові обмеження, яким повинні задовольняти введені змінні. Система повинна знайти значення цих змінних, які б задовольняли одночасно всім заданим обмеженням.

Наведемо приклад опису змінних в рамках методу. Розглянемо ЗВО, в якому працює Pr викладачів, ведеться N дисциплін. Визначимо множину таймслотів (пар) протягом тижня $T = 1,2,...,T\_max$. Нехай t_{ij} – пара, на якій i-й викладач проводить j заняття, $t_{ij} \in T$. Тоді обмеження «викладач не може одночасно вести більше одного заняття» можна представити як:

$$t_{ij} \neq t_{ij'},\ 1 \le i \le Pr,\ 1 \le j \le N,\ 1 \le j' \le N,\ j \neq j'. \qquad (3.1)$$

Аналогічно описуються й інші обмеження. Результатом роботи алгоритму є

множина значень змінних, що задовольняють заданим обмеженням. Основна перевага використання даного методу – скорочення простору пошуку за рахунок автоматичного виключення системою з розгляду безперспективних розв'язків.

На даний час для розв'язання задачі складання розкладу в університеті дуже поширене застосування метаевристичних методів. Застосовують такі метаевристичні методи як метод імітації відпалу [12], генетичні алгоритми [7,9,11], метод мурашиних колоній [14, 25] тощо. Велику кількість застосувань метаевристик до задачі про складання розкладу розглянуто в роботі [23].

Метаевристичні методи починають з одного або декількох початкових розв'язків і використовують стратегії пошуку, за допомогою яких намагаються уникнути локальних оптимумів. Ці алгоритми пошуку можуть забезпечувати високоякісні рішення, але часто мають значну обчислювальну вартість.

Метаевристичні методи зазвичай мають дві важливі особливості:

− у результаті їх роботи послідовно будуються кілька розв'язків;

− побудова кожного нового рішення ґрунтується на накопичених знаннях про якість попередніх отриманих рішень.

Застосування методу імітації відпалу розглядається в роботі [12]. В роботі [23] наводиться короткий огляд методів, що ґрунтуються на імітації відпалу.

Метод імітації відпалу [7,12] ґрунтується на дослідженні поведінки атомів металу при кристалізації речовини з рідкого стану в твердий під час процесу відпалу. У фізиці відпалом називається тепловий процес отримання низьких енергетичних станів тіла в тепловій бані (термостаті), який складається з підвищення температури термостату до максимального значення та поступового її зниження до впорядкування частинок в основний стан тіла (Kirkpatrick, 1983). У металі при температурі, що перевищує точку плавлення, атоми розташовуються випадковим чином. Під час охолодження метал переходить у більш низько-енергетичний стан до досягнення найнижчого із можливих станів – глобального мінімуму.

Розглянемо загальну схему роботи методу імітації відпалу:

а) формується допустимий початковий розклад $X=X_0$;

б) встановлюється початкове високе значення температури T_0 і задається операція мутації розкладу, яку можна реалізувати зміною таймслоту для навчального заняття або обміном місцями двох навчальних занять;

в) за допомогою операцій мутації на основі поточного розкладу формується новий допустимий розклад X';

г) визначається зміна цільової (штрафної) функції $\Delta f = f(X') - f(X)$:

1) при $\Delta f < 0$ (новий розклад краще попереднього) розклад X' приймається за поточний, тобто $X = X'$.

2) при $\Delta f > 0$ (новий розклад гірше попереднього) розклад X' приймається за поточний з ймовірністю $p(\Delta f, T)$. Як правило, в якості $p(\Delta f, T)$ вибирається точне значення відповідної фізичної величини $p = 1/(1 + e^{\Delta f / T})$ або її наближене значення $p = e^{-\Delta f / T}$.

д) задається зменшення поточної температури відповідно до заданого твірного сімейства імовірнісних розподілів $\varsigma(X,T)$, яке при фіксованих X і T задає випадковий елемент $G(X,T)$ в просторі S (S – множина енергетичних станів уявної фізичної системи).

Ймовірність прийняття розкладу з більшим значенням штрафної функції в якості поточного зменшується експоненціально на кожній ітерації. Функцію зміни температури можна пов’язати з кількістю проведених ітерацій або з величиною зміни штрафної функції. При високих температурах відбувається різка зміна штрафної функції, зі зменшенням температури її значення прямує до мінімуму.

е) якщо не виконується критерій зупинки, то перейти до п. 3. В якості критеріїв зупинки задаються такі умови: досягнення заданого числа ітерацій або виконання заданого числа ітерацій без істотного покращення значення штрафної функції $|\Delta f| < \varepsilon$).

Зменшення швидкості охолодження призводить до підвищення точності розв’язку, отриманого алгоритмом, оскільки при цьому забезпечується більш ретельне дослідження простору пошуку алгоритмом, але також до суттєвого збільшення часу роботи алгоритму.

Метод імітації відпалу ефективно працює для складання невеликих розкладів, але не підходить для складання розкладів великої розмірності.

Головний недолік даного методу – можливість порушення жорсткого обмеження під час операції мутації. Крім того, зміну температури необхідно підбирати окремо для кожної конкретної задачі, при цьому знаходження мінімуму функції не гарантується.

Метод мурашиної колонії.

Метод мурашиної колонії, запропонований М. Доріго в 1992 р. [14, 28,31,33], – це ймовірнісна техніка розв'язування обчислювальних задач, яка може бути зведена до пошуку кращих шляхів за допомогою графів з використанням штучних мурах, які наслідують поведінку колонії природних мурах.

Цей ітераційний алгоритм штучного інтелекту ґрунтується на ідеї послідовного наближення до оптимального рішення. Кожна ітерація – запуск штучної мурахи, який намагається за деяким правилом вибрати найкращий маршрут до «їжі» (оптимуму цільової функції), використовуючи мітки своїх попередників.

Кожна мураха виконує ланцюжок кроків:

а) для кожного заняття послідовно вибирається заняття *j* для постановки на місце *i* в розкладі, використовуючи ймовірності переходу. Мураха рухатиметься від вузла до вузла з ймовірністю:

$$p_{ij} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\Sigma_{j \in allowed\ nodes}(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}, \qquad (3.2)$$

де $\tau_{i,j}^{\alpha}$ – кількість феромонів у вершині *i, j* (накопичена статистична інформація про якість вибору для позиції і вимоги j);

α – параметр, який контролює вплив $\tau_{i,j}^{\alpha}$; при $\alpha = 0$ здійснюється вибір найзручнішої позиції, в основному це означає, що алгоритм стає жадібним.

$\eta_{i,j}^{\beta}$ – привабливість ребра (евристична інформація про те, наскільки хорошим

«здається» заняття *j* на місце *i* в розкладі);

β – параметр, який контролює вплив $\eta_{i,j}^{\beta}$; при β = 0 вибір здійснюється лише на основі феромону, що призводить до неоптимальних розв'язків.

Компромісне співвідношення між α та β знаходиться експериментальним шляхом.

б) здійснюється перерахунок (оновлення) феромонного сліду (вироблення і випаровування феромону) за формулами:

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \Delta\tau_{i,j} = (1-\rho)\tau_{i,j}(t) + \sum_{\substack{k \in \ ants\ that \\ used\ edqe\ (i,j)}} Q / L_k \qquad (3.3)$$

де $\tau_{i,j}$ – кількість феромону на вершині *i*, *j*;

ρ – швидкість випаровування феромону;

$\Delta\tau_{i,j}$ – кількість відкладеного феромона.

в) найкращим вважається рішення з найбільшим значенням локального феромонного сліду.

У різні часи було розроблено декілька розширень методу мурашиних систем [33] , до яких відносяться: метод мурашиних систем, заснований на елітній стратегії; метод мурашиних систем, заснований на ранжуванні (ASrank); метод системи мурашиних колоній; максімінний метод мурашиних систем (MAX-MIN AS – MMAS).

У даному випадку будемо використовувати метод системи мурашиних колоній (ACS), який акумулює переваги та поліпшує інші методи. ACS використовує псевдопропорційне правило для визначення наступної вершини для переміщення [33]: з ймовірністю q_0 агент переміщується у вершину, для якої добуток кількості феромонівта евристичної інформації є максимальним; з ймовірністю $1 - q_0$ використовується базовий підхід при визначенні наступної вершини для переходу, описаний у методі мурашиних систем.

При цьому використовується сувора елітна стратегія при відновленні феромонів на ребрах. Лише кращий агент (глобально або локально) виконує додавання феромонів після кожної ітерації методу.

Алгоритм роботи методу системи мурашиних колоній представлений на рисунку 3.2.

```
create_ants(outants[])
create_pheromone_matrix (outpheromone_matrix)
whileiterationδmax_iterationorfitness_functionδmax_fitnessdo
     forant = 1 tomax_antdo
          create_candidate_list (ant, outcandidate_list_ant)
          reset_ant_path(ant, outant_path_ant)
          set_ant_path(ant, pheromone_matrix, outcandidate_list_ant,
          outant_path_ant)
          calculate_local_pheromone (ant_path_ant, out
          local_pheromone_ant)
          local_search_algorithm(ant_path_ant)  //optional
     pheromoneevaporation(outpheromone_matrix)
     update_global_pheromone_trail(local_pheromone[], out
     pheromone_matrix)
endwhile
output_best_ant_path (ant_path[])
```

Рисунок 3.2 – Алгоритм методу мурашиних колоній

Метод мурашиних колоній детально розроблений для задачі комівояжера, але роботи, присвячені складанню розкладу навчального процесу даним методом, містить лише загальні теоретичні міркування щодо можливості використання цього метода для даної задачі.

Використання методу мурашиної колонії для розв'язання оптимізаційних задач розглянуто у [33]. У [14] розглядається використання Best-WorstAntSystem (BWAS) та Best-WorstAntColonySystem для складання розкладу.

Перевагами метода мурашиних колоній є гарантування збіжності до оптимального рішення, а також стохастичність пошуку, за рахунок чого зменшується можливість зациклення в локальному оптимумі.

Разом з тим у [6] відмічається невизначеність часу збіжності при тому, що збіжність гарантується; сильна залежність результатів роботи методу від

початкових параметрів пошуку, які підбираються експериментально.

Тому на нашу думку, ефективним підходом є використання гібридного підходу, зокрема, комбінації метода мурашиних колоній з генетичними алгоритмами.

Генетичні алгоритми.

Генетичні алгоритми – це стохастичні еволюційні методи оптимізації, основані на принципах біологічної еволюції видів [7]. Головним фактором еволюції є природний добір, який полягає в тому, що найбільш пристосовані особини до умов навколишнього середовища мають найбільше шансів на виживання та розмноження. При цьому завдяки передачі генетичної інформації наступні покоління успадковують від батьків спадкові характеристики.

Основні етапи генетичного алгоритму – послідовний вибір, схрещення та модифікація шуканих параметрів (генів хромосом популяції), при цьому робиться акцент на використання оператора «схрещення», який виконує рекомбінацію розв’язків-кандидатів, аналогічно ролі схрещення в живій природі.

Порядок роботи генетичного алгоритму:

а) розробляється структура хромосоми, в якій буде зберігатися розв’язок. У якості «хромосоми» розглядається розклад. Обрана структура повинна враховувати всі особливості й обмеження, які накладаються на розклад. З хромосом формується початкова популяція;

б) кожна особина популяції оцінюється за допомогою цільової функції, або функції придатності в термінах генетичних алгоритмів; кращі особини залишаються без змін, що дозволяє зберегти кращі рішення і забезпечити підвищену збіжність алгоритму;

в) кращі хромосоми вибираються із сукупності хромосом для подальшої репродукції і створення нового покоління з кращими показниками. Відбір хромосом здійснюється за допомогою цільової функції, яка визначає ступінь придатності хромосоми до формування наступного покоління;

г) відібрані хромосоми попарно схрещують між собою для отримання нової популяції. Якщо отриманий розклад не відповідає навчальному плану,

операціюсхрещування повторюють, поки не буде отриманий коректний розклад, у такому випадку краще передбачити евристичний механізм виправлення розкладу;

д) до популяції застосовують оператор мутації;

е) для отриманої популяції перераховується цільова функція.

Алгоритм роботи генетичного алгоритму представлений на рисунку 3.3.

```
create_start_population(outpopulation)
evaluate_fitness_function(population, outfitness_function[])
whileiterationδmax_iterationorfitness_functionδmax_fitnessdo
     select_individualsforcrossover(population, outindividuals)
     crossover(outindividuals)
     mutation(outpopulation)
     evaluate_fitness_function(population, outfitness_function[])
     create_next_population(fitness_function[], outpopulation)
endwhile
output_best_individuals(population)
```

Рисунок 3.3 – Алгоритм роботи генетичного алгоритму

Використання генетичних алгоритмів для складання розкладу представлено у [10], [7] [11],[24].

Генетичний алгоритм стійкий до локальних мінімумів, не вимагає інформації про поверхню відгуку, а також забезпечує відносно швидкий пошук оптимального розв’язку завдяки внутрішньому паралелізму.

Але застосування еволюційного методу для розв’язання задачі складання розкладу викликає проблему формування хромосоми.

Тому доцільно використовувати комбінацію розглянутих підходів – алгоритм мурашиної колонії для ітераційного наближення до оптимуму і початкового формування хромосоми, і генетичний алгоритм, який буде запускатися кожного разу після чергової ітерації методу мурашиних колоній, – для пошуку глобального оптимуму.

3.2 Гібридний алгоритм складання автоматичного розкладу в університеті

Як було показано в розділі 3.1, ефективним методом складання автоматичного розкладу в університеті є метод мурашиних колоній. Разом з тим даний метод має такі недоліки, як невизначеність часу збіжності алгоритму та сильна залежність ефективності роботи методу від початкових параметрів пошуку, які необхіднопідбирати експериментально. Для усунення вказаних недоліків пропонується використання гібридного алгоритму на основі методу мурашиних колоній, генетичного алгоритму та методу деформованого багатогранника. Використання генетичного алгоритму дозволить зменшити час роботи алгоритму та збільшити ймовірність попадання в глобальний оптимум. Метод деформованого багатогранника використовується для знаходження параметрів методу мурашиних колоній.

3.2.1 Параметри алгоритму

Початковими параметрами алгоритму, від яких залежить швидкість та якість розв’язку, є параметри впливу α, β, та привабливість вершини $\eta_{i,j}^{\beta}$. Розглянемо формування цих параметрів.

Параметри впливу α та β зазвичай підбираються експериментально. Використаємо для визначення цих параметрів метод деформованого багатогранника.

Метод деформованого багатогранника (метод Нелдера – Міда) – ефективний метод безумовної оптимізації (пошуку мінімуму) функції від декількох змінних, що дозволяє оптимізувати функції без використання градієнтів, і тому застосовний до оптимізації негладких функцій. Метод надійний і, як правило, забезпечує ефективний розв’язок, незважаючи на відсутність теорії

збіжності.

Алгоритм полягає в формуванні симплекса (simplex) і подальшому його деформуванні в напрямку точки екстремуму, за допомогою трьох операцій: відображення (reflection); розтягування (expansion); стиснення (contract).

Схема роботи методу деформованого багатогранника представлена на рисунку 3.4.

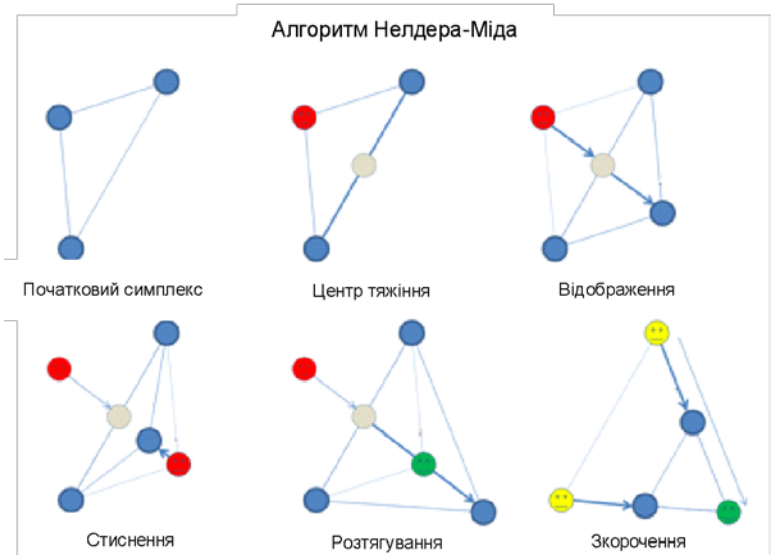


Рисунок 3.4 – Ілюстрація методу деформованого багатогранника

Для формування симплексу мурашину колонію розділимо на три групи, для кожної з яких задамо окремі параметри методу:

– α_1=random(0.5, 1), β_1= random(0, 0.5) – переважає вплив $\tau_{i,j}^{\alpha}$;

– α_2=0,5, β_2=0,5 – рівномірний вплив $\tau_{i,j}^{\alpha}$ та $\eta_{i,j}^{\beta}$;

– α_3=random(0, 0.5),β_3= random(0.5, 1) – переважає вплив $\eta_{i,j}^{\beta}$.

