Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Спосіб побудови освітньої програми за генетичним алгоритмом*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-1* *А.О. Загоруйко*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* *О.А. Пасічник*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* *Р.О. Багрій*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор *О.В. Бармак*
Підпис | Ініціали, прізвище

5 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
«6» 03 *2023* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Спосіб побудови освітньої програми за генетичним алгоритмом»*

2. Завдання видано студенту *Загоруйку Артуру Олеговичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КН Пасічник Олександр Анатолійович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «1» 03 *2023* р. № 5

5. Дата видачі завдання студенту: «3» 03 *2023* р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані: Провести аналіз методів побудови освітньої програми, її мети та завдань; провести аналіз можливостей, переваг та недоліків генетичного алгоритму; визначити послідовність застосування способу побудови освітньої програми за генетичним алгоритмом; реалізувати інформаційну технологію способу побудови освітньої програми за генетичним алгоритмом; провести експериментальне тестування інформаційної технології. Вихідними даними є навчальні дисципліни, що викладаються в Хмельницькому національному університеті, та їх атрибутивні характеристики.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*Грудень 2022*	*виконано*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*Січень 2023*	*виконано*
3	*Робота над розділом 1 – Характеристика предметної області та постановка задачі*	*Січень 2023*	*виконано*
4	*Робота над розділом 2 – Спосіб побудови освітньої програми за генетичним алгоритмом*	*Березень 2023*	*виконано*
5	*Робота над розділом 3 – Програмна реалізація способу побудови освітньої програми за генетичним алгоритмом*	*Квітень 2023*	*виконано*
6	*Оформлення пояснювальної записки згідно вимог*	*Травень 2023*	*виконано*
7	*Попередній захист кваліфікаційної роботи бакалавра*	*Травень 2023*	*виконано*
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*Червень 2023*	*виконано*

Виконавець: *студент 4 курсу, група КН-19-1* _____ *А.О. Загоруйко*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *О.А. Пасічник*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *Спосіб побудови освітньої програми за генетичним алгоритмом*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-1 Загоруйко Артур Олегович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Пасічник Олександр Анатолійович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
67	31	0	28	2

Мета кваліфікаційної роботи полягає в розробці способу побудови освітньої програми за генетичним алгоритмом. Для досягнення поставленої мети визначені наступні задачі дослідження: провести аналіз методів побудови освітньої програми, її мети та завдань; провести аналіз можливостей, переваг та недоліків генетичного алгоритму; визначити послідовність застосування способу побудови освітньої програми за генетичним алгоритмом; реалізувати інформаційну технологію способу побудови освітньої програми за генетичним алгоритмом; провести експериментальне тестування інформаційної технології.

Результатом виконання кваліфікаційної роботи бакалавра є розробка способу побудови освітньої програми за генетичним алгоритмом та його реалізація як інформаційної системи.

Ключові слова: генетичний алгоритм, освітня програма

Виконавець: *студент 4 курсу, група КН-19-1* (Курс, група виконавця) _____ (Підпис) *А.О. Загоруйко* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ОП	Освітня програма
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ЗВО	Заклад вищої освіти
ЄКТС	Європейська кредитно трансферна-накопичувальна система
ІС	Інформаційна система
ОК	Освітній компонент
ЗК	Загальна компетентність
ФК	Фахова компетентність
ПРН	Програмні результати навчання

Вступ

Кваліфікаційна робота бакалавра присвячена розробці способу побудови освітньої програми за генетичним алгоритмом.

Актуальність теми. Сучасна Україна швидко та наполегливо інтегрується у європейській та світовий освітній простір. Загальносистемними рисами сучасної вищої освіти є, з одного боку, широка автономія закладів освіти, а, з іншого, зростаючі вимоги здобувачів щодо результатів навчання з точки зору конкурентоспроможності на ринку праці усіх рівнів локалізації. Інтегруючим результатом можливостей закладу вищої освіти в частині надання освітніх послуг задля задоволення очікувань та потреб здобувачів є освітня програма. Освітня програма повинна забезпечувати можливість набуття здобувачем переліку компетентностей, визначених Державним стандартом для відповідної спеціальності, з урахування ряду обмежень, зокрема щодо тривалості навчання, інтервального навантаження здобувачів, забезпечення якості освіти шляхом введення граничної кількості дисциплін що вивчаються протягом семестру, урахування думок та пропозицій різноманітних категорій стейкхолдерів, змін та тенденцій на ринку праці тощо. Зазначені обставини роблять задачу гаранта та групи забезпечення по формуванню освітньої програми нетривіальним завданням, до того ж обтяженим достатньо короткими циклами перегляду, внесення змін та оновлення. Сучасні інтелектуальні інформаційні технології надають широкий інструментарій технологій, методів та засобів, зокрема генетичні алгоритми, які можуть бути застосовані для ефективного та оперативного вирішення задачі формування освітньої програми, зроблять роботу гаранта освітньої програми та групи забезпечення об’єктивно ефективною та якісною.

Мета і задачі роботи. Мета кваліфікаційної роботи полягає в розробці способу побудови освітньої програми за генетичним алгоритмом.

