Київський національний університет імені Тараса Шевченка

На правах рукопису

СІПКО ОЛЕНА МИКОЛАЇВНА

УДК 004.942:378.145

МОДЕЛІ ТА ЕВОЛЮЦІЙНИЙ МЕТОД СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ У ВИЩОМУ НАВЧАЛЬНОМУ ЗАКЛАДІ

05.13.06 – інформаційні технології

Дисертація на здобуття наукового ступеня кандидата технічних наук

Науковий керівник
Снитюк Віталій Євгенович
Доктор технічних наук, професор

Київ – 2016

ЗМІСТ

СПИСОК ПРИЙНЯТИХ СКОРОЧЕНЬ

NP – nondeterministically polynomial

АССР – автоматизована система складання розкладів

АСУ – автоматизована система управління

ВНЗ – вищий навчальний заклад

ГА – генетичний алгоритм

ЗСР – задача складання розкладу

ЗСРНЗ – задача складання розкладу навчальних занять

ІАС – інформаційно-аналітична система

ІМД – інфологічна модель даних

МАІ – метод аналізу ієрархій

МД – модель даних

НП – навчальний процес

ПАВ – послідовний аналіз варіантів

ПАС – програмно-алгоритмічні системи

РБД – реляційна база даних

РМД – реляційна модель даних

СКВ – середнє квадратичне відхилення

СУБД – система управління базами даних

ЧДТУ – Черкаський державний технологічний університет

ВСТУП

Актуальність теми. Ефективність функціонування сучасного виробництва значною мірою визначається рішеннями, прийнятими на етапах календарного планування та оперативного управління. Поряд з поліпшенням якості планових рішень все більш жорсткими стають вимоги до скорочення термінів виробництва, підвищення оперативності та гнучкості управління. Сьогодні у створенні автоматизованих систем управління виробництвом активно використовуються нові для вітчизняної практики технології: управління проектами, управління ресурсами, промислова логістика тощо. Їх спільною метою є оптимізація і використання наявних матеріальних ресурсів та кадрового потенціалу шляхом складання різного роду розкладів. Особливе місце серед них займає задача складання розкладу занять у навчальних закладах. Якість підготовки фахівців у вищих навчальних закладах і, особливо, ефективність використання науково-педагогічного потенціалу залежать певною мірою від рівня організації навчального процесу. Одна з основних складових цього процесу – розклад занять – регламентує трудовий ритм, впливає на творчу віддачу викладачів, тому його можна розглядати як фактор оптимізації використання обмежених трудових ресурсів – викладацького складу. Таким чином, вирішення проблеми розробки оптимальних розкладів занять у вищих навчальних закладах має ще і очевидну економічну складову.

Теорія розкладів є добре вивченою і описаною в багатьох роботах, починаючи з 60-х років минулого сторіччя, такими відомими вченими як Р.В. Конвей, Е.Г. Кофман, В.Л. Максвел, Л.В. Міллер; зокрема, проблемою складання розкладів навчальних занять займались такі вчені як Д. де Верра, М.М. Глибовець, Б.А. Лагоша, А.В. Петропавловська, К.Г. Самофалов, В.П. Симоненко, Т. Строхлейн, В.М. Томашевський, Г. Шмідт та інші.

Свого часу був сформульований клас NP-повних задач теорії розкладів, для розв’язання яких не знайдено ефективних алгоритмів. Саме до таких задач можна віднести більшість реальних задач виробничого планування, зокрема і

задачу складання розкладу навчальних занять. У порівнянні з серединою минулого століття в останні роки з'явилась значна кількість програмних і апаратних засобів (мови програмування високого рівня, сучасні персональні комп'ютери), які розширюють можливості розробників методів розв'язання задач теорії розкладів.

Проведений аналіз сучасного стану досліджень у теорії розкладів вказує на наявність ряду нерозв'язаних задач. Серед них основними є такі:

– здійснено формальні постановки задач й описані моделі і методи, що враховують різні обмеження, в той же час безпосереднє розв'язання практичних задач за допомогою цих моделей виявляється неможливим як внаслідок їх великої розмірності та NP-складності, так і тому, що в практичних задачах необхідно враховувати не лише формальні, але і якісні критерії та обмеження;
– проблеми розв'язання слабко структурованих задач на даний час розглядаються в межах окремих наукових дисциплін (починаючи від системного аналізу і теорії систем і закінчуючи теорією оптимізації), що веде до розрізненості досліджень, відсутності єдиного підходу;
– не вивчено проблеми залежності якості отриманих розв'язків від структурних характеристик та рівня визначеності середовища;
– недостатньо досліджено проблеми відображення системи переваг зацікавлених осіб та їх врахування в цільових функціях.

Оптимізація навчального процесу пов'язана з важливістю розв'язання вищевказаних задач, що вказує на актуальність розробки еволюційних технологій складання розкладів занять на основі суб'єктивних переваг.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконане відповідно до плану науково-дослідної роботи на тему: «Мультиагентні та еволюційні технології розв'язання високорозмірних задач» (номер державної реєстрації 0115U005344), що виконувалась у Черкаському державному технологічному університеті.

Мета і задачі дослідження. Метою дисертаційного дослідження є підвищення ефективності процесу диспетчеризації занять у вищих навчальних закладах шляхом розробки моделей і методів формування розкладу навчальних занять на основі суб'єктивних переваг та еволюційних технологій.

Для досягнення поставленої мети необхідно розв'язати наступні задачі:

- провести аналіз існуючих принципів, моделей, методів та програмно-алгоритмічного забезпечення для розв'язання задачі складання розкладу;
- здійснити формалізовану постановку та розв'язати задачу складання розкладу занять для підвищення ефективності освітнього процесу;
- побудувати математичні моделі, які дозволять визначати якість розкладу навчальних занять;
- розробити структуру потенційних розв'язків задачі складання розкладу занять;
- розробити еволюційний метод оптимізації цільової функції задачі складання розкладу занять;
- виконати верифікацію розроблених моделей та методу і сформувати структурні та функціональні елементи інформаційно-аналітичної системи.

Об'єктом дослідження є процеси диспетчеризації занять у вищих навчальних закладах.

Предмет дослідження – моделі, методи та інструментальні засоби формування розкладу занять у вищих навчальних закладах.

Методи дослідження. Аналіз технологій складання розкладів здійснено на основі системного підходу. Розробка моделей та методу складання розкладу навчальних занять виконані на основі технологій еволюційного моделювання, методу аналізу ієрархій та нечіткої логіки. При розробці структури та формуванні елементної бази інформаційно-аналітичної системи складання розкладу занять використані технології об'єктно-орієнтованого проектування.

Наукова новизна одержаних результатів. У дисертаційному дослідженні виконано формалізацію і розв'язано науково-технічну задачу розробки моделей, методів та інструментальних засобів складання розкладу навчальних за-

нять у вищих навчальних закладах на основі еволюційного моделювання та нечітких експертних висновків, що дозволило одержати теоретичні та практичні результати, які характеризують новизну дослідження та особистий внесок автора, зокрема:

вперше:

- побудовано математичні моделі для формування розкладу занять, в яких, на відміну від існуючих, диференційовано враховані вимоги суб'єктів освітнього процесу та штрафи за їх невиконання, що дозволило підвищити ефективність процесу диспетчеризації занять у вищому навчальному закладі;
- розроблено матрично-еволюційний метод оптимізації цільової функції формування розкладу занять, у якому забезпечено неперервність потенційних розв'язків-розкладів, що зменшує час їх побудови та верифікації;

удосконалено:

- цільові функції для розв'язання задачі складання розкладу навчальних занять шляхом об'єктивізації процесу визначення вагових коефіцієнтів переваг суб'єктів освітнього процесу, що дозволило здійснювати порівняльний аналіз ефективності розкладів;

дістали подальшого розвитку:

- методи еволюційної спрямованої оптимізації за рахунок їх адаптації до розв'язання дискретної задачі формування розкладів занять у вищих навчальних закладах, що дозволило більш повно задовольнити вимоги суб'єктів освітнього процесу та збільшити швидкість пошуку прийнятного (квазіоптимального) розкладу.

Практичне значення одержаних результатів. Отримані автором теоретичні результати доведені до конкретних методик і алгоритмів, розроблено автоматизовану систему складання розкладу навчальних занять. Основні положення, одержані в дисертації, спрямовані на подальший розвиток технологій складання розкладів навчальних занять. Розроблені моделі та методи складають методологічну базу для оптимізації освітнього процесу шляхом формування

покращених розкладів навчальних занять. Результати дисертаційної роботи впроваджені або використані в Черкаському державному технологічному університеті, Черкаському інституті пожежної безпеки імені Героїв Чорнобиля Національного університету цивільного захисту України та Черкаському національному університеті імені Богдана Хмельницького.

Особистий внесок здобувача. Всі основні результати, що виносяться на захист, отримані здобувачем особисто. У роботах, опублікованих у співавторстві, здобувачеві належать, зокрема, у статті [34] – структурні елементи цільової функції, у роботах [35, 121, 129, 131-134,] – матрично-еволюційний метод з використанням штрафної функції для оптимізації цільової функції, а також еволюційний метод формування оптимального розкладу, що базується на розробці та використанні цільової функції, в основі якої лежать врахування вимог викладачів і студентів.

Апробація результатів дисертації. Основні висновки, положення і результати дисертації доповідались і обговорювались на:

- IV Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених “Шевченківська весна” (м. Київ, 2006 р.);
- VIII Міжнародній конференції „Інтелектуальний аналіз інформації” (м. Київ, 2007 р.);
- VI, VII, VIII Всеукраїнських конференціях молодих науковців “Інформаційні технології в освіті, науці й техніці” (м. Черкаси, 2008 р., 2010 р., 2012 р.);
- XVI Всеукраїнській науковій конференції „Сучасні проблеми прикладної математики та інформатики ” (м. Львів, 2009 р.);
- IV Міжнародній науково-технічній конференції “Комп’ютерні науки та інформаційні технології” (м. Львів, 2009 р.);
- XII, XVI Міжнародних науково-технічних конференціях System Analysis and Information Technologies (м. Київ, 2010 р., 2014 р.);
- V, VI, VII Міжнародних школах-семінарах «Теорія прийняття рішень» (м. Ужгород, 2010 р., 2012 р., 2014 р.);

- I, II Міжнародних науково-технічних конференціях «Обчислювальний інтелект» (м. Черкаси, 2011 р., 2013 р.);
- IX Міжнародній науково-практичній конференції «Математичне та імітаційне моделювання систем» (м. Київ-с. Жукін, 2014 р.);
- XX Міжнародній конференції «Knowledge-Dialogue-Solution» (м. Київ, 2014 р.);
- II Міжнародній конференції «Інформаційні технології та взаємодії» (м. Київ, 2015 р.).

Публікації. Результати дисертації викладені в 24 публікаціях, серед яких 7 статей опубліковано у фахових виданнях, затверджених ДАК України, з них 2 – у наукових фахових іноземних виданнях, а також 17 тез доповідей у матеріалах конференцій та наукових семінарів.

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, 4 розділів, списку використаних джерел (145 найменувань), 3 додатків. Основний текст роботи викладено на 120 сторінках. Робота включає 14 рисунків, 15 таблиць.

Розділ 1. АНАЛІЗ ПРОБЛЕМИ ФОРМУВАННЯ ЕФЕКТИВНОГО РОЗКЛАДУ У ВИЩОМУ НАВЧАЛЬНОМУ ЗАКЛАДІ

Рух до інформаційного суспільства супроводжується дослідженнями, що охоплюють різні аспекти навчальної діяльності, пов’язані з автоматизацією навчального процесу (НП). Проблема підвищення якості підготовки фахівців у вищому навчальному закладі (ВНЗ) обумовлює необхідність постійного вдосконалення організації навчального процесу, зокрема, підвищення якості розв’язку основних задач, що регламентують підготовку фахівців: складання навчальних планів та формування розкладу занять.

Розв’язання задачі складання розкладів, у загальній постановці, є процесом виконання певної фіксованої системи завдань за допомогою деякої множини ресурсів [67]. Складання розкладу занять у ВНЗ є досить складною задачею навіть для групи фахівців [77].

Суб’єктивізм, який присутній у процесі складання розкладів, призводить до численних конфліктів, значних часових затрат та одержання неоптимальних за різними критеріями рішень. Автоматизація процесу розв’язання задачі складання розкладів (ЗСР) є досить складною проблемою, її алгоритмізація наштовхується на аспекти NP-повноти. Нагадаємо, що пошук для цих задач точних алгоритмів розв’язку, час роботи яких обмежено поліномом від розміру входу задачі, в даний час є безперспективним [78]. Експонентні алгоритми переборного типу вимагають значних обчислювальних витрат навіть при вирішенні прикладів середньої розмірності. Тому одним з важливих напрямків досліджень є побудова та аналіз наближених алгоритмів з гарантованою оцінкою точності для NP-повних задач. У даний час цей напрям завоював величезне число прихильників в середовищі дослідників, що займаються комп'ютерною математикою.

1.1 Формування розкладу у вищому навчальному закладі – необхідна умова ефективного навчального процесу

Задача складання розкладів є однією з найпоширеніших задач, що розв'язуються кожною людиною (свідомо чи ні) практично щодня. В даний час проводиться багато досліджень, що охоплюють різні аспекти навчальної діяльності, у тому числі й пов'язані з автоматизацією НП. Однак, більшість задач, що розв'язуються у рамках автоматизації НП, не мають тривіального розв'язку, що значно ускладнює їх остаточну реалізацію.

Від своєчасного подання інформації, організації лабораторних практикумів, практичних занять або семінарів залежить і рівень засвоєння інформації, рівень навченості. Вирішенню цих питань присвячено педагогічні теорії початку та середини XX століття, які отримали розвиток у наш час [145]. Спроби підвищення ефективності навчання виконувались навіть шляхом узгодження графіків занять з біологічними ритмами суб'єктів навчання з урахуванням їх психологічного типу та проводились у різних країнах і у різні часи.

Вперше задачу складання розкладів було розглянуто в 1784 році у Великобританії, коли було створено перший розклад для гужового транспорту та паровозів, що рухались за постійним маршрутом. Розклад із вказаним часом прибуття паровозів з'явився у 1839 році. Це була перша збірка розкладів залізничного руху у Великобританії складена Джорджем Бредшоу [7].

З розкладом пов'язаний і Середній час за Гринвічем: в 1847 році британські паровозні компанії об'єдналися і для загальнобританского розкладу було визначено єдиний час за часом Гринвіцької обсерваторії [39].

Однією з найперших конференцій, що присвячена суто проблемі складання розкладів була Міжнародна конференція з теорії та практики автоматизованих розкладів (International Conference of the Practice and Theory of Automated Timetabling), що відбулася в Единбурзі в 1995 році [33]. В березні 1996 року було створено Робочу групу з автоматизації розкладу (WATT), координаторами якої стали Едмунд Берк і Ян Шредер. Основними напрямами конференції є навчальні розклади, робочі розклади і, останнім часом, спортивні розклади.

Математики почали займатися питаннями методології побудови розкладів порівняно недавно. У 1967 році в США вийшла перша у світі книга по теорії розкладів. У СРСР вона з'явилася в перекладі у 1975 році [16]. Кількість літературних джерел, присвячених задачам теорії розкладів, сьогодні нараховує декілька десятків тисяч, друкуються статті у наукових спеціалізованих журналах і захищаються дисертації. Але в останні роки кількість публікацій зменшується і вони все більш орієнтовані на математиків вищої кваліфікації, які, в основному, зацікавлені в одержанні суто теоретичних результатів.

Розклад навчальних занять – це документ, що регламентує трудовий ритм, впливає на творчу віддачу викладачів, тому його можна розглядати як фактор оптимізації навчального процесу. Розклад повинен задовольняти педагогічним вимогам, що базуються на принципах аналітичної дидактики і кібернетичної аналогії [67].

Перші програми для складання розкладу навчальних занять з'явилися ще на початку 90-х років, однак, дотепер використовуються менш, ніж у чверті всіх навчальних закладів. Причин цьому досить багато. І однією з головних причин є те, що розроблювачі бачать і створюють програму за своїм задумом, а диспетчери хочуть бачити програму по-іншому. В результаті готова програма просто не подобається диспетчеру, вона не відповідає уявленню диспетчера про свою роботу.

Якість підготовки фахівців у ВНЗ і особливо ефективність використання науково-педагогічного потенціалу залежать деякою мірою від рівня організації НП. Одна з основних складових цього процесу – розклад занять – регламентує трудовий ритм, впливає на творчу віддачу викладачів, тому його можна розглядати як фактор оптимізації використання обмежених трудових ресурсів – викладацького складу. Технологію ж розробки розкладу варто сприймати не лише як трудомісткий технічний процес, об'єкт механізації й автоматизації з використанням персональних комп'ютерів (ПК), але й як акцію оптимального керування. Таким чином, це – проблема розробки оптимальних розкладів занять у ВНЗ з очевидним економічним ефектом. Крім цього, оптимальне керування та-

кою складною системою неможливе без нагромадження деякої статистичної інформації про процеси, що відбуваються у системі. Тому сама задача складання оптимального розкладу є лише частиною складної системи керування НП. Оскільки інтереси учасників НП різноманітні, задача складання розкладу – багатокритеріальна.

Багатокритеріальність цієї задачі і складність об'єкта, для якого складається математична модель, обумовлює необхідність серйозного математичного дослідження об'єкта для збільшення функціональних можливостей алгоритмів складання розкладів без значного ускладнення моделі і, як наслідок, збільшення обсягів використовуваної пам'яті і часу розв'язання задачі.

Оскільки задача складання розкладу навчальних занять (ЗСРНЗ) є NP-повною [13, 59] навіть для такої задачі, як розподіл N іспитів у T часових інтервалах, кількість можливих розв'язків дорівнює T^N. Це число досягає великих значень для реальних умов ВНЗ, що налічує невелику кількість студентів. Але ЗСР іспитів у порівнянні із ЗСР занять є суттєвим спрощенням. Наприклад, кожний іспит чітко пов'язаний з певною групою студентів і певним викладачем, чого не можна сказати про звичайне заняття з будь-якої дисципліни, адже кожну дисципліну може викладати група викладачів, а вивчати її певна кількість груп студентів. Таким чином, визначення того, який викладач у якої групи веде заняття, – це теж частина розкладу.

Детально зупинимося на тих аспектах НП, які відносяться до задачі складання розкладу, але спочатку визначимо основні поняття.

Розклад – це впорядкована в часі множина навчальних занять. Для кожного заняття повинна бути визначена група, для якої проводиться заняття, викладач, який його проводить, предмет, місце і час проведення заняття. Розклад вважається створеним, якщо для кожної групи розподілені за часом всі заняття з навчального плану і в розкладі немає ніяких розбіжностей або невідповідностей. Далі розкриємо докладно кожен із перелічених понять.

Групою, для якої проводиться заняття, може бути одна або декілька навчальних груп, а також частина навчальної групи – підгрупа. Для кожної навча-

льної групи відома назва, форма навчання, спеціальність, рік вступу і кількість осіб. Кількість підгруп в групі визначається виходячи з кількості студентів. В одній підгрупі не може бути більше 16 чоловік. Спеціальність та форма навчання групи дають інформацію про навчальний план, тобто про ті дисципліни, які студенти цієї спеціальності вивчають протягом усього терміну навчання. Рік вступу говорить про те, на якому курсі група навчається в даний час. Для кожної спеціальності визначена кафедра.

Викладач працює на кафедрі і веде різні дисципліни у різних груп. Крім того, викладач може мати переваги щодо часу занять.

Кожен предмет закріплений за певною кафедрою. Для кожного предмета існує кілька видів занять, наприклад: лекція, практична чи лабораторна робота. Дисципліну можуть вести кілька викладачів.

Під аудиторією будемо розуміти приміщення, де проводяться заняття. Для кожної аудиторії відомо кількість місць і тип, наприклад: лекційна, комп'ютерний клас, лабораторія, спортзал. Аудиторія закріплена за певною кафедрою або є загальною для інституту. Варто відзначити, що ЧДТУ має кілька корпусів, які знаходяться на деякій відстані один від одного.

Організація занять щодо часу залежить від форми навчання. Різні значення можуть мати такі параметри як кількість різних тижнів, кількість днів на тиждень, кількість занять на день. Наприклад, для очної форми навчання характерно чергування парного і непарного тижнів, а для заочної скороченої форми існує 3 різні тижні.

Для груп очної та заочної форм навчання розклад складається два рази на рік, перед початком кожного навчального семестру.

Складанням розкладу займаються диспетчери відділу планування та контролю навчального процесу.

Робочий план групи заснований на навчальному плані спеціальності. Він містить список предметів для даної спеціальності на даний семестр. Для кожного предмета визначені види занять і необхідну кількість годин на тиждень.

Бланк доручень викладачам кафедри містить список всіх викладачів кафедри. Для кожного викладача перераховані предмети, які він веде в семестрі, інформація про вид і кількість занять, вказівка на аудиторію, необхідну для занять, назву груп та кількість груп або підгруп, для яких проходить заняття, побажання.

Коли підготовлена потрібна інформація, починається складання розкладу. Диспетчер заповнює таблицю розкладу для груп і одночасно веде облік аудиторних карток і розклад викладачів.

З робочого плану спеціальності з'ясовується, які предмети повинні бути в розкладі для кожної групи. З бланка доручень викладачам тієї кафедри, яка веде конкретний предмет, для групи визначається викладач. Потім для предмета вибирається час і аудиторія. Вибраний час для викладача має бути вільним і не позначений як небажаний. Аудиторія повинна бути вільна і сумісна з предметом і типом заняття. Чисельність групи повинна бути менше або дорівнювати кількості місць в аудиторії. У бланку доручень викладачам кафедри може бути вказана конкретна аудиторія для заняття або необхідний тип аудиторії. Якщо конкретна аудиторія не вказана, то заняття проходять в аудиторії, яка закріплена за такою ж кафедрою, до якої відноситься група або в загальних аудиторіях інституту. Для груп і викладачів не повинно бути вікон у розкладі. Для груп існує максимально допустима кількість пар в день.

Задача складання розкладу може бути розв'язана різними способами, наприклад, перебором всіх варіантів або за допомогою теорії розкладів і методів оптимізації. Ці методи неприйнятні для розв'язання вручну через їх складність для людини. Задача розв'язується інтуїтивно, з використанням минулого досвіду складання. З огляду на велику ймовірність виникнення помилок і час створення розкладу, було прийнято рішення створити автоматизовану систему складання розкладу.

1.2 Аналітичний огляд принципів, моделей, методів і програмно-алгоритмічного забезпечення для складання розкладів занять

Дослідження низки наукових робіт показало велику зацікавленість науковців до задачі складання розкладів навчальних занять. Різноманітність досліджень в даній області визначається використанням як сучасних методів і підходів розв’язання даної задачі, так і розробкою нових або модифікованих методів.

Визначимо основні тенденції сучасного наукового пошуку в напрямку технологій розв’язання задач складання розкладів занять в навчальних закладах. Особливості процесу складання розкладу занять для дистанційного навчання розглянуті в статті [141]. Виконано аналіз різних методів складання розкладів: імітації відпалу, розфарбовування графа, імітаційного моделювання, логічного програмування з обмеженнями. Критерієм оптимальності розкладу визначено максимальне ущільнення викладацької навантаження і наголошено на використання генетичного алгоритму для максимізації цільової функції. Аспекти застосування генетичного алгоритму розглянуті і в роботі [61]. Той же генетичний алгоритм розглянуто і в [42], критеріальною функцією тут виступають усереднене значення рівня виконання вимог викладачів: небажання проводити пари в певний час, мінімізація кількості «вікон», рівномірність занять. Триває тема використання генетичних алгоритмів в статті [62] з ухилом в бік паралелізації обчислювального процесу і використання Grid-систем.

Метод формування розкладу, яке визначається побажаннями студентів, описаний в [125]. Запропоновані ідеї мають щось спільне з відомими технологіями, подібними методу Дельфі, оскільки присутні всі ті ж атрибути: заочность, багаторівневість, анонімність. Автор [100] пропонує інформаційну модель на базі обмежень зв'язків елементів розкладів, розклад формується з використанням методу аналізу ієрархій. В [53] тема використання такої моделі розвинена в напрямку використання генетичного алгоритму. У статті [148] запропоновано застосування елементів тензорного обчислення для опису процесів складання розкладів для навчальних комплексів. В роботі [110] зроблено висновок про недоцільність повністю автоматизованого складання розкладу через трудоміст-

кість побудови точних математичних конструкцій і пропонується використання діалогового процесу. Основні обмеження та види критеріальних функцій систематизовані в [82]. Автор стверджує, що існує два критерії пошуку цільової функції: мінімізація витрат і максимізація ефективності складеного розкладу. І, якщо з другим критерієм цілком можна погодитися за умови уточнення поняття «ефективного розкладу», то перший критерій викликає, як мінімум, здивування. Можливо, це мінімізація обчислювальних ресурсів, необхідних для отримання опорного або квазіоптимального рішення, або мінімізація витрат ручної праці диспетчера. Оптимізація структури інформаційної бази пропонується в [111]. Ще одним напрямком, превалюючим в деяких роботах [45, 58, 115] є використання елементів нечіткої логіки, що є природним, враховуючи природу вимог суб'єктів навчального процесу. Нечітка логіка дозволяє помітно спростити формалізацію вимог, але часто призводить до побудови розкладу, що має не кращі характеристики в результаті переходу від «жорстких» вимог до більш «м'яких».

Мілехіна Т.В. в своїй роботі [89] описує можливість використання методів підвищення ефективності кластерних систем при розв'язанні оптимізаційних задач за рахунок адаптації алгоритмів до особливостей їхньої архітектури. В результаті було розроблено паралельний алгоритм, що дозволяє автоматизувати процес пошуку близьких до оптимальних параметрів згортки, що забезпечують високу якість отримуваних розкладів.

Лугуєв Т.С. [83] пропонує використовувати теоретико-графові моделі і методи складання розкладів без переривань. Результатом роботи виступають розроблені комплекси програм для проведення обчислювальних експериментів, що дозволяють порівнювати швидкодію різних потокових методів, що використовуються при перевірці умов існування розкладів без переривань.

Не менш цікавим є використання так званих інтелектуальних методів розв'язання задачі складання розкладів занять. В їх основі лежить використання різних евристик і евристичних алгоритмів (Клеванський Н.Н., Костін С.А.). Проте, розв'язання задачі складання розкладу за допомогою евристик не гарантує знаходження глобального оптимуму [75].

В даний час для розв’язання задачі складання розкладу застосовується ще один новий підхід – нейронні мережі (Піліньский М., Рутковська Д.). Найважливішим недоліком застосування цього підходу є складність вибору початкового стану нейронної мережі.

В останні роки особливого поширення набули дослідження методів еволюційного пошуку (Єрунов В.П., Морковін І.І. Каширіна І.Л., Нізамова Г.Ф.) [67, 76]. Застосування методів еволюційного пошуку призводить до отримання прийнятних результатів, однак має місце висока обчислювальна трудомісткість і відносна неефективність на заключних етапах еволюції.

У роботі Нізамової Г.Ф. [76] використовуються методи системного аналізу, генетичних алгоритмів і теорії важливості критеріїв, що дозволяє спростити задачу, але призводить до жорсткої прив’язки складеного розкладу до викладацького складу.

Підводячи підсумки проведеного аналізу, можна дійти висновку про те, що значна кількість робіт присвячено огляду отриманих до цього часу результатів, або оптимізації процесу отримання ефективного розкладу з використанням сучасних обчислювальних технологій, таких як генетичні алгоритми. Треба зауважити, що вони є методами хоч і спрямованого, але випадкового пошуку і їх реалізація, і виконання найчастіше є ресурсоємним процесом, який, разом з необхідністю перевірки виконання значної кількості обмежень, часто не призводить до отримання прийнятного результату за прийнятний час.

Розглянемо класичні підходи до розв’язання задачі складання розкладів навчальних занять.

При формуванні розкладів встановлюють часовий період і це може бути один, два тижні чи місяць. Тоді розклад циклічно повторюється весь семестр. Але, у загальному випадку, цієї вимоги не дотримуються.

Якщо навчальні плани складені таким чином, що дисципліни можуть читатися лише один раз в тиждень або через тиждень, то задача складання оптимального розкладу може бути представлена як задача лінійного цілочисельного

програмування. При такому підході інтереси суб'єктів навчального процесу враховуються у вигляді обмежень або критеріїв оптимальності.

Розглянемо одну з класичних математичних моделей задачі складання розкладу навчальних занять [23]. Припустимо, що у ВНЗ є N навчальних груп, об'єднаних в R потоків, r – номер потоку, $r = 1,...,R$; k_r – номер навчальної групи в потоці r, $k_r = 1,...,G$. Для кожної групи k_r визначається множина номерів робочих днів цієї групи T_{kr} яка є підмножиною всіх робочих днів навчального закладу. Кожний робочий день розбивається на періоди навчання – пари, загальна кількість яких J, а $j = 1,...,J$ – номер визначеної пари.

На основі навчального плану для кожного потоку складається список із S_r лекційних занять, де $s_r = 1,...,S_r$ – номер конкретної дисципліни у списку. Для кожної групи k_r складається список із Q_{kr} запланованих практичних занять, $q_{kr} = 1,...,Q_{kr}$ – номер дисципліни у списку. Список усіх занять для кожної конкретної групи буде складатися з усіх занять, присутніх у списках. При цьому, якщо по дисципліні протягом тижня проводиться більше одного заняття, ця дисципліна згадується в списку лекцій чи практичних занять стільки разів, скільки їх передбачається навчальним планом для кожного потоку чи групи.

Нехай p – номер (ім'я) викладача, $p = 1,...,P$. Введемо булеві змінні:

$$\delta^{p}_{rs_r} = \begin{cases} 1, \textit{якщо на потоці } r \textit{ лекцію } s_r \textit{ читає} \\ \textit{викладач } p, \\ 0, \textit{в іншому випадку}. \end{cases}$$

$$\Delta^{p}_{rk_rq_{kr}} = \begin{cases} 1, \textit{якщо у групі } k_r \textit{ практичне заняття} \\ q_{kr} \textit{ проводить викладач } p, \\ 0, \textit{в іншому випадку}. \end{cases}$$

Навчальне навантаження викладачів планується до складання розкладу занять, внаслідок чого на даному етапі величини $\delta^{p}_{rs_r}$ і $\Delta^{p}_{rk_rq_{kr}}$ можна вважати заданими. Для кожного викладача p також задається його аудиторне навантаження.

Нехай A_1 – число аудиторій для лекцій, A_2 – для практичних занять. У загальному випадку, класифікація аудиторій може бути більш складною, тоді належність аудиторії до конкретного типу визначається за такими ознаками як кількість місць, тип лабораторії та ін.

При такому представленні початкових даних задача складання розкладу полягає у визначенні для кожного заняття з потоком чи групою дня тижня і пари у ньому з урахуванням виконання обмежень і оптимізації деякої цільової функції.

Введемо наступні булеві змінні:

$$\xi_j^{tk_r} = \begin{cases} 1, \textit{якщо в день } t \textit{ група } k_r \\ \textit{починає заняття з пари } j, \\ 0, \textit{в іншому випадку}; \end{cases} \quad \eta_j^{tk_r} = \begin{cases} 1, \textit{якщо в день } t \textit{ група } k_r \\ \textit{закінчує заняття парою } j, \\ 0, \textit{в іншому випадку}; \end{cases}$$

$$y_{rt_j}^{s_r} = \begin{cases} 1, \textit{якщо на потоці } r \textit{ на парі } j \\ \textit{читається лекція } s_r, \\ 0, \textit{в іншому випадку}; \end{cases} \quad x_{rk_rt_j}^{q_{kr}} = \begin{cases} 1, \textit{якщо на потоці } r \textit{ в день } t \\ \textit{на парі } j \textit{ у групі } k_r \textit{ прово –} \\ \textit{диться практ. заняття } q_{kr}, \\ 0, \textit{в іншому випадку}. \end{cases}$$

Тоді обмеження: „кожного дня на кожній парі для кожної групи може проводитися не більше одного заняття” є таким:

$$\sum_{q_{kr}=1}^{Q_{kr}} x_{rk_rt_j}^{q_{kr}} + \sum_{s_r=1}^{S_r} y_{rt_j}^{s_r} \le 1, \quad \forall r = 1,...,R, \quad \forall k_r = 1,...,G_r, \quad \forall t \in T_{kr}, \quad \forall j = 1,...,J. \tag{1.1}$$

Інша обов’язкова умова складання розкладу: „для кожної групи k_r повинні виконуватися всі види аудиторної роботи протягом тижня” буде такою:

$$\sum_{t \in T_{kr}} \sum_{j=1}^{J} \left(\sum_{q_{kr}=1}^{Q_{kr}} x_{rk_rt_j}^{q_{kr}} + \sum_{s_r=1}^{S_r} y_{rt_j}^{s_r} \right) = W_{kr}, \quad \forall r = 1,...,R, \quad \forall k_r = 1,...,G_r. \tag{1.2}$$

В конкретних задачах список обмежень може бути продовжено.

Після складання обмежень необхідно вибрати критерій оптимальності розкладу. Зокрема, можна розглядати критерій максимізації зваженої кількості ві-

льних від аудиторної роботи днів для всіх викладачів, що при фіксованій довжині робочого тижня еквівалентно максимальному сукупному ущільненню аудиторного навантаження. Тоді критерій буде таким:

$$\sum_{t \in T} \sum_{p=1}^{P} \Omega_p z_t^p \to \max, \tag{1.3}$$

де T і P – кількість робочих днів і кількість викладачів, відповідно; Ω_p – ваговий коефіцієнт, що визначається статусом викладача; z_t^p – булева змінна, що приймає значення „0”, якщо викладач має заняття в цей день і „1” в іншому випадку.

Побудована таким чином модель буде відображати основні фактори, що враховуються при складанні розкладу, і відноситься до класу задач лінійного булевого програмування, розв’язок яких може бути знайдений, наприклад методом гілок і границь [2, 4]. Розв’язок таких задач навіть відносно малої розмірності, зазвичай, вимагає великих витрат часу. Однак в деяких випадках це може виявитися необхідною платою за точне виконання всіх обмежень. В деяких випадках можливе розбиття задачі на систему підзадач суттєво меншої розмірності [81].

Представлена модель містить ряд недоліків, пов’язаних як з неповною адекватністю представленого розв’язку досліджуваній предметній задачі, так і значною трудомісткістю використання запропонованого комплексу програм, що вимагає участі кваліфікованого користувача.

Критеріальна функція, що визначає кількість вільних від аудиторних занять днів викладачів, не є єдиною. При складанні розкладів необхідно також оптимізувати аудиторний час роботи студентів, зокрема прагнути до рівномірного розподілу занять взагалі та лекцій зокрема по робочих днях, відсутності «вікон». До пріоритетних завдань належать також: проведення лекцій, практичних та лабораторних занять у спеціалізованих лабораторіях; максимізація кількості занять у першу зміну та інші.

Кожний ВНЗ має свою специфіку і тому моделі, які містять критеріальні функції та обмеження, мають певні відмінності. Разом із тим, очевидно, що раціональне розв'язання задачі складання розкладів базується на формуванні інтегральної функції, яка є певною згорткою окремих критеріальних функцій. Інтегральний критерій може бути адитивною функцією із ваговими коефіцієнтами, що визначаються експертами, він також може бути залежністю, структурна та параметрична ідентифікація якої здійснюватиметься на базі статистичної інформації. У будь-якому випадку, знаходження оптимуму такого критерію є складною науковою задачею, яка, у повному обсязі, залишається нерозв'язаною. Виконаємо аналіз методів, які дозволяють знаходити квазіоптимальні розв'язки.

Метод імітації відпалу. Ідея алгоритму імітації відпалу запозичена з досліджень поведінки атомів металу в процесі його випалювання. У металі, нагрітому до температури, яка перевищує точку його плавлення, атоми перебувають в хаотичному русі. При цьому, вони прагнуть до стану мінімуму енергії, але при високих температурах енергія атомного руху перешкоджає цьому. В процесі поступового охолодження металу виникають усе більш низькоенергетичні стани, поки не буде досягнутий найнижчий з можливих станів – глобальний мінімум [143].

При розв'язанні задачі складання розкладів в якості функції енергії приймають критеріальну функцію, що враховує штрафи, які додаються до поточного розкладу за кожний суперечливий у ньому момент, а в якості низькоенергетичного стану – коректний (хоч і невідомий) розклад. Загальну ідею алгоритму імітації випалювання для задачі складання розкладу можна представити наступною схемою:

Крок 1. На першій ітерації генеруємо деякий коректний початковий розклад X^0, який вважається поточним розв'язком задачі ($X = X^0$).

Крок 2. Задаємо початкове, високе значення температури T^0 і операції мутації розкладу (зміна часу проведення заняття; зміна аудиторії; обмін місцями в розкладі двох занять).

Крок 3. На основі введених операцій мутації і поточного розв'язку генеруємо новий коректний розклад X^t.

Крок 4. Шукаємо приріст цільової функції $\Delta f = f(X^t) - f(X)$ (якщо $\Delta f < 0$ (розв'язок не погіршився), то новий варіант розкладу стає поточним ($X = X^t$); якщо $\Delta f > 0$ (розв'язок погіршився), то новий розклад стає поточним з ймовірністю $p = e^{-\Delta f / T}$. Відповідно, з ймовірністю $(1 - p)$ попередній розклад зберігається в якості поточного. Такий підхід дозволяє виходити із локальних екстремумів.

Крок 5. Температура зменшується.

Крок 6. Якщо не виконується критерій зупинки (виконання заданого числа ітерацій; виконання заданого числа ітерацій без покращення значення цільової функції на задане значення), то перейти на крок 3.

Для підвищення ефективності алгоритму реалізовують збереження m кращих розв'язків, а також n останніх згенерованих розкладів. Це дозволяє запобігати зацикленню процесу складання розкладу.

Точність розв'язку збільшують за рахунок більш повільної зміни температури [38]. Незважаючи на зовнішню простоту, такий підхід виявляється цілком ефективним для складання невеликих розкладів, але є абсолютно неприйнятним в іншому випадку через велику кількість кроків у невірному напрямку.

Метод розфарбування графа. Задачу складання розкладу можна розглядати як задачу розфарбування графа, яку визначають як пошук хроматичного числа графа чи, іншими словами, пошук мінімального числа кольорів, необхідних для розфарбування вершин деякого графа з використанням для кожної пари сусідніх вершин різних кольорів. Сама задача пошуку хроматичного числа є NP-повною задачею [80, 139].

Для постановки задачі складання розкладу як задачі розфарбування графа будується граф, у якому кожна вершина є заняттям. Якщо між будь-якими двома вершинами можливі конфлікти, наприклад, два заняття проводяться в одній аудиторії, чи з одним викладачем, то вони з'єднуються ребром. Це еквівалентно

забороні одночасного проведення цих занять. Тоді задача складання розкладу є задачею мінімізації числа кольорів, необхідних для розфарбування графа. Кожний колір відповідає одному періоду розкладу.