Якщо протягом *iteration_max* ітерацій не досягнутий необхідний розв'язок, необхідно змінити параметри методу мурашиних колоній, обчисливши нові методом деформованого багатогранника.

При цьому в якості початкового симплексу вибирається поточне значення параметрів v_1=(α_1, β_1), v_2= (α_2, β_2), v_3= (α_3, β_3).

Оскільки при використанні метода Нелдера – Міда необхідно обчислити

значення цільової функції в кожній точці $f_1(\alpha_1, \beta_1)$, $f_2(\alpha_2, \beta_2)$, $f_3(\alpha_3, \beta_3)$, рівняння якої не існує, за дане значення приймається максимум локального феромонного сліду протягом *i* ітерацій в кожній групі мурах.

Сортуючи точки за значеннями функції в цих точках, отримуємо подвійну нерівність:

$$f(worst) \le f(good) \le f(best). \tag{3.4}$$

Застосовуємо операцію відображення – проекцію точки $worst$ через центр тяжіння:

$$new = mid + a(mid - worst), \tag{3.5}$$

$$mid = \left(\frac{\alpha_{good} + \alpha_{best}}{2}; \frac{\beta_{good} + \beta_{best}}{2} \right), \alpha=1.$$

Отримуємо 3 набори параметрів, які вибираємо в якості поточного симплексу v_1=*best*, v_2= *good*, v_3= *new*.

Алгоритм повторюється до отримання бажаного розкладу або протягом *iteration_max* ітерацій з новим набором параметрів.

Якщо протягом *iteration_max* ітерацій не досягнутий необхідний розв'язок, необхідно змінити параметри методу мурашиних колоній, обчисливши нові, аналогічно попередньому пункту, але замість операції відображення використати операцію розтягнення за формулою:

$$new = mid + \gamma(worst - min), \tag{3.6}$$

де γ=2.

Алгоритм повторюється до отримання бажаного розкладу або протягом *iteration_max* ітерацій з новим набором параметрів.

Якщо протягом *iteration_max* ітерацій не досягнутий необхідний розв'язок, необхідно змінити параметри методу мурашиних колоній, обчисливши нові, аналогічно попередньому пункту, але замість операції розтягнення використати операцію стиснення за формулою:

$$new = mid + \beta(worst - min), \tag{3.7}$$

де β=0,5.

Привабливість вершини $\eta_{i,j}^{\beta}$.

$\eta_{i,j}^{\beta}$ – евристична інформація про те, наскільки гарним «виглядає» розміщення заняття *i* на місце *j* в розкладі. Цей параметр обчислюється евристично за одним з правил:

а) за загальним правилом:

$\eta_{i,j} = \frac{1}{s\_max}$ (рівномірний розподіл, коли відсутня інформація о перевагах);

б) за правилом суб'єктивних переваг або нечіткої міри.

За допомогою нечіткої міри зручність розміщення заняття в день *d* на парі *p*визначається як суб'єктивна ймовірнісна міра (див. рис. 3.5).

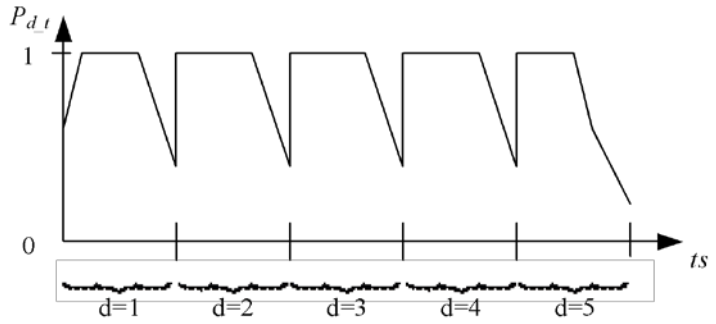


Рисунок 3.5 – Суб'єктивна ймовірнісна міра

Небажаним є проведення лекції на першій парі в понеділок, щоденно на

п’ятих-шостих парах та на останніх парах в останній день тижня, коли знижується працездатність та спостерігається найменша кількість студентів. Наприклад, p_{d_t}=0 при d=6 при 5-денному робочому тижні, p_{d_t}=0.2 при p=6 (небажане проведення заняття на останній парі).

При використанні суб’єктивних переваг визначається думка щодо бажаного часу навчання кожного учасника навчального процесу, яка за допомогою методу аналізу ієрархій і попарних порівнянь перетворюється на кількісну ймовірнісну міру для кожного таймслота.

Зручність розміщення заняття в *r*-й аудиторії μ_rвизначається відповідністю типу аудиторії типу заняття, ємнісними показниками аудиторії, належністю аудиторії до даної кафедри, наявністю необхідного лабораторного обладнання або програмно-апаратного забезпечення, а також суб’єктивними перевагами викладача. Найпростіше визначення: μ_r = 1 при повній відповідності аудиторії, μ_r = 0 при повній невідповідності аудиторії, μ_r = 0,5 при небажаному розміщенні заняття в даній аудиторії.

За μ_{d_t} та μ_r визначають параметр $\eta_{i,j}$:

$$\eta_{i,j} = max(\mu_{d\_t} \cdot \mu_r), \tag{3.8}$$

де $0 \le \mu_{d\_t} \le 1$ – зручність розміщення заняття в день *d* на парі *p*;

$0 \le \mu_r \le 1$ – зручність розміщення заняття в *r*-й аудиторії.

Таке представлення дозволяє врахувати побажання учасників навчального процесу при складанні розкладу. При цьому при $\beta>\alpha$ в алгоритмі послідовно на ітерації i = 1,...,*N* вибирається ще незайняте місце з найбільшим значенням $\eta_{i,j}$.

Феромонний слід.

У класичному алгоритмі мурашиних колоній, реалізованому, наприклад, для задачі комівояжера, феромон наноситься на дуги графа, оскільки дуга являє собою маршрут переходу від точки до точки.

У даному випадку основним джерелом інформації в графі виступають

вершини (комбінація таймслот-аудиторія), тому пропонується заносити феромон на вершини, а не на дуги графа.

3.2.2 Етапи гібридного метода

Розглянемо основні етапи гібридного метода.

Введемо такі змінні: *s_max* – загальна кількість занять згідно навчальних планів, *d_max*– кількість навчальних днів протягом тижня, *p_max* – максимальнакількість пар в день, $t\_max = d\_max \times p\_max$ – кількість пар (таймслотів) протягом тижня, *r_max* – розмірність аудиторного фонду, *S* – масив нерозміщених занять.

Етап 1. Ініціалізація методу.

а) створення масиву занять *S* розмірністю *s_max*та його сортування за зменшенням свободи розташування. У першу чергу розміщуються заняття, які мають обмежені можливості щодо часу проведення занять або для найбільших потоків, які потребують великі аудиторії;

б) для реалізації методу мурашиних колоній створення матриці феромонів $Ph\ \{s_i, t_j, r_k\}$ розмірністю $s\_max \times t\_max \times r\_max$ та її ініціалізація початковим значенням феромону на шляхах до початку моделювання,*Ph_start*.

в) створення колонії штучних мурах (агентів) *ants*[]. Для кожної мурахи визначається: *candidate_list* $\{r_i, d_j, p_k\}$ розмірністю $r\_max \times d\_max \times p\_max$, шлях мурахи*ant_tour* розмірністю *s_max*, які ініціалізуються нульовими значеннями.

г) розділення колонії мурах на три групи для формування симплекса методом Нелдера-Міда.

Для кожної групи мурах задаються окремі параметри методу:

$$\alpha_1 = \text{random}(0.5, 1),\ \beta_1 = \text{random}(0, 0.5);\ \alpha_2 = 0{,}5,\ \beta_2 = 0{,}5;$$

$$\alpha_3 = \text{random}(0, 0.5),\ \beta_3 = \text{random}(0.5, 1).$$

Етап 2. Ітераційний рух агентів

а) установка початкових значень шляху, ініціалізація нульовими значеннями *candidate_list*{r_i, d_j, p_k} та *ant_tour*.

б) для кожного заняття *i* послідовно вибирається місце *j* для постановки в розклад, використовуючи ймовірності переходу. Мураха рухатиметься від вузла до вузла з ймовірністю:

$$p_{ij} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\Sigma(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}, j \in \Omega, \tag{3.9}$$

де Ω – множина занять, ще не розміщених в розкладі.

Правило, за яким на позицію *i* вибирається місце *j* визначається таким чином:

$$j = \begin{cases} \arg\max_{h \in \Omega}(\tau_{i,h}^{\alpha})(\eta_{i,h}^{\beta}), q < q_0 \\ \text{j визначається випадковим чином згідно з розподілом ймовірностей}, q \geq q_0 \end{cases} \tag{3.10}$$

де $0 \leq q_0 \leq 1$ – параметр алгоритму ($q_0 = 0.5$), а значення q обчислюється випадковим чином на кожному кроці.

Перехід згідно розподілу ймовірностей здійснюється методом рулетки (roulette-wheelselection). Даний алгоритм функціонує наступним чином: на кожній ітерації кожній вершині ставиться у відповідність сектор колеса рулетки, величина якого встановлюється пропорційно значенню ймовірності переходу p_{ij} , усі сектори розміщуються лінійно на відрізку [0,1] з накопиченням ймовірностей, генерується випадкове число r ∈[0,1] згідно нормального розподілу. Сектор, який відповідає даному числу, буде визначати вершину для переходу (див. рис. 3.6 – 3.7).

Результат вибору фіксується у шляху мурахи (матриці *ant_tour*) та матриці *candidate_list*.

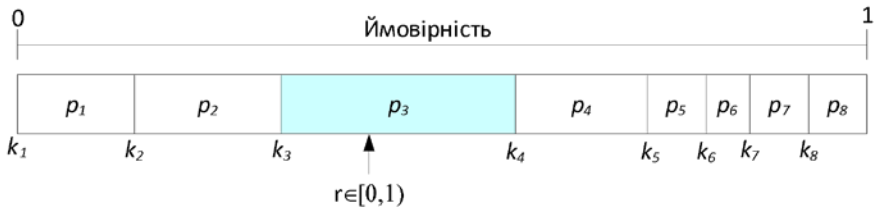


Рисунок 3.6 – Визначення вершини для переходу мурахи методом рулетки

```
place_probabilities_in_line_with_accumulation(k[])
generate_random_number(r)
i=0
while k(i)<r do
     increment(i)
endwhile
ouput(i)
```

Рисунок 3.7 – Алгоритм методу рулетки

Агент переміщується лише тими вузлами, які ще не були відвідані і для яких *candidate_list* $\{r_i, d_j, p_k\}$=0. У даному випадку матриця *candidate_list* відіграє роль списку табу і забезпечує виконання жорстких обмежень «відсутність накладок аудиторного фонду».

Крок б) повторюється доти, доки кожен агент не завершить шлях (див. рис. 3.8).

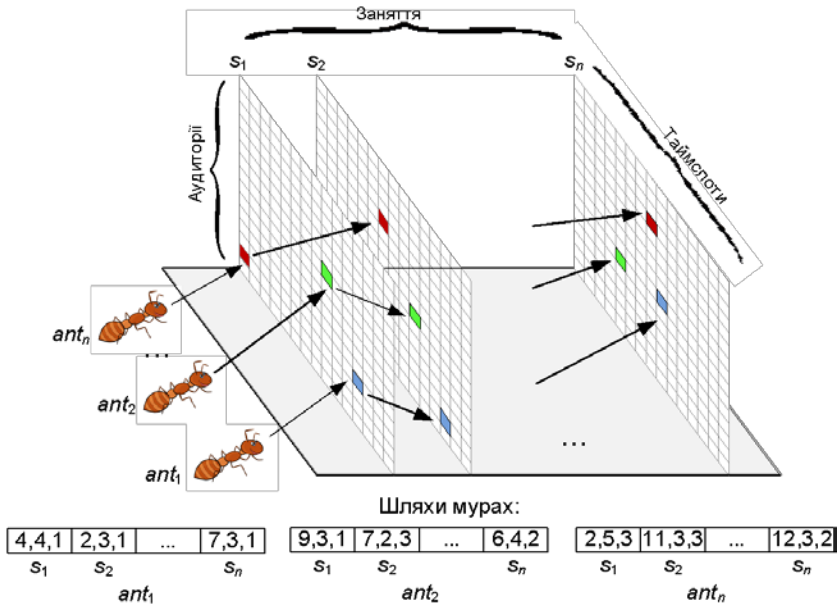


Рисунок 3.8 – Схематичне зображення шляху мурахи

в) розрахунок цільової (штрафної функції) F за формулою:

$$F = \sum_j \omega_j F_i, \tag{3.11}$$

де F – якість складеного розкладу (кількість порушень м'яких обмежень);

F_j – кількість порушень м'яких обмежень за j-им критерієм оцінки якості;

ω_j – ваговий коефіцієнт j-го критерію оцінки якості.

г) застосування методу локального пошуку та перерахунок цільової (штрафної функції) F.

Методи локального пошуку:

Варіант 1:

– у розкладі вибирається довільне заняття i, а також всі заняття j, що проходять одночасно з ним;

– у розкладі вибирається довільне заняття $S_0(r_0, d_0, t_0)$;

– розглядаються усі заняття $S_1(r_1, d_1, t_1)$ … $S_n(r_n, d_n, t_n)$ у межах даного таймслота (d_0, t_0).

– заняття i та j по черзі міняються місцями, при цьому перераховується цільова функція;

– фіксується заміна, що дає найкраще значення цільової функції, інші заняття повертаються на місце.

Така перестановка покращить розклад за критеріями «ефективне використання аудиторного фонду» та «рівномірне завантаження аудиторного фонду». Оскільки час заняття не змінюється, жорсткі вимоги при цьому не порушуються.

Варіант 2:

– у розкладі вибирається довільне заняття $S_i(r_i, d_i, t_i)$;

– заняття i по черзі переноситься на всі вільні позиції, при цьому перераховується цільова функція;

– фіксується заміна, що дає найкраще значення цільової функції, інші

заняття повертаються на місце.

Така перестановка покращить розклад за критеріями «компактність навантаження студентів та викладачів» та «рівномірне завантаження учасників навчального процесу».

Дані варіанти локального пошуку можна розглядати як генетичний алгоритм, який застосовується до однієї особини, варіант 1 – операція селекції, варіант 2 – операція мутації.

д) визначається локальний слід – кількість феромону, залишена на кожній вершині шляху *i*-ї мурахи, за формулою:

$$\Delta\tau_{i,j} = \frac{1}{1+F_i}, \tag{3.12}$$

де $\Delta\tau_i$ – кількість феромону, що залишила *i*-а мураха;

F_i – сума штрафів за порушення м'яких у обмежень у розкладі, складеним *i*-тою мурахою.

е) глобальний слід у матриці феромонів коригується за формулою:

$$\tau_i = (1-\rho)\tau_i + \Delta\tau_{i,} \tag{3.13}$$

якщо в кращому знайденому розкладі на позиції *j* знаходиться заняття *i*. В іншому випадку

$$\tau_i = (1-\rho)\tau_i, \tag{3.14}$$

де ρ – швидкість випаровування феромона, рівномірно розподілена в діапазоні [0,1].