Для того, щоб досягнути поставленої мети визначені наступні задачі і дослідження:

- проаналізувати методи та способи побудови освітньої програми;

- провести аналіз можливостей, переваг та недоліків генетичного алгоритму;
- визначити послідовність застосування способу побудови освітньої програми за генетичним алгоритмом;
- реалізувати інформаційну технологію способу побудови освітньої програми за генетичним алгоритмом;
- провести тестування інформаційної системи.

Об’єкт дослідження – спосіб побудови освітньої програми за генетичним алгоритмом.

Предмет дослідження – моделі, алгоритми та засоби для створення способу побудови освітньої програми за генетичним алгоритмом.

Ключові слова: спосіб побудови, генетичний алгоритм, освітня програма

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області та постановка задачі

Освітня програма – це комплексний набір освітніх компонентів, таких як дисципліни, індивідуальні завдання, контрольні заходи та інші компоненти, які сплановані і організовані з метою досягнення визначених програмних результатів навчання [1].

Освітні програми в Україні регулюються Міністерством освіти і науки України. Основними типами освітніх програм є:

– державні стандарти базової та повної загальної середньої освіти, які регулюють навчання учнів у загальноосвітніх навчальних закладах;

– державні стандарти вищої освіти, які регулюють навчання студентів у вищих навчальних закладах;

– навчальні плани та програми професійно-технічної освіти, які регулюють навчання учнів у професійно-технічних навчальних закладах.

– навчальні програми підвищення кваліфікації та професійного розвитку, які регулюють навчання працівників різних сфер діяльності [2].

ОП в Україні зазвичай мають певний набір обов'язкових дисциплін, а також вибіркові дисципліни, які студент може обрати самостійно. Особлива увага приділяється забезпеченню здобуття студентами необхідних компетенцій та практичних навичок у вибраних напрямах навчання [3]. Крім того, в ОП передбачені різноманітні форми контролю знань, такі як іспити, заліки, курсові та дипломні роботи, що дозволяє перевіряти рівень знань студентів та їх здатність застосовувати їх на практиці [4].

Формування та реалізація освітніх програм є основним завданням закладів вищої освіти. При оцінці освітньої діяльності конкретного навчального закладу, основна увага приділяється якості його освітніх програм. А саме освітні програми піддаються акредитації, яка є одним з ключових компонентів забезпечення якості вищої освіти [5].

Перш за все, ОП слід розглядати як цілісну систему освітніх компонентів. Інакше кажучи, освітня програма повинна бути охарактеризована цілісністю та повнотою своїх компонент. ОП повинна містити не тільки перелік навчальних дисциплін, але й порядок їх вивчення. Загалом, повноцінна освітня програма формується за допомогою обгрунтованого поєднання навчальних програм, за якими вивчаються різні навчальні дисципліни. В реальних обставинах, кожна освітня програма розробляється відповідним викладачем та схвалюється на рівні відповідного підрозділу закладу вищої освіти. Напрямок кожної навчальної програми в найкращому випадку повинен визначатися чітким розумінням місії, яку визначає для себе заклад вищої освіти та характеру відповідної ОП. Іншими словами, не можна на практиці пропонувати одну й ту саму навчальну програму для різних освітніх програм. Наприклад, навчальні програми з філософії для студентів філологічного та спортивного факультетів (відповідних до навчальних програм) повинні містити свої особливості, проте ці особливості не можуть бути істотними змінами [6].

Освітня програма повинна бути привабливою для здобувача вищої освіти та бути запропонованою закладом вищої освіти, який хоче заслуговувати на довіру людей, які бажають навчатись у цьому закладі. Приваблювати повинна саме освітня програма, а не зовнішні фактори, такі як регіональна наближеність до ЗВО, перспектива отримувати стипендію або уникнення проходження ЗНО шляхом вступу до навчальних закладів на основі неповної середньої освіти. По факту, у вище викладених випадках не йдеться про привабливість освітніх програм, а про зовнішні чинники при виборі таких програм. Низький рівень підготовки більшості студентів, що розпочали навчання за відповідною освітньою програмою, демонструє низьку якість самої програми, її недосконаліть і недостатнє фінансування з місцевого або державного бюджету. Тому, справедливими є нарікання на неприпустимість існування освітніх програм, які створені лише для забезпечення видимості якісної роботи закладу вищої освіти [7].

Оцінка привабливості освітної програми визначається можливістю, що надається після її завершення, а саме: подальше працевлаштування або

продовження навчання. ОП не може бути по-справжньому привабливою, якщо вона не дозволяє або знайти роботу, що й передбачає часто університетська освіта. Якщо ОП передбачає обов'язкове опанування додаткової освітньої програми в майбутньому, і ця ситуація обумовлена суспільними практиками, варто розглянути можливість відмовитися від цієї програми [8].

Основними складовими освітньої програми є:

– Мета освітньої програми: визначає, що повинен знати, розуміти та вміти студент після завершення навчального процесу;

– Зміст навчання: включає в себе теми, питання, проблеми та практичні навички, які мають бути охоплені в процесі навчання;

– Методи навчання: визначають, які підходи, методи та стратегії мають бути використані для забезпечення ефективного навчання студентів;

– Оцінювання навчальних досягнень: визначається система оцінювання, критерії та методи оцінювання для визначення рівня навчальних досягнень студентів;

– Ресурси та обладнання: включає в себе ресурси, необхідні для забезпечення навчального процесу, такі як підручники, комп’ютери, лабораторне обладнання та інше.