Застосування такого підходу для розв'язку реальних задач є малоефективним. У той же час, задача розфарбування графа при складанні розкладів виявляється корисною у випадку її комбінації з іншими алгоритмами [11].

Імітаційне моделювання. З огляду на NP-повноту задачі, для її розв'язку застосовують імітацію дій диспетчера при складанні розкладу.

У цьому випадку алгоритм оперує безпосередньо розкладом і списком занять, які необхідно включити в розклад. Процес складання розкладу починається з порожнього розкладу, коли всі заняття знаходяться в списку неврахованих занять. Далі згідно з алгоритмом переходять від одного незакінченого розкладу до іншого, прагнучи якнайкраще розставити всі заняття, включені в список. Процес продовжується доти, поки не буде сформовано повний розклад, чи виконається фіксована кількість ітерацій.

При реалізації алгоритму особлива увага надається розробці евристичних правил вибору чергового заняття зі списку, визначення найкращої для нього позиції в розкладі та оцінці одержуваного розкладу.

Позитивною рисою такого підходу є можливість детального врахування специфіки розв'язуваної задачі у випадку складання розкладу для конкретного ВНЗ. Недоліком є низька варіативність, що обмежує можливість застосування розробленої системи в інших навчальних закладах [29; 104].

Логічне програмування в обмеженнях. Складання розкладу є задачею виконання обмежень. Для розв'язку таких задач розроблено безліч алгоритмів і, навіть, виник цілий напрямок – програмування в обмеженнях (constraint programming).

Програмування в обмеженнях тісно пов'язано з традиційним логічним програмуванням, у рамках якого воно і сформувалося. Ідея розв'язання таким методом полягає у тому, що програміст визначає деяку множину змінних $x_1,...,x_n$ і області їх значень $\Omega_1,...,\Omega_n$, описує додаткові обмеження, яким повин-

ні задовольняти змінні, а програмна система знаходить значення змінних, що задовольняють одночасно всім заданим обмеженням.

Розглянемо ВНЗ, у якому працює M викладачів, у навчальному плані зафіксовано N предметів, аудиторний фонд нараховує L приміщень. Визначається множина $P = 1,2,...,D$, елементами якої є всі періоди навчання (пари) у ВНЗ протягом тижня, а D – число всіх пар протягом тижня.

Нехай y_{ij} – період навчання, у який i-й викладач веде j-й предмет, $y_{ij} \in P$ Тоді обмеження „кожен викладач у кожен момент часу може вести не більше одного заняття" є таким:

$$y_{ij} \neq y_{ij'}, \quad 1 \leq i \leq M, \quad 1 \leq j, \quad j' \leq N, \quad j \neq j'.$$

Нехай z_i – аудиторія, у якій проводить заняття i-й викладач, $1 \leq z_i \leq L$. Тоді обмеження „у кожній аудиторії в кожен момент часу може проводитися не більше одного заняття" є таким:

$$z_i \neq z_i', \quad 1 \leq i \leq M, \quad 1 \leq i' \leq M, \quad i \neq i'.$$

Результатом роботи алгоритму є множина значень кожної змінної, що не суперечать зазначеним обмеженням. При цьому область визначення змінних, може істотно скоротитися чи навіть містити єдине значення.

Основна перевага використання такого методу – скорочення простору пошуку, що досягається не шляхом оцінки кожного варіанта розкладу, а за рахунок того, що система сама виключає з розгляду „дороги, що свідомо ведуть у глухий кут" [36].

З моменту появи комп'ютерної техніки у навчальних закладах багато розробників програмного забезпечення намагалися, якщо не повністю розв'язати задачу складання розкладу, то хоча б частково полегшити цей процес. Але, зіткнувшись з великою складністю цієї задачі, більшість з них так і не змогли створити дійсно цінний програмний продукт, не дивлячись на те, що сьогодні ринок програмного забезпечення для складання розкладів достатньо великий.

На сьогодні відомо багато комп'ютерних програм складання розкладу занять у ВНЗ, що добре зарекомендували себе та успішно використовуються на практиці багатьох ВНЗ. Деякі програми розвиваються, досить часто оновлюються та вже є потужними системами складання розкладу занять. Технічно ці програми мають різні алгоритми складання та різні платформи, на яких вони побудовані. Отже, можна констатувати, що на сьогодні створений інформаційний інструментарій для складання розкладу занять, який може задовольнити вимоги найвибагливішого менеджера та найскладнішого, з точки зору організації, навчального закладу.

Розглянемо кілька таких програм: «Деканат», «Ректор-ВНЗ», «АВТОР-2+», «МЕТОДИСТ».

Програма обліку студентів «Деканат» є засобом автоматизації управління контингентом студентів. Система ІС «Деканат» має клієнт-серверну архітектуру і включає себе базу даних на SQL Server, клієнтський додаток ІС «Деканат» і допоміжну програму «Users Manager» для адміністрування системи. Перед встановленням програми передбачається наявність сервера бази даних MS SQL Server 2000 / 2005. ІС «Деканат» інтегрована з ІС «Плани», «Приймальна комісія», «Електронні відомості», «Семестрові графіки груп», «Картки студентів заочної форми навчання», «Visual Testing Studio».

«Ректор-ВНЗ» – програма, призначена для складання розкладу занять у вузах. Програма складається з чотирьох розділів: «Списки», «Навантаження», «Розклад» і «Заміни». Розділ «Списки» служить для введення, редагування та друку списків кафедр, спеціальностей, груп, дисциплін, аудиторій, викладачів і видів занять. Розділ «Навантаження» використовується для введення, редагування та друку навчальних планів по кожній спеціальності, навантажень викладачів, графіків розподілу годин по тижнях в межах семестру, звітів по завантаженню викладачів, кафедр і вузу в цілому. Розділ «Розклад» призначений для складання розкладу по групах, викладачам, аудиторіями і вузу в цілому. Розділ «Заміни» дозволяє оперувати замінами викладачів. Розклад занять можна складати в автоматичному, ручному або комбінованому режимі; переходити від од-

ного режиму до іншого можна в будь-який момент часу. При складанні розкладу в автоматичному режимі програма враховує всі сформульовані вимоги до розкладу. При складанні розкладу в ручному режимі програма підказує можливі варіанти розстановки пар вибраного викладача, можливі варіанти заповнення порожніх клітин в розкладі групи, стежить за кількістю місць в аудиторіях. Готовий розклад занять однієї, всіх або деяких груп і викладачів можна зберегти у форматах Microsoft Word, Excel або HTML.

Система АВТОР-2+ призначена для швидкого і зручного складання розкладів занять і супроводу їх протягом усього навчального року. Система допомагає легко будувати, корегувати і роздруковувати у вигляді зручних і наочних документів.

Програма відрізняється унікальним і дуже потужним алгоритмом побудови та оптимізації розкладу. Отриманий автоматично розклад практично не вимагає ручної доробки, тобто навіть при дуже складних і жорстких обмеженнях автоматично розміщуються ВСІ можливі заняття. Якщо у вихідних даних є нерозв’язні протиріччя, то їх можна виявити і усунути, використовуючи спеціальний блок аналізу. АВТОР-2+ дозволяє:

- оптимізувати "вікна" в розкладі;
- враховувати необхідний діапазон днів/годин як для груп, так і для викладачів;
- оптимально розміщувати заняття по кабінетах (аудиторіям) з урахуванням особливостей груп, предметів, викладачів і місткості кабінетів;
- розділяти групи на кількість підгруп до десяти;
- вводити (крім основних предметів) Спецкурси і Факультативи;
- оптимізувати рівномірність і трудомісткість розкладу.

«МЕТОДИСТ» – програмний комплекс, який поставляється в двох варіантах Стандарт і Virtual. Він дозволяє вирішувати такі основні задачі:

- розподіл і контроль навчального навантаження;

- облік методичних рекомендацій та особистих побажань викладачів («вікна», метод. дні, теніс по четвергах тощо);
- складання розкладу для будь-якого типу навчального закладу;
- формувати зміни/конфігурації з довільним складом груп зі своїм навчальним графіком ;
- закріплення спеціальних аудиторій;
- облік зайнятості викладачів і аудиторій;
- облік часу переходів між корпусами;
- можливість багаторазового автоматичного «покращення» розкладу.

Здійснивши порівняльний аналіз цих систем можна зробити висновок, що жодна система не відповідає необхідним вимогам в повному обсязі. На жаль, відсутність досвіду використання названих програм не дозволяє зробити їх повний аналіз. Для програм, розроблених для середовища Windows суттєвими недоліками є нерозподілені заняття, наявність «вікон» та тривалий час складання розкладу.

1.3 Формалізація задач підвищення ефективності роботи вищого навчального закладу

Організація навчального процесу в будь-якому ВНЗ базується на складанні розкладу занять, оскільки об’єднує в собі всі елементи інформаційного середовища, таких як викладачі, студенти, аудиторії, дисципліни, спеціальності тощо. Тому головною задачею автоматизації даної області на сьогодні є оптимальний розподіл усіх вищенаведених ресурсів в таблицях розкладу занять з дотриманням певних обмежень. В загальному вигляді функціональну модель інформаційної моделі ВНЗ можна представити так, як зображено на рис. 1.1.

Рис. 1.1 – Структурна функціональна модель інформаційної системи ВНЗ

Однією зі складових підвищення якості діяльності ВНЗ є використання інформаційно-комп'ютерного інструментарію в управлінні навчально-виховним процесом. Розв'язання цієї проблеми має велике значення для забезпечення оптимальності управлінських рішень за рахунок автоматизації рутинних процесів в управлінській діяльності керівників. В умовах розвитку в Україні ринкових відносин змінюються 2 принципи управління ВНЗ [68]. У вищій школі сьогодні активно розвиваються процеси гуманізації, що охопили взагалі все українське суспільство. Саме тому важливим завданням у рішенні зазначеної проблеми є наукова розробка технологічних основ забезпечення сприятливих умов для розкриття й розвитку особистісного потенціалу головних учасників навчально-виховного процесу − викладача та студента. Зміна навчальних програм і методик, впровадження нових педагогічних технологій, профілізація

ВНЗ вимагають динамічності у створенні розкладу занять та забезпечення можливості його оперативного регулювання.

Організація навчально-виховного процесу сьогодні вже не може спиратися на принципи та методи, що використовувалися в управлінні навіть 10-15 років тому. Але, слід визнати, що методи та інструменти планування навчально-виховного процесу залишаються поки переважно ручними і здійснюються частіше на емоційній основі. Більшість керівників не відчувають потреби використання автоматизованих способів прийняття управлінських рішень.

Інформатизація вищої освіти – сукупність взаємопов'язаних організаційних, управлінських, економічних, науково-технічних, навчальних, виховних процесів, що спрямовані на створення умов для задоволення інформаційних потреб всіх учасників освітнього процесу (студентів, викладачів, співробітників ВНЗ), розвитку їх інтелектуального потенціалу, самореалізації і самовдосконалення, на забезпечення підготовки до повноцінної професійної діяльності і життя в інформаційному суспільстві на основі створення, розвитку і використання сучасних інформаційно-комунікаційних систем, мереж, ресурсів та технологій. Проблема інформатизації – це стрижень, навколо якого сьогодні повинна будуватися вся система роботи сучасного ВНЗ. Розв'язання цієї проблеми надасть можливість виконати замовлення інформаційного суспільства на підготовку фахівців, які спроможні на сучасному рівні застосовувати інформаційно-комунікаційні технології у професійній діяльності та повсякденному житті [81].

З появою та розвитком обчислювальної техніки процеси генерації та вибору альтернатив рішень стали реалізовуватися на комп'ютерах. У зв'язку з цим з'явилися нові задачі формалізації і алгоритмізації процесу прийняття рішень, які на сьогоднішній день складають окрему велику область досліджень.

Формалізація тієї чи іншої задачі оптимізації, у загальному випадку, припускає опис усіх важливих факторів, які впливають на досягнення мети, їх взаємодії, обмежувальних умов та критерію оцінки якості рішення, що приймається, на базі якого можна здійснювати вибір між альтернативами. У якості крите-

рію оцінки виступає цільова функція, аргументами якої є кількісні характеристики, які описують становище факторів, що впливають на досягнення мети в розв'язуваній задачі [79]. При цьому, рішенню, яке приводить до найкращого результату, як правило, відповідає екстремальне значення цільової функції, тобто точка її максимуму або мінімуму.

Отже, процес генерації варіантів розв'язку та вибору найкращої з отриманих альтернатив зводиться до створення всіх можливих комбінацій значень характеристик, які впливають на цільову функцію, і пошук такої комбінації, яка приводить до її екстремального значення. Всі можливі комбінації аргументів при цьому утворять простір пошуку задачі, розмірність якого визначається числом аргументів цільової функції. А кожна з зазначених комбінацій утворює точку в даному просторі.

Слід відзначити, що цільова концепція при постановці задачі вибору може знаходити відображення як при визначенні множини допустимих альтернатив, так і при формулюванні вимог до отримання ефективних, зокрема строго оптимальних рішень.

Таким чином, метою задачі оптимізації як практичного, так і теоретичного характеру є вибір «найкращої» (припустимої або оптимальної) конфігурації з множини альтернатив для досягнення деякої поставленої мети. При цьому "найкращою" – в смислі забезпечення оптимуму (максимального або мінімального значення залежно від конкретної постановки задачі) заданої цільової функції при задоволенні певних обмежень.

Розклад навчальних занять – це документ, який пов'язує в єдину систему всі ланки і елементи навчального процесу і регламентує навчальну роботу професорсько-викладацького складу, студентів і навчально-допоміжного персоналу ВНЗ. Він складається навчальним відділом або навчально-методичним підрозділом факультету на семестр, затверджується ректором і доводиться до викладачів і студентів не пізніше, ніж за 10 днів до початку занять (зазвичай до початку семестру). У розкладі занять вказується група, дисципліна, вид заняття, час і місце проведення, прізвища викладачів.

До розкладу занять ставляться такі основні вимоги:

- точна і повна відповідність навчальному плану за обсягом, змістом, видом і часом занять, забезпечення закладених до навчальних планів і програм рівня сприймання матеріалу, систематичності і безперервності процесу навчання протягом дня та рівномірного розподілу навчальної роботи студентів впродовж тижня, місця і семестру;
- забезпечення на заняттях міжпредметних і внутрішніх логічних зв'язків кожної дисципліни, які визначаються її структурно-логічною схемою;
- забезпечення необхідних для самостійної роботи студентів часових інтервалів між лекціями і практичними видами занять з кожної дисципліни, чергування різних за рівнем складності дисциплін і видів занять протягом дня і тижня;
- дотримання принципів наукової організації праці викладачів і студентів, досягнення рівномірного навантаження викладачів з метою забезпечення їх підготовки до занять, планомірного проведення методичної, редакційно-видавницької і науково-дослідницької роботи (тут можуть враховуватися індивідуальні потреби викладача і рекомендації начальника кафедри);
- забезпечення ефективного використання аудиторного фонду і навчально-лабораторної бази.

Існують й інші вимоги ергономічного і організаційно-методичного характеру. До них, наприклад, відносяться розумне задоволення індивідуальних побажань викладачів, реалізація принципу індивідуального навчання, проведення занять з чисельними або малочисельними групами та інші. В ході наукових досліджень з методики складання розкладу занять за допомогою комп'ютера виявлено близько тридцяти обмежень, які необхідно враховувати при розробці оптимального розкладу.

Розв'язання задачі складання розкладів у загальній постановці є процесом виконання деякої фіксованої системи завдань за допомогою певної множини ресурсів чи обслуговуючих пристроїв [67]. При перенесенні загальної теорії розкладів на розклад навчальних занять формулювання задачі складання розкладу

занять звучить так: „Для заданого набору навчальних аудиторій (в даному випадку під навчальною аудиторією розуміється широке коло приміщень, в яких проводяться заняття (від комп'ютерної аудиторії до спортивного залу)) і заданого набору часових інтервалів (уроків або навчальних пар) побудувати такий розподіл навчальних занять для всіх об'єктів (вчителів і навчальних груп), для якого обраний критерій оптимальності є якнайкращим".

В роботі запропоновано створення такої математичної моделі розкладу занять у ВНЗ, яка дозволяла б ефективно (у задані терміни і з заданою мірою оптимальності) розв'язувати задачу автоматичного складання розкладу занять і була б гнучкою (властивість незначних структурних змін вихідної інформації) у випадку адаптації системи для розв'язання конкретної практичної задачі.

Для побудови математичної моделі розкладу занять у ВНЗ введемо змінні і визначимо обмеження. Нехай у ВНЗ є G навчальних груп, k – номер навчальної групи ($k = 1,...,G$). Заняття проводяться в робочі дні в півторагодинні інтервали, які називають парами. Позначимо t – номер робочого дня тижня ($t \in T$), де T – множина номерів робочих днів для групи k; j – номер пари ($j = 1,...,J$); J – загальна кількість навчальних пар. З кожною навчальною групою k протягом тижня у відповідності з навчальним планом проводиться S занять. Позначимо s – номер дисципліни в списку занять для групи k ($s = 1,...,S$). Передбачається, що заняття проводяться у всіх групах. Тоді, якщо по дисципліні протягом тижня проводиться більше одного заняття, ця дисципліна згадується в списку занять стільки разів, скільки їх передбачається навчальним планом для кожної групи. Нехай p – номер викладача ($p = 1,...,P$). Заняття кожного потоку можуть проводитися в наявному аудиторному фонді ВНЗ, який визначається до початку складання розкладу. Для того щоб повноцінно вести наукову, учбово-методичну роботу, готуватися до занять, викладач ВНЗ повинен мати вільний час. Ця умова не є достатньою, але є необхідною. Введемо вагові коефіцієнти, за допомогою яких повинен враховуватися статус викладача – його вчений ступінь і звання, займана посада, науково-суспільна

активність і т.п. Пріоритетне право викладачів на розподіл занять в алгоритмі складання розкладу визначається за методом ранжування посад .

Оптимально складений розклад навчальних занять не повинен змінюватися під час семестру чи навчального циклу, щоб не порушити враховані в розкладі обмеження та зв'язки між об'єктами. Планування послідовності проведення видів занять з дисципліни в навчальному процесі необхідно розглядати з боку студента і його психофізіологічних особливостей і розміщувати заняття так, щоб практичні заняття проводилися після лекцій, а лабораторні заняття – після практичних. Це підвищить ефективність використання навчального часу, що відводиться державним освітнім стандартом вищої професійної освіти на вивчення окремих дисциплін.

Основною задачею при складанні розкладу є планування і забезпечення методично правильного процесу вивчення усіх навчальних дисциплін навчального плану: їх взаємозв'язки, правильну послідовність і чергування усіх форм навчальної роботи з дисциплін на базі врахування можливостей студентів зі сприйняття і переробки навчальної інформації.

Ми існуємо в світі, де знання студентів оцінюють викладачі, ефективність роботи викладачів – адміністрація ВНЗ, роботу адміністрації – трудовий колектив і Міністерство освіти і науки і т.д., тобто система більш високого рівня ієрархії. Для оцінювання ефективності (якості) розкладу такими системами є колективи викладачів і студентів як деякі спільноти, що мають спільні інтереси та уподобання.

1.4 Постановка задачі дослідження та структурно-логічна схема

Яскравим прикладом задач багатокритеріальної оптимізації є задача складання розкладу занять у ВНЗ, яка відноситься до класу NP-повних. Як правило, вона розв'язується вручну людиною, що має великий досвід роботи в конкретному університеті. При складанні розкладу занять необхідно враховувати досить велику кількість критеріїв, що призначувана ступінь важливості яких має істотний вплив як на процес, так і на якість складеного розкладу. Процес скла-

дання розкладу занять у ручному режимі займає кілька тижнів. Безумовно, повністю повторити роботу людського мозку при розв'язанні багатокритеріальних задач поки неможливо. Однак це не означає, що автоматичні рішення будуть поступатися їм за якістю. Обчислювальні потужності сучасних комп'ютерів дозволяють за невеликий час синтезувати множину варіантів розкладів, які спираються різні показники.

Існує ряд факторів, що визначають вибір ручного способу складання розкладу навчальних занять.

До позитивних факторів було віднесено низьку собівартість такого способу, що не потребує додаткового витрачання на знаряддя праці та оплату роботи висококваліфікованого фахівця, а саме:

- інструмент праці залишається ручним: олівець, гумка, папір;
- завдання для виконання задачі по складанню розкладу є дуже складним, але додаткової оплати фахівцю за значне ускладнення та збільшення часу його роботи під час складання розкладу не передбачається;
- ручний спосіб складання розкладу дозволяє не оснащувати робоче місце керівника комп'ютерною технікою.

Отже, без витрат на знаряддя та маючи практично безкоштовну працю висококваліфікованого фахівця, навчальний заклад отримує розклад роботи на весь рік.

До позитивних факторів, що впливають на вибір ручного способу складання розкладу також можна віднести економію часу й зусиль на опанування нових технологій (комп'ютера та самої програми складання розкладу). Саме економією часу та зусиль керівники навчальних закладів переважно пояснюють своє небажання сідати за вивчення комп'ютерної техніки взагалі та спеціалізованих професійних програм зокрема.

До негативних факторів ручного вибору складання розкладу було віднесено такі:

- виснажлива ручна праця та нервове перевантаження;

– витрачання великої кількості часу при створенні одного, не найкращого варіанту розкладу;

– складності у використанні розкладу та маніпуляцій з ним під час роботи (наприклад, пошук замін);

– стагнація і зупинка професійного розвитку.

Отже, хоча і негативних факторів цього вибору було визначено більше, проте, низька собівартість та економія часу й зусиль на опанування нових технологій (два позитивних фактори) на практиці переважають у виборі ручного способу складання розкладу.

Проблема автоматизації складання розкладу навчальних занять – одна з найскладніших задач прикладної математики. Її розв'язання вимагає застосування нестандартних методів, оригінального творчого мислення і глибокого розуміння суті і складності проблеми.

Традиційно існує дві думки:

- багато програмістів впевнені, що цю задачу *легко* розв'язати. Проте на практиці більшість розробників створюють лише програми для ручного введення і редагування розкладів, оскільки їх автоматичні алгоритми не можуть побудувати справжній розклад: вони не справляються з реальними складнощами. Це видно навіть на прикладі демо-версій. Частка ручної праці в таких програмах складає 90-95%;
- диспетчери впевнені, що цю задачу неможливо вирішити. Так само вважають і деякі розробники, свідомо відмовляючись від автоматичного підходу. Пропоновані ними програми також засновані на ручній праці. Комп'ютер просто замінює лист ватману. Оскільки вибрати хороший продукт не так просто, багато навчальних закладів купують (зазвичай по рекламі) подібні програми. Але робота диспетчера при цьому не полегшується і якість роботи не підвищується. В результаті втрачається довіра до подібних програм.

Системний підхід у подоланні вищезазначених проблем полягає у формуванні логічної схеми дослідження ЗСР, як цілісної структури, яка буде визнача-

ти особливості проектування автоматизованої системи складання розкладів (АССР).

Серед тенденцій останнього часу, які мають відношення до теми дисертаційного дослідження, виділимо дві, на наш погляд, основні:

- використання сучасних інформаційно-аналітичних технологій розв'язку складних задач [12; 19], які могли б бути використані з метою зменшення людських втрат та матеріальних збитків від пожеж в умовах ресурсного та кадрового дефіциту;

- широке проникнення технологій Soft Computing [65, 108] у процеси розв'язування задач, пов'язаних із підтримкою прийняття рішень, що є особливо актуальним для ЗСР.

У відповідності до визначеної мети роботи подальше дослідження будується у відповідності із структурно-логічною схемою (рис. 1.2).

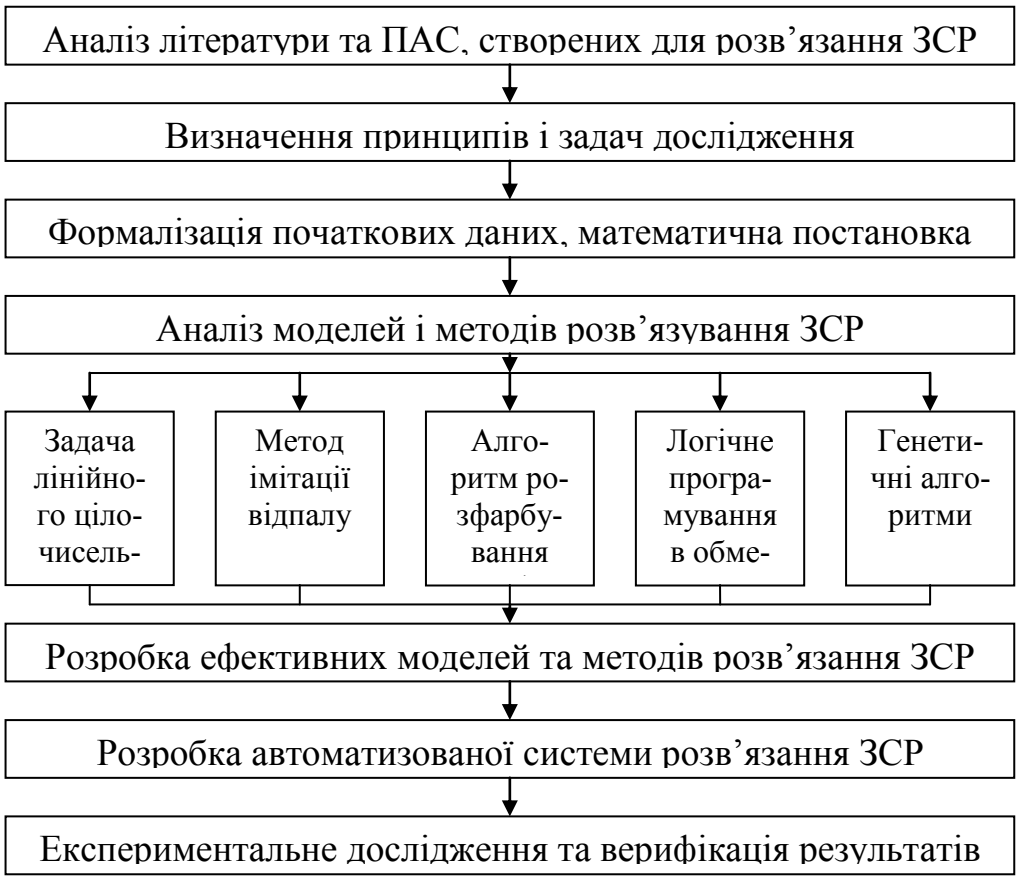


Рис. 1.2 – Структурно-логічна схема дослідження

На першому етапі виконуємо аналіз наукових літературних джерел, моделей і методів, які застосовувались для розв’язання ЗСР та програмно-алгоритмічних систем (ПАС). Виходячи з результатів аналізу формулюємо мету, принципи та задачі дослідження. Визначаємо інформаційні джерела, виконуємо попередній аналіз даних та здійснюємо формалізовані математичні постановки вказаних задач.

На другому етапі розробляємо еволюційний метод розв’язання ЗСР та алгоритм його застосування. Проводимо експериментальні дослідження та верифікацію отриманих результатів.

Висновки до розділу 1

Проблема підвищення якості підготовки фахівців у ВНЗ обумовлює необхідність постійного вдосконалення організації навчального процесу, зокрема, підвищення якості розв’язку основних задач, що регламентують підготовку фахівців: складання навчальних планів та формування розкладу занять. Це зумовлює необхідність розробки нових ефективних математичних моделей і методів розв’язання ЗСР навчальних занять.

У першому розділі визначено різні аспекти навчальної діяльності, пов’язані з автоматизацією НП. Зроблено аналіз наукових джерел, відомих програмних продуктів створення розкладу навчальних занять, а також представлені основні парадигми методів складання розкладу, кожна з яких має як сильні, так і слабкі сторони, тому вибір підходу для розв’язку кожної конкретної задачі визначається її специфікою.

Запропоновано принципи системного підходу у подоланні проблем, пов’язаних зі створенням розкладів навчальних занять. Побудовано структурно-логічну схему дослідження ЗСР, в основу якої покладено методологію системного аналізу.

Розділ 2. ЦІЛЬОВІ ФУНКЦІЇ ТА ВИД ПОТЕНЦІЙНИХ РОЗВ'ЯЗКІВ У ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДІВ

Проведений аналіз сучасного стану досліджень в галузі інформатики, теорії розкладів, теорії прийняття рішень, теорії штучного інтелекту та нечітких систем вказує на наявність важливої проблеми – теорія розкладів є визнаною дисципліною, в класичних роботах якої здійснено формальні постановки задач, що враховують достатньо різноманітні обмеження, в той же час безпосереднє розв'язання практичних задач за допомогою цих моделей виявляється неможливим як внаслідок їх великої розмірності та NP-складності, так і тому, що в практичних задачах необхідно враховувати не лише формальні, але і якісні критерії та обмеження.

Методика складання розкладу занять за допомогою комп'ютера перевірена у ряді ВНЗ. Виявлено, що при стабільній структурі навчального процесу застосовування комп'ютера дає відповідні позитивні результати. Але на практиці розклад занять поки що майже завжди складається вручну. Разом з тим, застосування комп'ютерів вже зараз значно полегшує працю упорядників розкладу занять і підвищує його якість.

У роботі розглянуто задачі, пов'язані з розподілом навантаження між викладачами, формуванням розкладу занять та розподілу ресурсів у ВНЗ. Представлені розробки скеровані на створення і вдосконалення методів та засобів розв'язання задач складання розкладу та розподілу ресурсів, впровадження процедур прийняття рішень на різних рівнях формування розкладу.

Для розв'язання вказаних задач пропонується використовувати методи еволюційного моделювання.

2.1 Домінуючий суб'єктивізм як принцип формування ефективного розкладу

Складання навчальних планів в більшості ВНЗ здійснюється вручну, що вимагає значних трудовитрат і часто здійснюється під впливом суб'єктивних переваг. Процес складання навчальних планів у ВНЗ, заснований на досвіді та інтуїції працівників вищої школи, потребує серйозного удосконалення та наукового обґрунтування прийнятих рішень.

Процес складання розкладу навчальних занять вимагає глибокого розуміння наукових основ навчання, структурно-логічних схем підготовки, діючих робочих навчальних планів і програм навчальних дисциплін, твердих знань численної інформації з усіх питань, які впливають на якість навчального процесу. На теперішній час процесу складання навчального розкладу занять притаманні такі характерні недоліки:

- відсутність погодженості між дисциплінами в семестрі та послідовність проведення занять;
- великі відриви в часі між проведенням практичних/лабораторних занять і відповідними лекціями;
- не враховується послідовність вивчення дисциплін, різних за складністю засвоєння та послідовність видів занять і методів роботи. Спостерігаються випадки, коли протягом одного дня плануються лише лекції, а інші дні зайняті тільки практичними/лабораторними заняттями;
- не завжди забезпечується достатня перерва між складанням заліків, іспитів, семінарів, практичних/лабораторних занять;

Трудомісткість складання розкладу зумовлена участю багатьох представників різних ланок управління, планування і забезпечення навчального процесу при підготовці, обробці і використанні великої кількості нормативної, навчальної, методичної документації тощо.

Тому великі часові витрати, помилки та суб'єктивізм є найсуттєвішими недоліками, що змушують звертатися до наукоємних інформаційних техноло-

гій, які в змозі звести до мінімуму існуючі негаразди шляхом автоматизації складання розкладу занять.

Огляд робіт, присвячених формалізованому складанню навчальних розкладів, показав, що автоматизоване складання навчального розкладу є складною комбінаторною задачею. Найчастіше при розв'язанні задачі автоматизованого формування розкладу занять використовуються евристичні алгоритми, а у випадках, коли застосовуються точні методи, наявні моделі не враховують цілий ряд істотних вимог, зокрема, умови безперервності вивчення дисциплін в різних семестрах, логічної послідовності вивчення дисциплін та ін. Евристичні методи не гарантують побудову допустимих і оптимальних розкладів. У багатьох системах передбачається можливість у разі тупикової ситуації почати наступну спробу побудови розкладу з самого початку зі зміненими вихідними даними. В інших випадках пропонується добудова частково згенерованого розкладу вручну. Необхідність коригування даних виникає також через порушення вимог, в зв'язку з неповнотою застосовуваної математичної моделі.

Сьогодні відома значна кількість досліджень, що пояснюють з певними припущеннями закономірності навчання для конкретних систем, у тому числі для формування необхідних показників навчального процесу [90]. Більшість досліджень враховують психофізіологічні властивості студента: забування, інерції, сприйняття та осмислення нової інформації тощо. При цьому вважається, що підвищення ефективності процесу навчання можна досягти лише вдосконаленням засобів управління на основі інформаційних технологій. Такий підхід не забезпечить необхідного результату без відповідної теорії управління. Урахування особливостей процесу навчання як керованого процесу вимагає його подання у вигляді відповідних математичних моделей об'єкта управління (моделі суб'єкта процесу навчання) і моделей управління, наприклад, на основі методів штучного інтелекту.

Однією з найбільш важливих особливостей навчального процесу у сучасному ВНЗ є його спрямованість на ефективну професійну підготовку майбутніх фахівців.

Якість підготовки будь-якого фахівця в сучасних умовах визначається не лише рівнем його знань, але й професійними вміннями, що дозволяють йому творчо вирішувати виникаючі проблеми, активно взаємодіяти з людьми на основі встановлення суб'єктних відносин. Система вузівської освіти з навчання фахівців повинна володіти широким набором засобів, що забезпечують розвиток умінь. Одним з найбільш важливих моментів у цьому є взаємодія між викладачем і студентом: беручи суб'єктні відносини і будучи їх активним учасником, студент починає сприймати реалізовані способи спілкування як норму, як свій індивідуальний вибір.

Тому важливим аспектом при взаємодії цих суб'єктів НП є створення сприятливих умов шляхом рівномірного розподілу навчального навантаження.

Навчальний процес у вищій школі – це система організації навчально-виховної діяльності, в основу якої покладено органічну єдність і взаємозв'язок викладання (діяльність викладача) і навчання (діяльність студента), спрямованих на досягнення цілей навчання, розвитку особистості студента, його підготовки до професійної діяльності.

Графік і план навчального процесу є основою для складання розкладу занять – важливого документу, яким регламентується академічна робота студентів і викладачів.

Під час складання розкладу навчальних занять необхідно враховувати:

- вимоги навчального плану;
- анатомо-фізіологічні і психологічні особливості учасників навчального процесу;
- можливості навчально-матеріальної бази ВНЗ;
- можливості аудиторного фонду;
- дидактичну доцільність віднесення навчальних дисциплін у розкладі на робочий тиждень і конкретний день.

При складанні розкладу виникає проблема оптимального управління ресурсами: викладацьким складом і аудиторним фондом. В процесі розв'язання

задачі необхідно враховувати обов’язкові обмеження, а також додаткові вимоги, які можуть порушуватися в деяких випадках.

Жорсткі обмеження – це обмеження, які повинні неодмінно задовольнятися; такі які фізично не можуть бути порушені. Приклад жорсткого обмеження: «В один і той же час у зазначеній аудиторії має бути заняття одного викладача з одного предмета». Очевидно, що крім загальних для всіх навчальних закладів такого типу обмежень, для кожного ВНЗ існують і свої жорсткі обмеження, викликані причинами різної природи. В результаті розв’язання задачі складання розкладу необхідно отримати розклад, що одночасно задовольняє всім жорстким обмеженням. Якщо це неможливо, то перелік таких обмежень необхідно змінити або провести деякі заходи, які дозволять отримати допустимий розклад.

М’які обмеження – це обмеження, які можна порушувати, але це порушення повинно бути зведене до мінімуму. Їхнє виконання не є таким же обов’язковим, як жорстких. На відміну від жорстких обмежень, що мають об’єктивну сутність, м’які обмеження мають об’єктивно-суб’єктивний і суб’єктивний характер. Так, обмеження «Лекція проводиться не в лабораторії» має об’єктивно-суб’єктивний характер, а обмеження «Доцент Горошко А. читає лекції в понеділок і вівторок» – суб’єктивний. Очевидно, що порушення м’яких обмежень призводить до погіршення розкладу, але не позбавляє його допустимості. Оскільки таких порушень може бути багато і вони мають різнобічний характер, то актуальність отримання допустимого оптимального (прийнятного) розкладу є незаперечною.

Наведемо жорсткі та м’які обмеження які необхідні для процесу складання розкладу.

Жорсткі обмеження:

- викладач не може бути присутнім на двох заняттях одночасно;
- в одній аудиторії не можуть проводитися різні заняття одночасно;

- місткість аудиторії не може бути меншою, ніж кількість присутніх у ній студентів;
- деякі заняття вимагають особливих аудиторій (наприклад комп'ютерних класів, лабораторій, спортзалів та інше);
- одна група не може вивчати декілька дисциплін на одному занятті одночасно;
- тижневе навантаження для студентів бакалаврату становить 30 годин;
- тижневе навантаження для студентів-спеціалістів 24 годин;
- тижневе навантаження для студентів-магістрів становить 18 годин;
- кількість занять на день не більше 4.

М'які обмеження:

- викладачі можуть віддавати перевагу конкретним часовим інтервалам;
- викладачі можуть віддавати перевагу певним аудиторіям;
- уникнення проміжних періодів в розкладі (вікон);
- рівномірний розподіл занять протягом тижня;
- місткість аудиторій не повинна бути набагато більшою ніж кількість присутніх студентів;
- більш складні заняття потрібно проводити на 1-2 парі;
- фізичне виховання рекомендується проводити 3-4 парою.

Крім визначення жорстких та м'яких обмежень самої задачі, шляхом опитування студентів і викладачів було визначено, що близько 98% опитуваних вважають, що розклад повинен складатись, в першу чергу, для студентів (співвідношення важливості суб'єктів НП: 60% – для студентів, 40% – для викладачів). Виходячи з цього, при розробці математичних моделей та цільових функцій будемо враховувати спочатку переваги студентів, а потім викладачів.

2.2 Інтегральна цільова функція та критерій ефективності розкладу навчальних занять

Кожний ВНЗ має свою специфіку і тому моделі, які містять критеріальні функції та обмеження, мають певні відмінності. Разом із тим, очевидно, що раціональне розв'язання задачі складання розкладів базується на формуванні інтегральної функції, яка є певною згорткою окремих критеріальних функцій. Інтегральний критерій може бути адитивною функцією із ваговими коефіцієнтами, що визначаються експертами, він також може бути залежністю, структурна та параметрична ідентифікація якої здійснюватиметься на базі статистичної інформації. У будь-якому випадку, знаходження оптимуму такого критерію є складною науковою задачею, яка, у повному обсязі, залишається нерозв'язаною.

Основними суб'єктами, для яких складається розклад, є студенти та викладачі. Дуже важливо створити розклад, який задовольняв би їх вимоги. Найкращим способом цього досягти є використання цільових функцій окремо для викладачів і студентів. Тоді з'являється наступна проблема – як визначити пріоритети між даними функціями.