Для кожної групи мурах розраховується та запам'ятовується найбільше значення локального феромонного сліду.

На 1-й ітерації у кожної вершини є можливість бути обраною. Щоб

поступово видалити вершини, які відповідають найгіршому розташуванню занять, до всіх вершин застосовується процедура випару феромону.

Якщо $|ant\_tour^{best} - ant\_tour^{worst}| < \varepsilon$, матрицю феромонів необхідно перезавантажити.

Розклади, отримані методом мурашиної колонії та покращені за допомогою локального пошуку, передаються до генетичного алгоритму в якості хромосом.

Етап 3. Генетичний алгоритм

а) на основі фітнес-функцій вибираються кращі хромосоми в кожній групі мурах для подальшої репродукції і створення нового покоління з кращими показниками;

б) відібрані хромосоми попарно схрещують між собою для отримання нової популяції, обмінюючись аудиторіями або таймслотами. Гени для схрещування (заняття) отримують методом рулетки. Отриману хромосому перевіряють на допустимість – виконання жорстких обмежень. Якщо отриманий розклад не відповідає навчальному плану або схрещування погіршило показники використання аудиторій, операцію схрещування повторюють, доки не буде отриманий коректний розклад;

в) до популяції застосовують оператор мутації – випадкову заміну аудиторії або тайслота.

г) для отриманої популяції перераховується цільова функція (фітнес-функція), яка фіксує виконання м’яких обмежень, і локальний феромонний слід.

Етапи 2 – 3 виконують до отримання бажаного розкладу або протягом *iteration_max* ітерацій.

Роботу алгоритму припиняють після досягнення бажаного результату або виконання *iteration_max* ітерацій, а також якщо протягом *i* ітерацій алгоритм не дає суттєвого поліпшення $(F_{global\_best} - F_i < \varepsilon)$.

Даний алгоритм можна схематично зобразити наступним чином (див.рис.3.9):

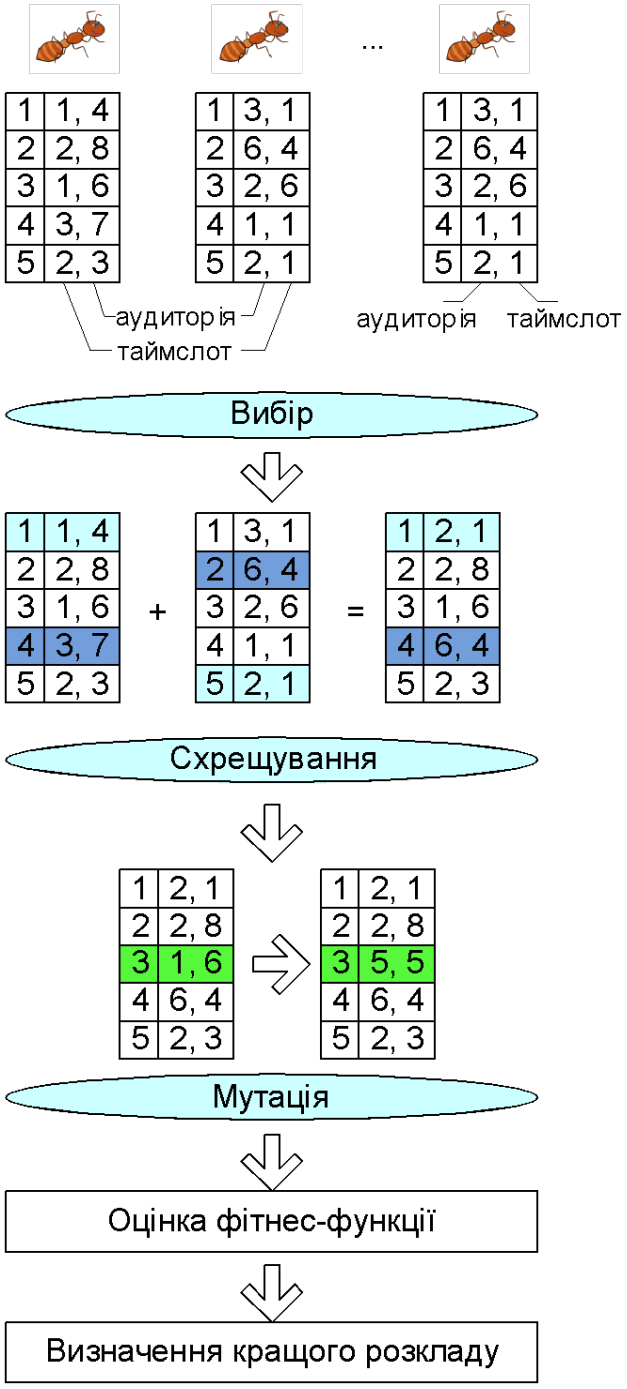


Рисунок 3.9 – Робота генетичного алгоритму

Загальна схема алгоритму представлена на рисунку 3.10.

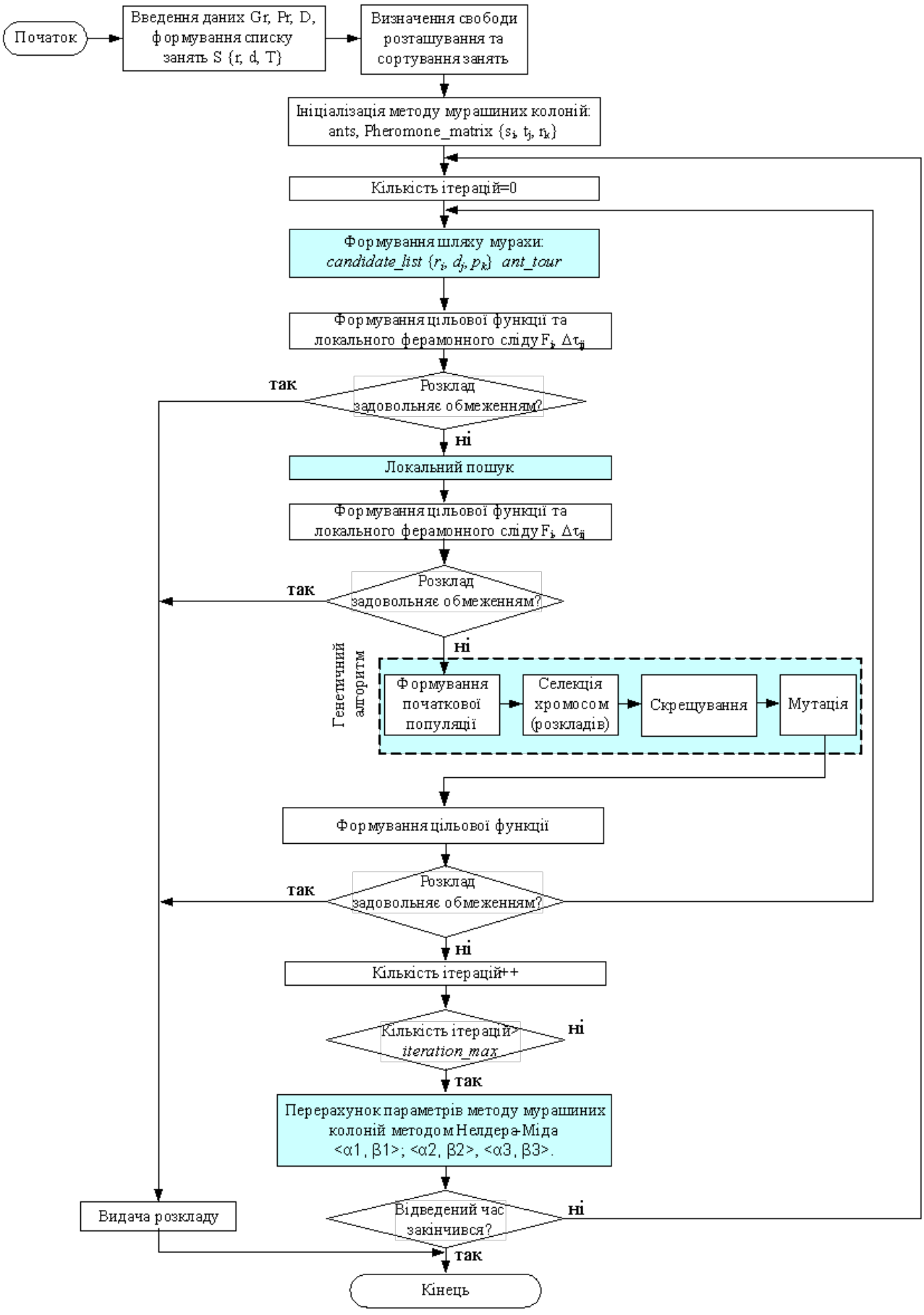


Рисунок 3.10 – Загальна схема роботи алгоритму

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОМАТИЧНОГО ФОРМУВАННЯ КОМП'ЮТЕРНОГО РОЗКЛАДУ

4.1 Вибір середовища програмування

Згідно поставленому технічному завданню на розробку програмного забезпечення необхідно використовувати середовище об'єктно-орієнтованого програмування і операційну систему Windows.

Розглянемо вибір мови програмування для реалізації системи автоматичного формування комп'ютерного розкладу, яка розробляється в атестаційній роботі.

Опишемо множину альтернатив:

– C#;

1. Java.

C# – об'єктно-орієнтована мова програмування, розроблена компанією Microsoft для платформи Microsoft .NET Framework. C# належить до сім'ї мов з C-подібним синтаксисом, але його синтаксис найбільш близький до C++ та Java. Дана мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів, у тому числі операторів явного і неявного приведення типу, делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML.

Узявши багато від своїх попередників – мов C ++, Delphi, Модула, Smalltalk і, особливо, Java, C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C+ не підтримує множинне успадкування класів, але допускає множинне спадкування інтерфейсів.

Java – об'єктно-орієнтована мова програмування, розроблена компанією SunMicrosystems, в подальшому придбаною компанією Oracle. Програми Java зазвичай компілюються в спеціальний байт-код, тому вони можуть працювати незалежно від комп'ютерної архітектури на будь-який віртуальній Java-машині (JVM). Іншою важливою особливістю технології Java є гнучка система безпеки, в

рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

Хоча існують реалізації з відкритим вихідним кодом, C# в основному використовується в розробці для платформ Microsoft – .NET Framework CLR і є найбільш широко використовуваної реалізацією CLI. На противагу Java має величезну екосистему з відкритим вихідним кодом.

Критерії вибору:

– кроссплатформеність;
– швидкодія;
– opensource;
– безпека типів;
– фокус.

Векторний опис альтернатив за критеріями представлений у таблиці 4.1.

Таблиця 4.1 – Векторний опис альтернатив «середовище програмування»

	Кроссплат форменість	Швидкодія	OpenSource	Безпека типів	Фокус
C#	–	+	– (Windows)	+	Веб-розробка та теорія ігор
Java	+	Низька	+	+	Мобільна розробка (Android)

Оскільки даний додаток передбачає роботу з великим обсягом даних (дані про факультети, кафедри, групи, викладачів, аудиторний фонд, навчальні плани, навчальне навантаження, переваги студентів та викладачів, календарі доступності ресурсів тощо), важливий критерій з найвищим пріоритетом, – швидкодія.

Програмне забезпечення, що розробляється, згідно технічного завдання на розробку орієнтоване на операційну систему Windows, тому критерій кросплатформеності має низький пріоритет і відсутність кросплатформеностіне впливає на даний вибір.

Отже, використаємо C#, набір технологій .NET та програмне середовище VisualStudio 2015.

4.2 Вибір системи управління базами даних

Опишемо множину альтернатив:

– Microsoft SQL Server;

– MySQL.

Microsoft SQL Server – система управління базами даних (СУБД), розроблена корпорацією Microsoft. Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства.

MySQL – вільна система керування реляційними базами даних. Розробка та підтримка сайтаMySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненою SunMicrosystems.

MySQL – це компактний багатопотоковий сервер баз даних, який характерризується великою швидкістю, стійкістю і легкістю у використанні. MySQL був розроблений компанією TcX для швидкої обробки дуже великих баз даних. MySQL є ідеальним рішенням для малих і середніх додатків. Найкраще можливості сервера проявляються на Unix-серверах, які забезпечують підтримкубагатопоточності, що дає значний приріст продуктивності.

Критерії вибору:

– платформа;

– підтримувані мови програмування;

– середовище;

– мова SQL;

– OpenSource;

– власник, ліцензія;

– система зберігання даних.

Векторний опис альтернатив за критеріями представлений у таблиці 4.2.

Таблиця 4.2 – Векторний опис альтернатив «СУБД»

	Платформа	Підтримувані мови	Середовище	Мова SQL	OpenSource	Власник, ліцензія	Система зберігання даних
Microsoft SQL Server	Windows	в т.ч. C#	.NET	T-SQL	-	Microsoft, комерційна ліцензія	Єдина система, розробка Microsoft
MySQL	Windows, Linux, Mac OS X	в т.ч. C#	Будь-яке	SQL	+	Вільна, подвійне ліцензування (GPL, власна комерційна ліцензія)	Множина движків

Оскільки в якості середовища розробки використовується Microsoft VisualStudio і система розробляється під ОС Windows, виберемо в якості системи керування базами даних Microsoft SQL Server, оскільки обидва програмних продукти належать одному виробнику. MySQL найбільше використовується у зв’язці з PHP.

4.3 База даних системи

При розробленні програмної системи складання розкладу необхідно створити базу даних, яка буде містити дані про навчальний заклад, необхідні для формування та зберігання розкладу. База даних системи складається з таких таблиць:

− Users – зберігає інформацію про адміністраторів системи (логін та пароль). Пароль зберігається у зашифрованому вигляді з використанням алгоритму шифрування md5;

− таблиці, що зберігають інформацію про структуру навчального закладу: Faculties – факультети/інститути, Departments – кафедри, Specialities – спеціальності;

− таблиці, що зберігають інформацію про навчальні плани: Courses – дисципліни, Curriculum_studies – закріплення дисциплін за викладачами;

− таблиці, що зберігають інформацію про створені розклади: Timetables – загальна інформація про розклад, Timetable_studies – заняття, розміщені в розкладі;

− таблиці, що зберігають інформацію про навчальні корпуси: Buildings – корпуси, Building_distances – відстані між корпусами;

− таблиці, що зберігають інформацію про аудиторії: Rooms – аудиторії, Room_calendar – календарі доступності аудиторій;

− таблиці, що зберігають інформацію про викладачів: Staff – штатне навантаження кафедр, Teachers – викладачі, Teacher_calendar – календарі доступності викладачів;

− таблиці, що зберігають інформацію про навчальні групи: Groups – навчальні групи, Group_calendar – календарі доступності груп;

− таблиця розкладу дзвінків: Timeslot_table.