Крім того, існує перевірка якості з боку роботодавців та інших зацікавлених сторін. В сучасних умовах економічного застою і фактичного зникнення окремих високотехнологічних секторів економіки за останні десять-п’ятнадцять років, можна висловити багато зауважень щодо неефективності цього контролю. Проте, марно сподіватися, що освіта може бути якісною і не відірваною від реальних секторів економіки та запитів суспільства. Саме тому ми повинні максимально користуватися можливостями, які гарантує чинне законодавство України, для залучення роботодавців, викладачів, студентів та представників інших зацікавлених груп на всіх рівнях забезпечення якості освіти починаючи з розробки стандартів і ОП до участі в їх акредитації [9].

Оцінка успішності навчання полягає у визначенні рівня компетентностей випускників. Компетентності відображають здатність особи якісно здійснювати

професійну та подальшу освітню діяльність, вони формуються шляхом поєднання знань, вмінь та практичних навиків, манери мислення, професійних, поглядів на життя та людських якостей, морально-етичних цінностей [10]. З практичної точки зору, для розуміння різниці між компетентностями та результатами навчання в контексті підходу, що визначений в законодавстві, також можна провести своєрідну аналогію зі створенням програмного забезпечення, де компетентності являють собою аналог вимог замовника або ринку, які є вихідними даними для розробки програмного продукту продукту, а програмні результати навчання є аналогом тих характеристик, за якими ми тестуємо готовий програмний продукт перед виходом на ринок [11].

Тому формулювання компетентностей та результатів навчання у стандарті можуть бути схожими. Це створює певні методологічні виклики. Загальні компетентності відповідно до методології Tuning [12]:

– Здатність до критичного мислення;
– Комунікативні навички;
– Вміння працювати в команді;
– Лідерські навички;
– Професійна етика та громадянська відповідальність;
– Навички ефективної комунікації з представниками галузі, які не є професіоналами;
– Уміння працювати в інтернаціональній команді;
– Навички самостійної роботи;
– Усвідомлення культурних особливостей та традицій різних країн світу;
– Управління проєктами та розробка проєктних рішень
– Прояв підприємницького духу та ініціативи;
– Навички самоосвіти та навчання протягом життя;
– Здатність до креативного мислення;
– Здатність до адаптації і гнучкості.

Під час визначення програмних результатів навчання, яких здобувач вищої освіти повинен досягнути, за рекомендаціями європейських фахівців з

великим досвідом, розподіляють їх на конкретні групи, котрі є важливими для конкретних інженерних спеціальностей. Зокрема, для інженерних спеціальностей існують такі групи програмних результатів, як фундаментальні та інженерні науки, інженерний аналіз, інженерне проєктування, інженерна діяльність, загальні вміння, такі як розуміння і знання, інженерний аналіз, проєктування, інженерна діяльність, припущення, комунікація та робота в команді, а також навчання впродовж життя [13].

Далі приведений приклад можливого переліку програмних результатів навчання для студентів бакалаврських освітніх програм, які навчаються на інженерних спеціальностях:

– поглиблені знання та розуміння дисциплін, що становлять основу спеціальності, на достатньому рівні для досягнення програмних результатів програми, включно із ознайомленістю з останніми досягненнями науки та техніки;

– рівень знань та поглиблене розуміння математики та інших точних наук, які є основою інженерної спеціалізації, достатній для досягнення інших цілей освітньої програми;

– здатність використовувати сучасні інженерні інструменти, технології та програмне забезпечення для моделювання, аналізу та вирішення інженерних проблем;

– здатність аналізувати складні інженерні продукти, процеси і системи відповідно до спеціалізації; обирати і застосовувати придатні типові аналітичні, розрахункові та експериментальні методи; правильно трактувати результати цих досліджень;

– вміння працювати у команді та спілкуватися ефективно з іншими фахівцями;

– усвідомлення етичних та професійних аспектів роботи в інженерній сфері;

– Здатність навчатися самостійно та постійно підтримувати свої професійні знання та навички у сфері інженерії [14].

1.2 Аналіз сучасних підходів до побудови освітньої програми

Сьогодні, побудова освітньої програми базується на різних підходах та технологіях, які орієнтовані на різні аспекти та потреби сучасного світу.

Основні з них описані нижче:

– Індивідуалізований підхід – передбачає створення програми, що відповідає конкретним потребам та інтересам кожного учня окремо. Цей підхід базується на розумінні того, що учні мають різні темпи [15];

– Інтегрований підхід – передбачає поєднання різних предметів в одній програмі з метою формування більш повного та комплексного розуміння предмета. Наприклад, при вивченні історії може бути включений курс літератури, який дасть учням можливість зрозуміти контекст того, що вивчається [16];

– Компетентнісний підхід – передбачає формування учнів таких компетенцій, як здатність до аналізу, синтезу, критичного мислення, розв'язання проблем тощо. Цей підхід передбачає активну роль учня в навчальному процесі, зосередженість на розвитку практичних навичок та вмінь, необхідних для успішного функціонування в різних сферах життя [17];

– Технологічний підхід – цей підхід передбачає використання сучасних технологій навчання та викладання, таких як E-learning, Blended learning, Flipped classroom тощо. Метою є покращення ефективності та якості навчання, а також забезпечення доступності та гнучкості [18];

– Інтердисциплінарний підхід – поєднання різних предметів у єдину навчальну програму з метою формування комплексних знань та умінь. Використовуються інтерактивні методи, дослідницькі проекти, спільні проекти різних факультетів та інші [19].