Традиційно для розв'язання задачі складання розкладу навчальних занять із застосуванням ГА використовують спрощений варіант цільової функції [22, 128, 138], яка на низькому рівні визначає ступінь придатності хромосоми до формування наступного покоління. Такі цільові функції враховують виконання лише жорстких обмежень, накладених на розклад.

Для організації оптимізуючого процесу необхідно визначити цільову функцію, чи, у термінах ГА, фітнес-функцію. Зазвичай, в її якості використовується адитивний показник оптимальності, що базується на штрафах. Перевагою такого вибору є можливість настройки алгоритму під конкретну задачу шляхом варіювання коефіцієнтів і, тим самим, зміни пріоритетів при пошуку оптимального розкладу.

На відміну від досліджень, які проводились раніше, пропонуємо вважати критеріальною функцією визначення оптимальності розкладу його

відповідність запитам студентів і викладачів. Оскільки у такому випадку присутні суб'єктивні оцінки в композиції з об'єктивними даними, та формування такої функції є слабко структурованою задачею. Здійснимо її формалізацію.

Позначимо $R = \{r_1, r_2, ..., r_n\}$ – скінченну множину можливих розкладів. Її скінченність гарантована скінченністю множини навчальних дисциплін $P = \{p_1, p_2, ..., p_m\}$, сукупності викладачів $L = \{l_1, l_2, ..., l_k\}$ та навчальних аудиторій $A = \{a_1, a_2, ..., a_v\}$. Задача формування оптимального розкладу з урахуванням вищенаведеного полягає у знаходженні

$$\max_{r \in R} F(r), \tag{2.1}$$

$$r \in \Omega(P, S, L, A) \in R, \tag{2.2}$$

де Ω – сукупність обмежень, які визначаються наявністю і спеціалізацією аудиторій, розподілом викладачів по дисциплінах, дисциплін по аудиторіях тощо. Виходячи з інтересів двох основних суб'єктних множин навчального процесу – студентів і викладачів, цільову функцію F вважатимемо векторною і представлятимемо як $F = \{F_S, F_L\}$, де F_S – цільова функція студентів, F_L – цільова функція викладачів.

Тоді задача (2.1)-(2.2) трансформується в таку:

$$\max_{r \in R} F_S(r), \ \max_{r \in R} F_L(r), \ r \in \Omega(P, S, L, A). \tag{2.3}$$

Згідно з теорією прийняття рішень, найчастіше задачу (2.3) розв'язують, використовуючи метод головного критерію, максимінний метод і метод адитивної згортки. У відповідності до методу головного критерію потрібно з множини критеріїв визначити основний, знайти його оптимум за умови, що значення інших критеріїв будуть не меншими деяких констант. Тоді задача (2.3) набуде виду:

$$\max_{r \in R} F_S(r), \ F_L(r) \geq d_L, \ r \in \Omega(P, S, L, A), \tag{2.4}$$

де d_L – константа, значення якої залежить від розмірності задачі, даних і встановлюється особою, що приймає рішення.

За іншим, максимінним критерієм, необхідно знайти такий критерій оптимальності, який має найменші значення на підмножині розкладів Ω_1, що задовольняють умову (2.2), а далі серед розкладів $r \in \Omega_1$ знайти такий, якому відповідає максимальне значення іншого критерію. Задача (2.3) запишеться так:

$$\max_{r\in\Omega_1} \min_{i\in\{S,L\}} F_i(r),\ \Omega_1 \subset \Omega. \tag{2.5}$$

Максимінний критерій дозволяє здійснити й іншу інтерпретацію. Зокрема, можна знайти підмножину розкладів, які є допустимими та яким відповідає мінімальне значення одного з критеріїв, і на цій підмножині знайти розклад, який є оптимальним за іншим критерієм. Задача (2.3) тоді буде такою:

$$r_* = \arg\min_{r\in\Omega} F_S(r) \in \Omega_1,\ r^* = \max_{r\in\Omega_1} F_L(r), \tag{2.6}$$

або

$$r_* = \arg\min_{r\in\Omega} F_L(r) \in \Omega_1,\ r^* = \max_{r\in\Omega_1} F_S(r), \tag{2.7}$$

Очевидно, що розв'язуючи задачі (2.6)-(2.7), орієнтовані на «найгірший» випадок і визначають гарантовану нижню оцінку для F_S або F_L. Максимінний критерій у порівнянні з методом головного критерію є більш детермінованим.

2.3 Визначення коефіцієнтів цільової функції відображення суб'єктивних переваг учасників навчального процесу

Третій критерій, який може бути використаний для розв'язання задачі складання розкладів, записується у вигляді адитивної згортки

$$\alpha_S F_S(r) + \alpha_L F_L(r) \to \max,\ r \in \Omega(P,S,L,A). \tag{2.8}$$

Вагові коефіцієнти α_S і α_L визначають вплив компонента цільової функції на загальний результат. Встановлення значень коефіцієнтів є емпіричною процедурою. Її формалізація може здійснюватись різними способами. Наведемо

один з них. Вважаючи студента домінуючим суб'єктом у вищому навчальному закладі (ВНЗ), апріорі раціонально встановити $\alpha_S = 0{,}6$, $\alpha_L = 0{,}4$. При подальшій корекції використати таке правило: якщо відношення кількості студентів до кількості викладачів відповідає нормативному значенню, то значення коефіцієнтів не змінюються, якщо реальне відношення відрізняється від нормативного, то α_S і α_L потрібно коригувати.

Позначимо N_S – кількість студентів у ВНЗ, N_L – кількість викладачів, Nom – номінальне значення відношення кількості студентів до кількості викладачів, визначене керівною інстанцією. Тоді, якщо виконується нерівність

$$\frac{1}{2}Nom \le \frac{N_S}{N_L} \le Nom, \text{ то } \alpha_S = 0{,}6 - \frac{1}{2}Nom\left(\frac{N_S}{N_L} - \frac{1}{2}Nom\right)\cdot 0{,}4 \quad (2.9)$$

в іншому випадку, якщо

$$Nom \le \frac{N_S}{N_L} \le \frac{3}{2}Nom, \text{ то } \alpha_S = 0{,}6 + \frac{1}{2}Nom\left(\frac{N_S}{N_L} - Nom\right)\cdot 0{,}3. \quad (2.10)$$

Одержуючи вирази (2.9) і (2.10), припускаємо, що реальне співвідношення між кількістю студентів та кількістю викладачів

$$\frac{N_S}{N_L} \in \left[\frac{1}{2}Nom, \frac{3}{2}Nom\right]. \quad (2.11)$$

Так, якщо нормативне значення відношення $\frac{N_S}{N_L} = 10$, то вважатимемо, що на практиці $\frac{N_S}{N_L} \in [5;15]$, і це відповідає дійсному стану речей. При $\frac{N_S}{N_L} > Nom$, очевидно, що кількість студентів є порівняно більшою і їх пріоритет у цільовій функції має збільшуватись. Тому, у цьому випадку $\alpha_S \in [0{,}6;0{,}9]$, а $\alpha_L \in [0{,}1;0{,}4]$. У протилежному випадку, при $\frac{N_S}{N_L} < Nom$, вважатимемо, що $\alpha_S \in [0{,}2;0{,}6]$ і $\alpha_L \in [0{,}4;0{,}8]$. Таким чином, використання виразів (2.9)-(2.10) дозволить адекватно реагувати на динаміку контингенту викладачів і студентів.

Розглянемо аспекти формування цільової функції F_S. Очевидно, що студентів можна розглядати як певну множину, розбиту на класи (групи по курсах і спеціальностях), які мають певні інтереси і переваги з однієї сторони, та як сукупність індивідів, кожен з яких має свій погляд на оптимальність розкладу та його аспектів. Крім того, існують певні нормативні вимоги, яких необхідно дотримуватись, формуючи розклад.

Кожен ВНЗ, а інколи й окремі факультети встановлюють певні обмеження на розклад занять. Як приклад, можна навести такі:

$Z_1^H = \{$*Заняття мають відбуватись у всі робочі дні*$\}$,

$Z_2^H = \{$*Кожен студент не може мати більше трьох пар на день*$\}$,

$Z_3^H = \{$*Кількість лекцій для студентської групи не повинна перевищувати дві на день*$\}$,

$Z_4^H = \{$*Лекційні заняття не проводяться в лабораторіях*$\}$ тощо.

Оскільки студенти мають індивідуальні переваги, які у тому числі залежать від курсу та групи, то раціонально ці індивідуальні переваги звести до групових переваг шляхом визначення їх сукупності, опитування студентів та інтелектуального аналізу його результатів.

Необхідно зауважити ще деякі аспекти формування F_S. По-перше, якщо існуючі вимоги є директивними та їх навіть незначне недотримання неможливе, то вважатимемо їх обмеженнями, що визначають область допустимих розкладів (розв'язків). Позначимо такі обмеження Z^d. Якщо ж існують вимоги, які можуть бути певним чином з різних причин недотриманими, то відповідні обмеження позначимо Z^v і ступінь порушення обмеження врахуємо шляхом додавання (віднімання) штрафних значень.

Припустимо, що $Z^d = \{Z_1^d, Z_2^d, ..., Z_k^d\}$, $Z^v = \{Z_1^v, Z_2^v, ..., Z_l^v\}$, де Z_i^d – директивні вимоги, Z_j^v – вимоги, які визначаються студентами, $i = \overline{1,k}$, $j = \overline{1,l}$. Вимоги із множини Z^d є апріорі відомими і визначеними директивними документами чи нормативами. Інші ж вимоги із Z^v не є заданими і їх визначення є однією з пе-

рших задач, розв'язання якої потрібне для формування ефективного розкладу. Пропонуємо такий спосіб. Студенти незалежно один від одного формують свої вимоги до розкладу. Далі усі вимоги вносяться до єдиного переліку. Якщо в ньому зустрічаються подібні вимоги – їх об'єднують. В іншому випадку авторам протилежних вимог пропонують дійти до узгодженого варіанту вимоги або вилучити свої вимоги взагалі. У випадку їх незгоди серед студентів проводиться голосування і та вимога, за яку проголосувала більшість, залишається у списку вимог, інша – вилучається.

Таким чином одержується множина вимог Z^v. Очевидно, що вимоги Z^v є різнопріоритетними для кожного студента. Для того, щоб узгодити індивідуальні переваги, кожному студенту надається можливість визначити перевагу кожної з вимог. Для цього за методом аналізу ієрархій [120] будують матриці попарних порівнянь, для яких знаходять максимальні власні числа та відповідні власні вектори. Нехай $\lambda_{\max}$ – максимальне власне число матриці попарних порівнянь, яку одержав i-й студент, $i=\overline{1,m}$, $x^i = (x_1^i, x_2^i, ..., x_l^i)$ – власний вектор, який відповідає цьому власному числу. Виконавши нормування елементів цього вектора за формулою

$$x_j^{iн} = \frac{x_j^i}{\sum_{j=1}^{l} x_j^i}, \tag{2.12}$$

одержимо, що $x_j^{iн} \in (0,1)$ і $\sum_{j=1}^{l} x_j^{iн} = 1$. Тоді можна стверджувати, що значення $x_j^{iн}$ буде вказувати на пріоритет j-го критерію для i-го студента. Оскільки всі студенти є рівнозначними (рівнокомпетентними), то пріоритетність вимог для їх колективу (найчастіше, групи) визначається як середнє значення пріоритетів критеріїв для кожного студента, тобто

$$x_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{iн}, \ j=\overline{1,l}. \tag{2.13}$$

Таким чином, для групи студентів одержано вектор пріоритетів вимог $X = (x_1, x_2, ..., x_l)$.

Без обмеження загальності, перша частина цільової функції (2.8) набуде виду:

$$\alpha_s \cdot F_s = \alpha_s \cdot \sum_{j=1}^{l} x_j \cdot \chi(A_j^v), \quad (2.14)$$

де коефіцієнт α_s визначається за (2.9) або (2.10), пріоритети x_i за формулою (2.13), функція $\chi(A_j^v)$ є індикатором і записується так:

$$\chi(A) = \begin{cases} 1, \textit{якщо } A \textit{ виконано}, \\ 0, \textit{в іншому випадку}. \end{cases} \quad (2.15)$$

де A – певна вимога.

У виразі (2.14) потрібно детальніше описати обмеження функції-індикатора (2.15). Але перш ніж це зробити, розглянемо особливості формування цільової функції викладачів F_L. Порівнюючи з процесом формування F_S, зауважимо, що викладачі не мають групових переваг і врахування їх вимог необхідно здійснювати індивідуально. Нехай M – кількість викладачів, які розбиваємо на групи:

- завідувачі кафедр;
- доктори наук, професори;
- доктори наук, доценти;
- кандидати наук, доценти;
- кандидати наук без вченого звання;
- старші викладачі;
- асистенти.

Очевидно, що на практиці потрібно враховувати пріоритети представників таких груп. Якщо перераховані не всі групи викладачів, то можна додати потрібні групи в перелік. Таким чином, нехай $\exists K$ груп викладачів, тобто $T = (T_1, T_2, ..., T_K)$. За пріоритетністю та порядковою шкалою:

$$T_1 \phi\, T_2 \phi \ldots \phi\, T_K. \qquad (2.16)$$

Водночас схема переваг (2.16) не дозволяє здійснювати кількісні оцінки. Тому особа, що приймає рішення, а це може бути проректор з навчальної роботи, фахівець-диспетчер, який складає розклади і знає міру впливу викладачів зазначених груп, формує матрицю попарних порівнянь: визначає пріоритети викладачів, представників груп

$$y = \{y_1, y_2, \ldots, y_K\}, \quad y_i \in (0,1), \quad \sum_{i=1}^{K} y_i = 1, \qquad (2.17)$$

Якщо адекватність такої процедури викликає сумніви, то її можна об'єктивізувати на основі аналізу посадових окладів представників груп.

Кожен викладач має свої переваги для формування розкладу, причому кількість таких переваг у різних викладачів буде різною. Позначимо $Z_i^T = \{Z_{i_1}^{T_j}, Z_{i_2}^{T_j}, \ldots, Z_{i_{n_i}}^{T_j}\}$ – вектор переваг i-го викладача з j-ї групи, n_i – кількість його елементів, $j = \overline{1, M}$, $i = \overline{1, K}$. Векторам Z_i^T відповідатимуть розраховані з використанням методу, наведеного вище, значення вектора пріоритетів $D_i^j = \{d_{i_1}^j, d_{i_2}^j, \ldots, d_{i_{n_i}}^j\}$, $i = \overline{1, M}$, $j = \overline{1, K}$. Тоді, другий доданок цільової функції (2.8) буде таким:

$$\alpha_L F_L = \alpha_L \sum_{j=1}^{K} y_j \cdot \sum_{i=1}^{M} \chi\{L_i \in T_j\} \cdot \sum_{l=1}^{n_i} d_{il}^j \cdot \chi\{Z_{il}^{T_j}\}, \qquad (2.18)$$

де $\chi\{L_i \in T_j\} = \begin{cases} 1, \textit{якщо викладач } L_i \textit{ належить групі } T_j, \\ 0, \textit{в іншому випадку.} \end{cases}$

Таким чином, з урахуванням (2.14) і (2.18), задача (2.8) перепишеться так:

$$F(r) = \alpha_S \sum_{j=1}^{l} x_j \chi\{Z_j^v\} + \alpha_L \sum_{j=1}^{K} y_j \sum_{i=1}^{M} \chi\{L_i \in T_j\} \sum_{l=1}^{n_i} d_{il}^j \cdot \chi\{Z_{il}^{T_j}\} \to \max, \qquad (2.19)$$

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

Зауважимо, що:

- на цьому етапі обмеження (2.2) залишаються без змін;

- розклад r явно в моделі та задачі (2.19) не присутній;
- оскільки значення α_S, α_L, x_j, y_j, d_{il}^j є відомими, то максимум цільової функції залежатиме від значень функцій-індикаторів, які вказують на виконання (невиконання) вимог студентів та викладачів.

Висновки до розділу 2

У другому розділі виконані математичні постановки задач дослідження.

Проведено аналіз моделей та методів, які могли б бути використані при розв'язанні задачі складання розкладів занять. У зв'язку із тим, що частина з них базуються на ітераційній техніці пошуку оптимальних рішень, застосування яких вимагає виконання повного перебору отриманих результатів, в якості базису дослідження вибрано еволюційне моделювання, складовими якого є генетичні алгоритми.

Визначено, що серед усіх еволюційних парадигм для розв'язання задачі теорії розкладів найбільш адекватно використовувати генетичні алгоритми.

Виконано порівняльний аналіз точності та швидкості класичних методів розв'язання ЗСР та методів, що базуються на еволюційному моделюванні, який показав, що результати останніх є значно точнішими.

Розділ 3. ЕВОЛЮЦІЙНА ТЕХНОЛОГІЯ СКЛАДАННЯ РОЗКЛАДІВ З ВИКОРИСТАННЯМ НЕЧІТКИХ ВИСНОВКІВ

Математична теорія нечітких множин (fuzzy sets) і нечітка логіка (fuzzy logic) є узагальненнями класичної теорії множин і класичної формальної логіки. Ці поняття були вперше запропоновані американським ученим Лотфі Заде (Lotfi Zadeh) в 1965 р. [47]. Основною причиною появи нової теорії стала наявність нечітких і наближених міркувань при описі людиною процесів, систем, об’єктів.

Перш ніж нечіткий підхід до моделювання складних систем отримав визнання у всьому світі, пройшло не одне десятиліття з моменту зародження теорії нечітких множин. І на цьому шляху розвитку нечітких систем прийнято виділяти три періоди.

Перший період (кінець 60-х – початок 70 рр.) характеризується розвитком теоретичного апарату нечітких множин (Л. Заде, Э. Мамдані, Беллман). У другому періоді (70-80-і роки) з’являються перші практичні результати в області нечіткого управління складними технічними системами (парогенератор з нечітким управлінням). Одночасно стала приділятися увага питанням побудови експертних систем, побудованих на нечіткій логіці, розробці нечітких контролерів. Нечіткі експертні системи для підтримки прийняття рішень знаходять широке застосування в медицині і економіці. Нарешті, в третьому періоді, який триває з кінця 80-х років і триває нині, з’являються пакети програм для побудови нечітких експертних систем, а сфери застосування нечіткої логіки помітно розширюються. Вона застосовується в автомобільній, аерокосмічній і транспортній промисловості, в області виробів побутової техніки, у сфері фінансів, аналізу і ухвалення управлінських рішень та багатьох інших.

3.1 Обґрунтування використання еволюційного моделювання в задачі складання розкладу навчальних занять

Одним із перспективних напрямків розвитку методів оптимізації, який на сьогодні інтенсивно розвивається, є еволюційне моделювання [94, 95, 96]. Цей напрямок охоплює математичні методи, в яких закладені принципи природних механізмів прийняття рішень, що забезпечують ефективну адаптацію флори та фауни до навколишнього середовища протягом мільйонів років [149]. Найбільш ефективним визнане використання еволюційних методів для розв’язання тих задач оптимізації, для яких застосування інших методів за якихось причин є неефективним.

Однак не дивлячись на те, що як самостійний науковий напрямок еволюційне моделювання склалося ще в 90-х роках минулого сторіччя, на сьогодні загально визнаного визначення поняття «еволюційне моделювання» не існує, як і не існує однозначного критерію, за яким той чи інший метод оптимізації може бути віднесений до класу еволюційних.

Так, одні науковці пов’язують еволюційне моделювання з теорією еволюції Ч. Дарвіна, а до еволюційних методів відносять виключно генетичні алгоритми [73], інші науковці вважають за доцільне розширити перелік еволюційних методів за рахунок тих, які не пов’язані з поглядами Ч. Дарвіна, однак засновані на природних алгоритмах, зокрема, це стосується методу моделювання відпалу [66] та мурашиних алгоритмів [128]. Окрім того існує думка, що методи ройового інтелекту не належать до еволюційних [137].

Такі розбіжності підтверджують висновок про те, що на сьогодні не існує єдиного підходу до визначення переліку методів, які можна віднести до класу еволюційних, а це відповідно свідчить і про відсутність узгодженого погляду на сутність самого поняття «еволюційне моделювання».

Така неузгодженість поглядів стосовно еволюційного моделювання як напрямку наукових досліджень призводить до стримування його розвитку і не надає змоги надати відповідь на запитання, чим же природні алгоритми оптимізації принципово різняться від алгоритмів математичного програмування, хоча

відповідь на нього може відкрити нові шляхи для удосконалення існуючих та розробки нових ефективних методів оптимізації.

Окремого розгляду потребує і питання стосовно класифікації методів, які можуть бути віднесені до класу еволюційних. Необхідність такої класифікації пояснюється декількома причинами: по-перше, систематизація знань про існуючі еволюційні алгоритми сприяє виявленню тих специфічних ознак, завдяки яким методи еволюційного моделювання забезпечують ефективне розв'язання складних оптимізаційних задач; по-друге, наявність класифікації дозволяє визначати шляхи удосконалення сучасних методів оптимізації; по-третє, така класифікація сприяє обґрунтованості вибору методу оптимізації для розв'язання конкретної задачі з урахуванням її специфіки.

На сьогодні група методів, які можуть розглядатись на відповідність ознакам еволюційності, складається з: генетичних алгоритмів [66; 73], моделювання відпалу [66], LARES [22], оптимізації роєм часток [138], мурашиною колонією [149; 138], колонією бактерій [9; 138] та роєм бджіл [138]. Причому останні чотири методи належать до методів ройового інтелекту, в основу якого покладені алгоритми поведінки соціальних істот [149; 138].

Аналіз особливостей цих методів, на нашу думку, надасть можливість уточнити поняття «еволюційне моделювання» та здійснити класифікацію еволюційних методів оптимізації.

Виходячи з аналізу наведених вище підходів до визначення особливостей, які відрізняють еволюційні методи від інших методів оптимізації, можна стверджувати, що на сьогодні не викликають сумніву лише два положення.

По-перше, всі еволюційні методи засновані на результатах вивчення процесів, що відбуваються в природі, і, по-друге, серед усіх природних процесів до еволюційних належать лише ті, які мають ознаки оптимізаційних.

При цьому під оптимізаційними ознаками розуміється досягнення природною системою такого стану, характеристики якого надають підстави визначити його принципові відмінності від усіх інших можливих станів.

Розглянемо з цієї точки зору деякі природні процеси, при цьому почнемо з процесу селекції, який є прообразом генетичних алгоритмів.

Як відомо, в основі селекції знаходиться ідея отримання кращих (за деякою ознакою) нащадків за рахунок відбору для схрещування кращих (за цією ознакою) особин з наявної популяції. Результатом багаторазового повторення цих операцій є закріплення у нащадків обраної ознаки та подальше формування однорідної за цією ознакою популяції. Яскравим прикладом практичних результатів застосування цієї ідеї є селекційна робота з отримання квітів форми та кольорової гами, які не є домінуючими у вихідній популяції.

Другим є приклад формування зграй птахів у період міграції, характерним для якого є те, що зграя як структура формується в певний період року з окремих особин, які раніше гніздувались та харчувались окремо.

Третім можна навести приклад утворення зграї вовків, особливістю якої є формування лише за складних погодних умов та суттєвої обмеженості звичної кормової бази.

Четвертим можна розглянути процеси пошуку та доставки їжі в колоніях мурах та роях бджіл. Для цих біологічних видів характерним є соціальна організація, відповідно до якої пошук їжі здійснюється розвідниками, а її доставка – фуражирами. При цьому розвідники досліджують місцевість, знаходять їжу та передають інформацію про неї фуражирам. Механізм передавання інформації у цих видах є різним (мурахи використовують феромони, а бджоли – танок), однак результат є однаковим – фуражири рухаються до джерела їжі цілеспрямовано, при цьому мурахи ще й забезпечують мінімальну довжину маршруту доставки.

Останнім прикладом може бути формування колонії бактерій, характерним для якого є те, що під впливом хімічного складу середовища форма цих колоній може утворювати правильні геометричні фігури, зокрема, подібні до сніжинок.

Зрозуміло, що наведені приклади не охоплюють всієї різноманітності природних систем, однак вже їх достатньо для того, щоб виділити деяку спільну ознаку, яка стала основою для розвитку еволюційного моделювання.

Для всіх розглянутих прикладів спільною ознакою є те, що деякий показник, який за одних умов є імовірнісним, під дією деякого впливу втрачає ознаки хаотичності та набуває ознак впорядкованості.

Зокрема, хаотичне забарвлення квітів під дією селекції перетворюється на однорідне, хаотичне розташування в просторі птахів та вовків під дією природних факторів призводить до їхньої локалізації в обмеженій ділянці простору, хаотичний рух розвідників за рахунок передавання здобутої інформації перетворюється в цілеспрямований рух фуражирів, хаотичне розташування бактерій під дією хімічних речовин набуває геометричних форм, близьких до правильних.

Тобто для природних систем, які стали основою для розробки різних еволюційних методів, загальним є перехід від хаотичного стану до впорядкованого внаслідок впливу зовнішніх факторів.

Однак подібні переходи мають місце не лише в природних, але й у фізичних процесах. Фізичними прикладами таких переходів є утворення регулярної структури у в’язкій рідині внаслідок критичної різниці температур між її верхніми та нижніми шарами (ефект Бенара) та виникнення лазерного випромінювання внаслідок досягнення критичного рівня енергії накачування [54, 60]. При цьому необхідно зазначити, що впорядкування елементів системи внаслідок досягнення зовнішнім фактором деякого критичного рівня притаманне дисипативним системам – відкритим термодинамічним системам, які перебувають далеко від рівноваги [54].

Той факт, що природні системи є дисипативними є достатньо відомим [146]. Однак з аналізу такої подібності між фізичними та розглянутими природними системами випливає запитання про те, який же спільний фактор в природних системах досягає критичного рівня та є причиною їх переходу від хаотичного до впорядкованого стану.

Для відповіді на це запитання більш детально розглянемо процеси пошуку та доставки їжі, які відбуваються в мурашиній колонії.

Відповідно до [138] особливістю мурах є те, що під час руху кожна з них залишає на поверхні землі феромони (особливі хімічні речовини, які мають специфічний запах), які з часом випаровуються, а для доставки їжі фуражири частіше обирають ті маршрути, на яких запах феромону сильніший. На всіх маршрутах руху концентрація феромону буде залежати від давнини його прокладення та кількості мурах, які пройшли по ньому. Коротший маршрут до джерела їжі на початку буде мати більшу концентрацію феромону внаслідок скорішого повернення розвідника, а потім ця концентрація буде підтримуватись рухом фуражирів. Маршрути, які прокладені, але не використовуються, зникають внаслідок випаровування феромону.

З викладеного зрозуміло, що формально фактором, який регулює рух мурах, є феромон, однак залишаються питання про те, яку фундаментальну фізичну величину він уособлює, та про те, що за критичний рівень є регулятором дисипативної системи.

Фундаментальними фізичними величинами, які характеризують стан дисипативної системи, є кількість та потоки речовини, енергії, ентропії та інформації, при цьому утворення впорядкованості пов’язується з обов’язковим експортом ентропії за межі системи та з наявністю малих флуктуацій стану окремих елементів системи [54].

Аналогічні механізми можна знайти і в інших природних системах, які стали основою для створення наведених вище методів оптимізації, тому всі ці методи можуть бути віднесені до еволюційних. При цьому і самі еволюційні методи оптимізації містять механізм впорядкування первинного випадкового розподілу пробних точок в просторі незалежних змінних.

Так, під час застосування генетичних алгоритмів та методів ройового інтелекту про це свідчить стягування вихідної популяції в одну точку, в методі LARES первинний хаотичний розподіл молекул по рівням енергії та пристроям штучного хімічного реактора в кінцевому підсумку трансформується в локалі-

зацію всіх молекул в пристрої сепарації з мінімальним значенням потенційної енергії, а в методі моделювання відпалу первинний хаотичний розподіл положення пробної точки в просторі незалежних змінних наприкінці перетворюється в одну точку.

Виходячи з наведеного, можна запропонувати такі визначення.

Еволюційне моделювання – це математичний метод моделювання процесу переходу системи від хаотичного до впорядкованого стану.

Еволюційний метод – це математичний метод, в основу якого покладені положення термодинаміки відкритих систем, далеких від рівноваги.

Еволюційний метод оптимізації – це математичний метод, в якому умови вихідної задачі оптимізації асоціюють з характеристиками дисипативної системи, в якій забезпечується експорт ентропії внаслідок критичної різниці між створюваною інформацією та інформацією, яка забувається.

Існуючі на сьогодні еволюційні методи оптимізації різняться за природними системами, поведінка яких покладена в їх основу, та за особливостями реалізації відповідних природних алгоритмів. Такі відмінності і можуть бути покладеними в основу їх класифікації.

Зрозуміло, що першою ознакою доцільно обрати галузь науки, формалізація знань якої покладена в основу того чи іншого методу. За цією ознакою сучасні методи еволюційного моделювання можуть бути класифіковані на фізичні, хімічні та біологічні.

Другою ознакою можна обрати спосіб, у який формується нова комбінація незалежних змінних під час пошуку оптимуму. За цією ознакою всі методи поділяються на дві групи: блочні та покоординатні. В блочних методах нова точка пошуку утворюється з окремих груп змінних, в той час як в покоординатних методах нова точка пошуку утворюється варіюванням значення по кожній незалежній змінній окремо.

Третьою ознакою класифікації оберемо обсяг інформації про цільову функцію, який використовується для вибору напрямку руху для ефективного пошуку оптимуму. За цією ознакою розглянуті вище еволюційні методи можна поді-

лити на прямі та квазіградієнтні. В прямих методах на кожному кроці використовується лише значення цільової функції, а в квазіградієнтних методах (до яких належать методи ройового інтелекту) внаслідок обміну знаннями між окремими агентами враховується додаткова інформація про рельєф поверхні, яка відповідає цільовій функції. Фактично такий обмін надає деяку оцінку значення градієнта, тому наступна точка обирається з урахуванням отриманої оцінки напряму покращення значення цільової функції.

Зрозуміло, що наведена класифікація не є вичерпною і в подальшому може бути уточнена. Окрім того, подальший напрям досліджень вбачається в удосконаленні сучасних методів еволюційного моделювання на основі визначених класифікаційних ознак.

Вважається, що історія еволюційного моделювання або еволюційних обчислень почалася з робіт Дж. Холланда, Л. Фогеля, А. Оуена і М. Уолша, І. Букатової, Л. Растригіна та інших дослідників [149]. Усі вони взяли за основу ряд перетворень живого матеріалу, існуючих в природі, спростили їх, побудувавши ряд принципів і моделей еволюційних процесів. З часом еволюційне моделювання перетворилося на теорію, на основі якої здійснюється пошук квазіоптимальних і оптимальних розв’язків задач, деякі з них вважалися до цього нерозв’язними.

Основна перевага еволюційного моделювання полягає у можливості розв’язання задач, що мають багато локальних оптимумів за рахунок комбінування елементів випадковості і спрямованості аналогічно тому, як це відбувається в природі. Іншим важливим чинником еволюційного моделювання є моделювання процесів селекції, розмноження і успадкування. При цьому одержувані за певними правилами альтернативні розв’язки можуть породжувати нові розв’язки, які будуть „наслідувати” кращі характеристики попередніх. Суть стратегії еволюційного моделювання полягає в реалізації цілеспрямованого процесу розмноження-загибелі, при якому розмноженню відповідає поява нових об’єктів, а загибелі – видалення об’єктів у відповідності з визначеними критеріями природного та штучного добору.

Сьогодні, згадуючи про еволюційне моделювання, найчастіше мають на увазі генетичні алгоритми (ГА). ГА використовують механізми природної еволюції, що у загальному вигляді можуть бути сформульовані так: чим вища пристосованість індивіда, тим вища імовірність того, що у його нащадків ця пристосованість буде виражена ще сильніше.

Головна перевага ГА полягає у тому, що вони дозволяють розв'язувати складні задачі, для яких ще не розроблені стійкі і прийнятні методи, особливо на етапі формалізації і структуризації системи, в когнітивних системах. ГА ефективні в комбінації з іншими класичними алгоритмами, евристичними процедурами, а також у тих випадках, коли про множину розв'язків є деяка додаткова інформація, що дозволяє настроювати параметри моделі, корегувати критерії відбору, еволюції [149].

Одним з головних етапів функціонування ГА є оптимізація цільової функції. Зазвичай, в її якості використовується адитивний показник оптимальності, що базується на штрафах, які встановлюються кожному розв'язку за будь-який конфліктний момент у розкладі [73]. Перевагою такого вибору є можливість настройки алгоритму під конкретну задачу шляхом варіювання коефіцієнтів і, тим самим, зміни пріоритетів під час пошуку оптимального розкладу.

Представимо формування розкладу як певний еволюційний процес. Його відображенням є модифікація та удосконалення автоматизованої системи складання розкладів (АССР). Процес моделювання еволюції АССР є складною слабкоструктурованою проблемою, тому що не існує строго визначеної процедури, методу, алгоритму її вирішення, а також визначення рівнів ієрархії. Такі системи, як відомо, є предметом вивчення у штучному інтелекті. Оскільки еволюція системи є процесом переходу із стану в стан у залежності від внутрішньої необхідності та зовнішніх умов, то організація запам'ятовування інформації, яка складається із вихідних даних, обмежень, моделей, методів, засобів, критеріїв оцінки одержаних рішень, становить основу для аналізу ефективності кожного етапу життєвого циклу системи та прогнозування майбутніх процесів і

прийняття рішень. Таким чином, моделювання еволюції є більш інформативним процесом ніж моделювання систем [66].

Важливо зауважити, що моделювання еволюції систем у переважній більшості випадків необхідно здійснювати на початкових етапах життєвого циклу. Верифікацію та оптимізацію результатів пропонується здійснювати за допомогою еволюційного моделювання.

Хід еволюції визначається спадковою змінністю, яка є передумовою еволюції; боротьбою за існування як контролюючим та направляючим фактором; природним відбором як перетворюючим фактором. Адаптуючи вказані аксіоми до проблеми еволюції АССР, зауважимо, що еволюційне моделювання у цьому випадку повністю виправдане, оскільки:

1. Спадкова змінність вказує на можливість АССР створювати кращий розклад (удосконалювати існуючий розклад), що і визначає еволюційні передумови.

2. Контролюючим та направляючим фактором еволюції АССР є виконання заданих обмежень задачі і визначає необхідність розв’язання задач оптимізації.

3. Природний відбір у цьому випадку визначається ефективністю результату розв’язання задачі.

Виконання наведених трьох умов свідчить про наявність моделі еволюції.

Об’єктом, що еволюціонує, є сама АССР. Критерієм еволюції є ефективність АССР, структурним елементом (розв’язком) є розклад. Таким чином, ми одержали три складові еволюційного моделювання: модель еволюції, об’єкт еволюції і критерій еволюції.

Таким чином, використання стратегій, принципів, методів і алгоритмів еволюційного моделювання при розв’язанні оптимізаційних задач є якісно новим перспективним підходом до створення і застосування складних штучних систем. Даний підхід дозволяє отримувати ефективні рішення в різних галузях науки і техніки за прийнятний час і з необхідною якістю.

3.2 Особливості формування структури розв’язку та початкової популяції

Першим кроком при розробці математичної моделі є розробка структури хромосоми, в якій буде зберігатися розв’язок. У нашому випадку такою „хромосомою” є розклад. Структура хромосоми повинна враховувати всі особливості й обмеження шуканого розв’язку.

Розглянемо будову хромосоми. Хромосома є набором генів (фрагментів розкладу (ФР)). Хромосома є динамічним масивом, кожним елементом якого є ФР. Один фрагмент в хромосомі – це одне заняття в розкладі. Наш ФР матиме структуру, що складається з чотирьох частин закодованої інформації:

- перша частина – день тижня;
- друга – номер навчальної пари;
- третя – номер групи, викладача і предмету;
- четверта – номер аудиторії.

Розглянемо приклад, де початковими даними для складання розкладу занять є: 5 днів тижня; 5 пар на день; 20 навчальних груп; 40 викладачів; 80 предметів; 100 аудиторій.

ФР, зазвичай, є бітовим рядком фіксованої довжини, на кодування якого (в нашому випадку) відводиться 32 біта, а саме: на кодування першої частини фрагмента відводиться 3 біти (5 днів тижня), другої – 3 біти (5 пар в день), третьої – 18 біт (20 груп, 40 викладачів, 80 предметів), четвертої – 7 біт (100 аудиторій). І таких фрагментів буде 500 (5 днів по 5 пар у 20 групах), тобто хромосома матиме 16000 біт. Наприклад, один фрагмент матиме вигляд: 010 011 10010 011000 0101110 1100001.

Якщо проаналізувати описане кодування, то можна побачити ряд недоліків, які, в свою чергу, істотно впливають на швидкість, а іноді й на збіжність алгоритму. По-перше, при такому кодуванні сусідні числа можуть відрізнятися в значеннях декількох бітів, так, наприклад, числа 7 (0111) і 8 (1000) у бітовому представленні розрізняються в 4-х позиціях, що ускладнює функціонування генетичного алгоритму і збільшує час його збіжності. Для того, щоб уникнути ці-

її проблеми можна використовувати кодування, при якому сусідні числа відрізняються меншою кількістю позицій, наприклад, використовуючи код Грея [9], але він також не є універсальним. По-друге, кодування такого роду призводить до невиправданої інформаційної надлишковості. На прикладі це можна пояснити так: використовуючи вищезгадані вихідні дані, для кодування першої частини ФР (дня тижня) нам знадобиться 3 біти, якими можна закодувати 7 чисел, але даних у нас менше – 5. Така надлишковість позначається на швидкості збіжності алгоритму, адже постійно потрібно перевіряти значення ФР на належність інтервалу. По-третє, якщо виникає потреба за значенням «гена» визначити значення відповідної йому ознаки, то попередньо закодовану інформацію розкодовують, щоб отримати числові або нечислові дані для виводу результатів, на що також витрачається процесорний час.

Тому пропонуємо інший метод представлення ФР: усі дані, необхідні для складання розкладу, представити у кодованому десятковому вигляді. Це дещо спростить кодування, адже, по-перше, не потрібно буде кодувати інформацію нечислових даних в числові, числові дані в бінарні, бінарні дані в код Грея, а, по-друге, потім усі ці дії проводити у зворотному порядку.

Початкова інформація про викладачів, групи, заняття та аудиторний фонд зберігається у таблицях зовнішньої бази даних і кодується автоматично при введенні. Отже, кодування частин ФР буде наступним: на кодування першої частини фрагмента відводиться 1 знак (дні тижня лежать в діапазоні [1;5]), другої – 1 знак (кількість пар – [1;5]), третьої – 6 знаків (20 груп, 40 викладачів, 80 предметів – [1..20;1..40;1..80]), четвертої – 3 знаки (аудиторії – [1;100]). При такому кодуванні ФР буде містити в собі 11 знаків, а хромосома складатиметься з 500*11=5500 знаків. Тобто, десяткове кодування дає можливість скоротити наш масив приблизно в 3 рази. Наприклад, вищезображений ФР в десятковому кодуванні буде таким: 1 3 18 24 46 097,

де 1 – день тижня (понедіок); 3 – номер пари (третя пара); 18 – код номера групи, 24 – порядковий номер викладача, 46 – номер предмета; 097 – номер ауди-

торії. При розкодуванні цього ФР, частина розкладу буде мати наступний вигляд (рис. 3.1).