Структура таблиць бази даних:

Users:

− Id_user – ID користувача;

− login – логін;

− password – пароль;

Faculties:

− Id_fac – ID факультета / інститута;

− name_fac – повна назва факультету;

− name_fac_short – скорочена назва факультету;

Departments:

− Id_dep – ID кафедри;

− name_dep – повна назва кафедри;

− name_dep_short– скорочена назва кафедри;

− id_fac – ID факультета, до якого належить кафедра;

Specialities:

− Id_spec – ID спеціальності;

− name_spec – повна назва спеціальності;

- name_spec_short– скорочена назва спеціальності;
- Id_ dep – ID кафедри, до якої належить спеціальність;

Courses:

- Id_disc – ID заняття в розкладі;
- id_group – ID групи;
- name_disc – назва дисципліни;
- lectures – кількість лекцій;
- labs – кількість лабораторних занять;
- practices – кількість практичних занять;
- consultations – кількість консультацій;

Curriculum_studies:

- Id_study– ID заняття в розкладі;
- study_type – тип заняття (лекція, лабораторне, практичне заняття, консультація);
- subgroup – підгрупа;
- id_teacher – викладач;
- room_type – тип аудиторії (лекційна, лабораторія, комп'ютерний клас тощо);
- id_disc – ID дисципліни;

Timetables:

- Id_timetable – ID розкладу;
- name_timetable – назва розкладу;
- year – рік;
- semestr – семестр;

Timetable_studies:

- Id – ID запису;
- id_study– ID заняття в розкладі;
- id_timetable – ID розкладу;
- id_room – ID аудиторії;
- day – день;

- time – час;

Buildings:

- Id_building – ID заняття в розкладі;
- name_building – повна назва корпусу;
- name_building _short– скорочена назва корпусу;

Building_distances:

- Id – ID запису;
- id_building1 – корпус1;
- id_building2 – корпус2;
- distance – відстань між корпусами;

Rooms:

- Id_room – ID аудиторії;
- n – номер/назва аудиторії;
- id_building – корпус, в якому знаходиться аудиторія;
- id _ dep – ID кафедри, до якої належить аудиторія;
- r_type – тип аудиторії;
- capacity – ємність аудиторії (кількість робочих місць);

Room_calendar:

- Id – ID запису;
- id_room – ID аудиторії;
- week – тиждень;
- day – день;
- time – час;
- nonavailable – інформація про недоступність аудиторії;

Staff:

- Id – ID запису;
- id _ dep – ID кафедри;
- id_teacher – ID викладача;
- position – посада;

Teachers:

- Id_teacher – ID викладача;
- surname – прізвище;
- name – ім’я;
- patronymic – по-батькові;
- degree – науковий ступінь;
- administration – керівник (керівники мають вищий пріоритет);

Teacher _calendar:

- Id – ID запису;
- id_teacher – ID викладача;
- week – тиждень;
- day – день;
- time – час;
- nonavailable – інформація про недоступність викладача;

Groups:

- Id_group – ID групи;
- name – назва/номер групи;
- id_spec – ID спеціальності;
- kurs – курс;
- capacity – ємність групи (кількість студентів).

Group_calendar:

- Id – ID запису;
- id_group – ID групи;
- week – тиждень;
- day – день;
- time – час;
- nonavailable – інформація про недоступність викладача.

Timeslot_table:

- first_start – початок першої пари;
- first_end – закінчення першої пари;
- second_start – початок другої пари;

- second_end – закінчення другої пари;
- third_start – початок третьої пари;
- third_end – закінчення третьої пари;
- forth_start – початок четвертої пари;
- forth_end – закінчення четвертої пари;
- fifth_start – початок п’ятої пари;
- fifth_end – закінчення п’ятої пари;
- sixth_start – початок шостої пари;
- sixth_end – закінчення шостої пари;
- seventh_start – початок сьомої пари;
- seventh_end – закінчення сьомої пари;
- eightth_start – початок восьмої пари;
- eightth_end – закінчення восьмої пари;

Максимально можлива кількість пар – 8.

Таблиці пов’язані між собою зв’язком «один до багатьох».

Схема бази даних представлена на рисунку 4.3.

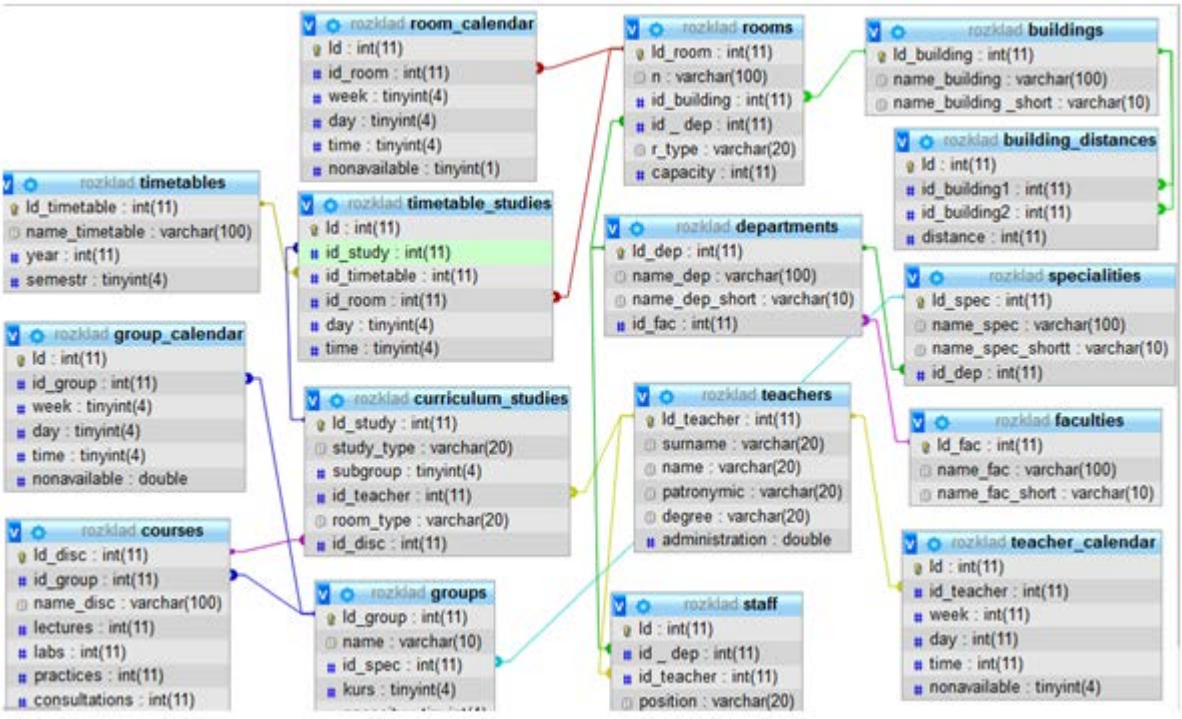


Рисунок 4.3 – Схема бази даних

4.4 Архітектура системи

Система складання розкладу, що розробляється, може використовуватися як окремий додаток або підмодуль модуля планування навчального процесу в автоматизованій інформаційній системі (АІС) ЗВО. На рисунку 4.4 представлене місце підсистеми складання розкладу в загальній структурі АІС ЗВО.

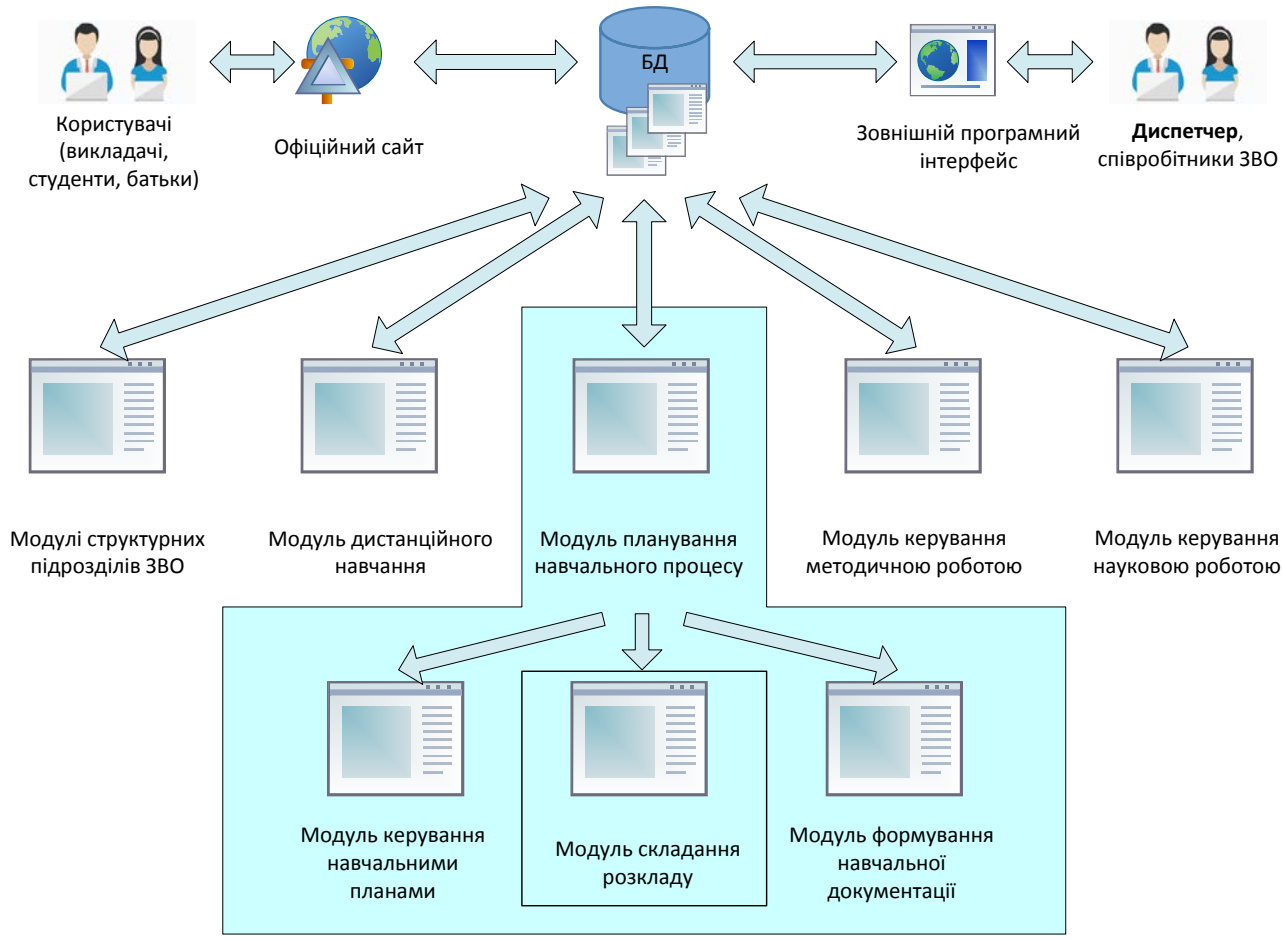


Рисунок 4.4 – Підсистема складання розкладу в загальній структурі автоматизованої інформаційної системи ЗВО

Структурна схема системи «Розклад» наведена на рисунку 4.5.

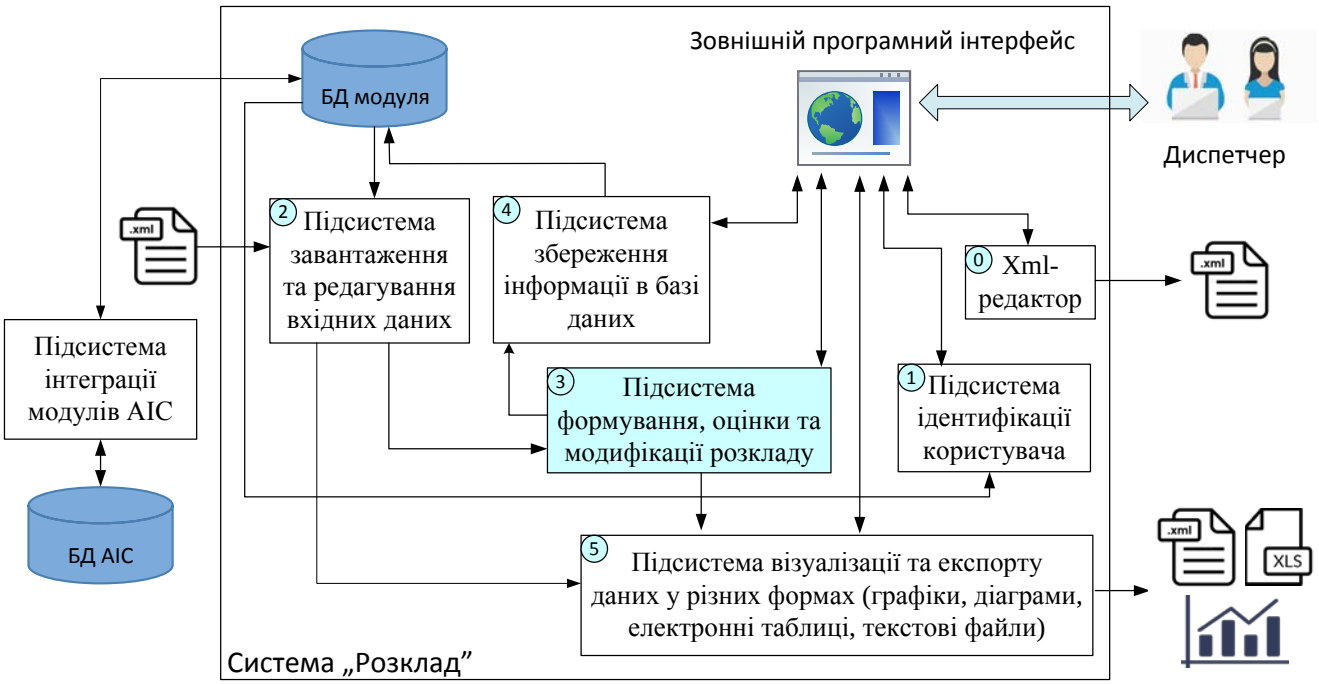


Рисунок 4.5 – Структурна схема системи «Розклад»

Діаграма класів, що описує відношення між класами, що репрезентують навчальні заняття та учасників навчального процесу, представлена на рисунку 4.6.

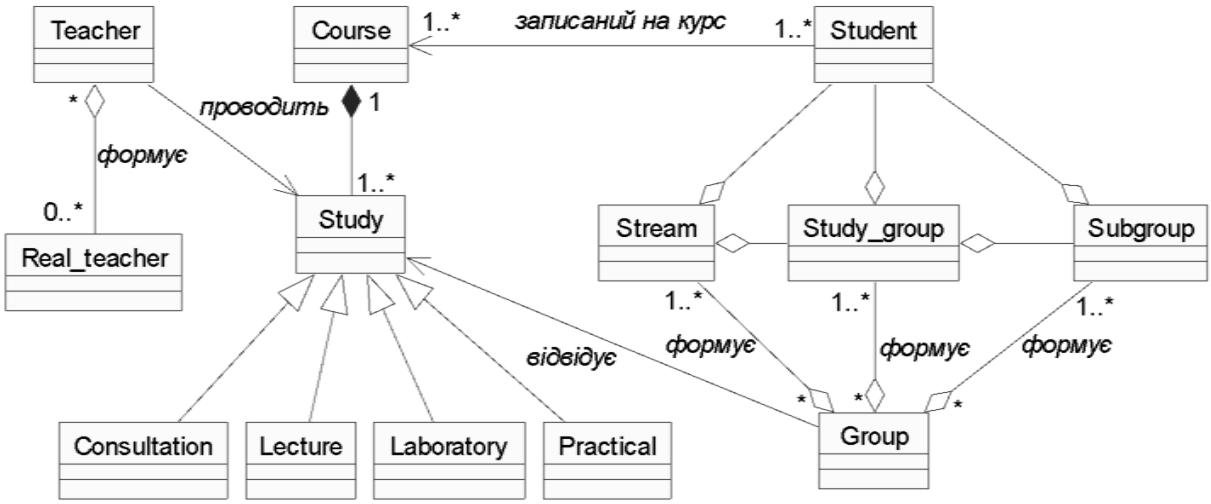


Рисунок 4.6 – Діаграма класів учасників навчального процесу

Загальна діаграма класів системи зображена на рисунку 4.7.