На сьогоднішній день в більшості університетів, особливо тих, що мають велику кількість факультетів та кафедр, існує значна кількість освітніх програм. Ось декілька прикладів:

– Київський національний університет імені Тараса Шевченка має більше 100 різних програм бакалаврату та магістратури на різних факультетах, таких як факультети філології, філософії, математики та інформатики та багатьох інших [20].

– Національний технічний університет «Харківський політехнічний інститут» має більше 150 різних програм бакалаврату та магістратури на різних факультетах, таких як факультети електроніки та комп'ютерних технологій, механічного та енергетичного інженерінгу та багатьох інших [21].

– Львівська національна музична академія імені Миколи Лисенка, має більше 30 різних програм бакалаврату та магістратури на факультеті музики та музичного мистецтва, що охоплюють різні напрямки музичної освіти [22].

Наведені приклади показують, що університети можуть мати велику кількість різних ОП на різних рівнях та спеціальностях.

Актуальною вимогою до здобувача освіти після завершення навчання є його конкуреннтоспроможність на ринку праці та можливість одразу після прийняття на роботу інтегруватися у наявні виробничі ланцюги. Здатність це забезпечити визначаються рівнем та обсягом набутих програмних результатів навчання, які, у свою чергу, обумовлюються загальними та фаховими компетентностями. Номенклатура компетентностей визначається Державними стандартами, а освітня програма повинна забезпечити повне та якісне їх набуття здобувачами освіти.

Якісно сформована освітня програма в частині переліку обов’язкових освітніх компонент фактично визначає програмні результати навчання в комплексі із здобутими компетентностями. Складність сучасного ринку освітніх послуг у поєднанні з різновекторними вимогами стейкхолдерів обумовлюють значну номенклатуру навчальних дисциплін, що робить питання формування освітньої програми нетривіальною задачею, обтяжену необхідністю дотримання вимог різноманітних нормативних документів, й, зокрема, щодо періодичності оновлення [23].

Зазначені обставини обумовлюють об’єктивну необхідність застосування автоматизованих підходів, як приклад інформаційних технологій, для

формування освітніх програм, що гарантує досягнення найбільш якісного результату у прийнятні терміни.

Сучасні інформаційні технології пропонують широку номенклатуру кібернетичних методів вирішення задач такого класу, й, зокрема, гентичний алгоритм.

1.3 Аналіз можливостей, переваг, недоліків та області застосування генетичного алгоритму

При вирішенні складних задач оптимізації, головною метою є пошук рішення, яке є «найкращим» у порівнянні з попереднім або початковим рішенням. Це пояснюється тим, що для складних систем часто необхідно знайти хоча б будь яке задовільне рішення, а досягнення оптимального результату стає менш пріоритетним. Методи, що базуються на математичних обчисленнях, а також перераховані методи, спрямовані на пошук оптимального рішення, часто складно застосовувати на практиці через їхню велику складність [24].

З огляду на це, доцільно використовувати генетичні алгоритми для вирішення складних оптимізаційних задач. Ефективність генетичного алгоритму у вирішенні конкретної оптимізаційної задачі залежить від різних факторів, таких як генетичні оператори, вибір відповідних значень параметрів мутації, функції придатності і спосіб представлення поставленої задачі на хромосомі.

Задача описується таким чином, щоб її рішення можна було представити у вигляді масиву, такий масив зазвичай називають «хромосомою». Початкова популяція, або деяка кількість початкових «індивидів», створюється в масиві випадковим чином. Індивиди оцінюються за допомогою функції придатності, яка присвоює кожній особині певне значення, вказуюче на її життєздатність. За використанням отриманих параметрів «пристосованості», обираються особини, допущені до схрещення. Генетичні оператори, зазвичай, включають схрещування (кросовер) та мутацію, і вони застосовуються до індивидів, створюючи наступне покоління. Індивиди наступного покоління також оцінюються з використанням генетичних операторів, після чого проводиться

селекція та мутація. Цей еволюційний процес відтворюється протягом кількох поколінь, поки не будуть виконані критерії зупинки алгоритму [25].

Таким критеріями зазвачай виступають:

– досягнення максимальної кількості часу, відведеного для еволюції.

– пошук глобального, або «найкращого» вирішення поставленої задачі;

– досягнення максимальної кількості поколінь, відведених для еволюції;

Генетичні алгоритми також можливо використати для пошуку рішень у великих і складних просторах пошуку оптимально рішення задачі, включаючи ситуації з великою кількістю можливих варіантів вирішення задачі [26].