	КТ-101	КТ-102	КТ-103
ПН, 8:30	Політологія Єрмілов Є.П. 408-1	Релігієзнавство Гудима І.П. 105-2	Інф. сист. в економ.
ПН, 10:00	Теорія алг. і мат. основи представл. знань Гамоц	ОАПСОіС Оксамитна Л.П. 313-2	Системотехніка Тимч
ПН, 11:50	ОАПСОіС Оксамитна Л.П. 313-2	Системотехніка Тимченко А.А. 103-4	Інф. сист. в економ.
ПН, 13:20	Ін.мова Скорик Л.М. 712-1	ОСАОПК Тимченко А.А. 105-2	
ПН, 14:50	Мат. моделі в розрах. на ЕОМ Стеценко І.В. 132-2		
ВТ, 8:30	Інфрм. мереж. технології Гамоцька С.Л. 307-2	Автоматиз. сист. керув. Сєркова Л.Е. 414-4	Інфрм. мереж. техно

Рисунок 3.1 – Візуалізація частини розкладу в розкодованому вигляді.

Далі з хромосом формують початкову популяцію. Популяція – досить велике співтовариство організмів, що схрещуються між собою. Популяції характеризуються набором ланцюжків генів кожного з об’єктів – хромосом, сукупність яких визначає генофонд популяції [138].

Для створення нового покоління з кращими показниками із сукупності хромосом необхідно вибрати кращі для подальшої репродукції. Сам процес відбору та репродукції хромосом здійснюється так: з поточної популяції випадково вибирається пара хромосом і з них визначається краща (таких хромосом буде значно менше загальної кількості хромосом в популяції), причому функція відбору вказує на хромосому лише один раз, що дає можливість замінювати найгірші хромосоми на кращі в наступній популяції. Щоб отримати нову популяцію, відібрані хромосоми попарно схрещують між собою.

Оператори кросовера характеризуються здатністю до руйнування (disruption) батьківських хромосом. Кросовер для цілочисельного кодування вважається більш руйнівним, якщо в результаті його застосування відстань по Хеммінгу [85] між хромосомами нащадків і хромосомами батьків велика. Іншими словами, здатність цілочисельного кросовера до руйнування залежить від того, наскільки сильно він „перемішує” (рекомбінує) вміст батьківських хромосом. Так, одноточковий кросовер вважається слабкоруйнуючим, а однорідний кросовер в більшості випадків є сильноруйнуючим оператором. Відповідно, двоточковий кросовер по руйнуючій здатності займає проміжну позицію по відношенню до одноточкового і однорідного кросоверам.

Для кросовера для дійсного кодування здатність до руйнування визначається тим, наскільки велика відстань в просторі пошуку між точками, що відповідають хромосомам батьків і нащадків. Таким чином, руйнуючий ефект двоточкового кросовера залежить від вмісту батьківських хромосом.

Відзначимо, що одночасно зі здатністю до руйнування говорять також про здатність до створення (creation, construction) кросовером нових особин. Тим самим підкреслюється, що, руйнуючи хромосоми батьків, кросовер може створити абсолютно нові хромосоми, що не зустрічалися раніше в процесі еволюційного пошуку.

Саме тому в задачі складання розкладів навчальних занять було обрано однорідний кросовер в якості оператора схрещування. Це характеризується тим, що при більшій руйнації структури хромосоми майже відпадає можливість використання мутації, принаймні ймовірність виникнення мутації можна звести до мінімуму.

Наш фрагмент розкладу представлений 4-ма частинами, кожна з яких – неподільна, але це не заважає нам точки кросоверу розміщувати не за границями фрагментів, а саме між частинами фрагмента. Отже, в даній задачі використовується однорідний кросовер [119]: хромосоми розбивають на частини, причому фрагмент теж розбивається на частини, і кожна ділянка хромосоми першого батька має 50-відсотковий шанс обмінятися з відповідною ділянкою хромосоми другого батька. Наприклад:

батьки: **2 3 18 24 46 097**

5 4 35 98 42 020

нащадки: 5 4 **18 24 46 097**

2 3 35 98 42 020

Далі представлена схема процедури „Crossover”, що використовується в еволюційній технології розв’язання задачі складання розкладів навчальних занять у вищому навчальному закладі (рисунок 3.2).

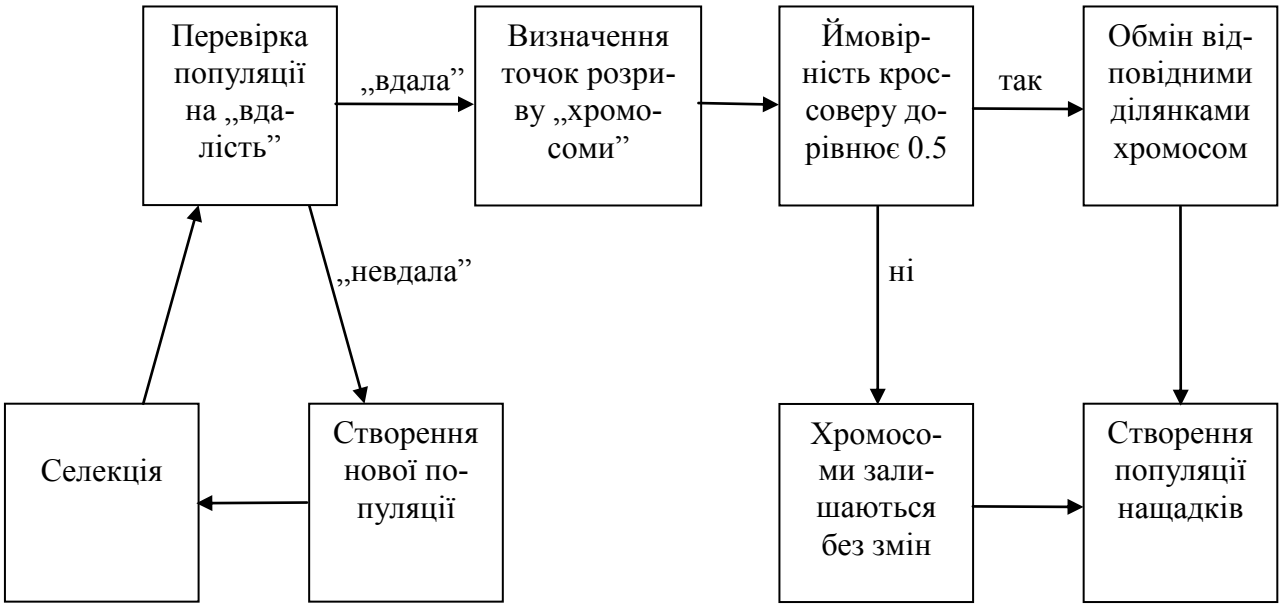


Рис. 3.2 – Схема процедури „ Crossover ”

Після кросоверу до популяції застосовують оператор мутації.

Застосування еволюційних алгоритмів при розв’язанні задачі складання розкладів пов’язане з необхідністю врахування «прокляття розмірності», оскільки середньостатистична кількість груп в середньостатистичному вищому навчальному закладі становить від 50 до 200. Зауважимо, що в таких алгоритмах виконується значна кількість операцій, які не ведуть до знаходження оптимального розв’язку, і їх практичне тестування і реалізація наштовхується на проблеми, пов’язані з часом виконання.

Скоротити кількість операцій можна, використовуючи деякі додаткові процедури, що оптимізують процес пошуку розв’язку. Певним чином вони пов’язані з реалізацією оператора мутації.

Зазвичай при реалізації ГА спочатку застосовують оператор схрещування, а потім оператор мутації, хоча можливі й інші варіанти. Існує думка, що оператор мутації є основним пошуковим оператором і відомі алгоритми, які не використовують інших операторів (кросинговер, інверсію тощо), крім мутації.

Дослідження показали, що в простих задачах, використовуючи ГА з мутацією (і без кросовера), знаходять розв’язок швидше. Також для такого методу потрібно менший розмір популяції. Якщо мають місце складні багатоекстрема-

льні функції, то краще використовувати ГА з кросовером, оскільки цей метод більш надійний, хоча і вимагає більшого розміру популяції.

Використовуючи теорему шаблонів, можна стверджувати, що мутація лише шкодить зростанню кількості представників прийнятних шаблонів, оскільки зайвий раз їх руйнує. Однак мутація просто необхідна для ГА з малим розміром популяції. Сутність цього твердження в тому, що для нечисленних популяцій властива передчасна збіжність (premature convergence). Це та ситуація, коли в деяких позиціях все індивіди мають один і той же біт, але такий набір бітів не відповідає глобальному екстремуму. При цьому кросовер практично не змінює популяції, тому що всі індивіди майже однакові. В цьому випадку мутація здатна інвертувати «застряглий» біт у одного з індивідів і знову розширити простір пошуку.

В якості оператора мутації найбільшого поширення набули випадкова і нерівномірна мутація Михалевича [27].

При випадкової мутації ген, який підлягає зміні, приймає випадкове значення з інтервалу своєї області визначення. У нерівномірній мутації значення випадково вибраного гена хромосоми або зменшується, або збільшується на деяке значення δ, яке залежить від номера поточного покоління так, що нащадок задовольняє обмеженням задачі:

$$c_i^* = \begin{cases} c_i + \delta(t, b_i - c_i), & при \quad \chi = 0, \\ c_i - \delta(t, c_i - a_i), & при \quad \chi = 1, \end{cases} \tag{3.1}$$

$$\delta(t, y) = y\left(1 - r^{\left(1 - \frac{t}{\varepsilon_{\max}}\right)^b}\right), \tag{3.2}$$

де χ – випадкове бінарне число, t – номер поточного покоління, b і a – допустимі права і ліва границі i-ї змінної, r – рівномірно розподілене випадкове число, $\varepsilon_{\max}$ – кількість поколінь, параметр b відображає ступінь залежності від номеру покоління. Функція $\delta(t, y)$ повертає значення в діапазоні [0, y], яке прямує до нуля зі збільшенням номеру поточного покоління.

Крім того, якщо протягом досить великого числа поколінь не відбувається збільшення пристосованості, то застосовуються «мала» і «велика» мутації покоління. При «малій» мутації покоління до всіх особин, крім 10% кращих, застосовується оператор мутації. При «великій» мутації кожна особина або мутує, або замінюється на випадково згенеровану. Число поколінь до реалізації «малої» і «великої» мутації під час роботи алгоритму є постійним. Також оператор мутації можна застосовувати тільки в тому випадку, якщо до даної пари батьківських особин не був задіяний оператор схрещування. Можна вибирати кілька точок в хромосомі для інверсії, причому їх число також може бути випадковим.

При збільшенні ймовірності мутації і при зменшенні впливу результатів відбору (наприклад, за рахунок використання інших стратегій відбору) розмноження представників пристосованих особин сповільнюється, але зате відбувається інтенсивний пошук інших особин. І навпаки, зменшення ймовірності мутації і збільшення впливу відбору веде до інтенсивного використання знайдених хороших особин, але тоді менше уваги приділяється пошуку нових [21].

У задачі складання розкладів розв’язком є оптимальний, в деякому сенсі, варіант розкладу. Оператор мутації призначений для модифікації одного з таких варіантів. Оскільки початкові розклади генеруються випадковим чином, то така генерація не обов’язково буде забезпечувати пошук по всьому простору розв’язків. Оператор мутації призначений саме для того, щоб внести в популяцію випадкові розв’язки, до яких важко або взагалі неможливо прийти, застосовуючи тільки оператор схрещування.

Ймовірність мутації залежить від самої задачі, але, зазвичай, має досить мале значення (від ≈0,001 до ≈0,01 [11, 12]). Оскільки в результаті мутації генеруються нові розв’язки і відсутня внесення хаосу в популяцію, в дослідженні використовується ймовірність виникнення мутації менше 0,01.

Традиційно використовувалася мутація, що базується на розігруванні випадкової величини, що має рівномірний розподіл. Результати проведених експериментів вказують на те, що «рівномірний» вибір фрагмента хромосоми і

«рівномірна» його мутація не спрямовані на скорочення часу пошуку оптимального розв'язку, оскільки однакові шанси для модифікації мають як «перспективні», так і «неперспективні» розв'язки. Причому мутація в перших може призводити до появи «неперспективних» розв'язків, а в других – до появи «перспективних», що вказує на випадковий ненаправлений характер пошуку оптимального розв'язку.

Зменшимо кількість «помилкових» кроків алгоритму, використовуючи замість рівномірного розподілу – нормальний. Реалізація такого підходу має такі етапи. Спочатку визначають ділянку потенційних розв'язків, яка є однаковою в кожній з них і якій відповідає максимальне або близьке до нього значення функції належності. Нагадаємо, що фенотип цієї ділянки є дійсне число, що належить інтервалу, який вказується на початку процесу кодування елементів початкової вибірки або початкової популяції розв'язків [50, 51]. На наступному етапі розігрується випадкове число, що має нормальний розподіл із середнім, що відповідає фенотипу ділянки, і середньоквадратичним відхиленням, таким, щоб інтервал $(m-3\sigma, m+3\sigma)$, де m – фенотип відповідної ділянки, σ – середньоквадратичне відхилення, що збігається з апріорним інтервалом зміни фенотипу. Цей інтервал грає певну роль при розрахунку величини мутації. Відзначимо також, що середньоквадратичне відхилення в процесі пошуку рішення змінюється, а саме, зменшується, що вказує на те, що в міру наближення до оптимального або квазіоптимального рішень ймовірність значних мутацій зменшується. Результати експериментів вказують на 30-50% скорочення часу пошуку рішення в порівнянні з використанням рівномірного розподілу.

Після реалізації мутації цільова функція перераховується.

Реалізація оператора мутації здійснюється за допомогою наступної процедури:

```
procedure mutation;
var i,j,n: Integer;
  temp1: array [3..5] of AnsiString;
  temp2: array [3..5] of AnsiString;
begin
  new_pop:='';
```

```
        new_hrom1:='';
        new_hrom2:='';
        for i:=1 to (5*5*t_g) do for j:=1 to 6 do for n:=1 to 20 do y[j,i,n]:='';
        for n:=1 to a do begin
         for j:=1 to (5*5*t_g) do begin
          for i:=1 to 2 do begin
           if Random<0.01 then rez1[i,j,n]:=IntToStr(random(5)+1);
          end;
          if Random<0.01 then begin

temp1[3]:=IntToStr(random(t_g)+1);
           if StrToInt(temp1[3])<10 then temp1[3]:='0'+temp1[3];

temp1[4]:=IntToStr(random(t_v)+1);
           if StrToInt(temp1[4])<10 then temp1[4]:='0'+temp1[4];

temp1[5]:=IntToStr(random(t_d)+1);
           if StrToInt(temp1[5])<10 then temp1[5]:='0'+temp1[5];
           rez1[3,j,n]:=temp1[3];
           rez1[4,j,n]:=temp1[4];
           rez1[5,j,n]:=temp1[5];
          end;
          if Random<0.01 then begin

rez1[6,j,n]:=IntToStr(random(t_a)+1);
           if StrToInt(rez1[6,j,n])<10 then rez1[6,j,n]:='0'+rez1[6,j,n];
           if (t_a>=100) then
            if StrToInt(rez1[6,j,n])<100 then rez1[6,j,n]:='0'+rez1[6,j,n];
          end;
         end;
        for j:=1 to (5*5*t_g) do begin
         for i:=1 to 2 do begin
          if Random<0.01 then rez2[i,j,n]:=IntToStr(random(5)+1);
          end;
          if Random<0.01 then begin

temp2[3]:=IntToStr(random(t_g)+1);
           if StrToInt(temp2[3])<10 then temp2[3]:='0'+temp2[3];

temp2[4]:=IntToStr(random(t_v)+1);
           if StrToInt(temp2[4])<10 then temp2[4]:='0'+temp2[4];

temp2[5]:=IntToStr(random(t_d)+1);
           if StrToInt(temp2[5])<10 then temp2[5]:='0'+temp2[5];
           rez2[3,j,n]:=temp2[3];
           rez1[4,j,n]:=temp2[4];
           rez1[5,j,n]:=temp2[5];
          end;
          if Random<0.01 then begin

rez2[6,j,n]:=IntToStr(random(t_a)+1);
           if StrToInt(rez2[6,j,n])<10 then rez2[6,j,n]:='0'+rez2[6,j,n];
           if (t_a>=100) then
            if StrToInt(rez2[6,j,n])<100 then rez2[6,j,n]:='0'+rez2[6,j,n];
          end;
        end;
```

```
        for i:=1 to (5*5*t_g) do                    end;
          for j:=1 to 6 do begin
                                                  new_pop:=new_pop+new_hrom1+new
new_hrom1:=new_hrom1+rez1[j,i,n];                 _hrom2;
                                                    end;
new_hrom2:=new_hrom2+rez2[j,i,n];                   p_p:='';
            y[j,i,2*n-1]:=rez1[j,i,n];              p_p:=new_pop;
            y[j,i,2*n]:=rez2[j,i,n];              end;
```

Таким чином, помістивши початкову популяцію у створене штучне середовище і реалізувавши процеси селекції, кросоверу і мутації, ми одержимо ітераційний алгоритм пошуку оптимального розв’язку, на кожній ітерації якого виконуються такі дії:

1. Кожен індивід популяції оцінюється за значенням фітнес-функції.
2. Кращі розв’язки (зазвичай, близько 5%) копіюються у нову популяцію без зміни. Такий принцип (принцип елітизму) запобігає втраті кращих розв’язків і забезпечує підвищену збіжність алгоритму.
3. На основі пропорційного відбору з поточної популяції вибираються два розв’язки, що піддаються рекомбінації. Для цього хромосоми батьків обмінюються відповідними ділянками.
4. Отриманий у попередньому пункті розклад може виявитися некоректним. У цьому випадку можна повторювати операцію рекомбінації доти, поки не буде отриманий коректний розклад, але краще передбачити евристичні механізми виправлення розкладу.
5. Якщо нова популяція сформована, то стара вилучається, після чого переходимо до п.1. В іншому випадку переходимо до п.3.

Розглянутий алгоритм є не лише стійким до локальних мінімумів, але й завдяки внутрішньому паралелізму, вираженому в роботі не з окремими розв’язками, а з цілими класами розв’язків, забезпечує відносно швидкий пошук оптимального розв’язку.

3.3 Модифікований метод розв'язання задачі складання розкладу з використанням штрафної функції

Як зазначалось вище, для оцінки придатності хромосом використовується цільова функція – адитивний показник оптимальності, що базується на штрафах.

Відомо, що методи штрафних функцій відносяться до групи прямих методів розв'язання задач нелінійного програмування [142]. Вони перетворюють задачу з обмеженнями в послідовність задач безумовної оптимізації деяких допоміжних функцій. Останні отримують шляхом модифікації цільової функції за допомогою функцій-обмежень таким чином, щоб обмеження в явному вигляді в задачі оптимізації не фігурували. Це забезпечує можливість застосування методів безумовної оптимізації. У загальному випадку допоміжна функція має вигляд:

$$F(x,a) = f(x) + \Phi(x,a), \tag{2.20}$$

де $f(x)$ – цільова функція задачі оптимізації; $\Phi(x,a)$ – «штрафна» функція; параметр $a > 0$. Точку безумовного мінімуму функції $F(x,a)$ будемо позначати через $x(a)$.

Залежно від виду $\Phi(x,a)$ розрізняють методи внутрішніх штрафних, або бар'єрних, функцій і методи зовнішніх штрафних функцій.

Методи внутрішніх штрафних функцій застосовуються для розв'язання задач нелінійного програмування з обмеженнями-нерівностями [69]. У цих методах функції $\Phi(x,a)$ підбирають такими, щоб їх значення необмежено зростали при наближенні до границі допустимої області G (рис. 2.1). Іншими словами, наближення до границі «штрафується» різким збільшенням значення функції $F(x,a)$. На границі G будується «бар'єр», що перешкоджає порушенню обмеження в процесі безумовної мінімізації $F(x,a)$. Пошук мінімуму допоміжної функції $F(x,a)$ необхідно починати з внутрішньої точки області G. При цьому в процесі оптимізації траєкторія спуску ніколи не вийде за межі допустимої об-

ласті. Всі перераховані властивості функції $\Phi(x,a)$ визначили найменування даної групи методів.

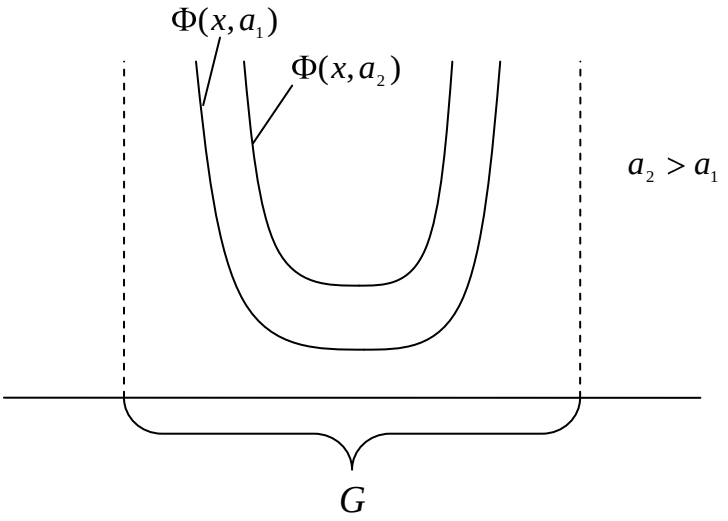


Рис. 2.1 – Внутрішня штрафна функція

Методи зовнішніх штрафних функцій застосовуються для розв'язання задач оптимізації в загальній постановці, тобто за наявності як обмежень-нерівностей, так і обмежень-рівностей. У цих методах функції $\Phi(x,a)$ вибирають такими, що їх значення рівні нулю всередині і на границі допустимої області G, а поза нею позитивними і зростають тим більше, чим сильніше порушуються обмеження (рис. 2.2). Таким чином, тут «штрафується» віддалення від допустимої області G.

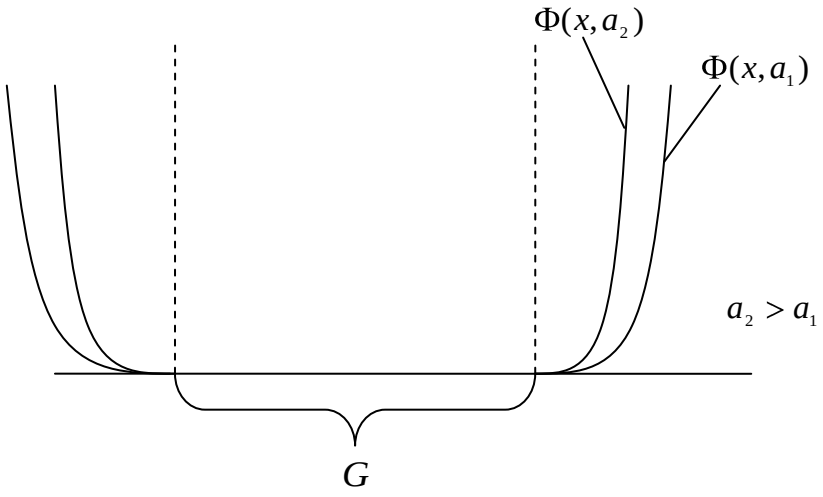


Рис. 2.2 – Зовнішня штрафна функція

Аналіз методів штрафних функцій дозволяє зробити наступні висновки про їх обчислювальні властивості. Відповідно до методів внутрішніх штрафних функцій ведуть пошук розв'язку, не виходячи за межі допустимої області. Це дуже важливо в тих випадках, коли цільова функція або обмеження не визначені за межами допустимої множини. Крім того, перервавши обчислення в будь-який момент часу, ми завжди отримаємо допустиме рішення. Однак для завдання в якості початкової деякої допустимої точки іноді потрібно розв'язувати задачу, яку за складністю можна порівняти з вихідною задачею нелінійного програмування. У цьому випадку метод зовнішніх штрафних функцій краще, оскільки він забезпечує розв'язання з будь-якої початкової точки.

Далі розглянемо модифікований метод розв'язання задачі складання розкладу з використанням штрафної функції. Він має такі кроки:

Крок 1. Визначити структуру S потенційного розв'язку r.

Крок 2. Визначити критерій E зупинки пошуку оптимального рішення.

Крок 3. Виконати кодування потенційного рішення.

Крок 4. Поки не виконаний критерій E:

Крок 4.1. Поки вибірка потенційних рішень Z неповна.

Крок 4.1.1. Згенерувати потенційне рішення r.

Крок 4.1.2. Якщо воно не допустимо ($r \notin \Omega_1(P,S,L,A)$ то перейти на крок 4.1.1.

Крок 4.1.3. Якщо рішення прийнятне ($r \in \Omega_2(P,S,L,A)$), то записати його у вибірку Z і перейти на крок 4.1.

Крок 4.1.4. Якщо рішення r неприйнятне хоча б по одному з обмежень з $\Omega_2(P,S,L,A)$, то реалізувати один з трьох варіантів:

A: Перейти на крок 4.1.1.

B: Якщо варіант A виконано більше $A_{\max}$ разів, то перейти до варіанту C.

C: Покласти

$$F(r) = \alpha_S F_1(r) + \alpha_L F_2(r) - \beta_{it}\varphi(F_1(r) \vee F_2(r)), \qquad (2.21)$$

де β – ваговий коефіцієнт, it – номер ітерації, $\varphi(*)$ – функція штрафу. Вважати r потенційним розв'язком і перейти на крок 4.1.1.

Крок 4.2. Для всіх потенційних рішень обчислити значення цільової функції $F(*)$, враховуючи те, що якщо рішення допустимо и прийнятне, то значення функції штрафу $\varphi(*) = 0$.

Крок 4.3. Виконати генерацію нових потенційних рішень, виходячи з значень цільової функції, застосовуючи операції кросовера і мутації (якщо методом оптимізації є генетичний алгоритм) або з використанням нормально розподілених чисел, якщо це еволюційна стратегія. У разі EvoMax застосовуються і інші технологічні елементи.

Крок 5. Обчислення значення критерію E.

У запропонованому методі застосовується штрафна функція. Звичайно, якби всі допустимі рішення були б прийнятними, необхідності в її використанні не було. Але в переважній більшості випадків на практиці це не так. Якщо розклад складається диспетчером вручну, то обов'язково знаходиться хоча б один викладач, який вважає розклад категорично для себе неприйнятним, і диспетчер починає вносити зміни в розклад, намагаючись задовольнити його вимоги і, звичайно ж, порушуючи переваги інших учасників освітнього процесу. Штрафом у цьому випадку є витрачений час і, найчастіше, моральна шкода. Крім того, немає об'єктивного показника для порівняння якості розкладів.

Але повернемося до задачі (2.19). Оскільки $\alpha_S > 0$, $\alpha_L > 0$ за визначенням і $F_1(*) \geq 0$, $F_2(*) \geq 0$ за будовою, то функція $F(*)$ в найгіршому випадку може приймати нульове значення, що відповідає допустимому, але неприйнятному рішенню для кожного з викладачів і для кожної студентської групи. Якщо вважати пріоритети студентських груп і викладачів заданими, то отримання оптимального значення $F_{\max}$ буде відповідати отриманню оптимальних значень $F_{1\max}$ і $F_{2\max}$. На практиці переваги груп студентів і викладачів найчастіше є антагоністичними, тому отримання значення $F_{\max} = \alpha_S F_{1\max} + \alpha_L F_{2\max}$ є бажаним, але недосяжним орієнтиром.

Зазвичай студентські групи є рівнопріорітетними, тому найбільш традиційний спосіб збільшити значення цільової функції – це першочергове виконання побажань і вимог викладачів з великими ваговими коефіцієнтами (ректори, проректори, завідувачі кафедрами і т.п.). Визначати вагові коефіцієнти викладачів пропонується методом аналізу ієрархій (МАІ).

Метод аналізу ієрархій, розроблений відомим американським математиком Томасом Сааті [106], з успіхом використовується для розв’язання багатьох практичних задач на різних рівнях планування. Цей метод набув широкого розповсюдження в останнє десятиріччя. Згідно з цим методом вибір пріоритетних рішень здійснюється за допомогою парних порівнянь. За допомогою МАІ можна порівняти відносну важливість будь-яких кількісно невизначених факторів.

МАІ полягає в декомпозиції (розкладанні) проблеми на більш прості складові частини і подальшій обробці послідовності тверджень особи, яка приймає рішення, за допомогою парних порівнянь. В результаті може бути виражений відносний ступінь взаємодії в ієрархії. Ці твердження потім виражаються чисельно.

МАІ включає процедури синтезу багатьох тверджень, отримання пріоритетності критеріїв та знаходження альтернативних рішень. Важливим є те, що отримані таким чином значення є оцінками в шкалі відношень, але відповідають так званим "жорстким" оцінкам.

Вирішення проблеми – процес поетапного становлення пріоритетів. На першому етапі виявляють найбільш важливі елементи проблеми, на другому – найкращий спосіб перевірки тверджень та оцінки елементів. Весь процес підлягає перевірці та переосмисленню доти, доки не буде встановлено, що він охопив усі важливі характеристики вирішення проблеми.

Для кожної пари елементів універсальної множини експерт оцінює перевагу одного елемента над іншим по відношенню до властивості нечіткої множини. В своїй задачі ми використовуємо висновки восьми експертів. Парні порівняння кожного експерта представимо наступними матрицями:

Експерт №1 (старший викладач)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	4	7	8
Доцент	1/4	1	3	7
Ст. викладач	1/7	1/3	1	2
Викладач	1/8	1/7	1/2	1

Вимоги експерта №1:

- не більше 4-х пар в день;
- не більше 2-х лекцій в день;
- вибір певної аудиторії;
- пари в першій половині дня;
- відсутність «вікон»;
- вибір певного дня тижня.

Експерт №2 (викладач)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	2	4	6
Доцент	1/2	1	2	4
Ст. викладач	1/4	1/2	1	2
Викладач	1/6	1/4	1/2	1

Вимоги експерта №2:

- мінімальна кількість зайнятих днів на тиждень;
- максимальна кількість пар в день;
- не більше двох лекцій в день;
- наявність «вікна»;
- не більше шести пар в день.

Експерт №3 (професор)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	3	6	7
Доцент	1/3	1	3	5
Ст. викладач	1/6	1/3	1	2
Викладач	1/7	1/5	1/2	1

Вимоги експерта №3:

- вибір певного дня тижня;

- вибір певної аудиторії;
- не більше 3-х пар в день;
- почерговість лекцій і лабораторних;
- відсутність «вікон».

Експерт №4 (викладач)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	5	7	9
Доцент	1/5	1	2	4
Ст. викладач	1/7	1/2	1	3
Викладач	1/9	1/4	1/3	1

Вимоги експерта №4:

- рівномірний розподіл пар на тиждень;
- не більше 3-х пар на день;
- вибір певної аудиторії;
- відсутність «вікон».

Експерт №5 (доцент)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	3	6	9
Доцент	1/3	1	2	4
Ст. викладач	1/6	1/2	1	2
Викладач	1/9	1/4	1/2	1

Вимоги експерта №5:

- пари у вівторок і середу;
- відсутність «вікон».

Експерт №6 (старший викладач)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	5	7	8
Доцент	1/5	1	2	4
Ст. викладач	1/7	1/2	1	2
Викладач	1/8	1/4	1/2	1

Вимоги експерта №6:

- рівномірний розподіл пар на тиждень;
- не більше 4-х пар на день;

- відсутність «вікон».

Експерт №7 (професор)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	6	7	9
Доцент	1/5	1	3	4
Ст. викладач	1/6	1/3	1	3
Викладач	1/9	1/4	1/3	1

Вимоги експерта №7:

- не більше 3-х пар на день;
- не більше 1 лекції на день;
- вибір певної аудиторії;
- відсутність «вікон».

Експерт №8 (викладач)

	Професор	**Доцент**	**Ст. викладач**	**Викладач**
Професор	1	3	5	7
Доцент	1/3	1	3	6
Ст. викладач	1/5	1/3	1	3
Викладач	1/7	1/6	1/3	1

Вимоги експерта №8:

- рівномірний розподіл пар на тиждень;
- мінімальна кількість пар на день;
- пари в першій половині дня;
- відсутність «вікон».

Цього виявилось достатньо для того, щоб визначити переваги (вимоги) викладачів і вагові коефіцієнти їх посад.

Таким чином, якщо не вдається сформувати вибірку допустимих прийнятних рішень, її доповнюють неприйнятними раніше рішеннями. Не виключено, що вибірка буде складатися виключно з неприйнятних рішень. Може статися так, що максимальне значення цільової функції для прийнятних рішень виявиться менше хоча б одного значення цільової функції для неприйнятних рішень, що відповідає випадку найбільш повного задоволення вимогам викладачів з великими значеннями вагових коефіцієнтів на шкоду молодим або «неос-

тепененним» викладачам. У кожному разі використання штрафних функцій підвищує різноманітність розглянутих потенційних рішень.

Розглянемо задачу конструювання функції штрафу. Як вже було зазначено,

$$\varphi(F_1(r) \vee F_2(r)) = \begin{cases} 1, & r \in \Omega_2(P,R,L,A), \\ f(\bigvee_{j=1}^{l} x_j \bigvee_{j=1}^{K} y_j . D, \gamma), & в \quad іншому \quad випадку. \end{cases} \tag{2.22}$$

У виразі (2.22) нульове значення функції штрафу відповідає випадку, коли не виконана хоча б одна з вимог студентських груп або хоча б одного викладача, D – значення, що інтегрує пріоритети вимог викладачів, γ – деякий параметр.

Оскільки «оштрафованими» можуть бути і рішення, що не задовольняють вимогам студентських груп, і рішення, що не задовольняють вимогам окремих викладачів, то доцільно вважати штрафну функцію адитивною і записувати в такому вигляді:

$$\varphi(F_1(r) \vee F_2(r)) = \beta_1 \varphi_1(F_1(r)) + \beta_2 \varphi_2(F_2(r)) = \beta_1 f_1(\bigvee_{j=1}^{l} x_j) + \beta_2 f_2(\bigvee_{j=1}^{K} y_j, D) \tag{2.23}$$

Очевидно, що чим більшою мірою порушуються обмеження, тим більшим має бути значення функції штрафу. За побудовою функція штрафу є невід'ємною функцією, а виходячи зі свого призначення, для неї справедлива нерівність

$$0 \le \varphi(*) \le F_{\max}. \tag{2.24}$$

Аналогічно і для її компонент:

$$0 \le \varphi_i(F_i(r)) \le F_{i\max} \text{ і } 0 \le f_1(\bigvee_{j=1}^{l} x_j) \le F_{1\max},\ 0 \le f_2(\bigvee_{j=1}^{K} y_j) \le F_{2\max}. \tag{2.25}$$

Розглянемо побудову штрафної функції $\varphi_1(*)$. Введемо функцію відстані від передбачуваного потенційного рішення r до області $\Omega_2(P,S,L,A)$. Раніше було описано, що для того, щоб отримати пріоритети вимог студентів $x_{j,}$ $j = \overline{1,l}$ використовувався метод аналізу ієрархій Т. Сааті [106]. При побудові матриці попарних порівнянь використовувалися лише прийнятні побажання і

вимоги. За таким же принципом побудуємо матрицю попарних порівнянь для неприйнятних варіантів розкладів. Розглянемо як приклад вимогу студентських груп: «Не більше трьох пар на день». Тоді неприйнятним для студентів є варіант розкладу, в якому існує «Більше трьох пар на день». Очевидно, доцільно вважати, що $T \cup \overline{T} = V$, де T – прийнятне вимога, $\overline{T}$ – неприйнятна вимога, V – множина можливих вимог про щось. Для викладача Петрова А. в розглянутому вище прикладі – це $\overline{T}$ = {*Одна пара в понеділок, дві – у вівторок і три в середу або одна пара у вівторок, чотири – в середу і одна в п'ятницю, або ...*}.

Таким чином, існує кілька підходів до визначення штрафній «студентської» функції. Для першого способу досить покласти

$$\overline{x_j} = 1 - x_j, \;\; j = \overline{1,l}, \;\; \sum_{j=1}^{l} x_j = 1, \;\; \sum_{j=1}^{l} \overline{x_j} = 1, \qquad (2.26)$$

і вважати, що

$$f_1(\bigvee_{j=1}^{l} x_j) = \sum_{j=1}^{l} (1 - x_j) = l - \sum_{j=1}^{l} x_j \,. \qquad (2.27)$$

Переваги функції (2.27) – простота отримання, недолік – припущення про те, що значення чогось небажаного обернено пропорційно значенню бажаного, що не завжди справедливо. Крім того, відсутня градація небажаних по якійсь вимозі розкладів.

Для другого способу необхідно по матриці парних порівнянь визначити небажаність виконання тієї чи іншої вимоги. Чим вимога є небажанішою, тим більше відповідне значення $\overline{x_{j,}} \;\; j = \overline{1,l}$. Вважаємо, що кожній бажаній вимозі відповідає небажана вимога. Штрафна функція буде мати такий вигляд:

$$f_1(\bigvee_{j=1}^{l} x_j) = \sum_{j=1}^{l} \overline{\overline{x_j}} \qquad (2.28)$$

Зауважимо, що в загальному випадку $\overline{\overline{x_j}} \neq \overline{x_j}$, $j = \overline{1,l}$.

Реалізація третього способу пов'язана з повним використанням можливостей методу аналізу анархій Т. Сааті [106]. Для визначення небажаності реаліза-

ції тієї чи іншої вимоги будуємо ієрархічну структуру вимог. На верхньому рівні знаходяться вимоги, аналогічні розглянутим у другому способі, і значення небажаності. Їх реалізації відомі: $\overline{\overline{x_j}}$, $j=\overline{1,l}$. На нижньому рівні необхідно побудувати матриці попарних порівнянь для підваріантів таких вимог.

Так, студентській вимозі $T=\{$*Одна лекція в понеділок*$\}$ відповідає небажана вимога $\overline{T}=\{$*Тільки не одна лекція в понеділок*$\}$. Очевидно, що $T=\overline{T_0}\ \mathrm{Y}\overline{T_2}\ \mathrm{Y}\overline{T_3}\ \mathrm{Y}\overline{T_4}$ де $T_i=\{i$ *лекцій в понеділок*$\}$. Для вимог T_i, $i=\overline{0,4}$, $i\neq 1$ будуємо матриці попарних порівнянь і знаходимо значення рівня їх небажаності. Для загального випадку – це значення $\overline{\overline{x}}_{ji}$, $i=\overline{1,k_j}$ Тоді функція штрафу буде такою:

$$f_1(\underset{j=1}{\overset{l}{\vee}} x_j)=\sum_{j=1}^{l} x_j \sum_{i=1}^{K_l} x_{ij}\cdot \chi(i-й \quad варіант \quad j-ї \quad вимоги \quad реалізовано). \quad (2.29)$$

Функція штрафу $f_2(\underset{j=1}{\overset{J}{\vee}} y_j, D)$ конструюється аналогічним чином. Єдиною принциповою відмінністю є необхідність врахування як пріоритетів викладачів, так і їхніх вимог.