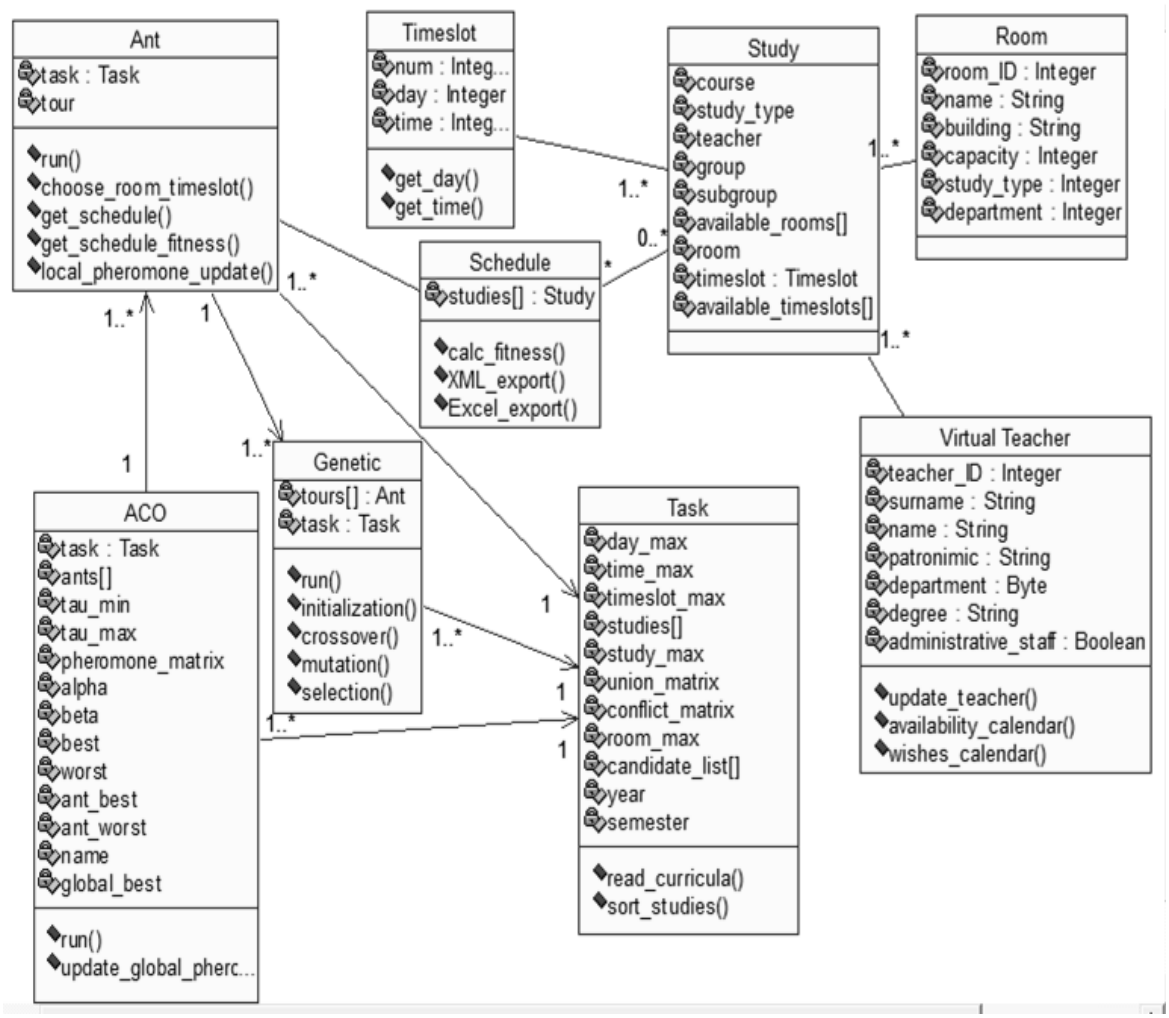


Рисунок 4.7 – Загальна діаграма класів системи

4.5 Загальна схема системи автоматичного формування комп’ютерного розкладу

Робота системи автоматичного формування комп’ютерного розкладу складається з таких етапів, представлених на рисунку 4.8.

Рисунок 4.8 – Загальна схема системи автоматичного формування розкладу

Представимо процес автоматичного формування комп'ютерного розкладу у вигляді IDEF0-діаграми (див. рис. 4.9).

Стандарт IDEF0 описує методику побудови функціональної моделі предметної області. Основна ідея даної методології полягає в поданні підприємства, організації або процесу, що моделюється, у вигляді сукупності взаємозалежних робіт (функцій). Роботи утворюють ієрархічну структуру, коренем якої є основна функція процесу, що моделюється.

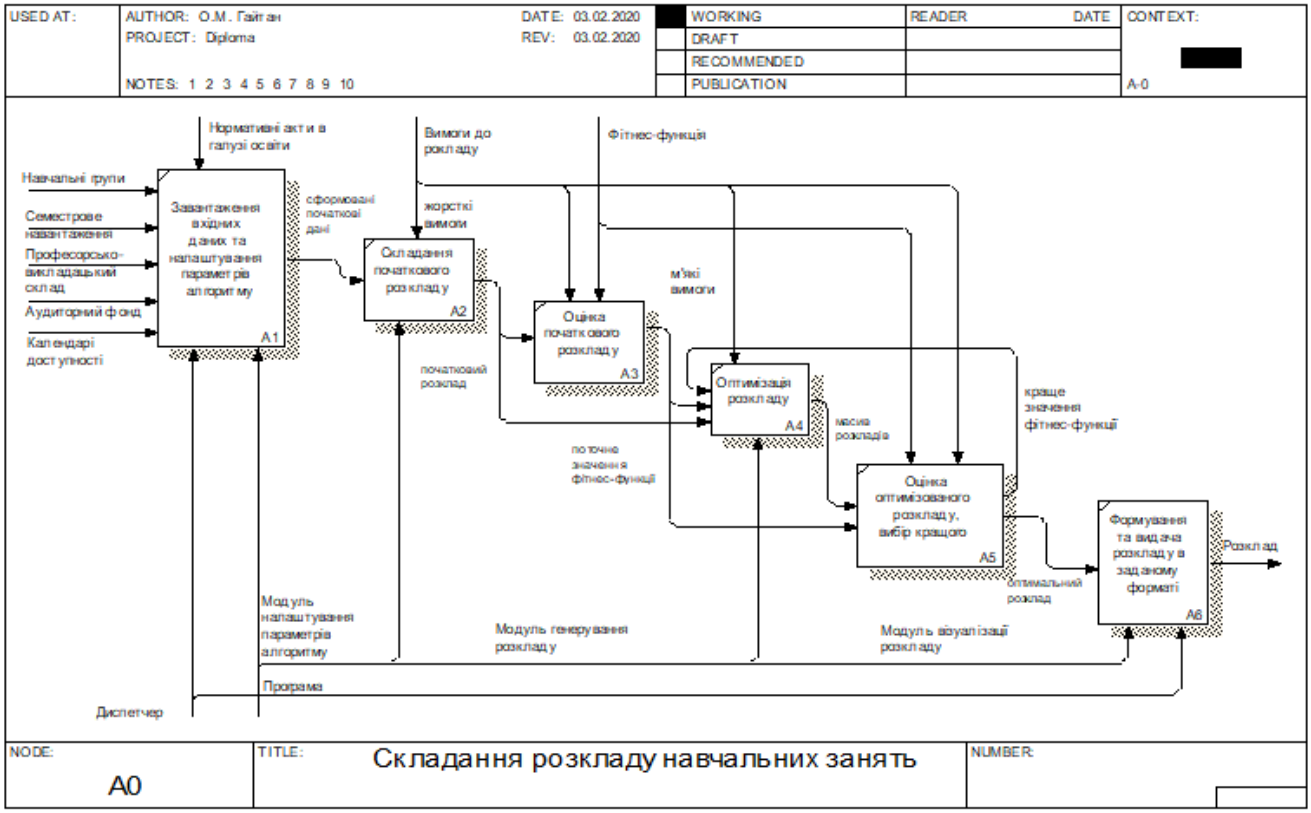


Рисунок 4.9 – IDEF0-діаграма процесу автоматичної генерації комп’ютерного розкладу

Представимо модуль завантаження вхідних даних та налаштування параметрів алгоритму у вигляді діаграми потоків даних.

Діаграми потоків даних (Dataflowdiagram, DFD) використовуються для опису документообігу й обробки інформації. DFD представляє модельну систему як мережу пов’язаних між собою робіт. Її можна використати для наочного відображення операцій в системах обробки інформації.

Використаємо для побудови діаграми потоків нотацію Гейна-Сарсона. Схема системи представлена на рисунку 4.10.

Рисунок 4.10 – Модуль завантаження вхідних даних
та налаштування параметрів алгоритму у DFD-нотації

Перед початком роботи з системою необхідно задати часову сітку розкладу у головному меню (пункт «Розклад», підпункт «Параметри часу»), а також налаштувати урахування критеріїв оптимальності та їхні вагові коефіцієнти. Ваги обмежень за замовчуванням дорівнюють одиниці, але є можливість налаштувати дані ваги, а також урахування окремих критеріїв якості у ручному режимі.

Стартова сторінка зображена на рисунку 4.11.

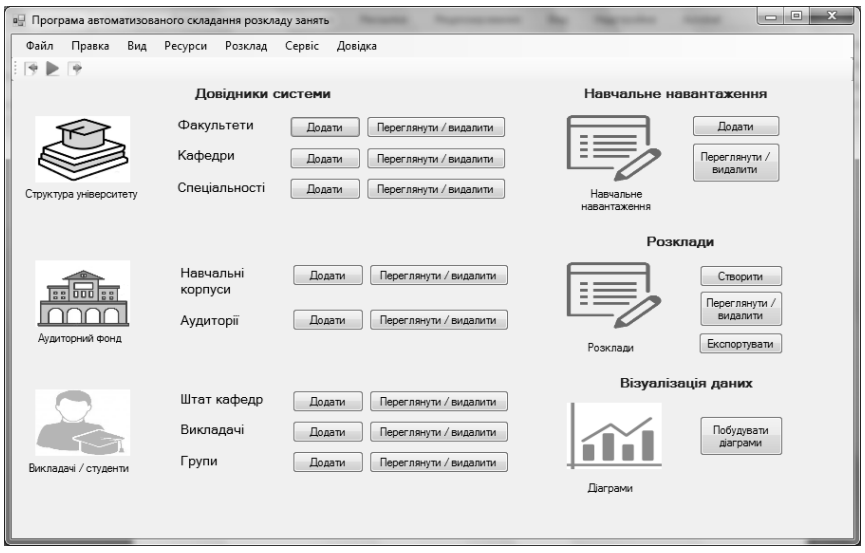


Рисунок 4.11 - Стартова сторінка

Розглянемо кожний етап роботи системи детальніше.

4.5.1 Оновлення постійних довідників

Постійні дані в системі зберігаються в sql-базі даних, доступ до якої реалізується за допомогою sql-запитів (див. схему на рисунку 4.12).

Покажемо роботу з базою даних на прикладі аудиторій.

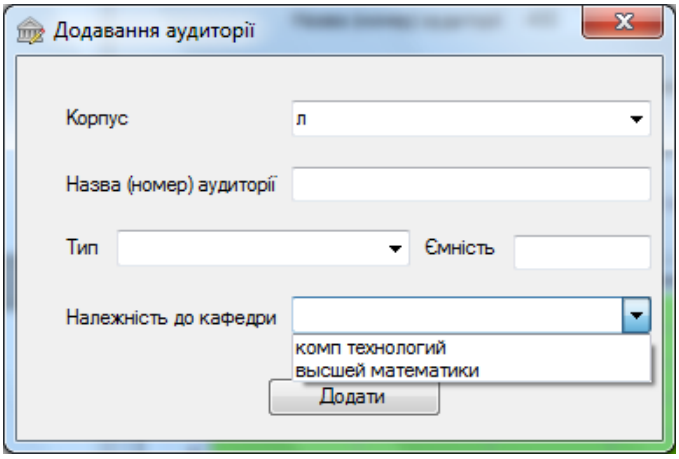


Рисунок 4.12 – Форма додавання аудиторії

Для попередження дублювання даних при спробі повторної реєстрації аудиторії (з однаковою назвою в одному корпусі) система видає відповідне повідомлення (див. рис. 4.13).

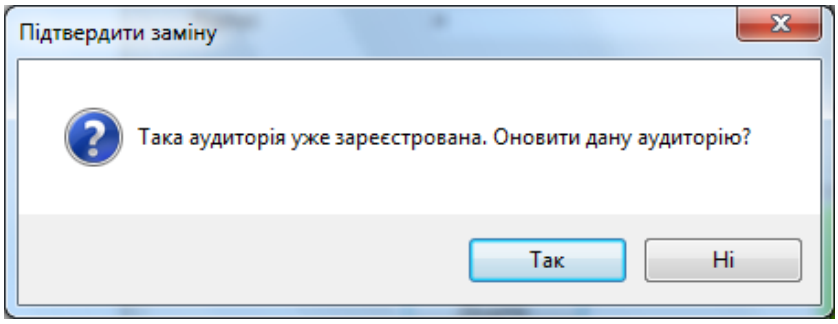


Рисунок 4.13 – Повідомлення при спробі повторної реєстрації аудиторії

При натисканні кнопки «Так» дані про аудиторію автоматично оновлюються, при натисканні «Ні» відбувається повернення до форми редагування даних.

Відображення та оновлення даних про аудиторії до системи.

Дані про аудиторії надаються у вкладці «Довідники», пункт «Аудиторний фонд». Є можливість переглянути та відредагувати аудиторії у всіх навчальних корпусах, а також з фільтрацією за корпусом (див. рис. 4.14).

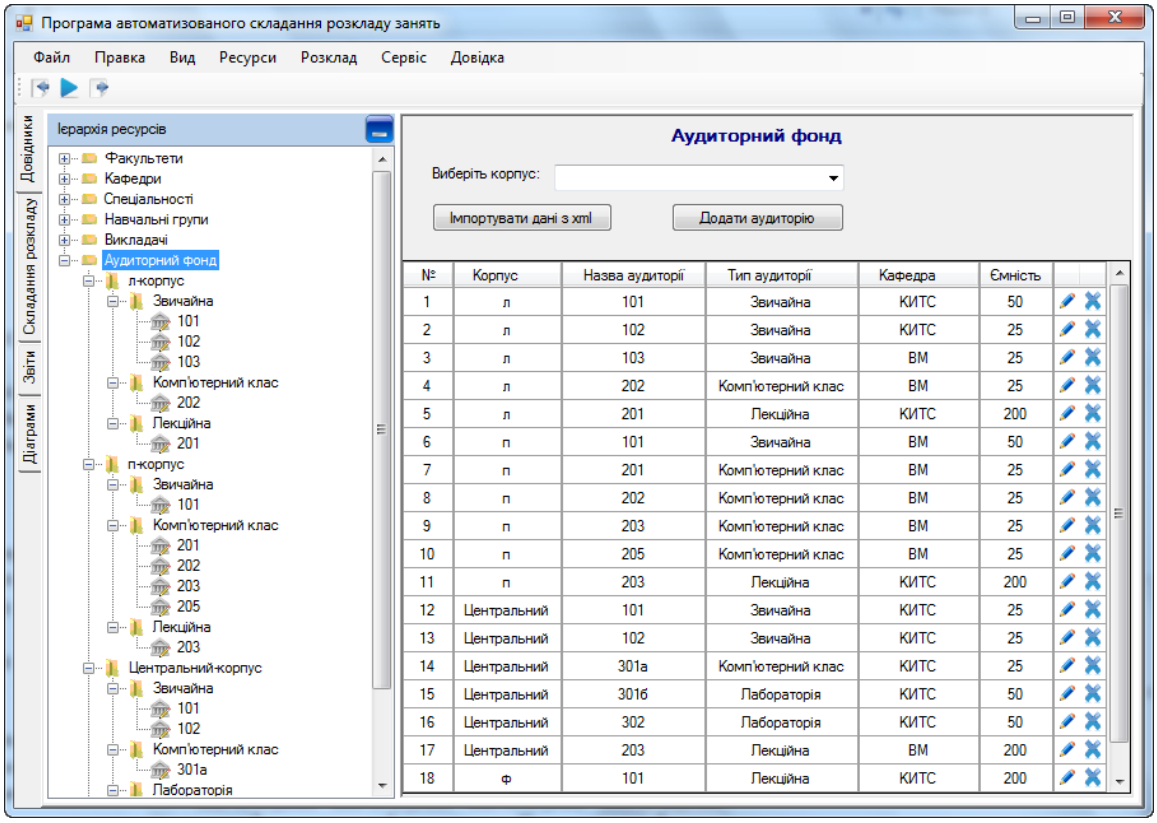


Рисунок 4.14 – Перелік аудиторій, зареєстрованих у системі

Поточна аудиторія вибирається із дерева ресурсів, що знаходиться на лівій вкладці. Користувач має можливість відредагувати текстові дані про аудиторію та задати календар доступності графічним методом (див. рис. 4.15).