1.4 Аналіз існуючих рішень для систем побудови освітньої програми

Хоча на сьогоднішній день існує багато програмних рішень та інструментів, які допомагають викладачам створювати та змінювати навчальні плани та ОП, автоматизовані рішення, які повністю замінять людську роботу , на сьогодні не існують.

Однак, варто відзначити, що певні елементи автоматизації вже використовуються в освіті, наприклад, платформи для електронного навчання, які дозволяють створювати та редагувати навчальний матеріал, а також контролювати його використання та результати навчання.

Деякі з найбільш відомих програмних рішень для організації навчального процесу наведені нижче:

aSc Timetables – ця програма розроблена для всіх типів початкових та середніх навчальних закладів [27].

Вона дозволяє користувачам вводити різні параметри, такі як кількість учнів, предметів, викладачів та кімнат, а потім автоматично генерує розклад на основі цих даних.

Програма також має вбудовані інструменти для оптимізації розкладу, забезпечуючи, що заняття не пересікаються, учні не мають вільного часу, викладачі не мають занадто багато занять підряд і т.д. Крім того, користувачі

можуть налаштовувати різні параметри, такі як час початку і закінчення занять, час на перерву між заняттями та інші параметри.

Рисунок 1.1 – Головна сторінка «aSc Timetables»

Moodle – це відкрите програмне забезпечення, яке дозволяє створювати освітні курси та взаємодіяти з учнями в режимі онлайн. Moodle має безліч функцій та можливостей для побудови освітніх програм, таких як розміщення завдань та іспитів [28].

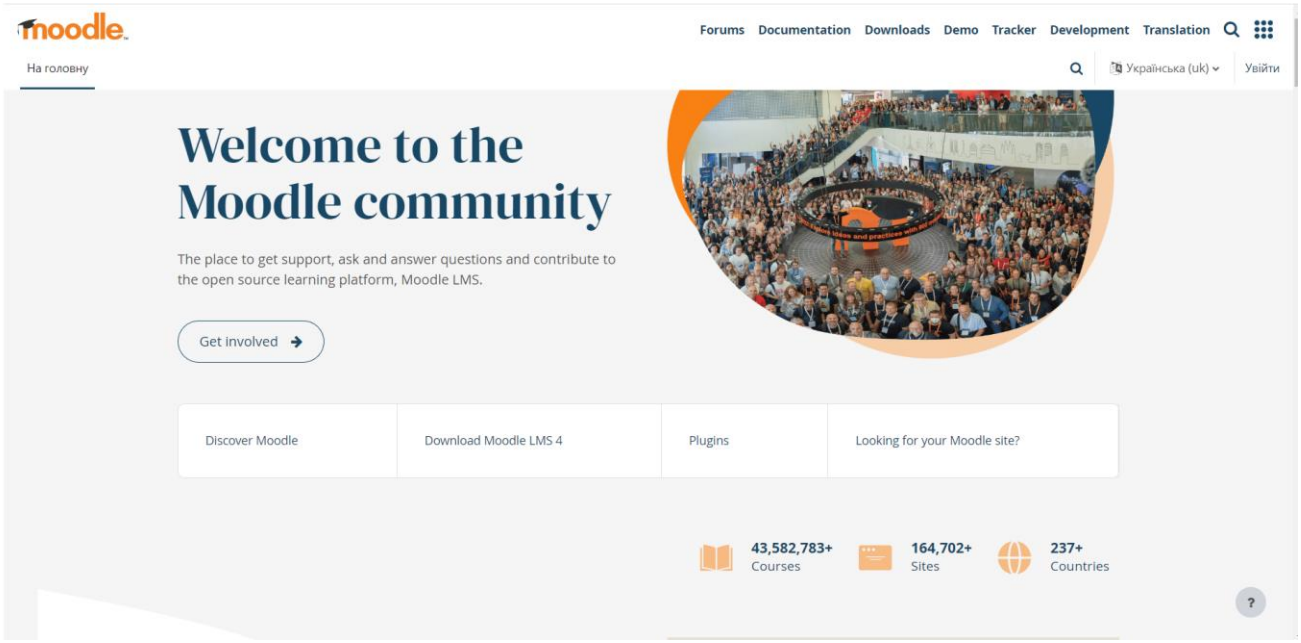


Рисунок 1.2 – Головна сторінка «Moodle»

Платформа Moodle має великий спектр функціональних можливостей, типовий для систем електронної освіти, систем управління навчанням або

віртуальних навчальних середовищ. Завдяки Moodle викладачі мають можливість створювати ефективні онлайн-курси для навчання, а учні, інтрументи для елективного навчання. Платформу можна використовувати як для шкільного та студентського навчання, так і для підвищення кваліфікації та бізнес-навчання.

1.5 Мета, задачі та вимоги до реалізації інформаційної системи

Наміром для створення інформаційної системи для побудови ОП за генетичним алгоритмом стала проблема, автоматизації процесу побудови освітньої програми з використанням методу генетичного алгоритму. Генетичний алгоритм полягає в застосуванні принципів еволюції до пошуку оптимального рішення.