Використання штрафних функцій направлено на збільшення різноманітності потенційних рішень, пошук рішень, відповідних великим значенням цільової функції, але які є неприйнятними для деяких учасників навчального процесу.

Як один із методів розв’язання задачі складання розкладу занять використовувався метод послідовного аналізу варіантів.

На основі узагальнення ідей теорії послідовних рішень і динамічного програмування В.С. Михалевич розробив загальну схему послідовного аналізу варіантів (ПАВ) [27]. З точки зору формальної логіки схема ПАВ зводиться до такої послідовності повторення процедур:

– розбиття множини варіантів розв’язання задачі на кілька підмножин, кожна з яких має специфічні властивості;

– використання цих властивостей для пошуку логічних протиріч в описі окремих підмножин;

– виключення з подальшого розгляду тих підмножин варіантів розв’язку, в описі яких є логічні суперечності.

Методика послідовного розвитку, аналізу і відсіву варіантів полягає в такому способі розвитку варіантів і побудови операторів їх аналізу, які дозволяють відсівати безперспективні початкові частини варіантів до їх повної побудови. Оскільки при відсівання безперспективних початкових частин варіантів відсівається тим самим і всі їхні збори продовжень, то відбувається значна економія обчислювальних витрат.

На базі цієї загальної схеми В. С. Михалевич і його співробітники розробили цілий ряд алгоритмів послідовного аналізу варіантів, які отримали широке застосування в практиці.

Наближені методи можна розбити на наступні групи: методи локальної оптимізації, модифікації точних методів, евристичні методи, які максимально враховують специфіку вирішуваних завдань, методи випадкового пошуку, а також методи, що поєднують локальну оптимізацію з випадковим пошуком. Відзначимо, що багато наближені алгоритми дозволяють вирішувати завдання дискретної оптимізації в діалоговому режимі. Це дає можливість в залежності від виділених ресурсів (часу, пам'яті комп'ютера і т.п.) послідовно покращувати отримане рішення шляхом зміни всіх або деяких вихідних даних.

Одними з найбільш розвинених наближених методів є методи локальної оптимізації, що мають на меті відшукання локально оптимальних рішень. Нерідко в цих методах на певних етапах виконання завдання використовуються методи випадкового пошуку, а також різні способи (евристики), які дають можливість скоротити хід варіантів і максимально враховують специфіку задачі. Слід зазначити, що алгоритми, в яких комбінуються різні ідеї, на практиці часто виявляються найбільш ефективними. За допомогою цих методів були розв’язані численні складні задачі класифікації об’єктів, розміщення, планування і проектування. Основна перевага цих методів – простота реалізації, а ос-

новний недолік це те, що вони не можуть адаптуватися до умов розв'язуваної задачі. Значно більш гнучкими є методи, у яких імовірнісний закон залежить від результатів попередніх випробувань і змінюється від ітерації до ітерації. Це методи випадкового пошуку з навчанням.

Деталізація схеми ПАВ відбувалася в декількох напрямках. У зв'язку з розв'язання задач лінійної або деревовидної структури був сформований узагальнений принцип оптимальності для монотонно рекурсивних функціоналів, на основі якого можна будувати схему розв'язків, вільну від деяких обмежень, властивих канонічним процедурам динамічного програмування.

Метод ПАВ базується на відсів безперспективних елементів як по обмеженням, так і по цільовій функції. Його ідеї розглянемо на прикладі такої задачі дискретного програмування [3].

Схему методу ПАВ викладемо для наступного задачі:

$$\min_{x \in D(\mathrm{X})} f(x), \tag{2.30}$$

при обмеженнях

$$g_i(x_1,...,x_n) \le 0, i = \overline{1,m}, \tag{2.31}$$

$$x_j = Q_j, j = \overline{1,n}, \tag{2.32}$$

де $Q_j = (q_{1j},...,q_{nj})$ – задані кінцеві множини. Вектор $x = (x_1,...,x_n)$ назвемо розв'язком, якщо його компоненти $x_j \in Q_j, j = \overline{1,n}$. Множину всіх розв'язків позначимо Ω. Розв'язок називається допустимим, якщо він задовольніє нерівності (2.31). Множину всіх допустимих розв'язків позначимо через Ω_j. Вектор $x_{(p)} = (x_1,...,x_n), p < n$ будемо називати частковим розв'язком, якщо $x_j \in Q_j$. Якщо при цьому він може бути побудований до допустимого розв'язку $(x_1,...,x_p,x_{p+1},...,x_n)$, то будемо називати його допустимим частковим розв'язком.

Нехай h – деяка множина часткових розв'язків. Для будь-якого впорядкованого набору елімінувальних тестів $\sigma = (\xi_0, \xi_1,..., \xi_i)$ введемо позначення

$$\sigma(h) = h^{(t+1)} \tag{3.33}$$

де

$$h^{(i)} = \frac{h^{(i-1)}}{\xi_{i-1}(h^{(i-1)})}, i = \overline{1, t+1}, h^{(0)} = h \tag{3.34}$$

Тут через $\xi_i(h)$ позначено множину часткових рішень, які виключаються тестом ξ_i. Викладемо загальний алгоритм методу послідовного аналізу рішень.

Нехай в результаті k кроків отримано множину повних і часткових розв'язків h_k $\{h_0 = x_{(0)}\}$, яке назвемо списком. В множині h_k за допомогою деякого правила μ вибираємо деяку підмножину $\mu(h_k)$ часткових розв'язків (множина $\mu(h_k)$ може складатися і з одного часткового вирішення). Кожне з часткових розв'язків $x^0_{(p)} \in \mu(h_k)$ заміняємо системою частковиз розв'язків виду

$$\beta\{x^0_{(p)}\} = \{x_{(p+1)} = (x_1, ..., x_{p+1}) \mid x_j = x^0_j,\ j = \overline{1, p},\ x_{p+1} \in Q_{p+1}\} \tag{3.35}$$

де оператор β реалізує покрокове конструювання часткових розв'язків. До отриманої множини

$$h^t_{k+1} = \left(\frac{h_k}{\mu(h_k)}\right) \mathrm{Y}\, \beta(\mu(h_k)) \tag{3.36}$$

Застосовуємо набір σ елімінувальних тестів, кожен з яких за певною ознакою здійснює елімінацію (відсів) безперспективних часткових рішень з безлічі h^t_{k+1}.

Отже, k-й крок методу ПАВ полягає в перетворенні списку h_k за формулами:

$$h^t_{k+1} = \left(\frac{h_k}{\mu(h_k)}\right) \mathrm{Y}\, \beta(\mu(h_k)), \tag{3.37}$$

$$h_{k+1} = \sigma(h^t_{k+1}). \tag{3.38}$$

В силу скінченності множини розв'язків і присутності в наборі σ тестів ξ_0, ξ_1 послідовність $h_0, h_1, ...$ сходиться за кінцеве число кроків до множини оптимальних розв'язків задачі (2.30)-(2.32).

Для розв’язання задачі складання розкладу навчальних занять за допомогою методу ПАВ замість формули (2.30) використовуємо функцію (2.19), а далі слідуємо схемою методу.

Застосувавши цей метод для розв’язання задачі складання розкладу навчальних занять і взявши в якості цільової функції інтегровану цільову функцію «викладача» і «студента», були отримані наступні результати. Запропонований ітераційний метод дає можливість отримувати більш точні рішення, в порівнянні з класичними ітераційними методами, але при цьому час розв’язання задачі збільшується в середньому в 0,46 рази. Виходячи з цього, можна сказати, що метод ефективний для задач невеликої розмірності, оскільки при великій розмірності задачі час розв’язання задачі буде істотно збільшуватися.

3.4 Алгоритмізація еволюційного методу розв’язання задачі складання розкладу навчальних занять

На етапі генерації розкладу спершу вводяться такі поняття, як: заняття для потоку, заняття для групи та заняття для підгрупи. Заняттям для потоку вважається лекція, яка проводиться одним викладачем з одного предмету на одному курсі, але для декількох груп. Заняттям для групи вважається заняття, яке проводиться для усіх підгруп певної групи одним викладачем з одного предмету. Заняття, яке проводиться для окремої підгрупи вважаються заняттям для підгрупи. На основі саме цих даних і відбувається генерація розкладу.

Принцип генерації наступний. Спочатку програма з навчального навантаження виділяє заняття для потоків, груп і підгруп. Далі випадковим чином, рівномірно по днях, але якомога ближче до першої пари, розміщуються заняття для потоків, для кожного заняття вибирається приміщення з набору придатних саме для цього заняття. Потім так само розміщуються на вільні місця заняття для груп та підгруп. При цьому враховуються наступні дані: а) кількість годин даного заняття за семестр; б) можливість проведення викладачем даної пари; в)

придатність вибраного приміщення для даного заняття; г) чи є вибране приміщення вільним для даної пари.

Загальний алгоритм складання розкладу занять за допомогою еволюційного моделювання складається з декількох етапів (рис. 3.3).

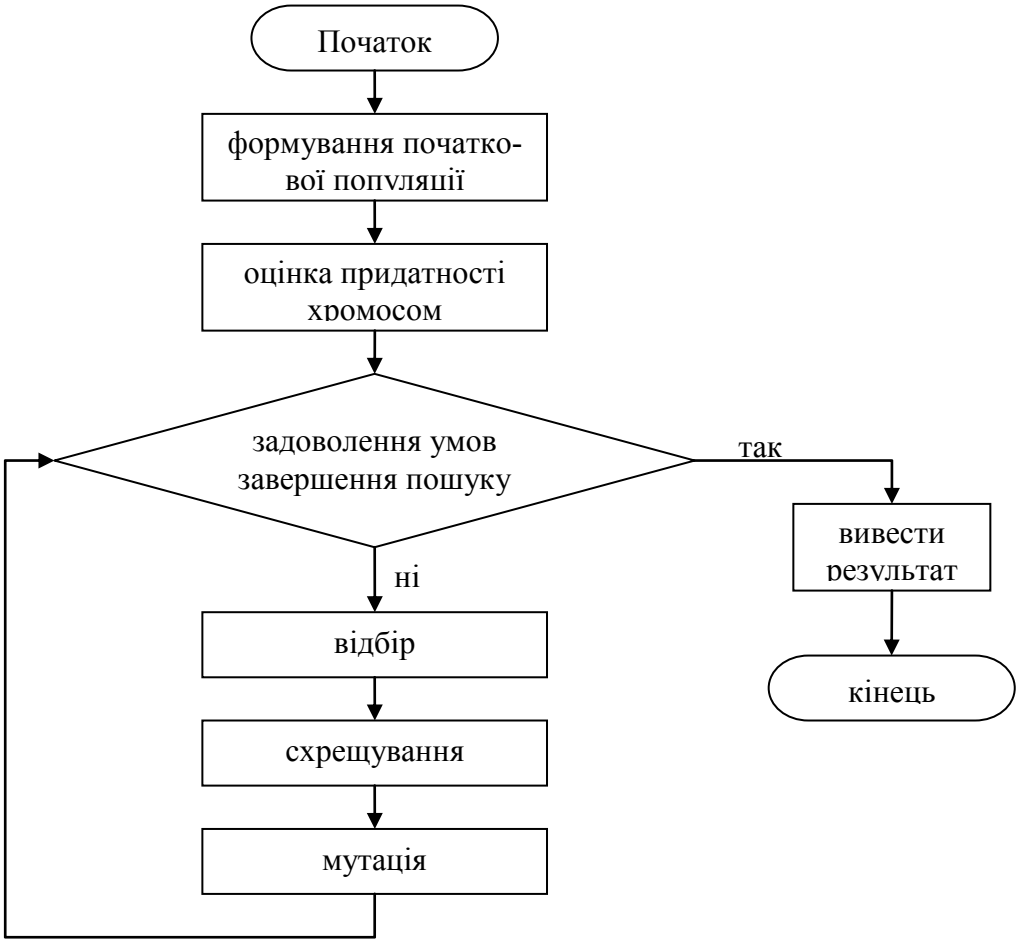


Рисунок 3.3 – Блок-схема роботи еволюційного алгоритму

У більшості випадків розклад занять формується диспетчером, починаючи із першого курсу, за технологією, схожою до одного із методів упаковки в контейнери. Визначається викладач з найвищим пріоритетом і формується розклад його занять згідно вимог. Далі вибирається наступний викладач (другий за пріоритетом), для нього формується відповідний його перевагам розклад, але з урахуванням розкладу занять першого викладача. Аналогічно далі, до тих пір, поки знайдеться викладач, для якого розклад занять не може бути сформованим на основі задоволення його вимог. Тоді потрібно переглянути останній розклад, який задовольняв усім вимогам попереднього викладача і здійснити його коригування, але таким чином, щоб він відповідав і вимогам попереднього виклада-

ча і вимогам останнього викладача. Якщо ж це не вдається, то потрібно коригувати розклад іншого викладача, який знаходиться вище за ієрархією пріоритетів. Такими є основні кроки і принципи формування розкладу занять у ВНЗ. Запишемо їх формально у вигляді методу, який має такі кроки:

Крок 1. Виконати ранжування викладачів за пріоритетами, тобто

$$L_1 \phi\ L_2 \phi\ ...\phi\ L_m, \text{ якщо } p_1 \geq p_2 \geq ... \geq p_m, \qquad (3.1)$$

де L_i – i-й викладач, p_i – його пріоритет, m – загальна кількість викладачів, $i = \overline{1,m}$, $m >> K$ й існує не менше $(K-1)$-ї пари (p_j, p_{j+1}), для якої в (3.1) виконується нерівність $p_j > p_{j+1}$, $j = \overline{1,m-1}$.

Крок 2. Для кожного викладача виконати ранжування його вимог за індивідуальними перевагами, тобто

$$V_i^1 \phi\ V_i^2 \phi\ ...\phi\ V_i^{n_i}, \text{ якщо } q_i^1 \geq q_i^2 \geq ... \geq q_i^{n_i}, \qquad (3.2)$$

де V_i^j – j-а вимога i-го викладача, n_i – кількість вимог у i-го викладача, q_i^j – значення пріоритетності j-ї вимоги i-го викладача, $i = \overline{1,m}$.

Крок 3. $i = 1$ (номер викладача).

Крок 4. Поки $i < m$ виконувати.

Крок 4.1. $l = 1$.

Крок 4.2. Для $j = 1$ до n_i виконати

Крок 4.2.1. Якщо j-а вимога i-го викладача виконується, то $l = l + 1$. КЯ.

КЦ.

Крок 4.3. Якщо $l = j$, то формуємо розклад i-го викладача, $i = i + 1$. КЯ.

Крок 4.4. Якщо $l < j$, і $n_i = 1$, і $i = 1$, то розклад створити неможливо (не виконується жодна вимога викладача з найвищим пріоритетом – вироджений випадок). КЯ.

Крок 4.5. Якщо $l < j$, і $n_i = 1$, і $i > 1$, то $i = i - 1$. КЯ.

Крок 4.6. Якщо $l < j$, і $n_i > 1$, то $n_i = n_i - 1$. КЯ.

КЦ.

Кроки наведеного методу реалізовуються у припущенні абсолютної домінантності. Це означає, що вимоги викладача з вищим пріоритетом будуть коригуватись тоді, коли не виконується жодна із вимог наступного за пріоритетом викладача. Зауважимо неєдиність такого припущення при складанні розкладів занять.

Висновки до розділу 3

У третьому розділі обґрунтовано використання еволюційного моделювання для розв’язання задачі складання розкладу навчальних занять. Представлено особливості формування структури розв’язку та початкової популяції.

При застосуванні традиційних методів оптимізації і пошуку, у разі навіть незначної зміни параметрів середовища всі обчислення доводиться проводити заново. Еволюційний підхід дозволяє проводити аналіз і адаптацію вже створеної популяції до нових умов середовища, чим скорочує час роботи алгоритму і реалізує принцип машинної адаптації і навчання.

Запропонований метод формування фрагмента потенційного розв’язку дозволяє прискорити обчислювальний процес за рахунок забезпечення неперервності одержуваних розв’язків. Зокрема, якщо буде виявлено, що деякий викладач не може провести певне заняття з даного предмету, то більш ймовірно, що в першу чергу на наступному кроці йому буде запропоновано змінити форму заняття або предмет.

Розділ 4. ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ ТА ВЕРИФІКАЦІЯ ОДЕРЖАНИХ РЕЗУЛЬТАТІВ

Існує думка, що досвідчений диспетчер зможе скласти розклад так, що він буде відповідати інтересам навчального процесу та суспільного життя навчального закладу. Однак з цим не можна погодитися. Ручне розв'язання задачі складання розкладу занять вимагає великих витрат часу, кваліфікованих фахівців, і, як результат, таке рішення часто виходить далеко не оптимальним. Після введення початкової інформації потрібно її узгодити, у той час, як неможливість отримання необхідного розкладу може бути визначена ще на етапі аналізу. Під час складання розкладу можливе виникнення безвихідних ситуацій. Все це вимагає зміни вихідних даних і спрощення обмежень, і тут без людини не обійтися. Без внесення даних змін розклад не матиме практичної цінності. Також слід врахувати той момент, що розклад може змінюватися і під час його використання, тобто після складання, і тут вельми важливий людський фактор. У цьому плані важлива підтримка даного процесу автоматизованими методами і процедурами. Основна перевага полягає в тому, що автоматизоване складання усуває масу рутинної роботи, такої як: пошук можливих варіантів внесення чергових елементів в розклад, перевірку виконання вимог, пошук випадкових помилок в готовому розкладі, оформлення розкладу на папері у вигляді різних таблиць (для викладачів, груп), залишаючи людині більше часу на більш інтелектуальні дії. Комп'ютер у даному випадку також є інструментом, що істотно посилює здібності людини, оскільки людина не в змозі перебрати і проаналізувати таку ж кількість варіантів розкладів, як комп'ютер.

4.1 Обґрунтування вибору засобів створення програмного продукту

Проектувальникові в даний час надається великий вибір СУБД, розроблених для різних конфігурацій і типів ЕОМ. Аналіз основних параметрів цих систем дозволяє одразу ж відкинути ряд СУБД, свідомо непридатних до викорис-

тання в розроблюваній інформаційній системі, залишивши для подальшого розгляду не більш 2-3-х систем-претендентів.

На вибір СУБД-претендентів найбільший вплив чинить узгодження ряду параметрів середовища реалізації і СУБД. Найбільш значущі параметри перераховані нижче:

- тип ЕОМ (IBM PC AT на базі процесора Intel 80386);
- операційна система (Windows XP);
- об'єм оперативної пам'яті (2Мб);
- об'єм дискової пам'яті (1,8 МБ);
- вибрана для реалізації модель даних (реляційна).

В основі проектування БД повинні бути враховані вимоги користувачів конкретного ВНЗ – концептуальні вимоги до системи. Саме користувач у своїй роботі приймає рішення з урахуванням одержуваної в результаті доступу до БД інформації. Від оперативності доступу до інформації, а також від її якості буде залежати ефективність роботи алгоритму складання розкладу.

Розглянемо найпоширеніші засоби створення користувальної програми-додатку з обробки даних: Access, SQL Server, Visual Basic, InterBase. Ці засоби можуть бути використані, як окремо – для розв'язання конкретної задачі, так і в якості інтегрованого набору, кожний компонент якого може бути застосований при розробці великих проектів масштабу ВНЗ. Характеристики цих продуктів, за винятком пакета InterBase, тому що він вимагає більш детального розгляду, наведені в таблиці 4.4.

Таблиця 4.4 – Характеристика СУБД

Назва продукту	**Основні переваги**	**Основне призначення**
Access	Простота освоєння. Можливість використання непрофесійним програмістом.	Створення звітів довільної форми за різними даними.
SQL-Server	Високий рівень захисту даних. Потужні засоби роботи з даними. Висока продуктивність.	Збереження великих масивів даних.
Visual Basic	Універсальність. Невисокі вимоги до потужності ЕОМ.	Створення додатків для інтеграції компонентів Microsoft Office.

Розглянемо більш детально перераховані в таблиці СУБД.

Які загальні риси мають аналізовані засоби розробки, що підтверджують твердження про можливість їх спільного використання для розробки програм-додатків різноманітного рівня складності? Це, в основному, такі нові технології, як OLE, ODBS, DAO, RDAO, Active та ін. Ці технології дають можливість використання однією програмою-додатком даних, що зберігаються в різних форматах. Це забезпечує розробку програм незалежних від даних.

Звичайно, при спільному використанні різноманітних засобів розробки програм-додатків нас більше всього будуть цікавити дані. У таблиці 4.5 наведений перелік типів даних, доступних в розглянутих засобах розробки.

Прочерки в передостанній колонці таблиці означають, що для цього типу даних задання конкретних розмірів не потрібно.

Таблиця 4.5 – Типи даних СУБД

Тип даних	**Access і Visual Basic**	**MC SQL Server**	**Займаний обсяг**
Binary	dbLongBinary	bynary(n)	до 1,2 Гбайт
Byte	dbByte	–	1 байт
Character	dbText	tinyint char(n)	4 байта
Count	dbLong	–	4 байт
Currency	dbCurrency	money	8 байт
DateTime	dbDate	datetime	8 байт
Logical (Yes/No)	dbBoolean	bit	1 байт
Numeric		float	від 1 до 20 байт
Integer Integer	dbInteger dbLong	Smallint int	2 байт 4 байт
Double	dbDouble	float	8 байт
Float		float	від 1 до 20 байт
General (OLE Object)	dbLongBinary	image	4байта
Memo	dbMemo	text	4 байт
Single	dbSingle	real	4 байт

Byte. Ціле позитивне число від 0 до 255.

Character. Символьний вираз може містити будь-які символи (до 254 для одного поля).

Count. Лічильник, що автоматично нарощує своє значення при додаванні запису. Початкове значення 1.

Currency. Грошовий вираз для числового розміру. Виводить число з чотирма десятковими розрядами і встановленим позначенням вибраної грошової одиниці.

DateTime. Вираз „дата і час” може містити час, день, місяць і рік.

Logical. Булевий вираз, що приймає значення „так” або „ні”.

Numeric. Числовий вираз може містити цілі або дробові числа зі знаком.

Integer (dbInteger). Ціле число в діапазоні від -32,768 до +32,767.

Integer (dbLong). Ціле число. Можна зберігати числа від -2147483647 до 2147483646.

Double. Числа з плаваючою точкою подвійної точності. Можна зерігати значення від 4,4065645841247E-324 до 1,9769313486232E308.

General. Поле для посилання на об’єкт OLE.

Memo. Поле приміток для посилання на блок даних.

Single. Число з плаваючою точкою одинарної точності. Можна зберігати від’ємне число від -3,402823E38.

Усі СУБД, як правило, мають подібний функціональний склад, у який входять діалогові засоби для роботи з даними – назвемо їх користувальними засобами, засоби розроблювача, що забезпечують можливість створення користувальної програми-додатку, і додаткові засоби, від складу яких, як правило, залежать функціональні можливості і потужність розроблювальних програм. У залежності від призначення засобу розробки склад різних засобів у конкретної СУБД може значно відрізнятися.

Наприклад, у Access користувальні засоби розвинуті значно сильніше, ніж у Visual Basic, де вони розглядаються як допоміжні функції. Тому необхідно розглянути засоби розробки окремо.

Опис СУБД MS Access. Microsoft Access – це найпопулярніша сьогодні система керування БД. Її успіх можна пов’язувати з чудовою рекламною компанією, організованою Microsoft, або включенням її в оточення сімейства Microsoft Office. Цілком можливо, що це так. Але корінь успіху, швидше за все, полягає в реалізації продукту, розрахованого як на починаючого, так і на кваліфікованого користувача.

СУБД Access для роботи з даними використовує процесор БД Microsoft Jet 3.0, об’єкти доступу до даних і засоби швидкої побудови інтерфейсу – Конструктор форм. Для створення звітів використовуються Конструктори звітів. Автоматизація рутинних операцій може бути виконана за допомогою макрокоманд. На той випадок, коли не вистачає функціональності візуальних засобів, користувачі Access можуть звернутися до створення процедур і функцій. При цьому як у макрокомандах можна використовувати виклики функцій, так і з коду процедур і функцій можна виконувати макрокоманди.

Незважаючи на свою орієнтованість на користувача в Access присутня мова програмування Visual Basic for Application, що дозволяє створювати масиви, свої типи даних, викликати DLL-функції, за допомогою OLE Automation контролювати роботу додатків, що можуть функціонувати як OLE-сервери. При необхідності можна створювати БД за допомогою кодування.

Головна перевага Access, що привертає до нього багатьох користувачів, – тісна інтеграція з Microsoft Office. Наприклад, скопіювавши в буфер графічний образ таблиці, відкривши Microsoft Word і застосувавши вставку з буфера, ми відразу одержимо в документі готову таблицю з даними з БД.

Вся робота з БД здійснюється через контейнери БД. Звідси здійснюється доступ до всіх об’єктів: таблицям, запитам, формам, звітам, макросам, модулям.

За допомогою драйверів ISAM можна одержати доступ до файлів таблиць деяких інших форматів: DBASE, Paradox, Excel, текстовим файлам, FoxPro 2.x, а за допомогою технології ODBC – і до файлів багатьох інших форматів.

Access може виступати як у ролі OLE-контролера, так і OLE-серверу. Це означає, що можна контролювати роботу програм-додатків Access із будь-якої програми, за умови, що вона може виступати в ролі OLE-контролера і навпаки.

Вбудований SQL дозволяє максимально гнучко працювати з даними і значно прискорює доступ до зовнішніх даних.

Користувачем, малознайомим із поняттями РБД Access дає можливість розділяти свої складні за структурою таблиці на декілька, зв'язаних ключовими полями.

Access – це типова настільна база даних, тобто всі користувачі можуть звертатися до однієї БД, встановленої на одній робочій станції, яка не обов'язково повинна бути сервером. Для того, щоб не виникли проблеми цілості і доступу до даних, має сенс скористатися засобами захисту, що надає Access. При цьому ви можете скористатися майстром, якщо не впевнені, що самі правильно встановите права й обмеження для користувачів.

СУБД Access має русифікований інтерфейс і частково переведений на російську мову файл контекстної допомоги.

Також Access має кращу вбудовану систему захисту серед усіх СУБД. Ви можете створювати групи, користувачів, привласнювати права доступу до всіх об'єктів, у тому числі і модулів. Система захисту доступна тільки при відкритій БД. Кожному користувачу можна надати індивідуальний пароль. Система захисту доступна як за допомогою візуальних засобів, так і програмним шляхом. Якщо ви хочете захистити вашу БД навіть від користувача з ім'ям Admin, то користуйтеся послугами настроювання Sequrity, що поставляється разом із Access Developer Toolkit. Крім цього ви можете закрити вашу БД від перегляду зовнішніми програмами.

Аналіз СУБД Visual Basic. Visual Basic є універсальним засобом програмування, проте розглядати його можливості тільки з погляду створення програм-додатків з обробки даних не можна.

На відміну від більшості пакетів програм Visual Basic не має головного вікна, що об’єднує всі інші елементи інтерфейсу розробника. Кожний елемент Visual Basic має своє незалежне вікно, яке можна закрити або розмістити незалежно від інших у будь-якому місці екрана.

Основні можливості Visual Basic, застосовувані в розробці додатків для обробки інформації, можуть бути реалізовані завдяки наявності в ньому об’єктів для доступу до даних – Data Access Object (DAO), 32-розрядного процесору даних – JET 3.0 і призначених спеціально для роботи з даними елементів управління.

Процесор даних Visual Basic підтримує всі стандартні операції по створенню, зміні і видаленню таблиць, індексів і запитів.

Формат БД процесора даних Visual Basic відповідає формату Access. JET 3.0 також забезпечує підтримку цілісності і перевірку вихідних і змінюваних даних на рівні полів і записів. Для зміни даних JET 3.0 дозволяє використовувати мову SQL.

Управління БД забезпечується процесором даних за допомогою об’єктів для доступу до даних. Ці об’єкти дозволяють розроблювачу програмним шляхом, за допомогою відповідних властивостей і методів DAO як маніпулювати даними, так і управляти структурою БД, включаючи її створення. У порівнянні з попередньою версією Visual Basic можливості об’єктів для доступу до даних тепер істотно розширені. У Visual Basic для роботи з даними можна застосовувати декілька робочих областей, підтримувати цілісність даних, включаючи каскадне видалення і відновлення і забезпечувати їх захист від несанкціонованого доступу. Крім цього застосування колекцій істотно скорочує програмний код.

Унікальною властивістю JET 3.0 є можливість створення копій даних (реплікацій БД). Для створення копій БД розроблювачу достатньо скористатися методом MakeReplica при заданні методу Synchronize виконується узгодження даних. Причому ці операції можуть виконуватися як із

файлами формату БД процесора даних, так і з БД інших форматів, підтримуваних через ODBC.

Не можна не відзначити, що JET 3.0 використовують індекси нової, більш компактної структури, що дозволяють зменшити час їх створення і прискорити процес пошуку даних.

У Visual Basic Enterprice Edition включені об’єкти для доступу до зовнішніх даних – Remote Data Object (RDO) і відповідні елементи управління – Remote Data Control (RDC). Це дозволяє використовувати всі можливості роботи з курсорами на сервері, досягаючи максимальної швидкості доступу до даних, мінімізуючи мережній графік.

Опис СУБД MS SQL Server. Microsoft SQL Server – одна з найбільш потужних СУБД архітектури клієнт-сервер. Ця СУБД дозволяє задовольняти такі вимоги як тиражування даних, підтримка великих БД на недорогих апаратних платформах при зберіганні несуміжного управління.

MS SQL Server не призначений безпосередньо для розробки користувальних програм-додатків, а виконує функції управління БД. Для користувальної програми-додатку SQL Server є потужним джерелом генерації і управління потрібними даними.

Сервер має засоби віддаленого адміністрування і управління операціями, організовані на базі объектно-орієнтованого розподіленого середовища керування. Microsoft SQL Server входить до складу сімейства Microsoft BackOffice, що об’єднує 5 серверних додатків, розроблених для спільного функціонування в якості інтегрованої системи.

Microsoft SQL Server підтримує широкий спектр засобів розробки і максимально простий в інтеграції з додатками, що працюють на ЕОМ.

SQL Server може тиражувати інформацію в БД інших форматів включаючи Oracle, BM DB2, Sybase, Microsoft Access і інші СУБД (при наявності ODBC драйвера, що відповідає визначеним вимогам).

Збережені процедури, що підтримують OLE Automation, дозволяють розроблювачу застосовувати практично будь-який інструмент із тих, що

підтримують OLE, із метою створення збережених процедур для SQL Server. Visual Basic підтримується посередництвом нової 32-розрядної DB-Library (OCX). Численні розширення мови Transact-SQL включають розширену підтримку курсорів, можливість використання команд визначення даних усередині транзакций і т.д.

Microsoft SQL Server містить інструмент, що дозволяє призначати основні процедури супроводу БД і визначати для них графік виконання. Операції супроводу БД включають перевірку розподілу сторінок, цілісності покажчиків у таблицях (включаючи системні) і індексах, відновлення інформації, необхідної оптимізатору, реорганізацію сторінок у таблицях і індексах, створення копій таблиць і журналів транзакцій. Всі ці операції можуть бути встановлені для автоматичного виконання по заданому адміністратором графіку.

Пакет Enterprise Manager включає утиліту, що дозволяє переносити об’єкти з однієї БД в іншу. Використовуючи цю утиліту розроблювач або адміністратор може:

- виконувати копіювання об’єктів будь-якого типу із вказівкою, якого типу об’єкти підлягають копіюванню (або копіювати всі об’єкти всіх типів);
- переносити схему БД разом із даними або без них;
- доповнювати або замінювати існуючі дані;
- знищувати об’єкти в базі перед копіюванням схеми;
- використовувати стандартні настроювання генерації коду створення/видалення об’єктів або використовувати власні;
- визначати момент виконання переносу об’єктів: негайно одноразово у визначений момент часу, багаторазово по визначеному графіку.

Основні характеристики системи InterBase. Як програмний продукт для створення автоматизованої системи складання розкладу занять ЧДТУ була вибрана розробка фірми Borland – мова програмування Delphi v.7.0 і бібліотека стандартних програм InterBase v.6.0 для реалізації обслуговування РБД.

InterBase – унікальний програмний засіб, що дозволяє програмістам мови Delphi в повному об’ємі використовувати архітектуру СУБД.

У більшості сучасних систем загальний доступ до даних можна здійснити лише при виконанні додаткових операцій – зазвичай файли одного операційного середовища перетворюються у формати іншого, і навпаки. Якщо при аналізі завдання з’ясовується, що початкові дані зберігаються в різних системах, то проблеми узагальнення стають важливою областю розв’язання задачі.

InterBase усуває цю проблему, відкриваючи структуру даних для програм на мові Delphi. Таким чином, прикладні програми на мові Delphi дістають можливість обробляти дані системи InterBase.

InterBase – це зручний засіб для розробників програм обслуговування РБД на традиційній мові програмування. Розробники можуть розширити функції системних програм InterBase, доповнивши їх програмами на мові Delphi.

Таблиці InterBase – це стандартні файли Windows з розширенням “.GDB”. Імена таблиць відповідають стандартним угодам про ідентифікацію файлів, прийнятим в ОС MS Windows.

Кожна таблиця може містити до двох більйонів записів, а при її записі на диск розмір таблиці обмежений лише розміром диска або об’ємом вільної пам’яті на диску.

Кожен запис може включати до 255 полів, а кожне поле до 255 символів.

У InterBase використовується п’ять типів полів: алфавітно-цифрового типу, які дозволяють використовувати повний набір символів коду ASCII, числові поля даних, що дозволяють використовувати до 15-ти значущих цифр, включаючи місце для десяткової точки в інтервал значень, а також тип неструктурних двійкових даних, короткі числові поля і поля дати.

Для впорядкування і пошуку ключових полів таблиці в InterBase передбачені спеціальні функції, що використовують індексний метод доступу. Індекс таблиці дозволяє прискорити процес пошуку потрібного запису. Первинний індекс впорядковує записи в порядку зростання ключових полів. InterBase зберігає записи відсортованими по ключу і не дозволяє дублювати ключі. Вторинні

індекси, як правило, організовуються для черг таблиць і служать для поліпшення представлення даних.

Для створення віконного інтерфейсу користувача використовується мова Delphi.

Звичайно, не можна однозначно сказати, що обрана СУБД ідеально відповідає поставленій перед розробниками задачі. Для цього порівняємо СУБД InterBase із СУБД MS Access.

Порівнюючи MS Access і Delphi, можна відзначити, що і той і інший володіють низкою переваг і недоліків.

Наприклад, MS Access володіє широким вбудованим апаратом для обслуговування БД, проте не володіє гнучкістю і управляючими можливостями мови Delphi. З іншого боку, Delphi, забезпечуючи максимально можливий рівень управління програмою, і, отже, її гнучкість, не володіє можливістю обслуговування БД, для цього їй потрібна InterBase.

MS Access вважається мовою високого рівня. Віна забезпечує можливість створення документів, вбудовану в інтерфейс разом з процедурними можливостями, а також є ідеальним засобом для швидкого макетування або розробки розрахованих на багато користувачів програм обслуговування БД.

В свою чергу, оскільки Delphi є традиційною мовою програмування, то її недоцільно застосовувати для програм обслуговування БД – вони складніші за аналогічні програми на мові MS Access. З іншого боку, програми обслуговування БД на мові MS Access більші за розміром і гірші по швидкодії, чим ті ж програми на мові Delphi, що використовують функції InterBase.

Переваги мови MS Access полягають в тому, що вінf володіє широким вбудованим апаратом і забезпечує негайний доступ до всіх областей системи. MS Access дозволяє розробляти програми з використанням опцій меню MS Access.

Переваги InterBase полягають в наступному: забезпечення високого рівня управління програмами; розширення функціональності системи InterBase (працюючи в InterBase, ви обмежені тільки можливостями власної програми і

доступністю допоміжних бібліотек); забезпечення максимальної ефективності програм при їх мінімальному розмірі.

З представлених вище міркувань виходить, що за допомогою будь-якого програмного продукту приблизно одного класу можна розв’язати поставлену задачу, вибравши різні підходи, мінімізуючи або максимізуючи ті або інші характеристики.

Можна зробити висновок, що на вибір СУБД вплинули наступні чинники:

1) наявність досвіду програмування на мові Delphi у розробників, що дозволяє понизити тимчасові і матеріальні витрати на їх перенавчання;

2) трудомісткість реалізації додатків, що дозволяє в той же час забезпечити вищу швидкодію виконання програм (наприклад, що дозволяє мінімізувати таку характеристику як час пошуку по БД, і, отже, зменшити час реакції на запит);

3) забезпечення „гнучкості” ПЗ і високого рівня управління програмами;

4) блокова структура програми на Delphi забезпечує як захист даних, так і високий рівень контролю за областями дії і видимості змінних;

5) можливість при реалізації автоматизованої системи складання розкладу занять ЧДТУ для створення віконного інтерфейсу на мові Delphi, що дає можливість приділити більше часу розв’язанню основної задачі.

4.2 Особливості представлення вхідної інформації та структура інформаційно-аналітичної системи складання розкладу занять

Перш ніж розпочати роботу над створенням автоматизованої системи складання розкладу занять, нами методом опитування 110 студентів було визначено їх переваги. Це необхідно для того, щоб адаптувати наш майбутній розклад до потреб студентів. Отже, найважливішими для себе перевагами студенти визначили:

- номер вільного дня тижня (якщо навантаження дозволяє);
- розподіл пар по днях.

В процентному співвідношенні вказані переваги були розподілені наступним чином (табл. 4.1 і табл. 4.2).

Таблиця 4.1 – Процентне співвідношення вільності дня тижня

Кіл-ть студ. / Дні тижня	9	28	12	13	11	10	9	9	9
пн	0	10	10	5	2	5	5	20	0
вт	0	10	5	5	1	0	5	0	0
ср	0	10	5	5	1	0	5	0	0
чт	0	10	10	15	1	20	5	0	5
пт	100	60	70	70	95	75	80	80	95

За результатами таблиці можна зробити висновок, що найбільша кількість студентів (25,5%) вважають, що пари повинні бути рівномірно розподілені в понеділок, вівторок, середу, четвер і п’ятницю залишити вільним днем.

Для визначення бажаного розподілу пар по днях студентам було запропоновано 4 можливих варіанти (табл. 4.2).