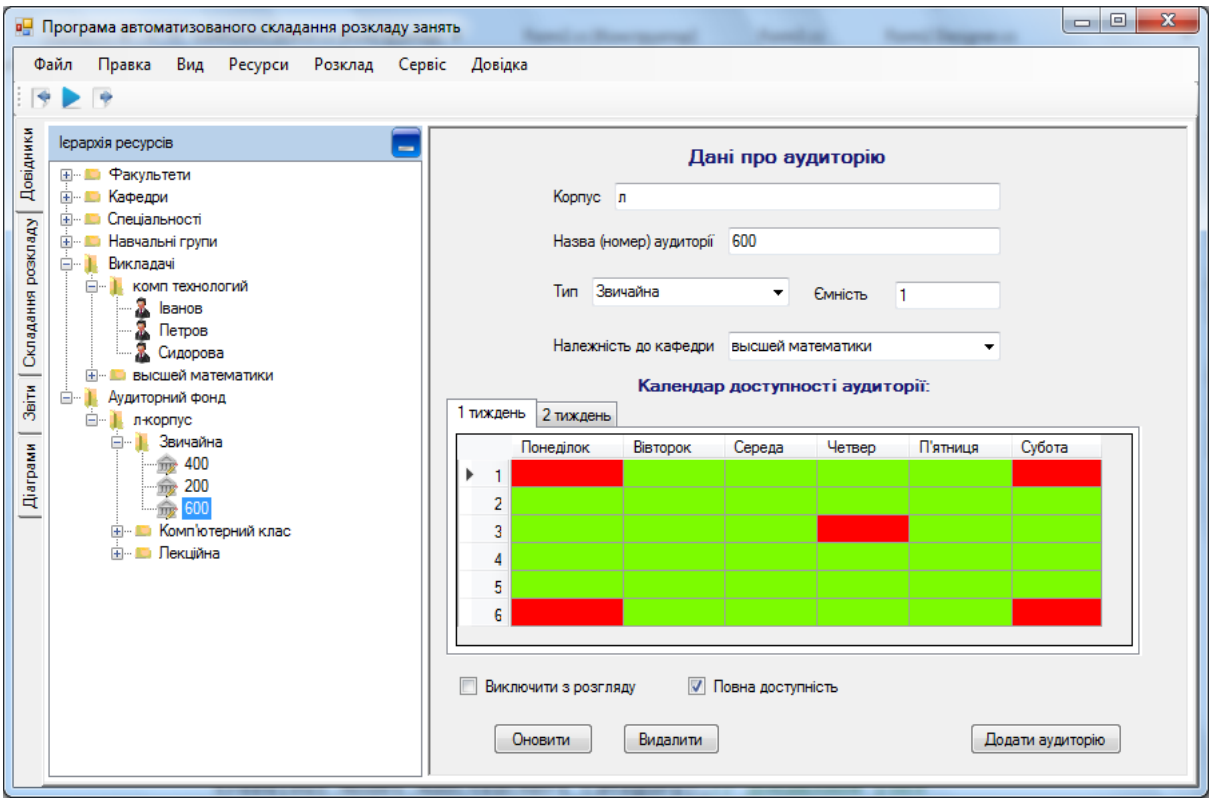


Рисунок 4.15 – Редагування даних про аудиторії

Для попередження випадкового видалення даних при натисканні кнопки «Видалити» система видає відповідне повідомлення – запитання (див. рис. 4.16).

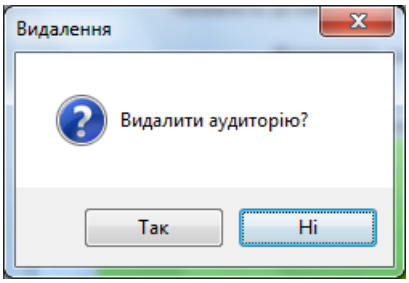


Рисунок 4.16 – Повідомлення при видаленні аудиторії

При натисканні кнопки «Так» дані про аудиторію автоматично видаляються із бази даних, при натисканні «Ні» відбувається повернення до форми редагування даних.

Аналогічно здійснюється робота з базою даних для інших довідників.

4.5.2 Імпорт оперативної семестрової інформації

Дані навчальних планів, про аудиторний фонд тощо зберігаються та завантажуються у вигляді файлів з розширенням XML. XML (англ. Extensible Markup Language) являє собою розширювану мову розмітки, запропонований консорціумом WorldWideWeb (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між додатками, зокрема, через Інтернет. XML-документскладаєтьсяізтекстових знаків і доступний длясприйняття людиною.

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації застосуванням простих тегів. Отже, даний стандарт визначає метамову, на основі якої за допомогою обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних, що робить XML ідеальним варіантом для зберігання оперативної інформації, необхідної для складання розкладу.

XML-дані про навчальне навантаження групи мають таку структуру:

```
<?xmlversion="1.0" encoding="UTF-8"?>
<groups>
<groupid="101-ТН">
<department>КІТС</department>
<speciality>Комп'ютерні науки</speciality>
<kurs>1</kurs>
<schedule>
<coursename="Вища математика">
<studyid="Лекція" teacher="Бутенко І.К." number="4"/>
<studyid="Лаб" subgroup="1" teacher="Бутенко І.К." number="2"/>
<studyid="Лаб" subgroup="2" teacher="Сомов Д.К." number="2"/>
</course>
<coursename="Фізика">
<studyid="Лекція" teacher="Кущ А.Р." number="2"/>
<studyid="Лаб" teacher="Редис Н.Е." number="2"/>
</course>
</schedule>
</group>
</group>
```

```
</groups>
```

XML-дані про аудиторний фонд мають таку структуру:

```
<?xmlversion="1.0" encoding="UTF-8"?>
<rooms>
<roomid="101" building="p">
<department>40</department>
<capacity>200</capacity>
<study_type>lecture</study_type>
</room>
<roomid="102"  capacity="20"  department="40"  study_type="computer"
building="p">
</room>
<roomid="103"  capacity="80"  department="40"  study_type="computer"
building="p">
<!- notavailableonMondaysandTuesdayson 4th lessonallweeks ->
<unavailableweeks="12" days="110000" lesson="4" />
<!- notavailableonFridayson 1th lessonoddweeksonly ->
<unavailableweeks="1" days="000100" lesson="1" />
</room>
</rooms>
```

Якщо користувач не має досвіду роботи з XML-файлами, для заповнення оперативної семестрової інформації передбачена наявність XML-редактора як окремого додатку у складі системи «Розклад» (див. рис. 4.17-4.19).

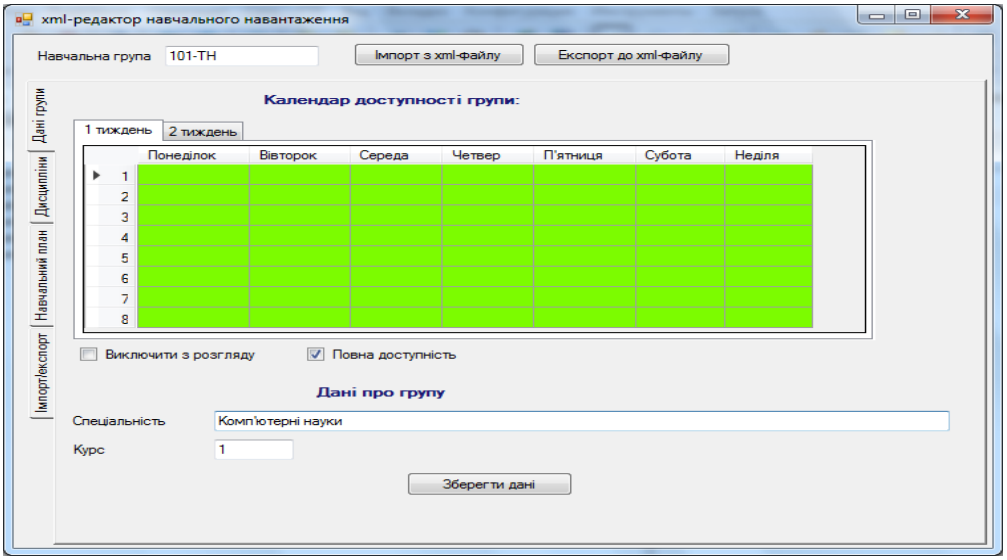


Рисунок 4.17 – Приклад заповнення даних про навчальну групу у XML-редакторі

Даний редактор дозволяє ввести дані у вигляді заповнення форм користувача та експортувати введені дані до XML-формату, а також відредагувати дані, що зберігаються, у XML-файлі за допомогою форм користувача, з наступним експортом до XML-файлу.

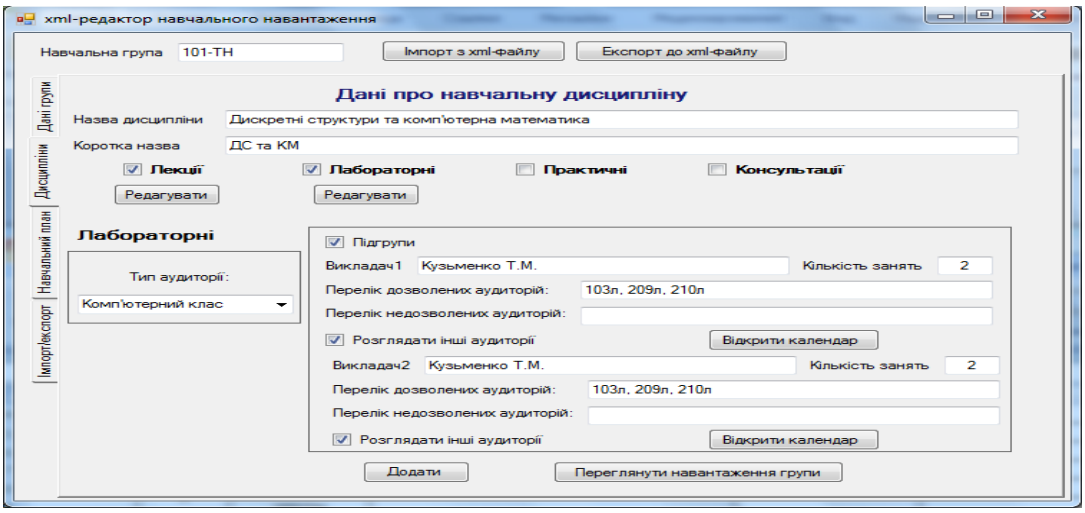


Рисунок 4.18 – Редагування навчального навантаження групи у XML-редакторі

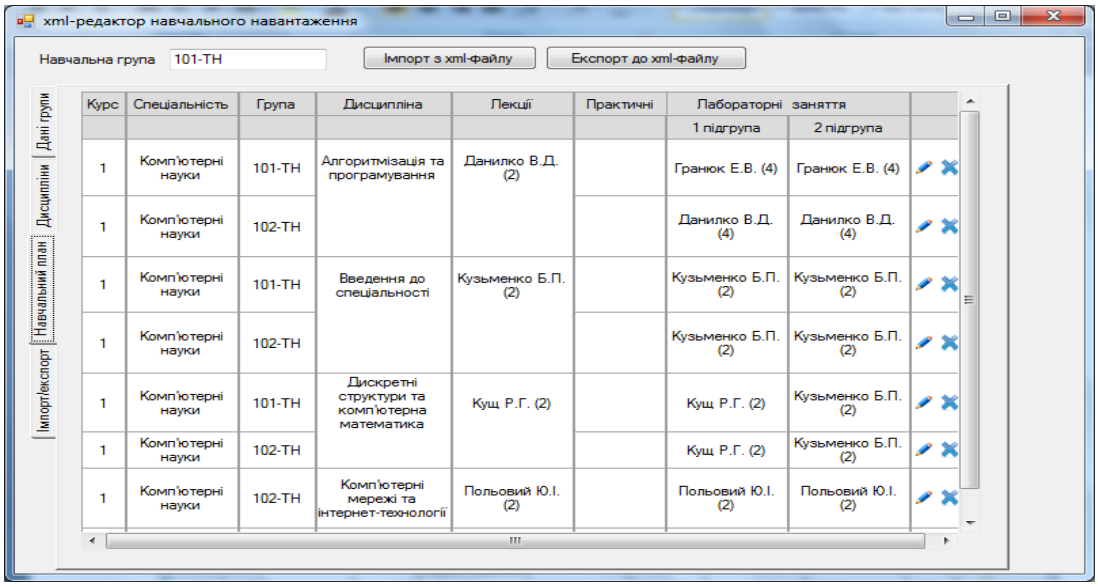


Рисунок 4.19 – Перегляд навчального навантаження групи у XML-редакторі

Перед експортом можна переглянути та відредагувати введені дані у вкладці «Навчальний план». Для зручності система автоматично сортує заняття за спеціальністю та курсом, а також візуально об’єднує лекційні заняття у потоки. Для експорту необхідно натиснути кнопку «Експорт до xml-файлу».

Для інших довідників XML-редактори працюють аналогічно.

4.5.3 Генерація розкладу

Складання розкладу може проходити у трьох режимах: автоматичному, ручному та автоматизованому.

При ручному способі складання розкладу вибирається відповідне заняття із дерева занять та методом Drag&Drop переноситься у сітку розкладу. При цьому система підсвічує доступні та заборонені таймслоти. Розмістити конфліктуючі заняття в один час або одній аудиторії неможливо.

Червоним кольором виділяються заборонені таймслоти, зеленим – такі самі заняття, уже розміщені в сітці розкладу, сірим – не рекомендовані варіанти, блакитним – рекомендовані варіанти. Рекомендовані та не рекомендовані варіанти розміщення визначаються типом занять, календарями завантаження учасників навчального процесу та даними про заняття, що містяться в імпортованих файлах. За замовчуванням усі таймслоти – рекомендовані, рекомендовані аудиторії – аудиторії, тип яких співпадає з типом заняття (див. рис. 4.20).

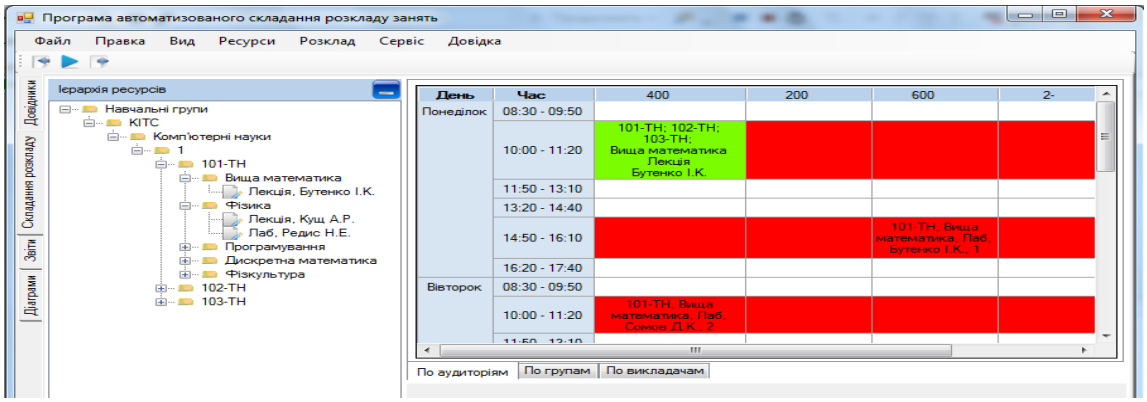
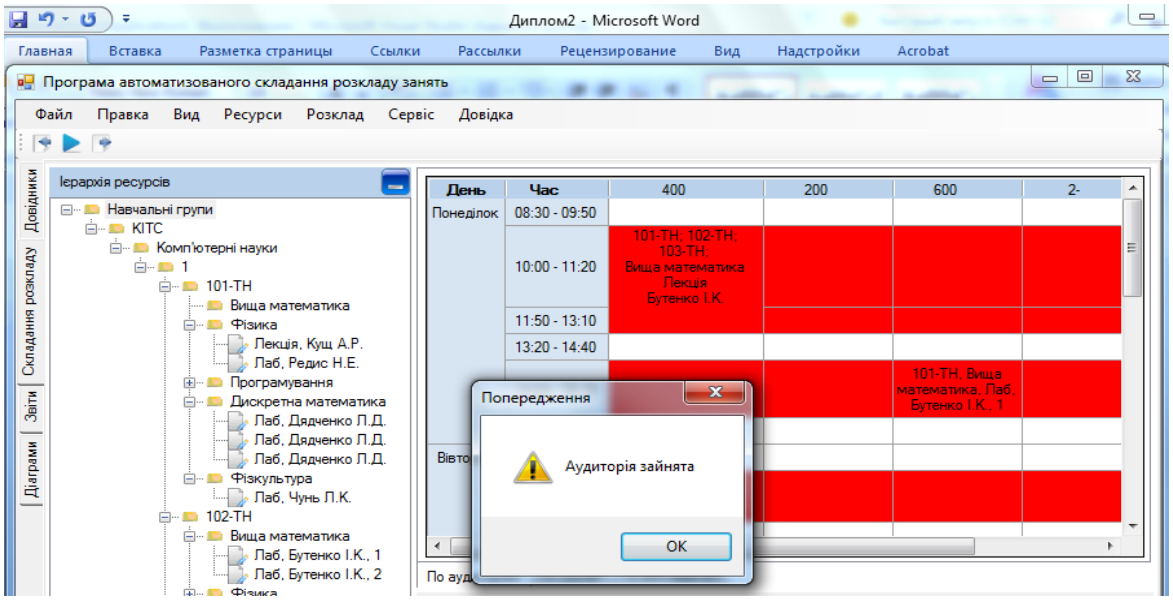