Для досягнення визначеної мети необхідно виконати наступні завдання:

- проаналізувати методи та способи побудови освітньої програми;
- проаналізувати можливості, переваги та недоліки застосування генетичного алгоритму для поставленого завдання;
- визначити послідовність застосування способу побудови освітньої програми за генетичним алгоритмом;
- реалізувати інформаційну технологію способу побудови освітньої програми за генетичним алгоритмом;
- провести експериментальне тестування інформаційної системи.

Розділ 2 Спосіб побудови освітньої програми за генетичним алгоритмом

2.1 Загальні положення щодо побудови освітньої програми

Освітня програма є ключовим документом, який визначає зміст та структуру освіти на певному рівні (наприклад, бакалаврату чи магістратури).

Загальні положення щодо структури освітніх програм ЗВО на території України, визначаються відповідно до чинного законодавства та нормативних документів, що затверджені Міністерством освіти і науки України. Також, в Україні існує державний стандарт вищої освіти, який містить вимоги до змісту, організації та проведення навчального процесу у вищих навчальних закладах.

Університети мають право самостійно формувати власні освітні програми на базі державного стандарту вищої освіти та інших нормативних документів, з урахуванням специфіки своєї діяльності та особливостей навчального процесу.

Згідно з державним стандартом, освітня програма повинна включати наступні складові (рисунок 2.1):

– Мета та завдання програми;

– Зміст та обсяг програми, який включає в себе навчальні дисципліни, практичні заняття, інші форми навчання та самостійну роботу студентів;

– Вимоги до знань, умінь та загальних і фахових компетентностей, які повинен мати випускник програми;

– Вимоги до кваліфікаційних рівнів та компетентностей викладачів, які займаються викладанням дисциплін у програмі;

– Обсяг та тривалість навчального процесу, який включає в себе кількість кредитів, тривалість та інтенсивність навчання, організацію практичної підготовки та наукової роботи студентів;

– Вимоги до матеріально-технічної бази, необхідної для забезпечення якісної підготовки студентів.

При розробці освітньої програми університети зазвичай залучають викладачів, науковців та фахівців у відповідній галузі знань. Вони враховують

актуальні тенденції, вимоги роботодавців, наукові досягнення та потреби студентів. Освітня програма повинна забезпечувати формування необхідних знань, умінь та компетентностей у випускників, що відповідають сучасним вимогам суспільства та ринку праці.

Рисунок 2.1 – Складові ОП

Крім того, побудова освітньої програми має враховувати вимоги до рівня підготовки випускників, визначені відповідними професійними стандартами та ринковими потребами. Наприклад, при побудові освітньої програми з медицини, потрібно враховувати міжнародні стандарти та вимоги до медичної підготовки, щоб забезпечити високий рівень компетентності випускників.

Важливо враховувати інноваційні технології та тенденції у світовій освіті при побудові освітньої програми. У світі існують різноманітні підходи до побудови ОП, які включають різні навчальні підходи, методи та форми організації навчання. Тому важливо вивчати досвід інших країн та організацій, які займаються розробкою та реалізацією ОП. Наприклад, у зв'язку зі зростаючими потребами у цифровій грамотності, в освітніх програмах можуть бути включені дисципліни з програмування, аналітики даних та інформаційних технологій.

Важливо також забезпечити реалізацію освітньої програми з використанням сучасних методів та технологій навчання, таких як онлайн-курси, відео-лекції, віртуальні лабораторії та інше. Загалом, побудова освітньої програми є складним та відповідальним процесом, який потребує залучення кваліфікованих фахівців.

Після схвалення освітньої програми університет здійснює її реалізацію шляхом проведення навчальних занять, практик, лабораторних робіт, самостійної роботи студентів та інших активностей, які сприяють засвоєнню матеріалу. Завершенням освітньої програми є отримання випускником кваліфікації або академічного ступеня, передбачених програмою.

Важливо зазначити, що освітня програма є динамічним документом, який може підлягати перегляду та оновленню з метою адаптації до змін у суспільстві, розвитку науки та технологій, а також з урахуванням фідбеку випускників та роботодавців. Постійне вдосконалення освітніх програм сприяє підвищенню якості освіти та готовності випускників до професійної діяльності.

2.2 Загальна послідовність побудови освітньої програми

Побудова освітньої програми в закладах вищої освіти складається з ряду етапів, які спрямовані на створення структурованого і збалансованого навчального плану, відповідного цілям та потребам студентів, роботодавців і суспільства. (рисунок 2.2):

Рисунок 2.2 – Етапи побудови освітньої програми в закладах вищої освіти

1. Аналіз потреб на ринку праці та вимог «Державного стандарту вищої освіти»;

На цьому етапі проводяться дослідження, що дозволяють з'ясувати потреби ринку праці та вимоги до освіти в конкретній галузі. Враховуючи ці

дані, визначаються основні пріоритети програми, її зміст, навчальні плани, вимоги до випускників та інші показники (рисунок 2.3).