Таблиця 4.2 – Процентне співвідношення розподілу пар по днях

Кіл-ть студ. / Варіант розподілу пар по днях	23	9	10	10	11	12	9	10	8	8
3-1-1-1-1	20	40	10	10	0	15	0	5	10	10
2-2-2-1-0	80	40	90	75	50	60	95	55	70	60
0-1-2-2-2	0	10	0	5	50	20	5	30	10	20
1-1-1-1-3	0	10	0	10	0	5	0	10	10	10

Отже, з таблиці видно, що найбільш вдалим (на думку студентів) є розподіл пар у вигляді: понеділок, вівторок, середа – по 2 пари, четвер – 1 пара, п’ятниця – вільний день.

Вся необхідна для розв’язання поставленої задачі інформація задається до початку ітерацій генетичного алгоритму розв’язання ЗСР. Для спрощення вважається, що задана інформація є постійною протягом усього періоду, для якого складається розклад.

Не втрачаючи певного ступеня загальності поставленої задачі, можна визначити деяку сукупність вхідних даних, необхідних для формування обме-

жень і розв’язання задачі, і в той же час загальних для всіх різновидів практичних реалізацій системи. Через специфіку поставленої задачі (можливість порівняно легкої адаптації математичної моделі для випадку практичної реалізації в рамках конкретного ВНЗ) форми документів вхідної інформації не розроблялися. Реквізити вхідної інформації описані в таблиці 4.3.

Таблиця 4.3 – Реквізити вхідної інформації

Найменування реквізитів вхідних документів	**Характеристика реквізитів**	
	Тип	**Макс. довжина**
Прізвище, ім’я, по батькові викладача	текст.	30
Посада, вчений ступінь, звання	текст.	20
Коефіцієнт статусу викладача	числ.	1/1
Назва групи	текст.	10
Чисельний склад групи	числ.	2
Назва дисципліни	текст.	50
Кількість годин	числ.	2
Номери аудиторій	текст.	10
Інформація про аудиторії (тип)	текст.	10
Семестр	числ.	2
Корпус	числ.	1
Місткість аудиторії	числ.	3

Для того, щоб розпочати автоматизацію складання розкладу навчальних занять на першому етапі необхідно заповнити базу даних. Для цього диспетчеру необхідно мати наступну інформацію:

- перелік аудиторного фонду із зазначенням місткості аудиторій і належність до факультету/кафедри;
- навчальний план;
- навантаження викладачів;
- спеціалізацію, посади і переваги викладачів;
- розклад дзвінків.

Вже на початковій стадії розробки знаходиться програмний модуль для диспетчера. Схема взаємодії його модулів показана на рис. 4.1. Така організація програми дозволяє оперативно вносити зміни в необхідних таблицях.

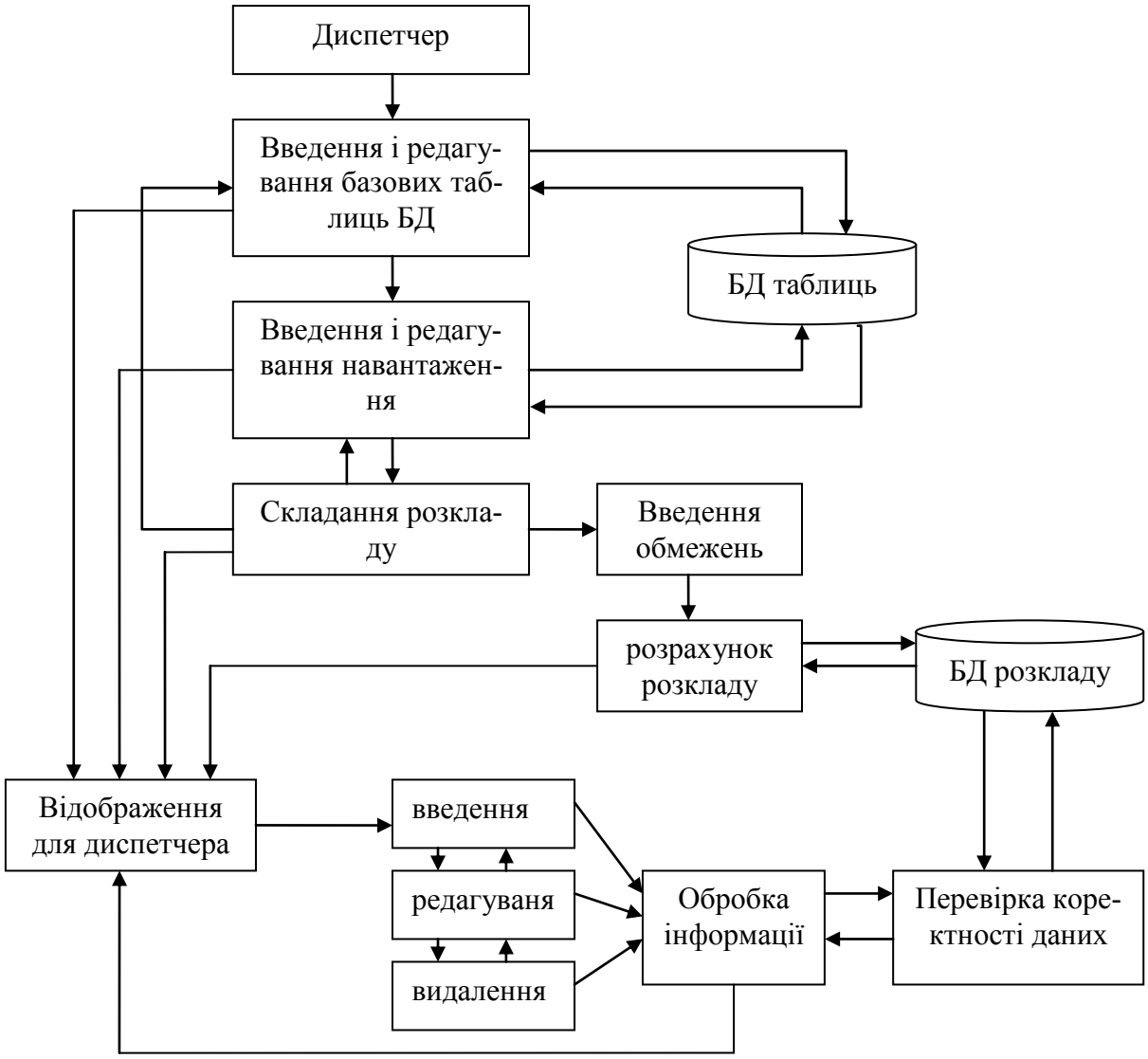


Рис. 4.1 – Структурна схема взаємодії модулів програми

4.3 Структура бази даних

База даних – це сукупність структурованих і взаємозалежних даних і методів, що забезпечують додавання у вибірку і відображення даних. З поняттям бази даних тісно пов’язане поняття системи управління базою даних. Це комплекс програмних засобів, призначених для створення структури нової бази, наповнення її вмістом, редагування вмісту і візуалізації інформації. Під візуалізацією інформації бази розуміється відбір відображуваних даних відповідно із

заданим критерієм, їх упорядкування, оформлення і наступна видача на пристрій виводу або передача по каналах зв’язку.

Вихідні дані доцільно зберігати в зовнішній БД в зручному форматі для заповнення і представлення. Це дозволить підключити модуль складання розкладу занять до складнішої системи управління НП університету. Але в роботі алгоритму використання такого формату неприйнятне через малу швидкість доступу до таких даних і їх структури, що не відповідає внутрішнім структурам алгоритму. Тому перед початком роботи алгоритму необхідно трансформувати зовнішні дані у внутрішній формат, а після закінчення – виконати зворотну трансформацію. Тому, насамперед, визначимося зі способом організації даних або моделлю даних.

Модель даних (МД) – це сукупність угод про способи і засоби формалізованого опису об’єктів і їхніх зв’язків, що мають відношення до автоматизації процесів системи. Вид МД і типи структур даних відбивають концепцію організації й обробки даних, використовуваних у системах управління базами даних (СУБД), що підтримує модель.

В рамках розв’язання поставленої задачі необхідно створити таку МД, при якій обсяг допоміжної інформації був би мінімальним, існувала принципова можливість багатокористувальницького доступу до даних і був би забезпечений високий рівень захисту даних.

В даний час існує три основних підходи до формування МД: ієрархічний, мережний і реляційний.

Ієрархічна база даних (ІБД) складається з упорядкованого набору „дерев”; більш точно – з упорядкованого набору декількох екземплярів одного типу „дерева” (рис. 4.2). Тип „дерева” складається з одного „кореневого” типу запису й упорядкованого набору з нуля чи більш типів „піддерев” (кожне з який є деяким типом „дерева”). Тип „дерева” в цілому являє собою ієрархічно організований набір типів запису.

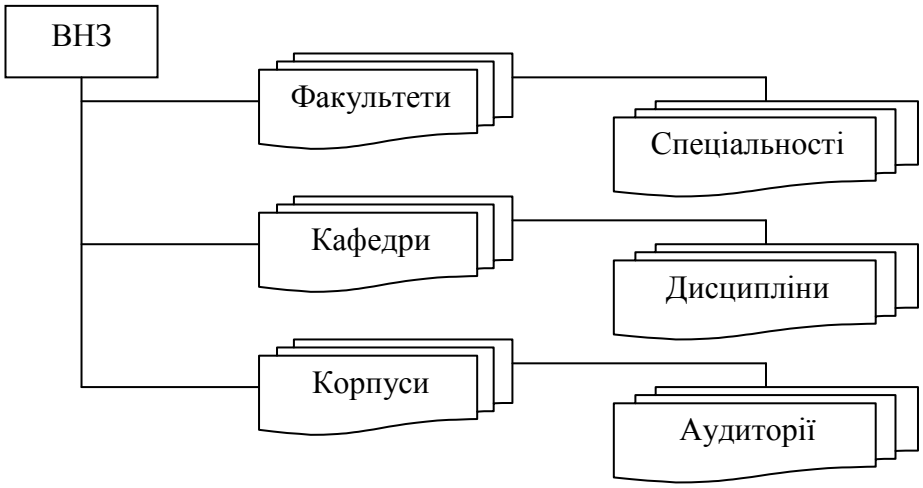


Рисунок 4.2 – Ієрархічний спосіб організації даних

Мережний підхід до організації даних є розширенням ієрархічного. В ієрархічних структурах запис-нащадок повинен мати в точності одного предка, у мережній структурі даних нащадок може мати будь-яке число предків.

Мережна база даних (МБД) складається з набору записів і набору зв’язків між цими записами, а якщо говорити більш точно, з набору екземплярів кожного типу з заданого у схемі БД набору типів запису і набору екземплярів кожного типу з заданого набору типів зв’язку (рис. 4.3).

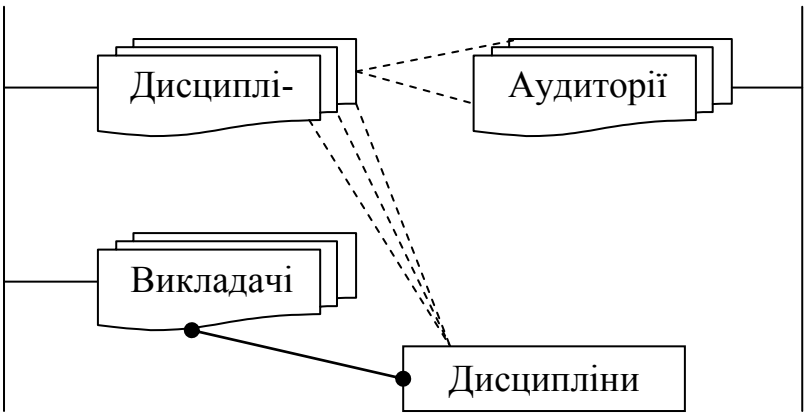


Рисунок 4.3 – Мережний спосіб організації даних

Тип зв’язку визначається для двох типів запису: предка і нащадка. Екземпляр типу зв’язку складається з одного екземпляра типу запису предка й упорядкованого набору екземплярів типу запису нащадка.

Основними недоліками ієрархічного і мережного типів МД є:

- занадто складно користуватися;
- фактично необхідні знання про фізичну організацію;

- прикладні системи залежать від цієї організації;
- їх логіка перевантажена деталями організації доступу до БД.

Недоліки ієрархічної і мережної БД призвели до появи нової, реляційної моделі даних (РМД), створеної Р. Коддом у 1970 році, що викликала загальний інтерес. РМД була спробою спростити структуру БД. У ній були відсутні прямі посилання на предків і нащадків, а всі дані були представлені у вигляді простих таблиць, розбитих на рядки і стовпчики.

У перших реляційних СУБД не були реалізовані деякі з ключових частин моделі Р. Кодда, і цей недолік був усунений тільки згодом. В міру росту популярності реляційної концепції реляційними стали називатися багато БД, які у дійсності такими не були.

У відповідь на неправильне використання терміну „реляційний” Р. Кодд у 1985 році написав статті, де сформулював 12 правил, яким повинна задовольняти будь-яка БД, що претендує на звання реляційної. З того часу дванадцять правил Кодда вважаються визначенням реляційної СУБД. Однак можна сформулювати і більш просте визначення:

Реляційною називається БД, у якій всі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями. Таке визначення не залишає місця вбудованим показникам, що є в ієрархічних і мережних СУБД. Незважаючи на це, реляційна СУБД також здатна реалізувати відносини предок-нащадок, однак ці відносини представлені винятково значеннями даних, що містяться в таблицях.

Найбільш розповсюджене трактування РМД, очевидно, належить К. Дейту [64], що відтворює її (з різними уточненнями) практично у всіх своїх книгах. Згідно К. Дейту РМД складається з трьох частин, що описують різні аспекти реляційного підходу: структурної частини, маніпуляційної частини і цілісної частини.

У структурній частині моделі фіксується, що єдиною структурою даних, використовуваної в РБД, є нормалізоване n-арне відношення.

У маніпуляційній частині моделі затверджуються два фундаментальних механізми маніпулювання РБД – реляційна алгебра і реляційне числення. Перший механізм базується в основному на класичній теорії множин (з деякими уточненнями), а другий – на класичному логічному апараті числення предикатів першого порядку. Основною функцією маніпуляційної частини реляційної моделі є забезпечення міри реляційності будь-якої конкретної мови РБД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра чи реляційне числення.

Нарешті, у цілісній частині РМД фіксуються дві базові вимоги цілісності, що повинні підтримуватися в будь-якій реляційній СУБД. Перша вимога називається вимогою цілісності сутностей. Друга вимога називається вимогою цілісності по посиланнях.

Після попереднього аналізу математичної моделі системи і способів організації даних, а також наявного на ринку ПЗ (ієрархічний і мережний способи організації припускають об’єктно-орієнтований підхід до організації даних і на сьогоднішній день маються такі СУБД (наприклад, Jasmin чи Informix Dynamic Server), але на момент розробки можливості їхнього використання не було, у той же час існують дуже „могутні” реляційні СУБД (наприклад, InterBase)) вибір був зроблений на користь реляційного способу організації збереження даних.

Проведемо аналіз початкової інформації з метою визначення складу і структури інформації для подальшої формалізації і побудов інформаційно-логічної моделі даних (ІЛМ). Описана в розділі 2 математична модель, а також додаткові відомості з опису предметної області дозволяють визначити роль реквізитів у взаємозв’язаній інформації, що міститься в документі. На основі такого аналізу встановимо функціональні залежності реквізитів відповідно до рекомендацій і вимог нормалізації даних, після чого проведемо саму нормалізацію. Мета нормалізації полягає в тому, щоб зменшити (але необов’язково усунути) надмірність даних. Проте іноді деяка надмірність даних створюється

навмисно, щоб підвищити ефективність роботи програми. Дамо визначення трьох форм нормалізації БД.

Таблиця знаходиться в першій нормальній формі (1NF), якщо вона має первинний ключ, всі атрибути є простими типами даних і відсутні атрибути, що повторюються. Щоб відповідати 1NF, домени атрибутів повинні бути атомарними значеннями і не повинно бути груп атрибутів, що повторюються. Усі групи атрибутів, що повторюються, повинні бути перенесені в нову таблицю.

Таблиця знаходиться в другій нормальній формі (2NF) тоді, коли вона знаходиться в першій нормальній формі і кожен не ключовий атрибут повністю функціонально залежить від первинного ключа (тобто, у 2NF кожен неключовий атрибут повинен повністю залежати від полів первинного ключа).

Таблиця знаходиться в третій нормальній формі (3NF), якщо вона знаходиться в 2NF і не містить транзитивних залежностей. Транзитивні залежності – це функціональні залежності між не ключовими атрибутами. Будь-який не ключовий атрибут, який функціонально залежить від іншого не ключового атрибуту тієї ж таблиці, створює транзитивну залежність і повинен бути переміщений в іншу таблицю.

Випливаючі функціональні залежності досить тривіальні і очевидно витікають з математичної моделі, тому в подальшому описі вони не наводяться. Також в подальшому викладі опускаються проміжні ступені нормалізації. Тому наведемо лише остаточну інфологічну модель БД (рис. 4.4).

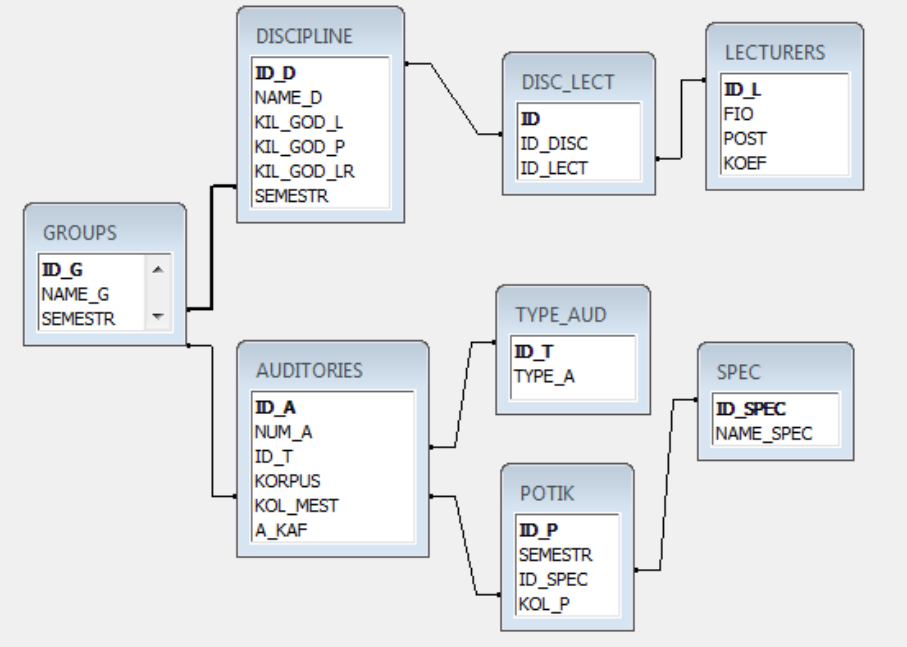


Рисунок 4.4 – Інфологічна модель бази даних «Розклад»

База даних включає в себе наступний перелік таблиць: групи, викладачі, дисципліни, зв'язок дисциплін з викладачами, групи, потоки, аудиторний фонд, тип аудиторії.

Розглянемо структуру бази даних, виходячи з якої формується розклад. База даних DB_1 складається з чотирьох основних таблиць, тобто

$$DB_1 = \langle T_1, T_2, T_3, T_4 \rangle, \tag{4.1}$$

де T_1 – база навчальних дисциплін, T_2 – база викладачів, T_3 – база груп, T_4 – база аудиторій. У свою чергу

$$T_1 = \langle T_1^d, T_1^g, T_1^k, T_1^\Lambda, T_1^P, T_1^l, ... \rangle, \tag{4.2}$$

де T_1^d – назва дисципліни, T_1^g – курс, на якому вивчається дисципліна, T_1^k – група, в якій вивчається дисципліна, T_1^Λ – кількість лекційних годин з цієї дисципліни, T_1^P – кількість годин практичних занять, T_1^l – кількість годин лабораторних робіт тощо;

$$T_2 = \langle T_2^v, T_2^s, T_2^z, T_2^p, T_2^{d_1}, T_2^{d_{1\lambda}}, T_2^{d_{1p}}, T_2^{d_{1l}}, ..., T_2^{d_{k\Lambda}}, T_2^{d_{kp}}, T_2^{d_{kl}} \rangle, \tag{4.2}$$

де T_2^v – прізвище, ім'я, по-батькові викладача, T_2^s – його науковий ступінь, T_2^z – вчене звання, T_2^p – посада викладача, $T_2^{d_i}$ – назва i-ї навчальної дисципліни, яку має читати цей викладач, $T_2^{d_{ia}}$ – кількість лекційних годин з i-ї навчальної дисципліни, $T_2^{d_{ip}}$ – кількість годин практичних занять з i-ї навчальної дисципліни, $T_2^{d_{il}}$ – кількість годин лабораторних робіт з i-ї навчальної дисципліни, $i = \overline{1, w_v}$;

$$T_3 = \left\langle T_3^g, T_3^{ks} \right\rangle, \tag{4.3}$$

де T_3^g – назва (шифр) групи, T_3^{ks} – кількість студентів в цій групі;

$$T_4 = \left\langle T_4^a, T_4^t, T_4^{km} \right\rangle, \tag{4.4}$$

де T_4^a – назва (номер) аудиторії, T_4^t – тип аудиторії, T_4^{km} – кількість місць в цій аудиторії.

Моделі (4.1)-(4.4) носять теоретико-множинний характер і використовуються для початкового змістовного уявлення про об'єкти і суб'єкти навчального процесу. Очевидно, що використовувати інформацію з DB_1 у такому вигляді, як вона там записана, неможливо. Саме тому, згідно певних законів (формул) дані з DB_1 кодуються і формується база даних DB_2, де усі дані мають числовий характер.

Дамо визначення розкладу занять у ВНЗ, яке не претендує на повноту, але містить переважну більшість його атрибутів. Розкладом (Sh) будемо вважати таблицю даних із кортежами

$$Sh = \langle Day, Year, Group, Room, Course, Type, Lecturer, Time \rangle, \tag{4.5}$$

де Day – день проведення занять, $Year$ – курс, на якому навчаються студенти, $Group$ – шифр групи студентів, $Room$ – аудиторія, в якій відбувається навчання, $Course$ – навчальна дисципліна (предмет), $Type$ – тип заняття (лекція, семінар, практичне заняття, консультація, іспит тощо), $Lecturer$ – прізвище, ім'я, по-батькові викладача, $Time$ – номер пари. Для того, щоб мати можливість проводити подальшу обробку даних, необхідно здійснити перетворення даних із різ-

них форматів до формату натуральних чисел. Матимуть місця такі відображення:

$Day : String \to Integer$, $\{1,2,...,6\}$; $D_m = 6$;

$Year : Integer \to Integer$, $\{1,2,...,6\}$; $Y_m = 6$;

$Group : String \to Integer$, $\{1,2,...,200\}$; $G_m = 200$;

$Room : Integer \to Integer$, $\{1,2,...,300\}$; $R_m = 300$;

$Course : String \to Integer$, $\{1,2,...,1000\}$; $C_m = 1000$;

$Type : String \to Integer$, $\{1,2,...,7\}$; $Ty_m = 7$;

$Lecturer : String \to Integer$, $\{1,2,...,200\}$; $L_m = 200$;

$Time : Integer \to Integer$, $\{1,2,...,6\}$; $Ti_m = 6$.

Вище вказано, яких значень набуватимуть атрибути (4.5). Справа позначені константи – їх максимальні значення, які записані для попереднього орієнтування. Для кожного ВНЗ такі значення будуть різними. Загальна можлива кількість розкладів визначається формулою

$$N_{Sh} = D_m \cdot Y_m \cdot G_m \cdot R_m \cdot C_m \cdot Ty_m \cdot L_m \cdot Ti_m. \qquad (4.6)$$

Якщо в (4.6) підставити значення-орієнтири, то кількість розкладів складе $N_{Sh} = 1\ \ 440\ \ 000\ \ 000\ \ 000$. Очевидно, що оцінка за цільовою функцією (2.8) такої кількості розкладів є надто тривалою. Але на практиці потрібно оцінювати меншу їх кількість.

Яким же чином можна скоротити і зменшити кількість можливих варіантів? Один із шляхів (на першому етапі):

- припустити, що в лекційних аудиторіях проводяться виключно лекції, в аудиторіях для семінарських занять – семінари, в лабораторіях – лабораторні роботи і т.і.;
- навчальні дисципліни асоціювати з викладачами.

Такі спрощуючі припущення не завжди і не у повному обсязі можуть спрацювати, зокрема:

– не обов'язково семінари проводяться в спеціальних аудиторіях, вони можуть відбуватись і в лекційних аудиторіях або лабораторіях;

– лекції може вести один викладач, а практичні заняття чи лабораторні роботи – інший;

– на лабораторні роботи група може ділитись на підгрупи.

Для оптимізації процесу формування розкладу здійснимо такі кроки.

1. Виконаємо об'єднання кортежів *Course*, *Lecturer* і *Type*, тобто в одному атрибуті буде зосереджена інформація про навчальну дисципліну (її назву), викладача (ПІП) та тип заняття, яке він проводить. Новий атрибут позначимо INCLT. Має місце відображення

$$<Course, Lecturer, Type> \rightarrow INCLT, \qquad (4.7)$$

$$INCLT : String \times String \times String \rightarrow Integer, \{1,2,...,7000\}, \; I_m = 7000 .$$

Максимальне значення атрибуту *INCLT* є 7000, яке пояснюється тим, що кількість викладачів дорівнює 200, кожен з яких може вести заняття не більше ніж з 5 навчальних дисциплін, кожна з яких має не більше 7 форм занять. Очевидно, що I_m взято для значень-орієнтирів з надлишком. З урахуванням (4.7) кількість можливих розкладів складає 90 720 000 000, що майже в 16 раз менше, ніж у попередньому варіанті.

2. Вважаємо, що базовими документами для формування розкладу є навчальний план, який має вигляд таблиці бази даних з такими кортежами:

$$Lp =< Course, Sem, Ex, Le, \Pr, La, Nw >, \qquad (4.8)$$

де Sem – номер семестру, в якому викладається навчальна дисципліна $Course$; Ex – екзамен або залік; Le – кількість лекційних годин з курсу $Course$ в семестрі Sem; Pr – відповідна кількість практичних занять; La – кількість годин на лабораторні заняття; N_w – кількість тижнів у семестрі Sem,

і документ, який визначає навчальне навантаження викладачів та містить кортежі

$$Np =< Lecturer, Course, Year, Group, N_{Le}, N_{\Pr}, N_{La}, N_{Co}, N_K >, \qquad (4.9)$$

де $N_{Le}, N_{\Pr}, N_{La}, N_{Co}, N_K$ – кількість годин, відведених на лекції, практичні заняття, лабораторні роботи, консультації, іспит або залік, відповідно, з дисципліни $Course$ в групі $Group$.

3. У першому наближенні вважатимемо, що на лабораторні роботи група не ділиться на підгрупи, але така можливість буде передбачена.

Таким чином, розклад (4.5) з урахуванням вищенаведеного набуде вигляду таблиці

$$Sh^* = < Day, Year, Group, Room, INCLT, Time >, \qquad (4.10)$$

і його формування буде здійснюватись в результаті реалізації перетворення

$$Lp \times Np \to Sh^* \qquad (4.11)$$

за умови виконання обмежень (2.2).

Оскільки визначення структури потенційного розв'язку задачі формування розкладу має велике значення для оптимізації обчислювального процесу, то зауважимо, що перетворення тріади <Викладач-Предмет-Форма_заняття> в ціле число повинне здійснюватися по деякому алгоритму. Наведемо його особливості. На першому етапі виписуємо всі форми занять по одному предмету певного викладача. Далі, те ж саме для іншого предмета, але того ж викладача. На наступному кроці виписуємо предмети і форми заняття іншого викладача і т.д. Відзначимо, що викладачі впорядковані за посадами, ступенями і званнями. Якщо зазначені атрибути збігаються, то викладачі виписуються за алфавітом. Після отримання списку у вказаному форматі виконуємо кодування, так, що першому запису відповідатиме 1, а останньому – 800000.

У другому варіанті у вузлах решітки можуть бути кодовані тільки предмети і форми занять, оскільки за першими двома атрибутами і базою даних можна однозначно встановити викладача. Однак цей процес може бути більш ресурсоємним, що належить встановити додатково.

Запропонований метод формування фрагмента потенційного розв'язку дозволить прискорити обчислювальний процес за рахунок забезпечення безпе-

рвності одержуваних рішень. Зокрема, якщо буде виявлено, що деякий викладач не може провести певне заняття з даного предмету, то більш імовірно, що в першу чергу на наступному кроці йому буде запропоновано змінити форму заняття або предмет.

Одночасно з побудовою паралелепіпеда розкладу раціонально будувати ще один паралелепіпед такої ж розмірності, але у вузлах решітки якого знаходитимуться одиниці, які вказують, що в такий-то день на такій-то парі для групи студентів в аудиторії проходить заняття і нуль, якщо там заняття немає. Такий паралелепіпед необхідний для прискорення процесу побудови потенційних розв'язків-розкладів і перевірки їх адекватності, а також для обчислення значень цільової функції.

4.4 Принципи функціонування автоматизованої системи та експериментальна верифікація одержаних результатів

Принципи функціонування автоматизованої системи полягають в розробці оптимального алгоритму для складання розкладу і використання його в системі. Довідники бази даних системи будуть заповнюватися диспетчером, також на виході розкладу буде можливість його зміни вручну. Постійними новими даними для системи буде навчальне навантаження, яке і буде основним ключем при формуванні розкладу. Система буде володіти можливістю редагування, додавання і видалення всіх передбачених даних, також система повинна бути багатокористувацької з різними рівнями доступу.

Під час практичної реалізації системи особлива увага зверталась на задачу написання "ядра" системи – методам розв'язання задачі. Оскільки не ставилася задача написати повнофункціональний комерційний продукт, інтерфейсна частина була написана для цілей тестування ядра і визначення меж застосовності алгоритмів, тому включає мінімум функціональних можливостей. Методи розв'язання написані з використанням об'єктно-орієнтованих технологій, що дозволить в майбутньому легко інкапсулювати їх в подальші модифікації системи, не порушуючи цілісності взаємодії різних алгоритмів.

Ядро системи і інтерфейсна частина були написані на мові програмування Delphi 7.0. Методи розв’язання написані з використанням об’єктно-орієнтованих технологій, що дозволить в майбутньому легко інкапсулювати їх в подальші модифікації системи, не порушуючи цілісності взаємодії різних алгоритмів. Текст об’єктів методів розв’язання задачі приведений в додатку А. БД була реалізована на СУБД InterBase 6.0.

Початкові дані задачі заносяться в таблиці БД за допомогою форм. Одна з таких форм приведена на рис. 4.5.

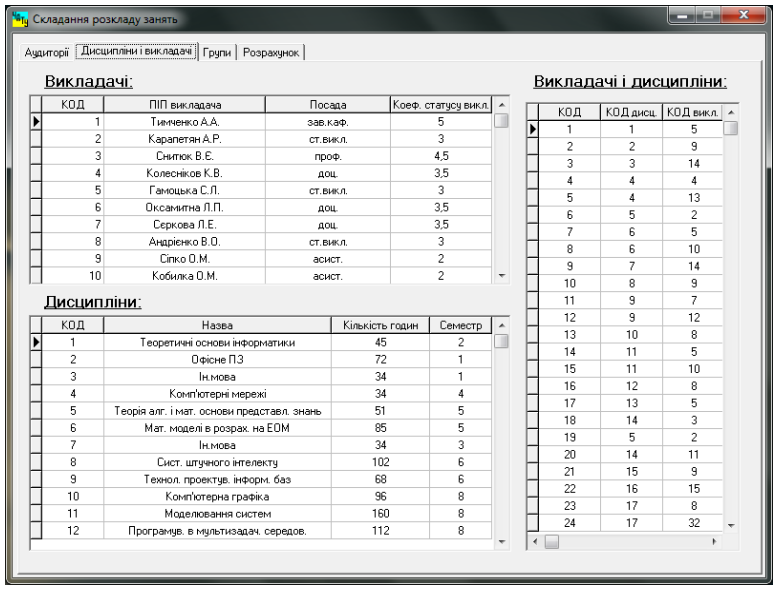


Рисунок 4.5 – Форма занесення початкових даних

Остаточний розклад занять виводиться у вигляді таблиці, приклад якої приведений на рис. 4.6.

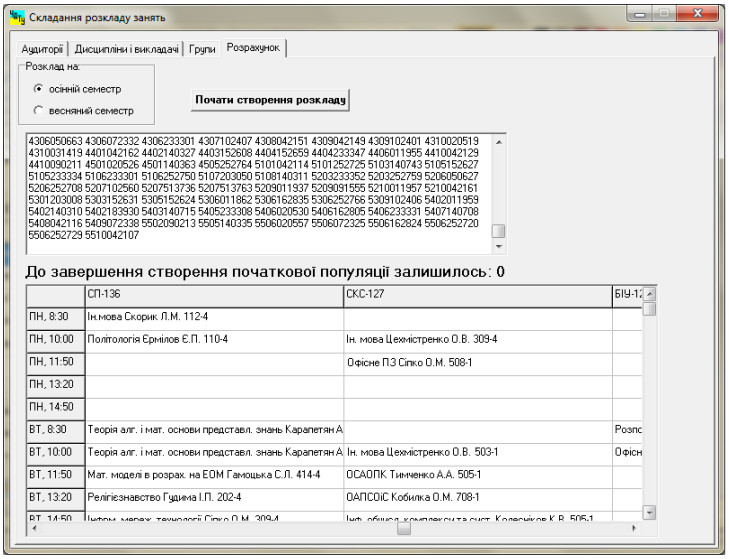


Рис. 4.6 – Приклад розкладу занять

Алгоритм розв’язання задачі тестувався на різних вибірках вихідних даних. У якості тестових початкових даних були використані як реальні дані про групи, викладачів і предмети денної форми навчання Черкаського державного технологічного університету, так і випадково сформовані початкові дані (дисциплінам випадковим чином призначали аудиторії для проведення занять).

Для визначення ефективності й доцільності розроблених моделей, методів і цільових функцій, а також актуальності створеної автоматизованої системи, було проведено розрахунки на створених автоматично і «вручну» розкладах занять. Дослідження проводились на розкладах для факультету інформаційних технологій і систем Черкаського державного технологічного університету, створених на різних вибірках початкових даних: для різних семестрів (осінній, весняний), для різних навчальних років (2014-2015 н.р., 2015-2016 н.р.). Результати порівняльної характеристики значень цільових функцій для різних початкових даних наведені в табл. 4.4.

Таблиця 1 – Значення цільової функції (ЦФ) для різних вибірок початкових даних

Спосіб створення розкладу	Осінній семестр 2014-2015 н.р. (76 викладачів, 43 групи, 49 аудиторій)	Весняний семестр 2014-2015 н.р. (75 викладачів, 43 групи, 46 аудиторій)	Осінній семестр 2015-2016 н.р. (74 викладачі, 45 груп, 49 аудиторій)	Весняний семестр 2015-2016 н.р. (74 викладачі, 45 груп, 47 аудиторій)
вручну	643,4	488,5	604,7	543,4
автоматично	682,3	521,4	663,1	571,2
Відносний приріст ЦФ у відсотках	≈5,7%	≈6,3%	≈8,8%	≈4,9%

Розрахунок значень цільових функцій, отриманих при автоматичному та «ручному» методах створення розкладу навчальних занять, показав, що ефективність розроблених моделей та методів складає близько 5-9% в порівнянні з методом «ручного» складання розкладу занять у ВНЗ.

Крім дослідження оптимальності автоматично складеного розкладу занять, було проведено дослідження залежності часу розв’язання задачі складання оптимального розкладу від розмірності задачі. В середньому, проводилося від 5 до 10 тестів для кожної розмірності початкових даних. В результаті отримано дані, наведені в табл. 4.5.

Таблиця 4.5 – Залежність часу розв’язання задачі складання розкладу від розмірності задачі

Розмірність задачі – (кількість пар) · (кількість груп) · (кількість днів тижня)	Час розв’язання (хв.)					
	max (ГА)	min (ГА)	середній (ГА)	max (МСО)	min (МСО)	середній (МСО)
50	0,8	0,05	0,24	0,6	0,04	0,30
250	3,2	0,9	1,96	2,3	0,6	1,48
450	5,4	2,1	3,5	3,6	1,5	2,57
650	12	3,1	5,8	9	2,2	5,53
850	14	5,2	7,6	10	3,5	6,71
1050	25	10	14,05	16,5	7,3	11,9
1250	39	14,5	18,1	27,7	9,7	18,72
1450	46	19	26,5	32	12,9	22,33
1650	51	25	32	34,7	17,3	25,97
1850	*	30	*	52	21,9	36,95
2050	*	37	*	58	26,3	42,14

де * – задача не розв’язана через перевищення заданого періоду часу (60 хв).

На рис. 4.7 зображений графік залежності середнього часу розв’язання задачі від розмірності задачі (кількості пар на тиждень і кількості груп).

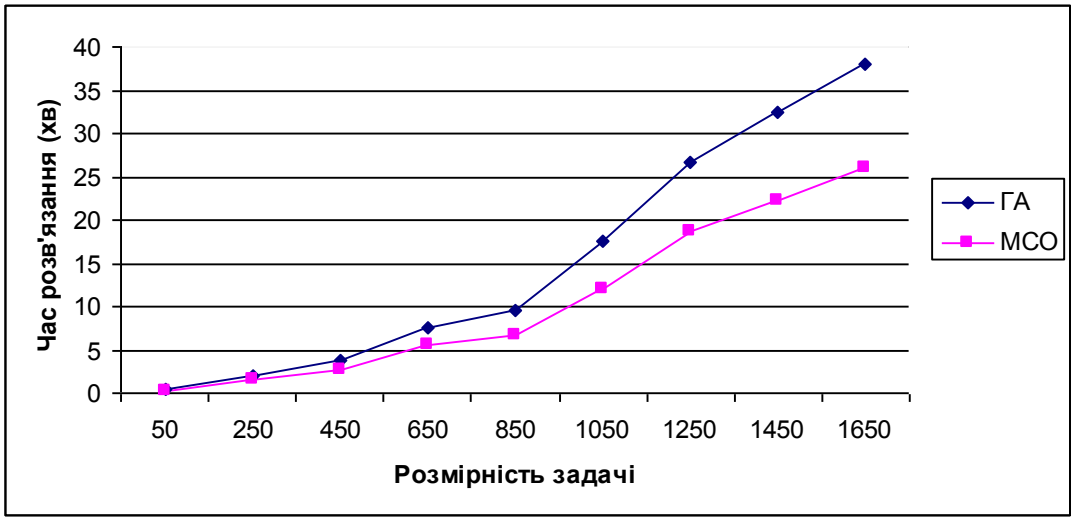


Рисунок 4.7 – Часова динаміка процесу розв’язання задачі

Як видно, модифікований метод спрямованої оптимізації з використанням штрафної функції в порівнянні з генетичним алгоритмом дає 28-31% приросту швидкості знаходження розв’язку за рахунок направленості пошуку оптимуму.