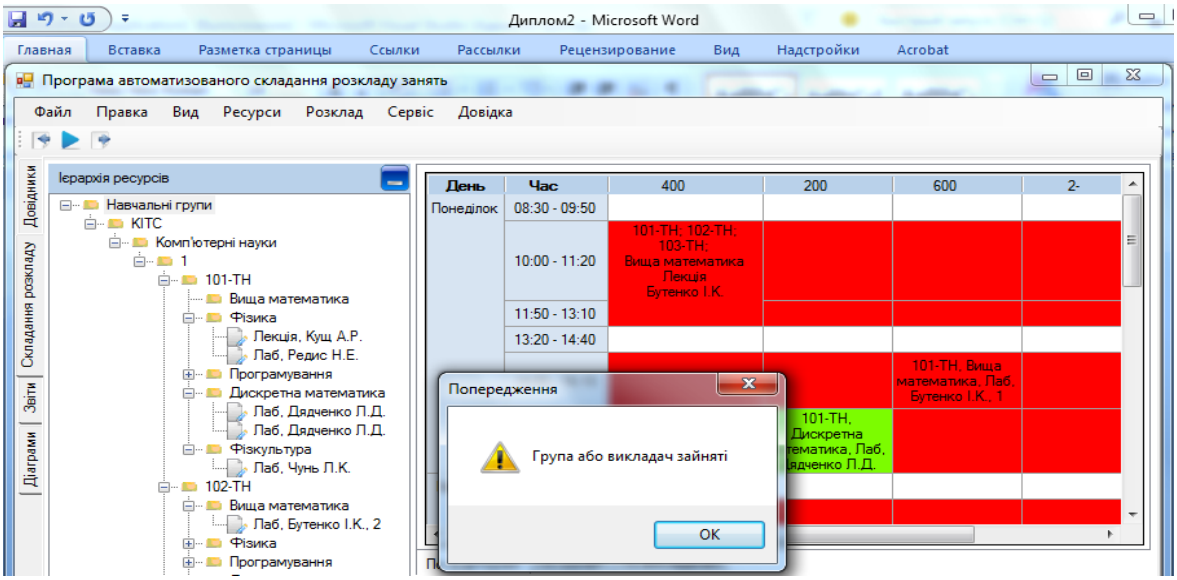
Рисунок 4.20 – Розклад по аудиторіям

Перенесення занять у сітці розкладу методом Drag&Drop відбувається за допомогою трьох обробників подій: методу MouseDown джерела, який запускає

процес переносу, методу DragOver приймача, який контролює можливість використання поточної комірки та методу DragDrop приймача, який завершує процес переносу при успішній перевірці можливості використання поточної комірки або видає відповідне повідомлення в іншому випадку (див. рис. 4.21).



а)

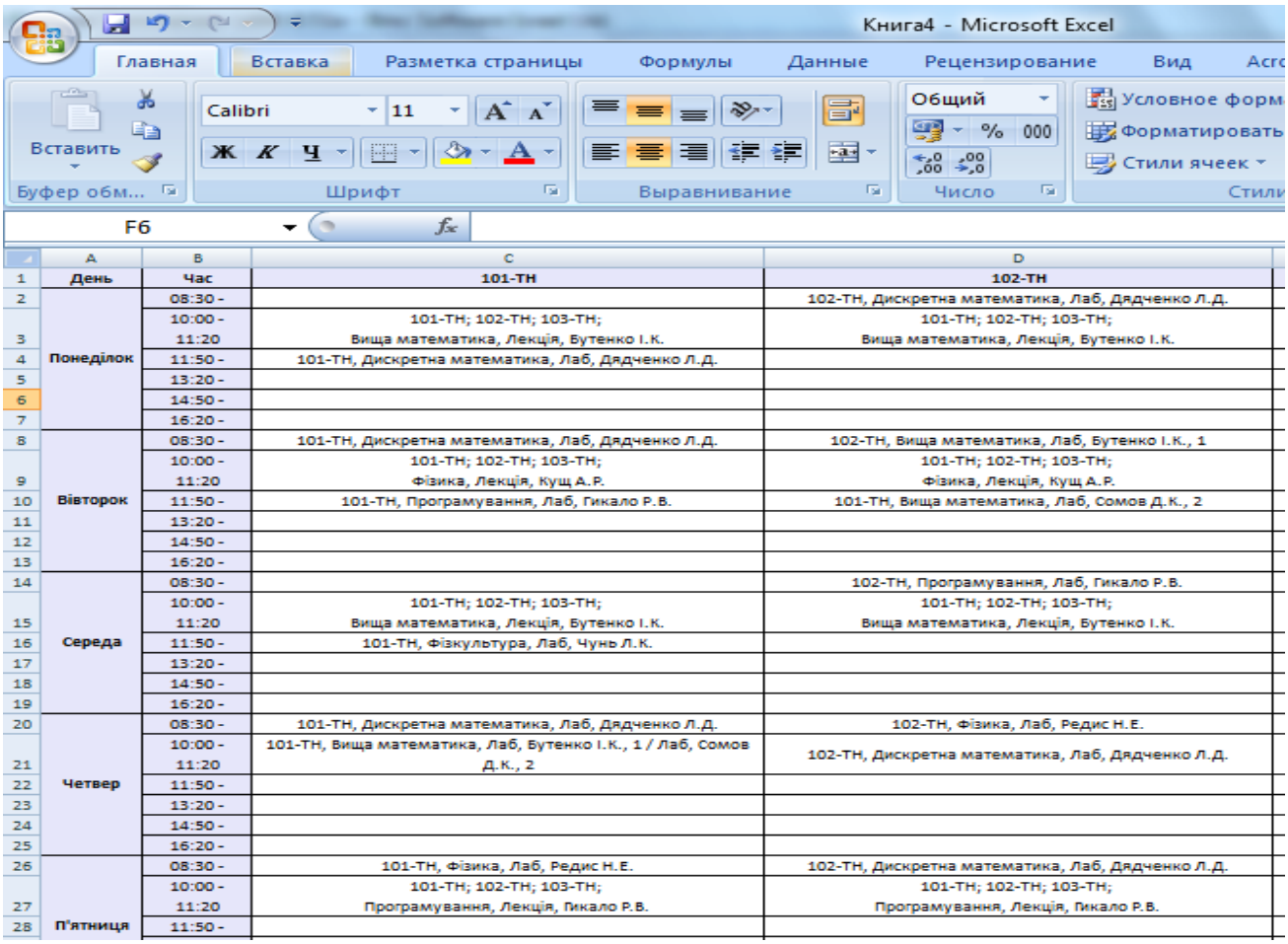


б)

Рисунок 4.21 – Заборона розміщення заняття у комірці

При автоматичній генерації згенерований розклад поміщається до аналогічної таблиці, яку користувач може за потреби відредагувати.

Приклад автоматично згенерованого та експортованого до Excel-файлу розкладу наведено на рисунку 4.22.



	A	B	C	D
1	День	Час	101-ТН	102-ТН
2		08:30 -		102-ТН, Дискретна математика, Лаб, Дядченко Л.Д.
3		10:00 - 11:20	101-ТН; 102-ТН; 103-ТН; Вища математика, Лекція, Бутенко І.К.	101-ТН; 102-ТН; 103-ТН; Вища математика, Лекція, Бутенко І.К.
4	Понеділок	11:50 -	101-ТН, Дискретна математика, Лаб, Дядченко Л.Д.	
5		13:20 -		
6		14:50 -		
7		16:20 -		
8		08:30 -	101-ТН, Дискретна математика, Лаб, Дядченко Л.Д.	102-ТН, Вища математика, Лаб, Бутенко І.К., 1
9		10:00 - 11:20	101-ТН; 102-ТН; 103-ТН; Фізика, Лекція, Кущ А.Р.	101-ТН; 102-ТН; 103-ТН; Фізика, Лекція, Кущ А.Р.
10	Вівторок	11:50 -	101-ТН, Програмування, Лаб, Гикало Р.В.	101-ТН, Вища математика, Лаб, Сомов Д.К., 2
11		13:20 -		
12		14:50 -		
13		16:20 -		
14		08:30 -		102-ТН, Програмування, Лаб, Гикало Р.В.
15		10:00 - 11:20	101-ТН; 102-ТН; 103-ТН; Вища математика, Лекція, Бутенко І.К.	101-ТН; 102-ТН; 103-ТН; Вища математика, Лекція, Бутенко І.К.
16	Середа	11:50 -	101-ТН, Фізкультура, Лаб, Чунь Л.К.	
17		13:20 -		
18		14:50 -		
19		16:20 -		
20		08:30 -	101-ТН, Дискретна математика, Лаб, Дядченко Л.Д.	102-ТН, Фізика, Лаб, Редис Н.Е.
21		10:00 - 11:20	101-ТН, Вища математика, Лаб, Бутенко І.К., 1 / Лаб, Сомов Д.К., 2	102-ТН, Дискретна математика, Лаб, Дядченко Л.Д.
22	Четвер	11:50 -		
23		13:20 -		
24		14:50 -		
25		16:20 -		
26		08:30 -	101-ТН, Фізика, Лаб, Редис Н.Е.	102-ТН, Дискретна математика, Лаб, Дядченко Л.Д.
27		10:00 - 11:20	101-ТН; 102-ТН; 103-ТН; Програмування, Лекція, Гикало Р.В.	101-ТН; 102-ТН; 103-ТН; Програмування, Лекція, Гикало Р.В.
28	П'ятниця	11:50 -		

Рисунок 4.22 – Приклад автоматично згенерованого розкладу

4.5.4 Візуалізація даних

За результатами побудови розкладу система забезпечує побудову графіків для навантаження учасників навчального процесу та вікон протягом тижня з можливістю вибірки даних в розрізі факультетів, кафедр, навчальних груп, викладачів, навчальних корпусів та навчальних аудиторій для окремих об’єктів та для групи об’єктів для порівняння. Є можливість експорту побудованих графіків до Excel-файлу (див. рис. 4.23).

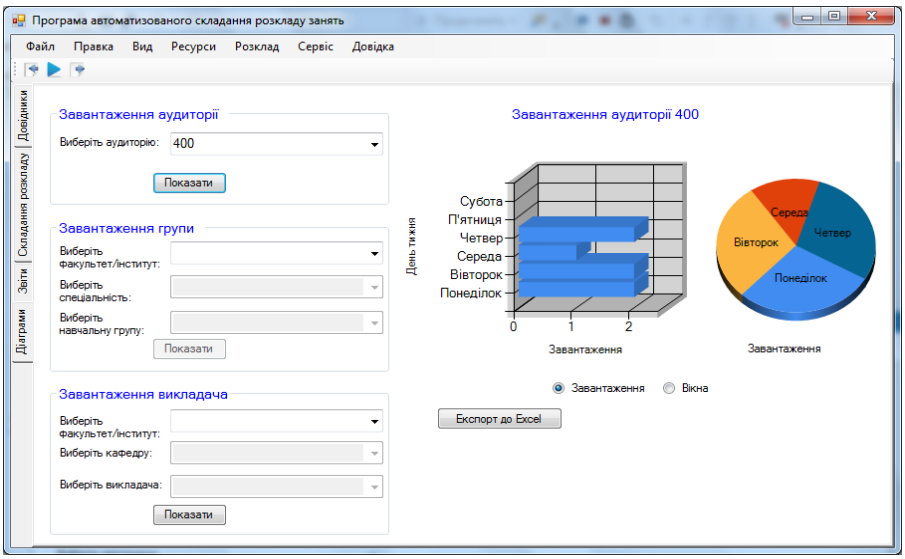


Рисунок 4.23 – Візуалізація навантаження навчальної аудиторії протягом тижня

4.5.5 Експорт даних

У системі передбачено експорт отриманих таблиць для визначених груп до тестових файлів xml, xlsx та txt форматів (див.рис. 4.24).

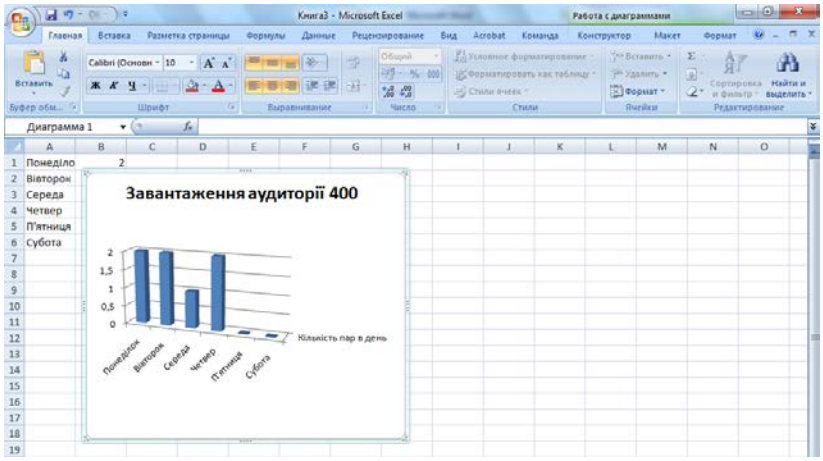


Рисунок 4.24 –Результати експорту побудованого графіку до Excel-файлу

Експорт даних до Excel надає можливість зручного зберігання та розповсюдження даних у звичному для звичайного користувача форматі, а також можливість статистичної обробки отриманих результатів.

5 ТЕСТУВАННЯ СИСТЕМИ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

Тестування, як завершальний етап розробки програмної системи, відіграє життєво важливу роль в процесі створення якісного програмного забезпечення.

Після завершення основних робіт зі створення програмної частини додатку, був виконаний аналіз коду та проектної документації, підготовлено чек-листи у вигляді Google-таблиць та тестові випадки у середовищі TestLink для тестування інтерфейсу користувача та функціонального тестування додатку згідно спеціально розробленої методики.

Під час виконання функціонального тестування додатку було перевірено:

– роботу всіх обов'язкових функцій додатку;
– працездатність форм користувача на додатку;
– реєстрацію та авторизацію користувачів;
– функціональні можливості додатку (введення вхідних даних, редагування вхідних даних, формування розкладу, візуалізацію даних, експорт даних, робота xml-редактора).

Всі знайдені дефекти були занесені до системи управління дефектами Mantis та виправлені, що було підтверджено під час регресійного модульного тестування.

Для оцінки ефективності роботи гібридного алгоритму складання розкладу навчального процесу було проведено тестування розробленої системи на реальних даних Національного університету «Полтавська політехніка імені Юрія Кондратюка», в Навчально-науковому інституті інформаційних технологій і механотроніки на кафедрі комп’ютерних та інформаційних технологій і систем.

Структура Навчально-наукового інституту інформаційних технологій і механотроніки у складі університету наведена на рисунку 5.1.

Кількісні показники експериментальних даних до моделювання наведені у таблиці 5.1. Експериментальна база включає: 1 інститут у складі університету, 1 кафедру, 14 викладачів, 13 груп, 25 аудиторій.