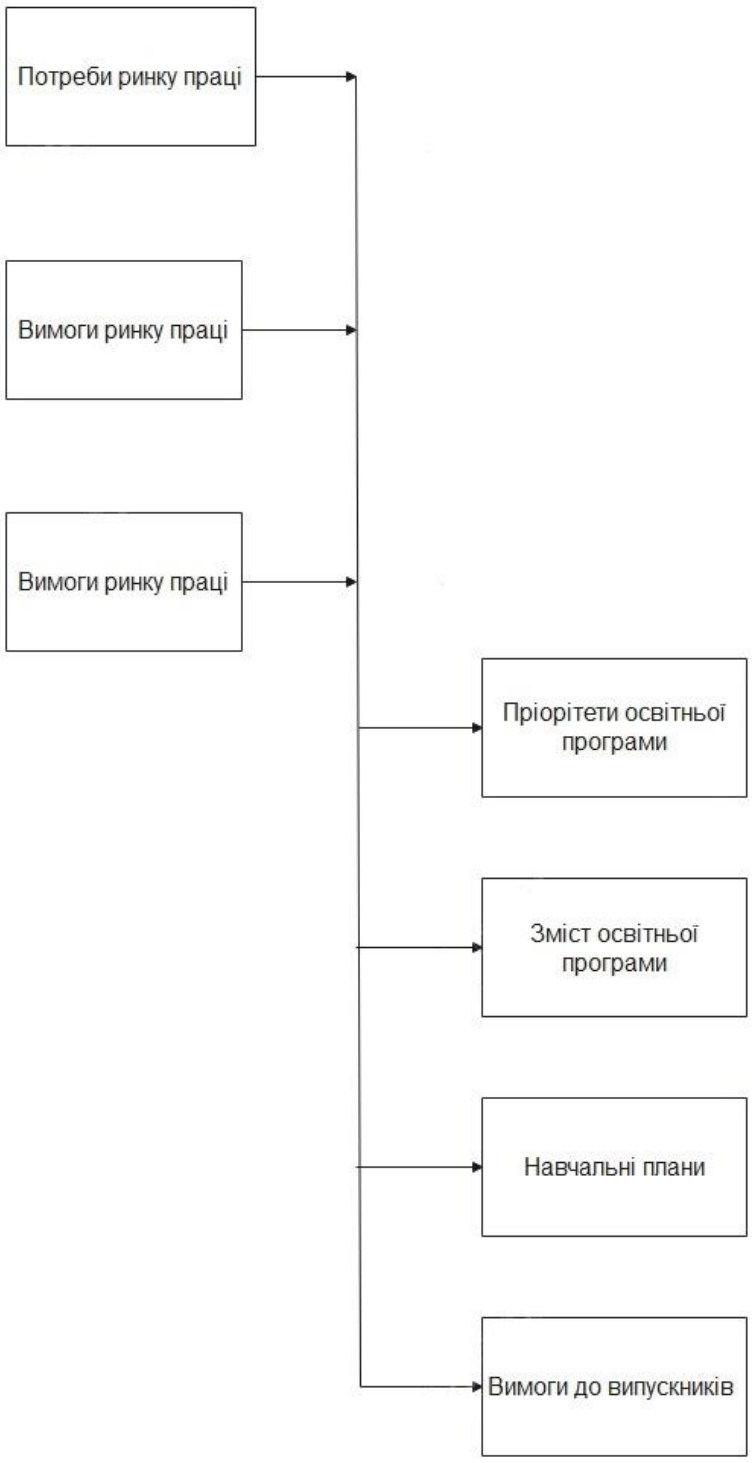


Рисунок 2.3 – Аналіз потреб та вимог

2. Визначення спеціалізіції освітньої програми;

Під час другого етапу виконуються наступні дії (рисунок 2.4):