Висновки до розділу 4

В четвертому розділі обґрунтовано вибір засобів створення автоматизованої системи складання розкладу занять, представлено структурний та функціональний опис автоматизованої системи складання розкладу.

Здійснено опис та аналіз проведених експериментів на різних початкових вибірках, який показав доцільність використання модифікованого методу спрямованої оптимізації з використанням штрафної функції в порівнянні з ГА з штрафами, а саме скорочення часу на пошук оптимального розв’язку в середньому на 28-31%.

ВИСНОВКИ

У дисертаційному дослідженні розв’язана науково-технічна задача підвищення ефективності функціонування вищого навчального закладу шляхом розробки моделей і методів формування розкладу навчальних занять, зокрема:

1. Проведений аналіз принципів, моделей, методів та інструментальних засобів свідчить про домінування механістичного підходу до формування розкладу занять у вищих навчальних закладах з мінімальним урахуванням вимог суб’єктів навчального процесу.

2. Виконано формалізовану постановку та розв’язано задачу складання розкладу занять з урахуванням вимог викладачів і студентів та штрафів за їх невиконання, що дозволило здійснювати порівняльний аналіз ефективності розкладів навчальних занять як ступеня задоволення таких вимог.

3. Побудовано математичні моделі, в яких диференційовано враховано вагові коефіцієнти студентів та викладачів, а також пріоритетність вимог студентських груп та індивідуальні переваги викладачів, оптимізація ступеня задоволення яких і визначає процес формування оптимізованого розкладу занять.

4. Сформовано структуру потенційних розв’язків задачі складання розкладів та їх подання у вигляді вузлів решітки прямокутного паралелепіпеда, що дозволило зменшити кількість таких розв’язків. Їх генерація здійснюється з урахуванням «жорстких» обмежень, а нові розв’язки отримують шляхом варіації можливих варіантів, що залишились, з урахуванням їх перевірки на суперечливість. Це дозволяє скоротити час пошуку оптимізованого розкладу на 28-31%.

5. Розроблено інтегральний матрично-еволюційний метод оптимізації цільової функції задачі складання розкладів, синергетичний ефект використання якого досягається через скорочення розмірності задачі та спрямований пошук оптимального значення складної негладкої поліекстремальної залежності, які дозволяють визначати ефективність розкладу як ступінь задоволення вимог студентів і викладачів.

6. Результати експериментальної верифікації свідчать про те, що запропоновані моделі та метод не дозволяють гарантувати отримання оптимального розв’язку, вони спрямовані на пошук розкладів, ефективність яких можна порівняти і, відповідно, вибрати найкращий з них. У порівнянні з методом «ручного» складання розкладу занять у ВНЗ застосування запропонованої технології дозволяє одержати розклад, який на 5-9% є ефективнішим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A study of control parameters affecting online performance of genetic algorithm for function optimisation [Text] / [Schaffler J. D., Caruana R. A., Escheman L. J., Das R.] // Proceedings of the Third International Conference on Genetic Algorithms and their Applications. – San Mateo, CA: Morgan Kaufmann Publishers, 1989. – P. 51-60.

2. Agin N. Optimum Seeking with Branch and Bound [Text] / N. Agin. // Management Science. – 1966. – № 4. – Pp. 176-185.

3. Aloulou M.A. Evaluation Flexible Solutions in Single Machine Scheduling via Objective Function Maximization: the Study of Computational Complexity [Text] / M.A. Aloulou, M.Y. Kovalyov, M.-C. Portmann // RAIRO Oper. Res. – 2007. – №41. – P. 1-18.

4. Balas E. A Note on the Branch-and-Bound Principle [Text] / E. Balas. // Operations Research. – 1968. – №2. – P. 442-444.

5. Baptiste Ph. Constraint-based scheduling: applying constraint programming to scheduling problems [Text] / Ph. Baptiste, C. Le Pape, W. Nuijten. – Kluwer Academic Publishers, 2001.– 198 p.

6. Bellman R. Mathematical aspects of scheduling theory [Text] / R. Bellman // Journal of the Society of Industrial and Applaid Mathematics. – 1956. – Vol. 4. – P. 168-205.

7. Bradshaw's Railway Time Tables and Assistant to Railway Travelling, 1839 [Text] // Encyclopaedia Britannica (11th ed.). – New York: Encyclopaedia Britannica Co. – 1910.

8. Brooks G.N. An algorithm for finding optimal or near – optimal solutions to the production scheduling problem [Text] / G.N. Brooks, C.R. White // J. Ind. Eng. – 1965. – Vol. 16. – №1. – P. 34-40.

9. Brucker P. Resource-constrained project scheduling: Notation, classification, models, and methods [Text] / [P. Brucker, A. Drexl, R. Möhring, K. Neumann] // Europian J. Oper. Res. – 1999. – Vol. 112. – №1. – P. 3–41.

10. Brucker P. Scheduling Algorithms [Text] / P. Brucker. – Springer-Verlag, 2001. – 365 p.

11. Burke E.K. A University Timetabling System Based on Graph Colouring and Constraint Manipulation [Text] / E.K. Burke, D.G. Elliman, R.F. Weare // Journal of Research on Computing in Education. – 1993. – № 28 – P. 46-62.

12. Davis L. Adapting operator probabilities in genetic algorithms [Text] / L. Davis // Proceedings of the Third International Conference on Genetic Algorithms. – La Jolla, CA: Morgan Kaufmann Publishers, 1989. – P. 60-69.

13. Du J. Minimizing total tardiness on one processor is NP-hard [Text] / J. Du, J. Y.-T. Leung // Math. Operation Research. – 1990. – Vol. 15. – P. 483-495.

14. Dubois D. Fuzzy Sets and Systems: Theory and applications [Text] / D. Dubois, H. Prade. – New York: Academic Press, 1980. – Vol. 144. – 1980. – 393 c.

15. Egwali A.O. Synthesis-Algo: An Inclusive Timetable Generating Algorithm [Text] / A.O. Egwali, F.A. Imouokhome// African Journal of Computing & ICT. – 2012. – Vol. 5. – №4. – P. 92-100.

16. Evan S. On the complexity of timetable and multicom-modify flow problems [Text] / S. Evan, A. Itai, A. Shamir. // SIAM J. of Comp. – 1976. – №4. – P. 691–703.

17. Filev D.P. Operations on fuzzy numbers via fuzzy reasoning [Text] / D.P. Filev, R.R. Yager // Fuzzy Sets and Systems 91. – 1997. – P. 137-142.

18. Gafarov E.R. On Lower and Upper Bounds for the Resource-Constrained Project Scheduling Problem [Text] / E.R. Gafarov, A.A. Lazarev, F. Werner // Otto-von-Guericke-Universitat Magdeburg. – 2010. – 27 p. (Preprint 8/10, FMA, Otto-von-Guericke-Universitat Magdeburg).

19. Grefenstette J.J. Optimization of control parameters for genetic algorithms [Text] / J.J. Grefenstette // IEEE Transaction on systems, man end cybernetics. SMC-16(1). – 1986. – P. 122-128.

20. Gröbner M. A General View on Timetabling Problems [Text] / M. Gröbner, P. Wilke. // Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling. – 2002. – P. 234–241.

21. Holland J. H. Adaptation in natural and artificial systems [Text] / J. H. Holland. – Ann Arbor: Univ. of Michigan Press, 1975. – 183 p.

22. Irizarry R. LARES: An Artificial Chemical Process Approach for Optimization [Текст] / R. Irizarry // Evolutionary Computation. – Vol. 12. – № 4 (2004). – P. 435-459.

23. Jian Ni. Genetic Algorithm and its Application in Scheduling System [Text] / Jian Ni, Ning-Ning Yang // Telekomnika. – 2013. – Vol. 11. – № 14. – P.1934-1939.

24. Karaboga D. A comparative study of Artificial Bee Colony algorithm [Text] / D. Karaboga, B. Akay // Applied Mathematics and Computation. – 2009. – № 214. – P. 108-132.

25. Mastrolilli M. Efficient Approximation Schemes for Scheduling Problems with Release Dates and Delivery Times [Text] / M. Mastrolilli // J. of Scheduling. – 2003. – V. 6. – №6. – P. 521-531.

26. Merkle D. Ant Colony Optimization for Resource-Constrainted Project Scheduling [Text] / D. Merkle, M. Middendorf, H. Schmeck// IEEE Transactions on Evolutionary Computation. – 2002. – Vol. 6. – №4. – P. 333-346.

27. Michalewicz Z. Genetic Algorithms, Numerical Optimization and Constraints [Text] / Z. Michalewicz // Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh. – 1995. – P. 151-158.

28. Mingozzi A. An exact alforithm for project scheduling with recource constraints based on new mathematical formulation [Text] / [A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco] // Management Science. – 1998. – Vol. 44. – P. 714-729.

29. Muller T. Some Novel Approaches to Lecture Timetabling [Text] / T. Muller // In Proc. of the 4-th Workshop of Constraint Programming for Decision and Control “CPDC’2002”. – Gliwice. – 2002. – P. 352-369.

30. Novak V. Fuzzy logic, fuzzy sets, and natural languages [Text] / V. Novak // General Systems – №20 (1). – 1991. – 83-97.

31. Odim O.M. On the Fitness Measure of Genetic Algorithm for Generating Institutional Lecture Timetable [Text] / O.M. Odim, B.O. Oguntunde, O.O. Alli // Journal of Emerging Trends in Computing and Information Since. – 2013. – Vol. 4 – № 4. – pp. 436-444.

32. Potts C.N. A branch-and-bound algorithm for the weighted tardiness problem [Text] / C.N. Potts, L.N. Van Wassenhove // Operations Research. – 1985. – Vol. 33. – P. 363-377.

33. Practice and Theory of Automated Timetabling: First International Conference [Text] [Selected Papers: editors Burke E., Ross P.]. – Edinburgh: UK, August 29 – September 1, 1995. – Vol. 115. – 1995. – 381 p.

34. Sipko O.M. Aspects of formulation of the objective function in the problem of scheduling in higher educational institutions based on subjective preferences [Text] / Sipko O.M., Snytyuk V.Ye. // Nauka i Studia. – Polska, Przemyśl: Sp. z o.o. «Nauka i studia». – 2014. – № 15 (125). – P. 49-51.

35. Snytyuk V.Ye. Aspects of formulation of the objective function in the problem of scheduling in higher educational institutions based on subjective preferences [Text] / V.Ye. Snytyuk, O.M. Sipko // Proceedings of XX-th International conference «Knowledge-Dialogue-Solution». – Київ, 2014. – С. 136-137.

36. Stamatopoulos P. Nearly Optimum Timetable Construction Through CLP and Intelligent Search [Text] / P. Stamatopoulos, E. Viglas, S. Karaboyas// International Journal on Artificial Intelligence Tools. – Vol. 7. – №.4. – 1998. – P. 58-71.

37. Syswerda G. Uniform crossover in genetic algorithms [Text] / G. Syswerda // Proceedings of the Third International Conference on Genetic Algorithms. – San Mateo, CA: Morgan Kaufmann Publishers, 1989. – P. 2-9.

38. Thompson J. Variants of simulated annealing for the examination timetabling problem [Text] / K. Dowsland // Annals of Operational Research. – 1996. – № 63. – P. 142-156.

39. Walford E. Somers Town and Euston Square [Text] / E. Walford // Old and New London: Volume 5. – 1878. – P. 340-355.

40. Zadeh L. Fuzzy sets [Text] // Information and Control. – 1965. – №8. – P. 338-353.

41. Ануфрієв М.І. Вищий заклад освіти МВС України: Науково-практичний посібник [Текст] / М.І. Ануфрієв, О.М. Бандурка, О.Н. Ярмиш. – Харків: Ун-т внутр. справ, 1999. – 369 с.

42. Астахова И.Ф. Составление расписания учебных занятий на основе генетического алгоритма [Текст] / И.Ф. Астахова, А.М. Фирас // Вестник ВГУ, серия: Системный анализ и информационные технологии. – 2013. – № 2. – С. 93-99.

43. Балтак С. В. Построение расписаний учебных занятий на основе раскраски вершин графа [Текст] / С. В. Балтак, Ю. Н. Сотсков // Информатика. Объединенный институт проблем информатики НАН Беларуси. – 2006. – № 3(11).

44. Балтак С.В. Построение расписаний учебных занятий на основе раскраски вершин графа [Текст] / С. В. Балтак, Ю. Н. Сотсков // Информатика. Объединенный институт проблем информатики НАН Беларуси. – 2006. – № 3(11). – С. 56-69.

45. Бевз С.В. Розробка автоматизованої системи формування розкладу магістратури / [С.В. Бевз, В.В. Войтко, С.М. Бурбело, А.М. Шоботенко] // Наукові праці ВНТУ. – Вінниця. – Вип.1. – 2009. – С. 1-10.

46. Безгинов А.Н. Обзор существующих методов составления расписаний [Текст] / А.Н. Безгинов, С.Ю. Трегубов // Информационные технологии и программирование. Межвузовский сборник статей. – 2005. – №2 (14). – С. 5-18.

47. Беллман Р., Принятие решений в расплывчатых условиях [Текст] / Р. Беллман, Л. Заде // Вопросы анализа и процедуры принятия решений. – М.: Мир. – 1976. – С.172-215.

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

49. Берзин Е.А. Оптимальное распределение ресурсов и элементы синтеза систем [Текст] / Е.А. Берзин. – М.: Сов.радио, 1974. – 304 с.

50. Бойко О.М. Еволюційна технологія розв’язування задачі складання розкладів навчальних занять [Текст] / О.М. Бойко // Искусственный интеллект. – 2006. – №3. – С. 341-348.

51. Бойко О.М. Еволюційна технологія складання розкладу занять у ВНЗ [Текст] / О.М. Бойко // Збірник тез Четвертої Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених “Шевченківська весна”. – Київ, 2006 р. – С. 34.

52. Верёвкин В.И. Автоматизированное составление расписания учебных занятий вуза с учётом трудности дисциплин и утомляемости студентов [Текст] / В.И. Верёвкин, О.М. Исмагилова, Т.А. Атавин // Доклады ТУСУРа, часть 1. – Томск. – 2009. – №1 (19). – С. 221-225.

53. Вишнякова И.Н. Моделирование и автоматизация составления расписания учебных занятий ВУЗов / И.Н. Вишнякова, С.Ю Разумов. Режим доступа: stp.diit.edu.ua/article/download/17545/15284

54. Волькенштейн М.В. Энтропия и информация [Текст] / М.В. Волькенштейн. – М.: Наука, 1986. – 192 с.

55. Воронин И.В. Информационная система учебно-методического управления университета [Текст] / И.В. Воронин [и др.] // Региональная информатика – 2006 (РИ – 2006): материалы X междунар. конф. – СПб: СПИСУ. – 2006. – С. 201-202.

56. Воронков Ю.В. Организация электронного расписания с использованием Microsoft Excel и VBA [Текст] / Ю.В. Воронков // Современные наукоемкие технологии. – 2009. – №11. – С.64-68.

57. Гаврилова Т.А. Базы знаний интеллектуальных систем [Текст] / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2000. – 384 с.

58. Галузин К.С. Математическая модель оптимального учебного расписания с учетом нечетких предпочтений: Дис. ... канд. физ.-мат. наук: 05.13.18 / Галузин Константин Станиславович. – Пермь. – 2004. – 148 с.

59. Гафаров Е.Р. Доказательство NP-трудности частного случая задачи минимизация суммарного запаздывания для одного прибора [Текст] / Е.Р. Гафа-

ров, А.А. Лазарев // Известия АН: Теория и системы управления. – 2006. – №3. – С. 120-128.

60. Гершуни Г. З. Конвективная устойчивость несжимаемой жидкости [Текст] / Г. З. Гершуни, Е. М. Жуховицкий. – М.: Наука, 1972. – С. 37.

61. Глибовец Н.Н. Генетические алгоритмы и их использование для решения задач составления расписания [Текст] / Н.Н. Глибовец, С.А. Медвидь // Кибернетика и системный анализ. – 2003. – № 1. – С. 95-108.

62. Годлевський М.Д. Розробка та налаштування паралельних генетичних алгоритмів для розв'язання задачі створення розкладу занять вузу на основі GRID-системи [Текст] / М.Д. Годлевський, О.О. Абабілов // Сборник научных трудов "Вестник НТУ "ХПИ": Системний аналіз, управління та інформаційні технології, №67. – Вестник НТУ "ХПИ". – 2010. – С. 17-23.

63. Губаев А. Справедливая зарплата [Текст] / А. Губаев, Ю. Бекенская // The Chief (Шеф). – 2005. – № 3(38). – С. 34-41.

64. Гурин Л.С. Задачи и методы оптимального распределения ресурсов [Текст] / Л.С. Гурин, Я.С. Дымарский, А.Д. Меркулов. – М.: Сов. радио, 1968. – 464 с.

65. Дейт К. Введение в системы баз данных [Текст] / К. Дейт. – М.: Наука, 1980. – 348 с.

66. Емельянов В.В., Теория и практика эволюционного моделирования [Текст] / В.В. Емельянов, В.В. Курейчик, В.М. Курейчик. – М.: ФИЗМАТЛИТ, 2003. – 432 с.

67. Ерунов В. П. Формирование оптимального расписания учебных занятий в вузе [Текст] / В. П. Ерунов, И. И. Морковин. // Вестник ОГУ. – 2001. – №3. – С. 55–63.

68. Єльникова О.В. Вимірювання рівня інформатизації навчального закладу [Електронний ресурс] / О.В. Єльникова // Народна Освіта: Електронне наукове фахове видання АПН України. – Випуск 2(5). – 2008. // Режим доступу: http://www.narodnaosvita.kiev.ua/vupysku/5/statti/4elnikova.htm

69. Жалдак М. І. Основи теорії і методів оптимізації: Навчальний посібник [Текст] / М. І. Жалдак, Ю. В. Триус. – Черкаси: Брама-Україна, 2005. – С. 485-499.

70. Інформаційно-аналітична система управління навчальним процесом ВНЗ [Текст] / [Триус Ю.В., Стеценко І.В., Герасименко І.В., Гриценко В.Г.] // Інформаційні технології в освіті: Збірник наукових праць. Випуск 9.– Херсон: Видавництво ХДУ, 2011. – С. 39-48.

71. Казиев В.М. Введение в анализ, синтез и моделирование систем [Текст] / В.М. Казиев // Интеллектуальные системы. – М.: Изд-во „Открытые системы”. – 2006. – С. 46-49.

72. Караковский В.А. Программы для составления школьного расписания: возможности и опыт применения [Текст] / В.А. Караковский, О.Л. Матусевич // XV Международная конференция ”Применение новых технологий в образовании”. – Троицк. – 2004. – С. 376-380.

73. Каширина И.Л. Введение в эволюционное моделирование [Текст] / И.Л. Каширина. – Воронеж: ВГУ, 2007. – 40 с.

74. Кисляков А.В. Генетические алгоритмы: операции скрещивания и мутации [Текст] / А.В. Кисляков // Информационные технологии. – 2001. – № 1. – С. 29-34.

75. Клеванский Н.Н. Формирование оптимального расписания занятий ВУЗ’а и средства его визуализации [Текст] / Н.Н. Клеванский, С.А. Костин // Материалы XV Международной конференции «Применение новых технологий в образовании». – Троицк: «Байтик», 2004. – С. 380-382.

76. Композиционный генетический алгоритм составления расписания учебных занятий [Текст] / [Кабальнов Ю.С., Шехтман Л.И., Низамова Г.Ф., Земченкова Н.А.] // Вестник УГАТУ, Т.7. – Уфа. – 2009. – №2 (15). – С. 99-107.

77. Конвей Р. В. Теория расписаний [Текст] / Р. В. Конвей, В. Л. Максвел, Л. В. Миллер. – М.: Наука, 1975. – 360 с.

78. Кононов А.В. Актуальные задачи теории расписаний: вычислительная сложность и приближенные алгоритмы [Текст]: дис. … доктора физ.-мат. на-

ук: 01.01.09 / Александр Вениаминович Кононов. – Новосибирск, 2014.– 196 с.

79. Корнеев В.В. Базы данных. Интеллектуальная обработка информации [Текст] / В.В. Корнеев, А.Ф. Гареев, С.В. Васютин, В.В. Райх. – М.: “Нолидж”, 2000. – 352 с.

80. Коффман Э. Г. Теория расписаний и вычислительные машины [Текст] / Э. Г. Коффман. – М.: Наука, 1984. – 335 с.

81. Лагоша Б.А. Комплекс моделей и методов оптимизации расписания занятий в вузе [Текст] / Б.А. Лагоша, А.В. Петропавловская // Экономика и математические методы. – 1993. – № 4. – С. 48-56.

82. Леонова М. В. Моделювання задач складання розкладу занять у ВНЗ: огляд та різні підходи до розв’язування / М. Леонова // Вісник Запорізького національного університету. – 2013. – № 1. – С. 52-59.

83. Лугуев Т.С. Теоретико-графовые модели и методы составления расписаний без прерываний [Текст]: автореферат дис. на соискание научн. степени кандадата физ.-мат. наук: 05.13.18 «Математическое моделирование, численные методы и комплексы программ» / Тимур Садыкович Лугуев. – Астрахань, 2011. – 17 с.

84. Люгер Ф.Дж. Искусственный интеллект. Стратегии и методы решения сложных проблем [Текст] / Ф.Дж. Люгер. – М.: “Вильямс”, 2003. – 864 с.

85. Мак-Вильямс Ф.Дж. Теория кодов, исправляющих ошибки: Пер. с англ. [Текст] / Ф.Дж. Мак-Вильямс, Н. Дж. А. Слоэн. – М.: Связь, 1979. – 744с.

86. Мельник О. О. Одноетапні задачі теорії розкладів у багаторівневій системі планування [Текст]: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 05.13.06 «Інформаційні технології» / Олена Олексіївна Мельник, Національний університет «Львівська політехніка». – Львів, 2013. – 26 с.

87. Мельников А.Ю. Автоматизированное составление расписания занятий в высшем учебном заведении [Текст] / А.Ю.Мельников, М. Сусяк // Современные проблемы информатизации в информационных системах и телекоммуникациях: Сборник трудов. – 2006. – №11. – С. 344–345.

88. Метод декомпозиций для решения комбинаторных задач упорядочения и распределения ресурсов [Текст] / [Д.И. Батищев, Э.Д. Гурман, И.П. Норенков, М.Х. Прилуцький] // Информационные технологии. – 1997. – №1. – С. 29-33.

89. Милехина Т.В. Повышение эффективности кластерных систем обработки информации при решении оптимизационных задач (на примере задачи составления расписания занятий) [Текст]: автореф. дис. на соискание научн. степени канд. техн. наук: спец. 05.13.01 «Системный анализ, управление и обработка информации (в приборостроении)»/ Татьяна Викторовна Милехина. – М., 2011. – 23 с.

90. Митюшкин Ю.И. Soft Computing: идентификация закономерностей нечеткими базами знаний [Текст] / Ю.И. Митюшкин, Б.И. Мокин, А.П. Ротштейн. – Вінниця: УНІВЕРСУМ-Вінниця, 2002. – 145 с.

91. Павлов А.А. Метод группового учёта аргументов и анализа иерархий (МГУАиАИ) в задачах принятия решений [Текст] / А.А. Павлов, А.А. Иванова, Р.А. Зигура. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – К.: ВЕК+. – 2007. – №47. – С. 207-216.

92. Плотнинский Ю.М. Математическое моделирование динамики социальных процессов: Учебное пособие [Текст] / Ю.М. Плотнинский. – М.: МГУ, 1992. – 133 с.

93. Попов Ю.Д. Методи оптимізації [Електронний ресурс] / Ю.Д. Попов, В.І. Тюптя, В.І. Шевченко. – К.: Ел. вид. Ел. бібл. фак-ту кібернетики КНУ ім. Т. Шевченка, 2003. – 215 с. – Режим доступу: http://cyb.univ.kiev.ua/library/books/popov-30.pdf

94. Потьомкін М.М. Удосконалення методу оптимізації роєм часток та його застосування для розв’язання сепарабельних задач нелінійного програмування [Електронний ресурс] / М. М. Потьомкін // Матер. шостої всеукраїнської наук.-практ. інтернет-конференції "Україна наукова" (21-23 грудня 2009 р.), Ч.6. – К.: ТОВ "ТК Меганом", 2009. – С. 48-50. – Режим доступу до журн.: http://intkonf.org/kand-tehnichn-nauk-potomkin-mm-udoskonalennya-metodu-

optimizatsiyi-kolonieyu-bdzhil-ta-yogo-zastosuvannya-dlya-rozvyazannya-zadach-neliniynogo-programuvannya-z-separabelnimi-funktsiyami/.

95. Потьомкін М. М. Класифікація методів еволюційного моделювання [Електронний ресурс] / М. М. Потьомкін // Матер. шостої Всеукр. наук.-практ. інтернет-конф. «Науковий потенціал України 2010» (22-24 березня 2010 р.), Ч. 3. – К.: ТОВ "ТК Меганом", 2010. – С. 33-38. – Режим доступу до журн.: http://intkonf.org/kand-tehnichn-nauk-potomkin-mm-klasifikatsiya-metodiv-evolyutsiynogo-modelyuvannya/.

96. Потьомкін М. М. Шляхи підвищення ефективності еволюційних методів оптимізації [Електронний ресурс] / М. М. Потьомкін // Матер. шостої Всеукр. наук.-практ. інтернет-конф. "Українська наука XXI століття" (16-18 червня 2010 р.), Ч. 4. – К.: ТОВ "ТК Меганом", 2010. – С. 42-47. – Режим доступу до журн.: http://intkonf.org/kandidat-tehnichnih-nauk-potomkin-mm-shlyahi-pidvischennya-efektivnosti-evolyutsiynih-metodiv-optimizatsiyi/.

97. Прилуцкий М.Х. Многокритериальное распределение однородного ресурса в иерархических системах [Текст] / М.Х. Прилуцкий // Автоматика и телемеханика. – 1996. – №2. – С. 24-29.

98. Прилуцкий М.Х. Распределение и упорядочение работ в многостадийных системах [Текст] / М.Х. Прилуцький, Д.В. Попов // Межвузовский тематический сборник научных трудов ВГАВТ: Моделирование и оптимизация сложных систем. – 1999. – С. 123-130.

99. Радченко В.Г. Биология пчел [Текст] / В.Г. Радченко, Ю.А. Песенко. – СПб.: Зоол. ин-т РАН. – 1994. – 350 с.

100. Разумов Ю.А. Спеціалізована модель задачі про призначення для складання розкладу занять вищих навчальних закладів / Ю.А. Разумов. Режим доступу: http://stp.diit.edu.ua/article/viewFile/14094/11909.

101. Ризун Н.О. Применение методов декомпозиции при решении много критериальной задачи автоматизации составления расписания учебных занятий в вузе [Текст] / Н.О. Ризун // Восточно-Европейский журнал передовых технологий. – 2010. – №2/4 (44). – С. 59-70.

102. Романченко І. С. Еволюційні методи оптимізації та їх використання у військовій галузі досліджень: монографія [Текст] / І. С. Романченко, С. Л. Борисюк, М. М. Потьомкін // Центр. НДІ Збройних Сил України. – Житомир: Рута, 2015. – 127 с.: рис., табл. – Бібліогр.: с. 118-127.

103. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети [Текст] / А.П. Ротштейн. – Винница: УНИВЕРСУМ-Винница, 1999. – 320 с.

104. Рубальская О.Н. Автоматизированные системы составления учебных расписаний [Текст] / О.Н. Рубальская. – М.: НИИВО. – 2001. – 68 с.

105. Рубин А.Б. Биофизика: В 2-х кн.: Учеб. для биол. спец. вузов. Кн. 1. Теоретическая биофизика [Текст] / А.Б. Рубин. – М.: Высш. шк., 1987. – 319 с.

106. Саад Алла Ібрагім. Паралельна реалізація генетичних алгоритмів для задач теорії розкладів, заданих на перестановках [Текст]: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 01.05.02 «Математичне моделювання та обчислювальні методи»/ Саад Алла Ібрагім; Харків. нац. ун-т радіо-електрон. – Х ., 2007. – 20 с.

107. Саати Т. Принятие решений. Метод анализа иерархий [Текст] / Т.Саати. – М.: Радио и связь,1993. – 278 с.

108. Самгин Э.Б. Освещение рабочих мест [Текст] / Э.Б. Самгин. – М.: МИРЭА, 1989. – 268 с.

109. Севастьянов С.В. Геометрические методы и эффективные алго- ритмы в теории расписаний [Текст]: дис. … доктора физ.-мат. наук: 01.01.09 / Сергей Васильевич Севастьянов. – Новосибирск, 2000. – 283 с.

110. Семенов С.П. Сравнительный анализ подходов к автоматизации составления расписаний учебных занятий в образовательных учреждениях / С.П. Семенов, Я.Б. Татаринцев // Известия Алтайского государственного университета. – 2010. – С. 103-105.

111. Семенюта И.С. Методика анализа информационной структуры базы данных автоматизированной системы составления расписаний / И.С. Семенюта // Научный журнал КубГАУ. – 2011. – №73(09). – С. 1-11.

112. Сибаров Ю.Б. Охрана труда в вычислительных центрах [Текст] / Ю.Б. Сибаров. – М.: Машиностроение, 1990. – 354 с.

113. Сипко Е.Н. Метод последовательного анализа вариантов решения задачи составления расписания занятий [Текст] / О.М. Сіпко // Искусственный интеллект – Донецк, 2011. – № 1. – С. 243-250.

114. Сипко Е.Н. Оператор мутации в еволюционной технологии решения задачи составления расписаний [Текст] / О.М. Сіпко //International Book Series, Information Science And Computing, Book 7 Artificial Intelligence and Decision Making. – Varna, 2008. – P. 111-116.

115. Сироджа И.Б. Нечеткий дедуктивный вывод в системе квантов знаний для поддержки принятия решений в условиях – неопределенности / И.Б. Сироджа, С.В. Россоха // Открытые информационные и компьютерные интегрированные технологии. – 2006. – Вып. 30. – С. 50-56.

116. Сіпко О.М. Аналіз методів рекомбінації в еволюційній технології розв’язання задачі складання розкладів навчальних занять [Текст] / О.М. Сіпко // Матеріали IV Міжнародної науково-технічної конференції “Комп’ютерні науки та інформаційні технології”. – Львів, 2009. – С. 72.

117. Сіпко О.М. Аналіз результатів та перспективи застосування еволюційного алгоритму розв’язання задачі складання розкладів [Текст] / О.М. Сіпко // Матеріали VI Всеукраїнської конференції молодих науковців «Інформаційні технології в освіті, науці і техніці». – Черкаси: ЧНУ, 2008. – С. 42.

118. Сіпко О.М. Аспекти еволюційної технології для визначення області компромісу між значеннями цільових функцій „викладача” і „студента” у задачі складання розкладів [Текст] / О.М. Сіпко // Матеріали VIII-ї міжнародної конференції „Інтелектуальний аналіз інформації”. – Київ, 2007. – С. 128.

119. Сіпко О.М. Вибір типу кросоверу в еволюційній технології розв’язання задачі складання розкладів навчальних занять [Текст] / О.М. Сіпко // Матеріали XVI Всеукраїнської наукової конференції „Сучасні проблеми прикладної математики та інформатики ”. – Львів, 2009. – С. 193.

120. Сіпко О.М. Визначення статусу суб’єктів навчального процесу на основі аналізу ієрархій [Текст] / О.М. Сіпко // Матеріали Першої Міжнародної науково-технічної конференції ОБЧИСЛЮВАЛЬНИЙ ІНТЕЛЕКТ – 2011 (результати, проблеми, перспективи). – Черкаси, 2011. – С. 240.

121. Сіпко О.М. Врахування вимог студентів і викладачів в задачі складання розкладу занять [Текст] / Сіпко О.М., Котик О.В. // VII Міжнародна школа-семінар «Теорія прийняття рішень»: праці школи-семінару, 29 вересня – 4 жовтня 2014 р. – Ужгород: УжНУ, 2014. – С. 236.

122. Сіпко О.М. Груповий та індивідуальний підходи для визначення вимог суб’єктів навчального процесу [Текст] / О.М. Сіпко // Матеріали Другої Міжнародної науково-технічної конференції “Обчислювальний інтелект-2013”. – Черкаси, 2013. – С. 241.

123. Сіпко О.М. Імплементація цільової функції задачі складання розкладів в метод послідовного аналізу варіантів [Текст] / О.М. Сіпко // Матеріали V Міжнародної школи-семінару «Теорія прийняття рішень». – Ужгород, 2010. – С. 205.

124. Сіпко О.М. Метод послідовного аналізу варіантів розв’язання задачі складання розкладу навчальних занять [Текст] / О.М. Сіпко // Праці VI міжнародної школи-семінару «Теорія прийняття рішень». – Ужгород, 2012. – С. 177-178.

125. Сіпко О.М. Методи обробки нечітко заданих початкових даних в задачі складання розкладу занять [Текст] / О.М. Сіпко // Матеріали Міжнародної науково-практичної конференції «Інформаційні технології в освіті, науці і техніці». – Черкаси, 2012. – С. 63.

126. Сіпко О.М. Порівняльний аналіз операторів мутації в оптимізаційних задачах [Текст] / О.М. Сіпко // Матеріали VI Всеукраїнської конференції молодих науковців ІТОНТ-2010. – Черкаси, 2010. – С. 51.

127. Сіпко О.М. Случайная мутация в задаче составления расписания учебных занятий [Текст] / О.М. Сіпко // Матеріали Міжнародної науково-технічної конференції “SAIT 2010”. – Київ, 2010. – С. 315.

128. Снитюк В.Е. Аспекты эволюционного моделирования в задачах оптимизации [Текст] / В.Е. Снитюк // Искусственный интеллект. – № 4. – 2005. – С. 284-291.

129. Снитюк В.Е. Об особенностях формирования целевой функции и ограничений в задаче составления расписания занятий [Текст] / В.Е Снитюк., Е.Н. Сипко // Математичні машини і системи. – 2014. – № 3. – С. 88-95.

130. Снитюк В.Е. Программирование жизненного цикла сложных систем в условиях неопределенности [Текст] / В.Е. Снитюк // Искусственный интеллект. − 2007. − № 4. − С. 562-567.

131. Снитюк В.Е. Штрафные функции в задаче составления расписания занятий [Текст] / В.Е. Снитюк, Е.Н. Сипко // Математичні машини і системи. – 2015. – № 3. – С. 158-164.

132. Снитюк В.Є. Аспекти формування цільової функції в задачі складання розкладу занять у ВНЗ [Текст] / В.Є. Снитюк, О.М.Сіпко // Матеріали Дев'ятої міжнародної науково-практичної конференції «Математичне та імітаційне моделювання систем. (МОДС – 2014)». – Київ-Жукін, 2014. – С. 349-353.

133. Снитюк В.Є. Еволюційний метод розв'язання задачі складання розкладу занять з використанням функції штрафу [Текст] / В.Є. Снитюк, О.М. Сіпко // Матеріали II Міжнародної конференції «Інформаційні технології та взаємодії» КНУ ім..Тараса Шевченка. – Київ, 2015. – С. 109-111.

134. Снитюк В.Є. Формування цільової функції в задачі складання розкладу занять у ВНЗ на основі суб'єктивних переваг [Текст] / В.Є. Снитюк, О.М. Сіпко // Матеріали 16-ї Міжнародної конференції System Analysis and Information Technologies (SAIT 2014). – Київ, 2014. – С.263-264.

135. Соуса Ф. Полностью эвристическое расписание занятий, управляемое пожеланиями студентов [Текст] / Ф. Соуса, А. Алвес // Наукові праці ВНТУ. – 2009. – № 2. – С. 1–4.

136. Стариков А. Генетические алгоритмы – математический аппарат [Електронний ресурс] / А. Стариков – Режим доступу до ресурсу: http://www.basegroup.ru/genetic/math.htm.

137. Субботин С.А. Метод оптимизации на основе моделирования перемещения бактерий с применением эволюционных операторов [Текст] / С.А. Субботин, А.А. Олейник // Бионика интеллекта. – № 2 (69). – 2008. – С. 137-144.

138. Субботін С.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей [Текст] / С.О. Субботін, А.О. Олійник, О.О. Олійник. – Запоріжжя: ЗНТУ, 2009. – 375 с.

139. Сухарев А.Г. Курс методов оптимизации [Текст] / А.Г. Сухарев, А.В. Тимохов, В.В. Федоров. – М.: Наука, 1986. – 326 с.

140. Сычёв Е. В. Организационно-технический подход к составлению расписания учебных занятий в ВУЗе [Текст] / Е. В. Сычёв. // Современные наукоемкие технологии. – 2009. – №11. – С. 87–89.

141. Томашевський В.М. Складання розкладів занять у дистанційних системах навчання [Текст] / В.М. Томашевський, Ю.Л. Новіков, П.А. Камінська // Вісник НТУУ КПІ. – К.: ВЕК+ (Серія Інформатика, управління та обчислювальна техніка). – Вип. 52. – 2011. – С. 118-130.

142. Трифонов А.Г. Постановка задачи оптимизации и численные методы ее решения [электронный ресурс] / А.Г. Трифонов. // SoftLine Co. Режим доступа: http://matlab.exponenta.ru/optimiz/book_2/index.php (дата обращения: 25.03.2013).

143. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика [Текст] / Ф. Уоссермен. – М.: Мир, 1992. – 240 с.