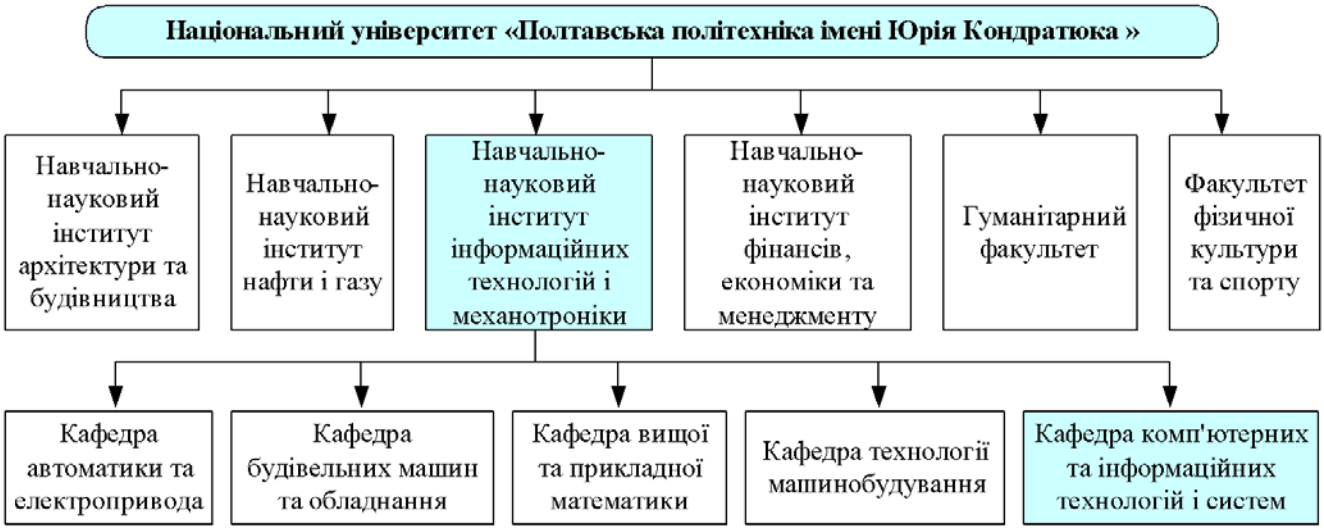


Рисунок 5.1 – Структура Навчально-наукового інституту інформаційних технологій і механотроніки у складі університету

Таблиця 5.1 – Експериментальні дані до моделювання

Кількісні показники	Дисципліни	Групи	Підгрупи	Кафедри	Аудиторії	Викладачі	Розмірність розкладу		
							Кількість днів	Кількість пар в день	Максимальне навантаження в день
Кількісні дані	35	13	21	1	25	14	5	6	4

Було задано наступні параметри методу: розмірність мурашиної колонії – 100 мурах, кількість ітерацій – 50 ітерацій. У результаті моделювання були отримані розклади всіх навчальних груп, в яких жорсткі обмеження були виконані повністю, а м’які – частково, оскільки задовольнити всі побажання учасників навчального процесу неможливо (див. табл. 5.2).

Таблиця 5.2 – Результати експериментів для штрафної функції

Показник	Максимальне значення	Мінімальне значення	Середнє значення, average	Середньоквадратичне відхилення, std
Значення	73	60	64	5.4

На рисунок 5.2 представлено графік залежності значення штрафної функції від ітерації. Як показали експерименти, найбільше зменшення штрафної функції у пропорційному відношенні було зафіксовано протягом 20 ітерацій, що свідчить

про можливість встановлення меншої кількості ітерацій. Але необхідна кількість ітерацій вимагає додаткових досліджень.

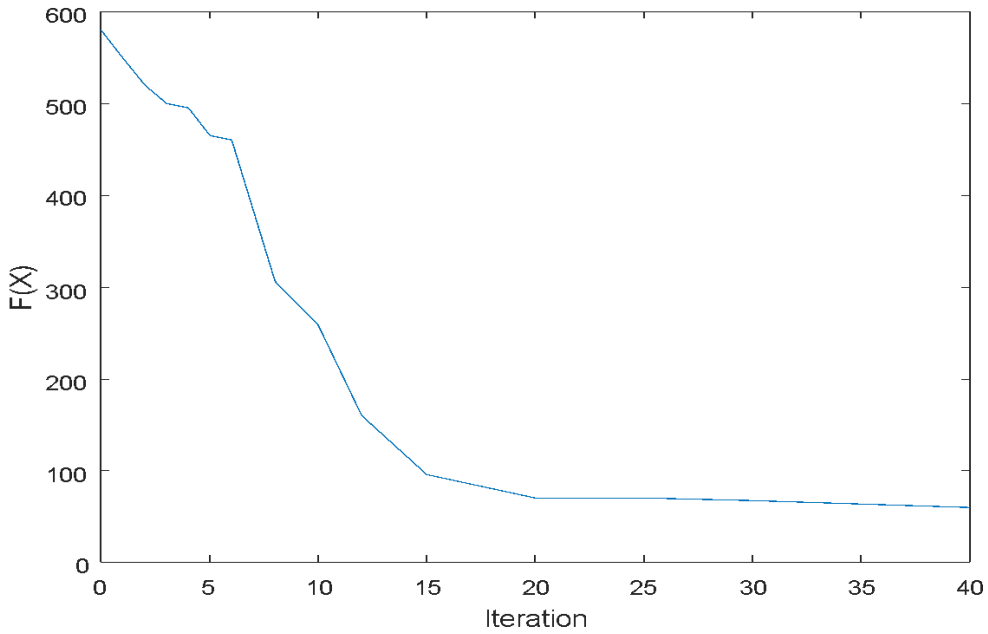


Рисунок 5.2 – Графік залежності значення штрафної функції від ітерації

Порівняння згенерованих розкладів з розкладами, створеними вручну працівниками диспетчерського відділу, показали зменшення значення штрафної функції, а отже, підвищення ефективності складання розкладу, на 12,1%. Але для наукового обґрунтування ефективності роботи даного алгоритму необхідне дослідження на вибірці, яка включає весь навчальний процес університету.

Обчислювальні експерименти для оцінки ефективності роботи гібридного алгоритму складання розкладу навчального процесу проводилися на базі процесора AMD Athlon(tm) II X3 455 з тактовою частотою 3.3 Ггц під керуванням операційної системи Microsoft Windows 7. Час обрахунку склав 5 с.

Окремо було досліджено вплив кількості доступних аудиторій на ефективність роботи алгоритму. Як показали дослідження, зменшення кількості лекційних аудиторій незначно впливає на результат, у той час як зменшення кількості комп’ютерних класів на 30% призвело до 250% росту штрафної функції, що узгоджується з реальними даними, оскільки лабораторні занятті пред’являють підвищені вимоги до аудиторного фонду (див. табл. 5.3).

Таблиця 5.3 – Дослідження впливу кількості доступних аудиторій на ефективність роботи алгоритму

Показник	max	min	average	std
Початкові значення	73	60	64	5.4
Зменшення кількості лекційних аудиторії на 25%	91	67	75	7,0
Зменшення кількості комп'ютерних класів на 25%	200	155	170	12.8
Зменшення кількості лекційних аудиторії на 50%	110	89	95	3,2
Зменшення кількості комп'ютерних класів на 50%	∞ [1)]	280	350 [2)]	36,9 [2)]

Моделювання різних варіантів використання аудиторного фонду на комп'ютері дозволяє врахувати економічний ефект або втрати при введенні /виведенні в/з навчальний процес окремих корпусів або аудиторій, наприклад, з метою оптимізації аудиторного фонду або під час ремонту певної лабораторії.

Таким чином, результати тестування підтвердили ефективність запропонованого метода (підвищення ефективності складання розкладу, на 12,1%) і дозволили промоделювати різні варіанти використання аудиторного фонду.

[1)] Частина колонії не забезпечила виконання жорстких вимог, у такому випадку штрафна функція не розраховується;

[2)] Розраховано лише для мурах, які склали допустимий розклад.

ВИСНОВКИ

У результаті атестаційної роботи розглянуто адаптивні методи автоматичного складання розкладу закладу вищої освіти та розроблено гібридний підхід до розв’язання даної задачі на основі методу мурашиних колоній, генетичного алгоритму та методу деформованого багатогранника. Метод мурашиної колонії є основою даного алгоритму, що формує початкову популяцію для генетичного алгоритму. Комбінація даного метода з генетичним алгоритмом та методом деформованого багатогранника спрямована на усунення таких недоліків даного метода як невизначеність часу збіжності алгоритму та сильна залежність ефективності роботи методу від початкових параметрів пошуку, які необхідно підбирати експериментально. Метод деформованого багатогранника використовується для знаходження параметрів методу мурашиних колоній. Використання генетичного алгоритму дозволить зменшити час роботи алгоритму та збільшити ймовірність попадання в глобальний оптимум.

Розроблений гібридний метод є основою програмної реалізації системи складання розкладу навчального процесу в університеті, яка на основі постійних даних, що зберігаються в sql-базі даних, а також оперативної семестрової інформації, імпортованої з xml-довідників, забезпечує можливість ручного та автоматизованого складання розкладу навчального процесу у ЗВО з використанням розглянутих методів та з використанням Drag-and-drop технологій. Ваги обмежень за замовчуванням дорівнюють одиниці, але є можливість налаштувати дані ваги, а також урахування окремих критеріїв якості у ручному режимі.

Розроблена система була протестована на реальних даних кафедри комп’ютерних та інформаційних технологій і систем Навчально-наукового інституту інформаційних технологій і механотроніки Національного університету «Полтавська політехніка імені Юрія Кондратюка» з наступними даними: 1 інститут у складі університету, 1 кафедра, 14 викладачів, 13 груп, 25 аудиторій.

Було зафіксовано підвищення ефективності складання розкладу на 12,1%, таким чином результати тестування підтвердили ефективність запропонованого метода.

Очікується, що використання даної системи в університеті дозволить полегшити та підвищити ефективність роботи диспетчерів, які займаються складанням розкладу, та якість навчального процесу за рахунок максимального врахування побажань учасників навчального процесу при складанні розкладу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хасухаджиев А.С., Сибикина И.В. Обобщенный алгоритм составления расписания в вузе с учетом новых требований федеральных государственных образовательных стандартов // Вестник АГТУ. Сер.: Управление, вычислительная техника и информатика. 2016. № 3. С. 78–86.

2. PATAT Conferences. URL: https://patatconference.org (дата звернення: 01.03.2020).

3. Бурнасов П.В. Математическая постановка задачи составления расписания занятий // Вестник ИрГТУ. 2014. №4. С. 12-18.

4. Бульонков М.А., Емельянов П.Г., Пак Е.В. К стандартизации данных для составления расписания в учебных заведениях. Открытое образование. 2010. № 3. С. 45–57.

5. Снитюк В.Є. Сіпко Є.Н. Про особливості формування цільової функції та обмежень в задачі складання розкладу занять//Математичні машини і системи, 2014. № 3. С. 67–76.

6. Дворянкин А.М. Чалышев В.С. Обзор методов составления расписания вузов // Изв. ВолгГТУ. Серия Актуальные проблемы управления, вычислительной техники и информатики в технических системах: межвуз. сб. науч. ст. 2011. Вып. 11. № 9. С. 110-113.

7. Томашевський В.М., Новіков Ю.Л., Камінська П.А. Складання розкладів занять у дистанційних системах навчання // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка, 2010. №. 52. С. 118-130.

8. Bania Kumar Rubul, Duarah, Pinkey. Exam Time Table Scheduling using Graph Coloring Approach // International Journal of Computer Sciences and Engineering, 2018. №6. Pp. 84-93.

9. Jha S.K. Exam Timetabling Problem using Genetic algorithm // International Journal of Research in Engineering and Technology. 2014. Vol.3, №5, Pp. 649-655.

10. Астахова И.Ф., Фирас А.М. Составление расписания учебных занятий на

основе генетического алгоритма // Вестник ВГУ, серия: Системный анализ и информационные технологии. 2013. № 2. С. 93-99.

11. Юрчак І.Ю., Москович Т.Р. Дослідження генетичних алгоритмів та їх застосування їхнього в автоматизованій системі розподілу навантаження для викладачів і студентів. URL: http://eom.lp.edu.ua/sntk/doc/ksm2018/moskovytch.pdf (дата звернення 01.05.2020).

12. Leite Nuno, Melicio Fernando, Rosa Agostinho. A fast simulated annealing algorithm for the examination timetabling problem // Expert Systems with Applications, 2018. Vol. 122.

13. Гусейн А. Разработка механизма интеллектуального управления отношениями «студент-преподаватель» в пространстве виртуального образования с применением нейронных сетей // Open education. V. 22. № 5. 2018. Pp. 94-103.

14. Thepphakorn T, Pongcharoen P, Hicks C. An ant colony based timetabling tool // International Journal of Production Economics, 2014, № 149(3). С. 131-144.

15. Verma O.P., Garg. R., Bisht V.S., Optimal Time-Table Generation by Hybridized Bacterial Foraging and Genetic Algorithm // Proceedings of International Conference on Communication Systems and Network Technologies (CSNT‟12), (2012), Pp. 919-923.

16. ITC 2019: International Timetabling Competition. URL: https://www.itc2019.org/home (дата звернення: 01.03.2020).

17. Галактика: Расписание учебных занятий. URL: http://galaktika.ua/manuals/ruz/index.html?ruz_raspisanie.htm (дата звернення: 01.03.2020).

18. 1С: Автоматизированное составление расписания. Университет. URL: https://solutions.1c.ru/catalog/asp_univer (дата звернення: 05.03.2020).

19. Расписание занятий: «Ректор-ВУЗ». URL: http://rector.spb.ru/raspisanie-vuz-4u.php (дата звернення: 01.03.2020).

20. Система для составления расписаний в вузах. URL: https://ru.osvita.ua/vnz/53319/

21. Мулява І.Я. Система формування розкладу навчального занять з використанням суб’єктивних переваг // Міжнародний науковий журнал, 2016.

№ 7. С. 22-27.

22. Лагоша Б.А., Петропавловская А.В. Комплекс моделей и методов оптимизации описания занятий в вузе // Экономика и математические методы, 1993. № 4. С. 48-56.

23. Безгинов, А.Н Трегубов С.Ю. Обзор существующих методов составления расписаний // Информационные технологии в программировании: межвуз. сб. ст. 2005. Вып. 2(14). С.5-19.

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

25. Blum C. Ant colony optimization: Introduction and recent trends // Physics of Life Reviews 2, 2005. С. 353-373.

26. Абухания А.Ю. Модели, алгоритмы и программные средства обработки информации и принятия решений при составлении расписаний занятий на основе эволюционных методов: Дис. …канд. техн. наук: 05.13.01 // Новочеркасск, 2016. 231 с.

27. Жукова М.Ю., Аль-Габри В.М. Автоматизация построения расписания экзаменов ВУЗа с использованием генетического алгоритма // Инженерный вестник Дона, 2017. №3.

28. Титов Ю.П. Модификации метода муравьиных колоний для разработки программного обеспечения решения задач многокритериального управления поставками // Современные информационные технологии и ИТ-образование, 2017. Том 13. № 2. С. 64-74.

29. Аль-Габри В.М. Обзор литературных источников по теме «Автоматизация составления расписания занятий и экзаменов в высших учебных заведениях»// Вестник Донского государственного технического университета. 2017, №1 (88), С. 132-143.

30. Береговых Ю.В., Васильев Б.А., Володин Н.А. Алгоритм составления расписания занятий // Искусственный интеллект. 2009. № 2. С. 50-56.

31. Skakalina E. Застосування мурашиних алгоритмів в рішенні задачі маршрутизації // Системи управління, навігації та зв'язку. Збірник наукових

праць, 2019. Т. 6 (58). С. 75-83.

32. Леснік С.В., Хижняк Т.А. Застосування методу лінійної згортки для вибору джерела альтернативної енергії. URL: http://elc.kpi.ua/old/article/download/158162/187431 (дата звернення 01.05.2020).

33. Устенко С.В., Бібко О.О. Використання методу мурашиної колонії для розв`язання оптимізаційних задач // Науковий вісник НЛТУ України. Вип. 25.3, 2015. С. 351-359.