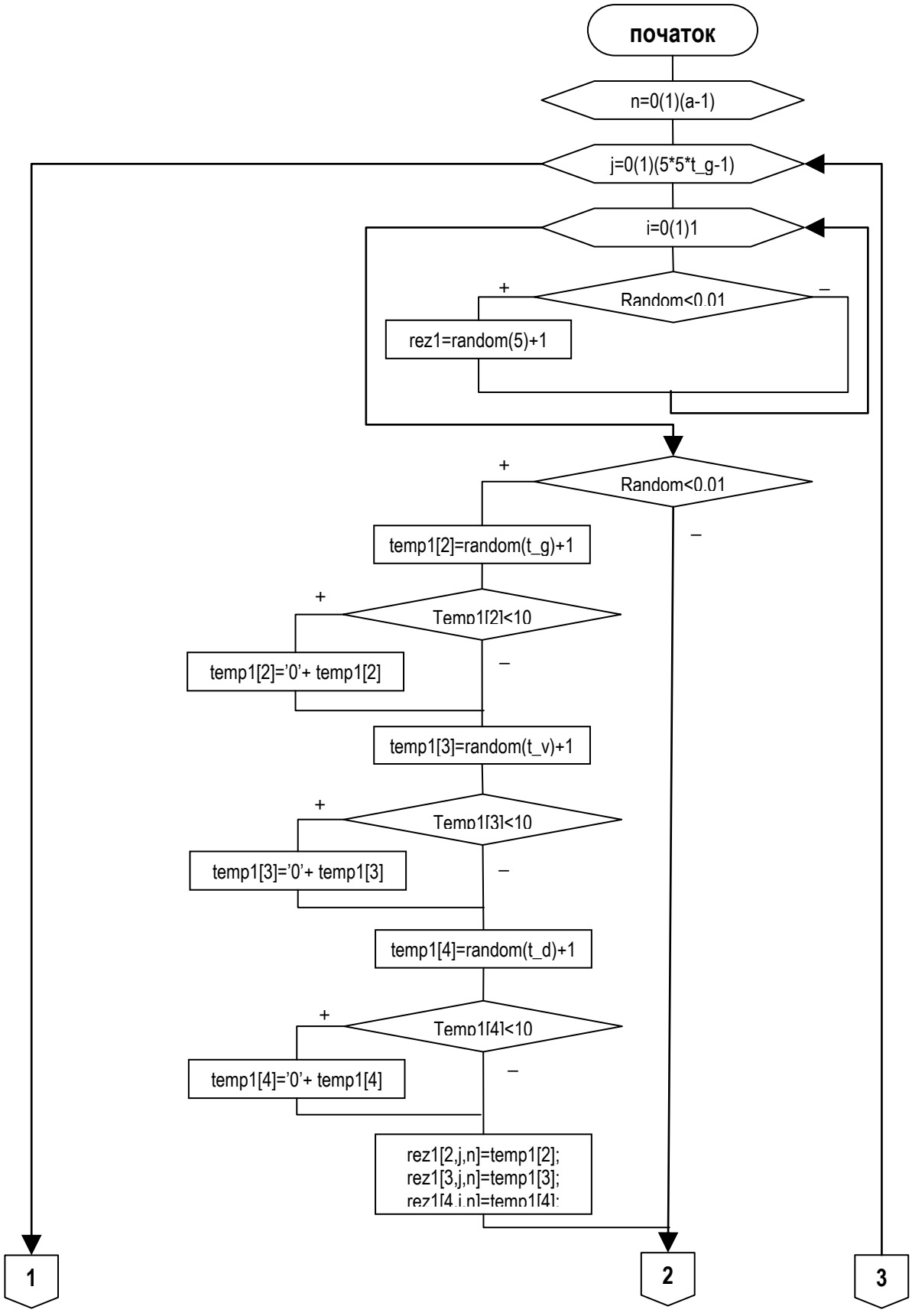
144. Цой Ю.Р. Генетический алгоритм [Текст] / Ю.Р. Цой, В.Г. Спицын. Представление знаний в информационных системах: учебное пособие. – Томск: Изд-во ТПУ, 2006. – 146 с.

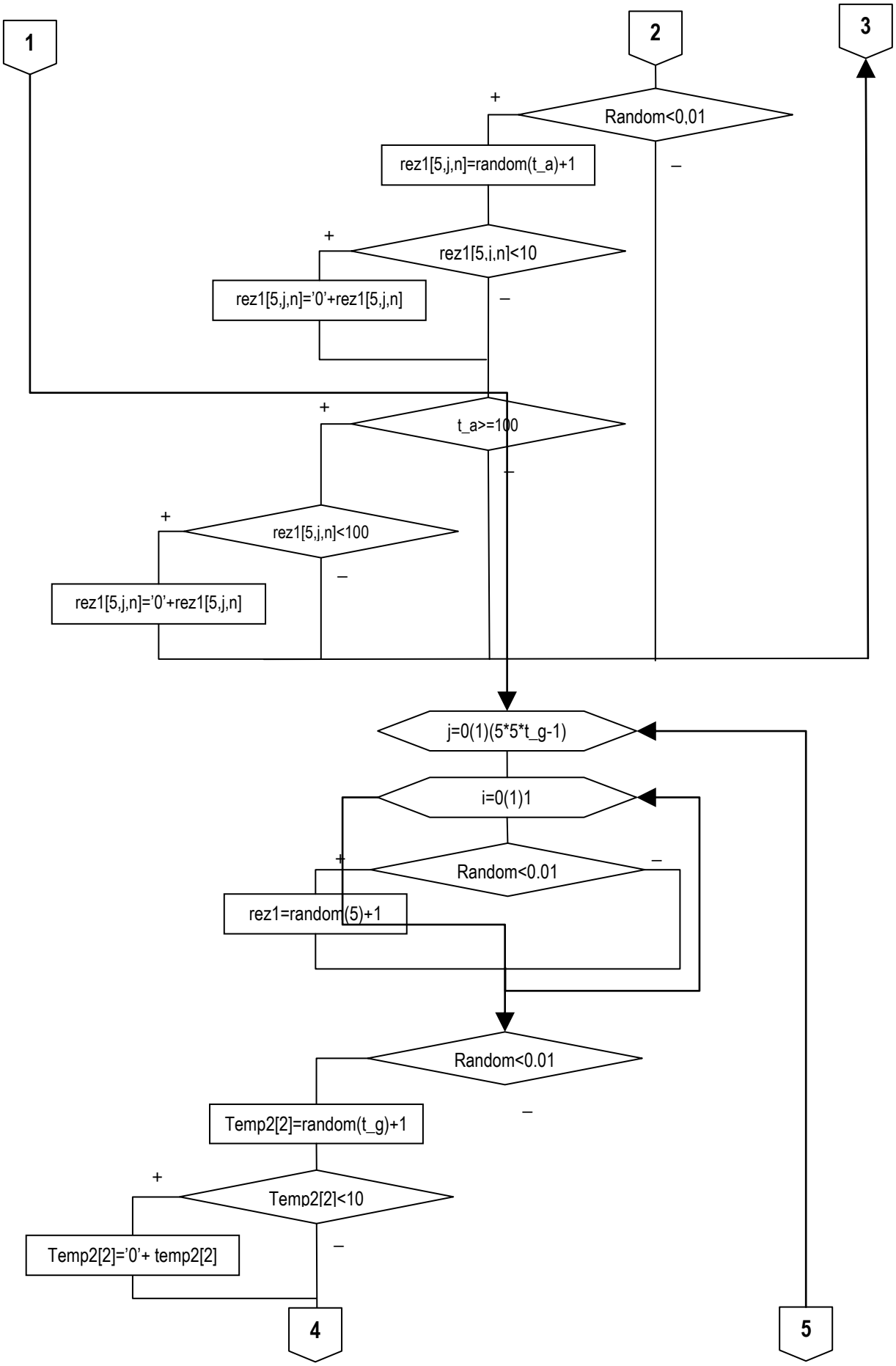
145. Чорний О.П. Формування графік а процесу навчання фахівців інженерних спеціальностей [Електронний ресурс] / О.П. Чорний, Ю.В. Лашко, Т.П. Коваль // Інженерні та освітні технології в електротехнічних і комп'ютерних системах. – 2013. – №4. – Режим доступу до журналу: http://eetecs.kdu.edu.ua.

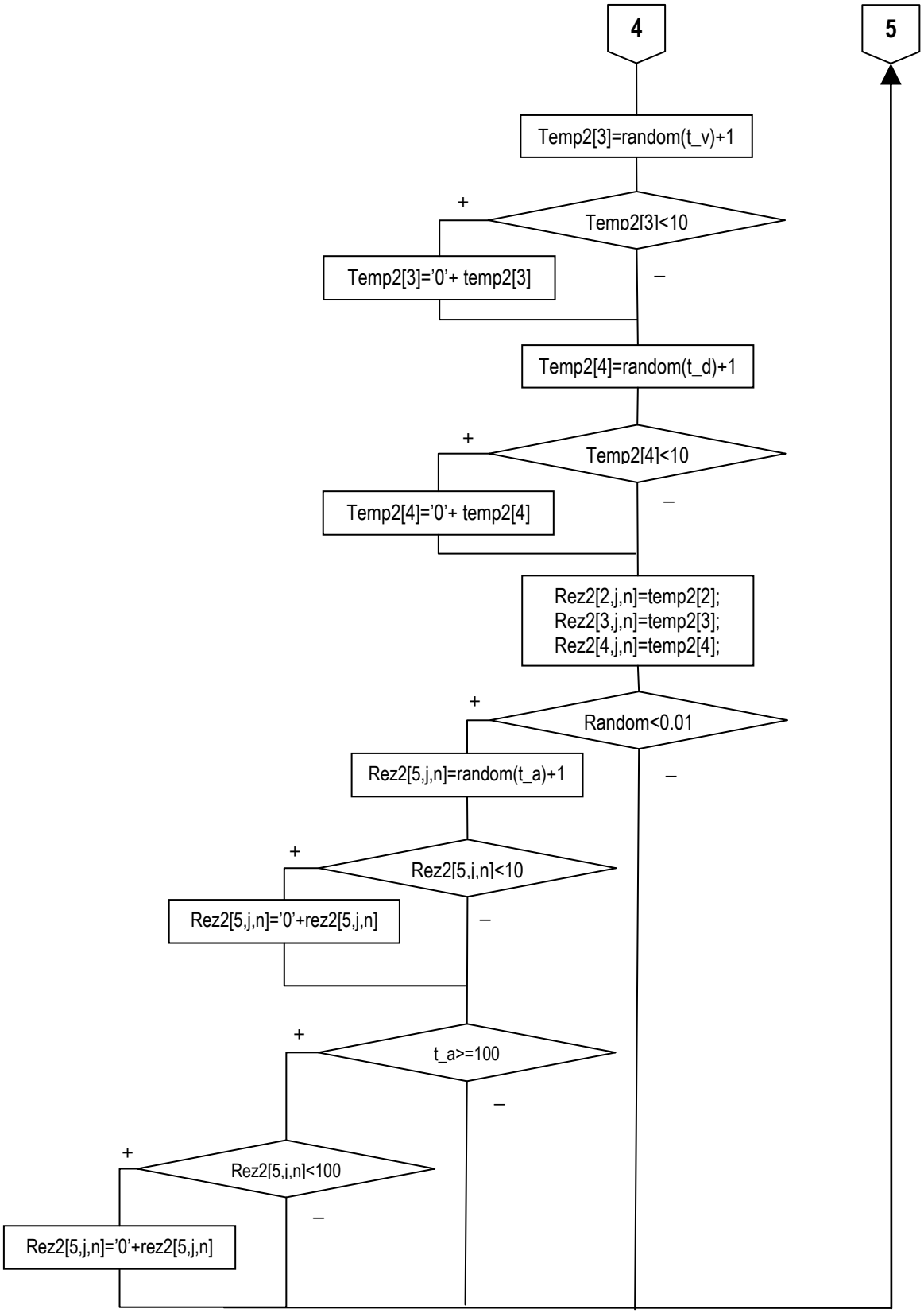
ДОДАТКИ

Додаток А

Блок-схема процедури мутації




1
2
3
Random<0,01
+
–
rez1[5,j,n]=random(t_a)+1
rez1[5,i,n]<10
rez1[5,j,n]='0'+rez1[5,j,n]
t_a>=100
rez1[5,j,n]<100
rez1[5,j,n]='0'+rez1[5,j,n]
j=0(1)(5*5*t_g-1)
i=0(1)1
Random<0.01
rez1=random(5)+1
Random<0.01
Temp2[2]=random(t_g)+1
Temp2[2]<10
Temp2[2]='0'+ temp2[2]
4
5


4
5
Temp2[3]=random(t_v)+1
Temp2[3]<10
+
–
Temp2[3]='0'+ temp2[3]
Temp2[4]=random(t_d)+1
Temp2[4]<10
+
–
Temp2[4]='0'+ temp2[4]
Rez2[2,j,n]=temp2[2];
Rez2[3,j,n]=temp2[3];
Rez2[4,j,n]=temp2[4];
Random<0.01
+
–
Rez2[5,j,n]=random(t_a)+1
Rez2[5,j,n]<10
+
–
Rez2[5,j,n]='0'+rez2[5,j,n]
t_a>=100
+
–
Rez2[5,j,n]<100
+
–
Rez2[5,j,n]='0'+rez2[5,j,n]

Додаток Б

Програмний код для створення автоматизованої системи

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, TabNotBk, Grids, DBGrids, DB, DBTables, StdCtrls,
  Mask, DBCtrls, IBDatabase, IBCustomDataSet, IBTable;

type
  TForm1 = class(TForm)
    TabbedNotebook1: TTabbedNotebook;
    Button1: TButton;
    Memo1: TMemo;
    DataSource2: TDataSource;
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    DataSource4: TDataSource;
    Label1: TLabel;
    DataSource5: TDataSource;
    DBGrid5: TDBGrid;
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    DataSource6: TDataSource;
    DataSource3: TDataSource;
    StringGrid1: TStringGrid;
    Label2: TLabel;
    Label3: TLabel;
    DBGrid3: TDBGrid;
    DBGrid6: TDBGrid;
    DBGrid4: TDBGrid;
    Label4: TLabel;
    DBGrid2: TDBGrid;
    Label5: TLabel;
    Label6: TLabel;
    IBDatabase1: TIBDatabase;
    IBTransaction1: TIBTransaction;
    IBTable1: TIBTable;
    IBTable2: TIBTable;
    IBTable3: TIBTable;
    IBTable4: TIBTable;
    IBTable5: TIBTable;
    IBTable6: TIBTable;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;
  g: array [1..6] of String;
  hromosoma: array [1..500] of Int64;
  p_p: AnsiString;
  gen: String;
  y: array [1..6,1..500,1..20] of AnsiString;
  f1: array [1..20] of Real;
  x1,x2,i,j,r1,r2,r: Int64;
  d1,d2: array [1..30] of Int64;
  b1,b2,w,imov: Real;
  rez1: array [1..6,1..500,1..20] of AnsiString;
  rez2: array [1..6,1..500,1..20] of AnsiString;
  new_hrom1,new_hrom2,new_pop: AnsiString;
  qwerty: array [1..20] of Int64;
  qwe,z,a,fory: Integer;
  temp,t_a,t_v,t_d,t_g,t_vd,t_d_ob: int64;
  grups: array [1..10] of int64;
  discip: array [1..100] of int64;
  disc_lect: array [1..100] of int64;
  gr_disc: array [1..10,1..20] of int64;
  lectors: array [1..10,1..20] of int64;
  kod_dis: array [1..100] of int64;
  tim2: array [1..100] of int64;
  f1_temp: Real;
  Q_temp: Int64;

implementation

uses DateUtils, Math;

{$R *.dfm}

procedure create_hromosoma;
var
i,d,per2,per3,p,dni,par,j,q,w,e,l,per2_temp,per4,sch,a1,q1,per1_temp: Integer;
   s1,s2,s3,s4,s5: Byte;
   per1: array [1..10] of byte;
   per3_temp: Boolean;
begin
```

```
 randomize;
 dni:=1; fory:=1;
{*********************************
*}
 with Form1.IBTable5 do begin
//групи
   First;
   repeat
     par:=1;
     repeat
       p:=1;
       q:=0;
       repeat
         for d:=1 to 6 do g[d]:='';
         for i:=1 to 2 do begin
//дні тижня та пари (по 5)
           g[i]:=IntToStr(random(5)+1);
         end;
{*********************************
*}
        with Form1.IBTable4 do begin
//дисципліни
          per3_temp:=false;
          First;
          repeat
            per3:=Random(t_d)+1;
            for l:=1 to t_d do if (per3=tim2[l])
then per3_temp:=true;
          until (per3_temp=true);
          per2_temp:=0;
          with Form1.IBTable4 do begin
            First;
            repeat
              if
(per3=Form1.IBTable4.FieldByName('ID_
D').Value) then
per2_temp:=Form1.IBTable4.FieldByName
('SEMESTR').Value;
              Next;
            until (per2_temp<>0);
          end;
          per2:=0;
          s1:=0; a1:=1; per1_temp:=0;
          for q1:=1 to t_g do per1[q1]:=0;
          repeat
            if (grups[a1]=per2_temp) then
begin
              Inc(s1);
              per1[a1]:=s1;
            end;
            Inc(a1);
          until (a1=t_g+1);
          with Form1.IBTable5 do begin
            First;
            repeat
              per4:=Random(s1)+1;
              for q1:=1 to t_g do if
(per4=per1[q1]) then per1_temp:=q1;
              sch:=0;
              repeat
                if
(per2_temp=Form1.IBTable5.FieldByName
('SEMESTR').Value) and

(per1_temp=Form1.IBTable5.FieldByName
('ID_G').Value) then
per2:=Form1.IBTable5.FieldByName('ID_
G').Value;
                Next;
                Inc(sch);
              until (per2<>0) or (sch=t_g);
            until (per2<>0);
          end;
          g[3]:=IntToStr(per2);
          g[5]:=IntToStr(per3);
          if StrToInt(g[3])<10 then
g[3]:='0'+g[3];
          if StrToInt(g[5])<10 then
g[5]:='0'+g[5];
        end;
        with Form1.IBTable6 do begin
//викладачі
          Form1.IBTable6.First;
          repeat
            if
(per3=Form1.IBTable6.FieldByName('ID_
DISC').Value)
              then
g[4]:=IntToStr(Form1.IBTable6.FieldByNa
me('ID_LECT').Value);
            Form1.IBTable6.Next;
          until (g[4]<>'');
        end;
        if StrToInt(g[4])<10 then
g[4]:='0'+g[4];
{*********************************
*}
        g[6]:=IntToStr(random(t_a)+1);
//аудиторії
        if StrToInt(g[6])<10 then
g[6]:='0'+g[6];
        if (t_a>=100) then
          if StrToInt(g[6])<100 then
g[6]:='0'+g[6];
```

```
{**********************************
*}

gen:=g[1]+g[2]+g[3]+g[4]+g[5]+g[6];
        y[1,fory,r]:=g[1];
        y[2,fory,r]:=g[2];
        y[3,fory,r]:=g[3];
        y[4,fory,r]:=g[4];
        y[5,fory,r]:=g[5];
        y[6,fory,r]:=g[6];
        hromosoma[fory]:=StrToInt64(gen);
{**********************************
*}
        Form1.IBTable5.Next;
        Inc(fory);
        Inc(p);
       until (p=11);
       Inc(par);
     until (par=6);
     Inc(dni);
   until (dni=6);
   for      j:=1      to      (5*5*t_g)      do
p_p:=p_p+IntToStr(hromosoma[j])+' ';
  end;
end;

procedure poch_pop;
var k: Integer;
   den,para,vyk,my,zminna1: Integer;
begin
  Randomize;
  k:=20; r:=1;
  Form1.Memo1.Lines.Add('Створюється
початкова популяція! Зачекайте...');
  temp:=1;
  repeat
   create_hromosoma;
{**********************************
*********}
   x1:=0; Q_temp:=0;
   den:=1;                          //оцінка і-ої
хромосоми
   repeat
     para:=1;
     repeat
       vyk:=1;
       repeat
         my:=1;
         repeat
           if (den=StrToInt(y[1,my,r]))and
              (para=StrToInt(y[2,my,r]))and
              (vyk=StrToInt(y[4,my,r]))then
              begin
               x1:=1;
               zminna1:=1;
              end
              else  begin  x1:=0;  zminna1:=0;
end;
          Q_temp:=Q_temp+x1;
          Inc(my);
        until          (zminna1=1)          or
(my=5*5*t_g+1);
         Inc(vyk);
       until (vyk=t_v+1);
       Inc(para);
     until (para=6);
     Inc(den);
   until (den=6);
   f1_temp:=1;
   if              (Q_temp>0)              then
f1[temp]:=Q_temp*Form1.IBTable3.FieldB
yName('KOEF').Value
     else f1[temp]:=1;
{  i:=1;
  repeat                          //перевірка
на жорсткі обмеження
   j:=i+1;
   repeat                                {аудито-
рії}
{      if (StrToInt(y[1,i,r])=StrToInt(y[1,j,r]))
and
       (StrToInt(y[2,i,r])=StrToInt(y[2,j,r]))
and
       (StrToInt(y[4,i,r])=StrToInt(y[4,j,r]))
then
         begin
           fory:=j;
           create_hromosoma;
           i:=1; j:=i+1;
         end;
     Inc(j);
   until (j=5*5*t_g);
   Inc(i);
  until (i=(5*5*t_g-1));
  i:=1;
  repeat
    j:=i+1;
    repeat                                {групи}
{      if (StrToInt(y[1,i,r])=StrToInt(y[1,j,r]))
and
       (StrToInt(y[2,i,r])=StrToInt(y[2,j,r]))
and
       (prov[3,i,r]=prov[3,j,r]) then
         begin
```

```
          fory:=j;
          create_hromosoma;
          i:=1; j:=i+1;
         end;
     Inc(j);
   until (j=5*5*t_g);
   Inc(i);
  until (i=(5*5*t_g-1));
  i:=1;
  repeat
   j:=i+1;
   repeat                                {викла-
дачі}
{      if (StrToInt(y[1,i,r])=StrToInt(y[1,j,r]))
and
       (StrToInt(y[2,i,r])=StrToInt(y[2,j,r]))
and
       (prov[4,i,r]=prov[4,j,r]) then
         begin
          fory:=j;
          create_hromosoma;
          i:=1; j:=i+1;
         end;
     Inc(j);
   until (j=5*5*t_g);
   Inc(i);
  until (i=(5*5*t_g-1));
  i:=1;
  repeat
   j:=i+1;
   repeat                                {дисци-
пліни}
{      if (StrToInt(y[1,i,r])=StrToInt(y[1,j,r]))
and
       (StrToInt(y[2,i,r])=StrToInt(y[2,j,r]))
and
       (prov[5,i,r]=prov[5,j,r]) then
         begin
          fory:=j;
          create_hromosoma;
          i:=1; j:=i+1;
         end;
     Inc(j);
   until (j=5*5*t_g);
   Inc(i);
  until (i=(5*5*t_g-1));
{**********************************
*********}
   Form1.Refresh;
   Dec(k);
   Inc(r);
   Form1.Label1.Caption:='До завершення
створення початкової популяції залиши-
лось: '+IntToStr(k);
   Inc(temp);
  until temp=21;
  Form1.Memo1.Text:='';
  ShowMessage('Початкову популяцію
створено!');
  Form1.Memo1.Lines.Add('Початкова по-
пуляція:');
  Form1.Memo1.Lines.Add(p_p);
  Form1.Memo1.Lines.Add('Пошук опти-
мального розкладу! Зачекайте...');
end;

procedure vidbir;
var i,j,t,m,c,n,h:integer;
   b: Real;
   qw: array [1..20] of integer;
begin
  Randomize;
  b1:=0; b2:=0; qwe:=1; b:=0;
  for n:=1 to 20 do qw[n]:=0;
  for n:=1 to 20 do qwerty[n]:=0;
  for t:=1 to 20 do
   begin
    if b<f1[t] then b:=f1[t];
   end;
  for j:=1 to 20 do if f1[j]>=(b*imov) then
begin    // Відбір хромосом для схрещу-
вання
   for n:=1 to 20 do d1[n]:=0;
   for h:=1 to 20 do d2[h]:=0;
   b1:=0; b2:=0; m:=0; c:=0;
   for i:=1 to 20 do begin          //вибір
першої кращої хромосоми для схрещу-
вання
    if (i<>qw[i]) then if (i<>d1[i]) then if
(i<>d2[i]) then begin
     if f1[i]>=(b*imov) then if b1<f1[i]
then begin
      for h:=1 to 20 do d1[h]:=0;
      b1:=f1[i];
      d1[i]:=i;
      m:=i;
     end;
    end;
   end;
   for i:=1 to 20 do begin          //вибір дру-
гої кращої хромосоми для схрещування
    if (i<>d1[i]) then if (i<>d2[i]) then
begin
```

```
      if (i<>qw[i]) then if f1[i]>=(b*imov)
then if b2<f1[i] then begin
        for n:=1 to 20 do d2[n]:=0;
        b2:=f1[i];
        d2[i]:=i;
        c:=i;
       end;
     end;
   end;
   if (m<>0) then qwerty[qwe]:=m;
   if (c<>0) then qwerty[qwe+1]:=c;
   Inc(qwe,2);
   if (m<>0) then qw[m]:=m;
   if (c<>0) then qw[c]:=c;
  end;
end;

procedure crossover;
var i,j,n: Integer;
begin
  z:=0;
  for i:=1 to 20 do if qwerty[i]<>0 then z:=i;
  a:=z div 2;
  if a=0 then begin
    ShowMessage('Невдала початкова по-
пуляція!');
    Form1.Memo1.Text:='';
    p_p:='';
    create_hromosoma;
    poch_pop;
    vidbir;
    crossover;
  end
  else begin
    for n:=1 to a do begin
     r1:=qwerty[2*n-1];
     r2:=qwerty[2*n];
     Randomize;
     for j:=1 to 2 do begin
      i:=1;
      repeat
       if Random<0.5 then
         begin
          rez1[j,i,n]:=y[j,i,r2];
          rez2[j,i,n]:=y[j,i,r1];
         end
         else begin
          rez1[j,i,n]:=y[j,i,r1];
          rez2[j,i,n]:=y[j,i,r2];
         end;
         Inc(i);
      until (i=5*5*t_g+1);
     end;
     i:=1;
     repeat
      if Random<0.5 then
        begin
         rez1[3,i,n]:=y[3,i,r2];
         rez2[3,i,n]:=y[3,i,r1];
         rez1[4,i,n]:=y[4,i,r2];
         rez2[4,i,n]:=y[4,i,r1];
         rez1[5,i,n]:=y[5,i,r2];
         rez2[5,i,n]:=y[5,i,r1];
        end
        else begin
         rez1[3,i,n]:=y[3,i,r1];
         rez2[3,i,n]:=y[3,i,r2];
         rez1[4,i,n]:=y[4,i,r2];
         rez2[4,i,n]:=y[4,i,r1];
         rez1[5,i,n]:=y[5,i,r2];
         rez2[5,i,n]:=y[5,i,r1];
        end;
        Inc(i);
     until (i=5*5*t_g+1);
     i:=1;
     repeat
      if Random<0.5 then
        begin
         rez1[6,i,n]:=y[6,i,r2];
         rez2[6,i,n]:=y[6,i,r1];
        end
        else begin
         rez1[6,i,n]:=y[6,i,r1];
         rez2[6,i,n]:=y[6,i,r2];
        end;
        Inc(i);
     until (i=5*5*t_g+1);
    end
  end;
end;

procedure mutation;
var i,j,n: Integer;
   temp1: array [3..5] of AnsiString;
   temp2: array [3..5] of AnsiString;
begin
  new_pop:='';
  new_hrom1:='';
  new_hrom2:='';
  for i:=1 to (5*5*t_g) do for j:=1 to 6 do for
n:=1 to 20 do y[j,i,n]:='';
  for n:=1 to a do begin
    for j:=1 to (5*5*t_g) do begin
     for i:=1 to 2 do begin
```

```
      if Random<0.01 then
rez1[i,j,n]:=IntToStr(random(5)+1);
    end;
    if Random<0.01 then begin
      temp1[3]:=IntToStr(random(t_g)+1);
      if StrToInt(temp1[3])<10 then
temp1[3]:='0'+temp1[3];
      temp1[4]:=IntToStr(random(t_v)+1);
      if StrToInt(temp1[4])<10 then
temp1[4]:='0'+temp1[4];
      temp1[5]:=IntToStr(random(t_d)+1);
      if StrToInt(temp1[5])<10 then
temp1[5]:='0'+temp1[5];
      rez1[3,j,n]:=temp1[3];
      rez1[4,j,n]:=temp1[4];
      rez1[5,j,n]:=temp1[5];
    end;
    if Random<0.01 then begin
      rez1[6,j,n]:=IntToStr(random(t_a)+1);
      if StrToInt(rez1[6,j,n])<10 then
rez1[6,j,n]:='0'+rez1[6,j,n];
      if (t_a>=100) then
        if StrToInt(rez1[6,j,n])<100 then
rez1[6,j,n]:='0'+rez1[6,j,n];
    end;
  end;
  for j:=1 to (5*5*t_g) do begin
    for i:=1 to 2 do begin
      if Random<0.01 then
rez2[i,j,n]:=IntToStr(random(5)+1);
    end;
    if Random<0.01 then begin
      temp2[3]:=IntToStr(random(t_g)+1);
      if StrToInt(temp2[3])<10 then
temp2[3]:='0'+temp2[3];
      temp2[4]:=IntToStr(random(t_v)+1);
      if StrToInt(temp2[4])<10 then
temp2[4]:='0'+temp2[4];
      temp2[5]:=IntToStr(random(t_d)+1);
      if StrToInt(temp2[5])<10 then
temp2[5]:='0'+temp2[5];
      rez2[3,j,n]:=temp2[3];
      rez1[4,j,n]:=temp2[4];
      rez1[5,j,n]:=temp2[5];
    end;
    if Random<0.01 then begin
      rez2[6,j,n]:=IntToStr(random(t_a)+1);
      if StrToInt(rez2[6,j,n])<10 then
rez2[6,j,n]:='0'+rez2[6,j,n];
      if (t_a>=100) then
        if StrToInt(rez2[6,j,n])<100 then
rez2[6,j,n]:='0'+rez2[6,j,n];
    end;
  end;
  for i:=1 to (5*5*t_g) do
    for j:=1 to 6 do begin
      new_hrom1:=new_hrom1+rez1[j,i,n];
      new_hrom2:=new_hrom2+rez2[j,i,n];
      y[j,i,2*n-1]:=rez1[j,i,n];
      y[j,i,2*n]:=rez2[j,i,n];
    end;

new_pop:=new_pop+new_hrom1+new_hro
m2;
  end;
  p_p:='';
  p_p:=new_pop;
end;

procedure fitness;
var temp,j,i: Integer;
    den,vyk,para,my,zminna1: Integer;
begin
  temp:=1; w:=0;
  Form1.Refresh;
  for i:=1 to 20 do f1[i]:=0;
  j:=1;
  repeat
    x1:=0; Q_temp:=0;
    den:=1;                          //оцінка і-ої
хромосоми
    repeat
      para:=1;
      repeat
        vyk:=1;
        repeat
          my:=1;
          repeat
            if (den=StrToInt(y[1,my,j]))and
               (para=StrToInt(y[2,my,j]))and
               (vyk=StrToInt(y[4,my,j]))then
               begin
                x1:=1;
                zminna1:=1;
               end
               else begin x1:=0; zminna1:=0;
end;
            Q_temp:=Q_temp+x1;
            Inc(my);
          until (zminna1=1) or
(my=5*5*t_g+1);
          Inc(vyk);
        until (vyk=t_v+1);
        Inc(para);
```

```
    until (para=6);
    Inc(den);
  until (den=6);
  f1_temp:=1;
  if (Q_temp>0) then
f1[temp]:=Q_temp*Form1.IBTable3.FieldB
yName('KOEF').Value
    else f1[temp]:=1;
  Inc(j);
 until (j=2*a+1);
end;

procedure TForm1.Button1Click(Sender:
TObject);
var u,i,j,t,h,i2,i3,t1,t2,t3,cel: integer;
  df,i1: Boolean;
  sem,tim,tim1: Integer;
  azs,azs1,ce1,ce2,ce3,ce4,sort2,sort3,s,ry:
integer;
  c2,c3,c4: String;
  sort1: Int64;
begin
 Memo1.Text:='';
 new_pop:='';
 p_p:='';
 t_a:=0; t_v:=0; t_d:=0; t_g:=0; t_vd:=0;
cel:=0;
 with Form1.IBTable6 do begin
//кількість записів у табл. "Викладачі і
Дисципліни"
  First;
  while not EOF do begin
   inc(t_vd);

kod_dis[t_vd]:=IBTable6.FieldByName('ID
_DISC').Value;
   Next;
  end;
 end;
 with Form1.IBTable5 do begin
//кількість записів у табл. "Групи"
  First;
  while not EOF do begin
   inc(t_g);

grups[t_g]:=IBTable5.FieldByName('SEME
STR').Value;
   Next;
  end;
 end;
 with Form1.IBTable1 do begin
//кількість записів у табл. "Аудиторії"
  First;
  while not EOF do begin
   inc(t_a);
   Next;
  end;
 end;
 with Form1.IBTable3 do begin
//кількість записів у табл. "Викладачі"
  First;
  while not EOF do begin
   inc(t_v);
   Next;
  end;
 end;
 with Form1.IBTable4 do begin
//кількість записів у табл. "Дисципліни"
  First;
  while not EOF do begin
   inc(t_d_ob);
   Next;
  end;
 end;
 if Form1.RadioButton1.Checked=true then
begin //кількість записів у табл. "Дисци-
пліни"
  with Form1.IBTable4 do begin
   First;
   while not EOF do begin //на
осінній семестр

sem:=IBTable4.FieldByName('SEMESTR')
.Value;
    if (sem mod 2)=1 then begin
     inc(t_d);

tim2[t_d]:=IBTable4.FieldByName('ID_D').
Value;
      discip[t_d]:=sem;
    end;
    Next;
   end;
  end;
 end else begin
  with Form1.IBTable4 do begin
   First;
   while not EOF do begin //на ве-
сняний семестр

sem:=IBTable4.FieldByName('SEMESTR')
.Value;
    if (sem mod 2)=0 then begin
     inc(t_d);
```

```
tim2[t_d]:=IBTable4.FieldByName('ID_D').
Value;
        discip[t_d]:=sem;
       end;
       Next;
     end;
   end;
  end;
  for tim:=1 to t_g do begin
   azs:=grups[tim];
   h:=0;
   for tim1:=1 to t_d do begin
    azs1:=discip[tim1];
    if (azs=azs1) then begin
     Inc(h);
     gr_disc[tim,h]:=tim2[tim1];
    end;
   end;
  end;
  u:=1; a:=10; z:=0; r:=1; fory:=1;
  imov:=0.95;
  df:=false;
  randomize;
  poch_pop;
  vidbir;
  crossover;
  mutation;
  fitness;
  imov:=imov-0.1;
  repeat
   vidbir;
   for i:=1 to 20 do if qwerty[i]<>0 then
z:=i;
   a:=z div 2;
   if (a<>0) then begin
    crossover;
    mutation;
    fitness;
    if (imov<0.85) then imov:=imov+0.05
else imov:=imov-0.1;
   end else df:=true;
   Inc(u);
   Form1.Refresh;
//                ShowMessage('Виконано
'+IntToStr(u)+' рази!');
  until (df=true);
  p_p:='';
  for ry:=1 to 5*5*t_g do begin

gen:=y[1,ry,1]+y[2,ry,1]+y[3,ry,1]+y[4,ry,1
]+y[5,ry,1]+y[6,ry,1];
   hromosoma[ry]:=StrToInt64(gen);
  end;
  for j:=1 to (5*5*t_g) do begin
//сортування хромосоми методом вставки
   sort1:=hromosoma[j];
   sort2:=1;
   while (sort1>hromosoma[sort2]) do
Inc(sort2);
   for sort3:=j-1 downto sort2 do
hromosoma[sort3+1]:=hromosoma[sort3];
   hromosoma[sort2]:=sort1;
  end;
  for j:=1 to (5*5*t_g) do
p_p:=p_p+IntToStr(hromosoma[j])+' ';
  Form1.Memo1.Lines.Add(p_p);
//  ShowMessage('Відсортовано!');
  i2:=1;
  repeat
   c4:=''; c2:=''; c3:='';
   ce2:=StrToInt(y[4,i2,1]);
   ce3:=StrToInt(y[5,i2,1]);
   ce4:=StrToInt(y[6,i2,1]);
   with Form1.IBTable1 do begin
    First;
    while (c4='') do begin
     if
(ce4=IBTable1.FieldByName('ID_A').Valu
e) then begin

c4:=Form1.IBTable1.FieldByName('NUM_
A').Value+'-
'+IntToStr(IBTable1.FieldByName('KORP
US').Value);
     end;
     next;
    end;
   end;
   with Form1.IBTable3 do begin
    First;
    while (c2='') do begin
     if
(ce2=Form1.IBTable3.FieldByName('ID_L'
).Value)                    then
c2:=Form1.IBTable3.FieldByName('FIO').
Value;
     next;
    end;
   end;
   with Form1.IBTable4 do begin
    First;
    while (c3='') do begin
```

```
      if
(ce3=Form1.IBTable4.FieldByName('ID_D'
).Value)                              then
c3:=Form1.IBTable4.FieldByName('NAME
_D').Value;
      next;
     end;
    end;
    t1:=1;
    repeat
      t2:=1; i1:=false;
      repeat
        Inc(cel);
        if       (StrToInt(y[1,i2,1])=t1)       and
(StrToInt(y[2,i2,1])=t2)
          then begin

Form1.StringGrid1.Cells[StrToInt(y[3,i2,1])
,cel]:=c3+' '+c2+' '+c4;
           i1:=true;
           Form1.Refresh;
          end;
        Inc(t2);
      until (t2=6);
      Inc(t1);
      if (i1=true) then t1:=6;
    until (t1=6);
    cel:=0;
    Inc(i2);
  until (i2=5*5*t_g+1);
  ShowMessage('Виконано!!!');
end;

procedure        TForm1.FormCreate(Sender:
TObject);
var p: Byte;
begin
  randomize;
  Memo1.Text:='';
  p_p:='';
  r:=1; fory:=1;
  imov:=0.95;
  StringGrid1.Cells[0,1]:='ПН, 8:30';
  StringGrid1.Cells[0,2]:='ПН, 10:00';
  StringGrid1.Cells[0,3]:='ПН, 11:50';
  StringGrid1.Cells[0,4]:='ПН, 13:20';
  StringGrid1.Cells[0,5]:='ПН, 14:50';
  StringGrid1.Cells[0,6]:='ВТ, 8:30';
  StringGrid1.Cells[0,7]:='ВТ, 10:00';
  StringGrid1.Cells[0,8]:='ВТ, 11:50';
  StringGrid1.Cells[0,9]:='ВТ, 13:20';
  StringGrid1.Cells[0,10]:='ВТ, 14:50';
  StringGrid1.Cells[0,11]:='СР, 8:30';
  StringGrid1.Cells[0,12]:='СР, 10:00';
  StringGrid1.Cells[0,13]:='СР, 11:50';
  StringGrid1.Cells[0,14]:='СР, 13:20';
  StringGrid1.Cells[0,15]:='СР, 14:50';
  StringGrid1.Cells[0,16]:='ЧТ, 8:30';
  StringGrid1.Cells[0,17]:='ЧТ, 10:00';
  StringGrid1.Cells[0,18]:='ЧТ, 11:50';
  StringGrid1.Cells[0,19]:='ЧТ, 13:20';
  StringGrid1.Cells[0,20]:='ЧТ, 14:50';
  StringGrid1.Cells[0,21]:='ПТ, 8:30';
  StringGrid1.Cells[0,22]:='ПТ, 10:00';
  StringGrid1.Cells[0,23]:='ПТ, 11:50';
  StringGrid1.Cells[0,24]:='ПТ, 13:20';
  StringGrid1.Cells[0,25]:='ПТ, 14:50';
  p:=1;
  with Form1.IBTable5 do begin
   First;
   repeat

StringGrid1.Cells[p,0]:=IBTable5.FieldByN
ame('NAME_G').Value;
    next;
    Inc(p);
   until (p=11);
  end;
end;

end.
```