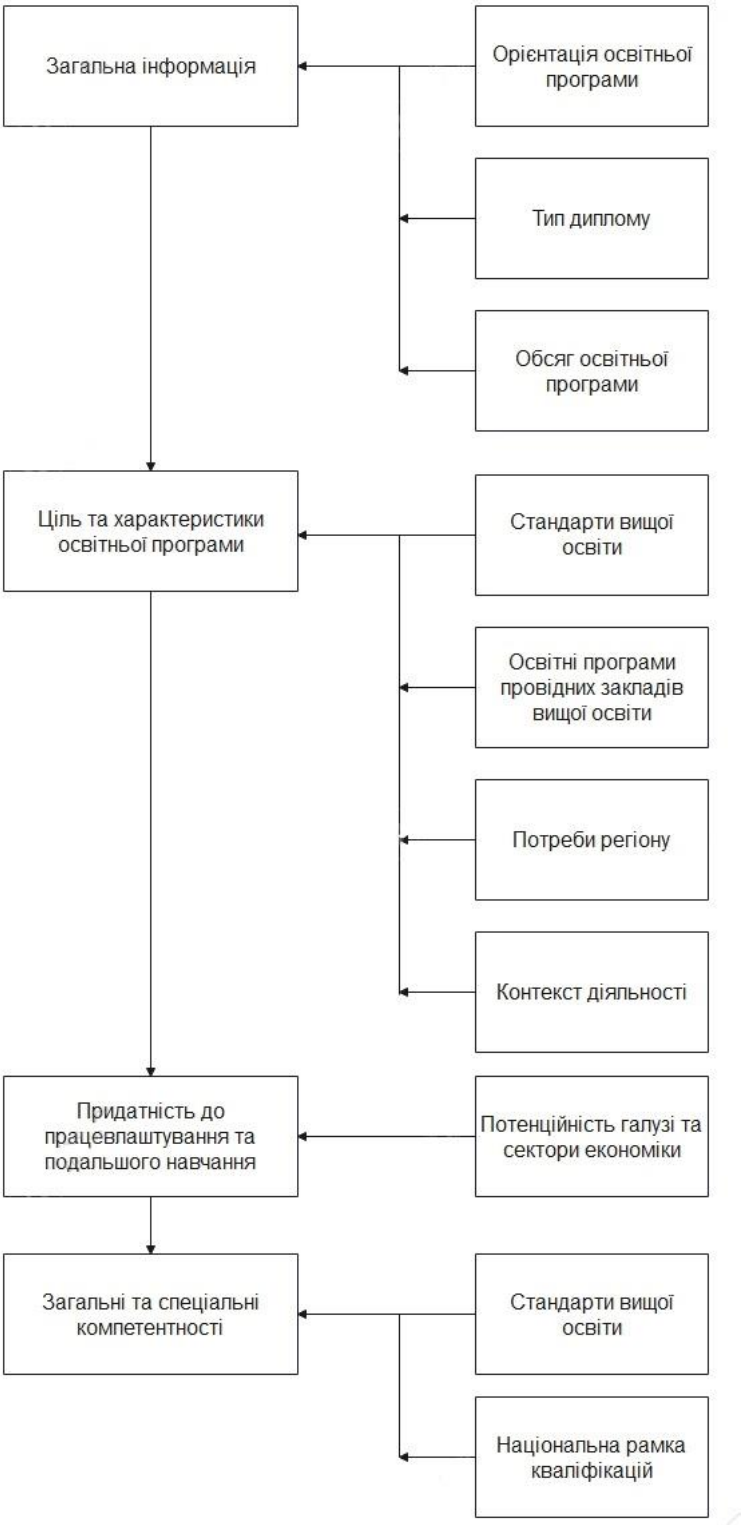


Рисунок 2.4 – Визначення профілю програми та компетентностей

– Проводиться визначення спеціалізації та типу освітньої програми, а також об’єм освітньої програми, тобто надається вичерпна інформація про освітню програму;

– Встановлюються цілі та задачі ОП, керуючись затвердженими Міністерством освіти, стандартами вищої освіти. Також, рекомендується

ознайомитися з освітніми програмами провідних університетів України та світу для виявлення переваг та недоліків розроблюваної програми. Також враховується специфіка регіону та в контексті діяльності відповідної кафедри навчального закладу, наприклад інституту чи університету з метою формулювання основного напрямку програми, що задовольняє потреби ринку;

– Враховуючи динамічний характер сучасного ринку праці, освітні програми повинні відповідати потребам та вимогам суспільства і підприємницького середовища. Дослідження ринкових тенденцій, прогнозування майбутніх потреб та аналіз кар'єрних можливостей є важливими кроками у визначенні придатності випускників для працевлаштування та подальшого навчання;

– Опис потенційних галузей та секторів економіки, де випускники можуть бути затребувані, має включати огляд актуальних технологій, трендів та потенційних ринкових можливостей. Цей опис може включати перелік професійних навичок, знань та компетенцій, які випускники отримають під час навчання, та їх застосування у реальних ситуаціях;

– Визначаються та описуються галузі та сектори економіки, де випускники можуть мати можливість працевлаштування та подальшого навчання з урахуванням потенційного попиту;

– У відповідності до затвердженого стандарту вищої освіти для даної спеціальності, або відповідно до кваліфікаційних навичок, визначаються загальні та фахові компетентності. Ці компетентності розподіляються на загальні та фахові відповідно до окреслених для відповідного освітнього рівня в Національній рамці кваліфікацій. При цьому враховуються вимоги, які найбільш точно відповідають запропонованій освітній програмі та визначають відмінності освітньої програми від інших.

3. Визначення освітніх компонентів програми;

На третьому етапі відбувається складання списку освітніх компонентів, які сприяють досягненню запланованих освітньою програмою, програмних результатів навчання. При складанні цього переліку враховуються вимоги, що

кожнен ОК має бути кредитованим, тобто мати кредитний вимір. Кредитний вимір ОК повинен відповідати реальному навчальному навантаженню здобувача вищої освіти (рисунок 2.5).

Рисунок 2.5 – Вимоги до освітніх компонент

Для освітніх компонентів освітньої програми встановлюється кількість кредитів, з урахуванням того, що один семестр складається з тридцяти кредитів, а навчальний рік з шестидесяти кредитів.

Команда забезпечення навчального процесу має завдання об'єднати встановлені компетентності та результати навчання з відповідними освітніми компонентами, зокрема вирішити наступні питання (рисунок 2.6):

– в межах яких ОК буде досягнуто програмниих результатів навчання, які мають на меті розвиток конкретних компетентностей у здобувача вищої освіти.

– обов’язкові ОК;

– зміст ОК;

– рекомендовані для вибору здобувачем вищої освіти для поглиблення рівня знань в професійній діяльності.

У результаті проведена робота узагальнюється у формі списку ОК, які відповідають компетентностям та програмним результатам навчання, для подальшого заповнення матриць взаємозв’язків.

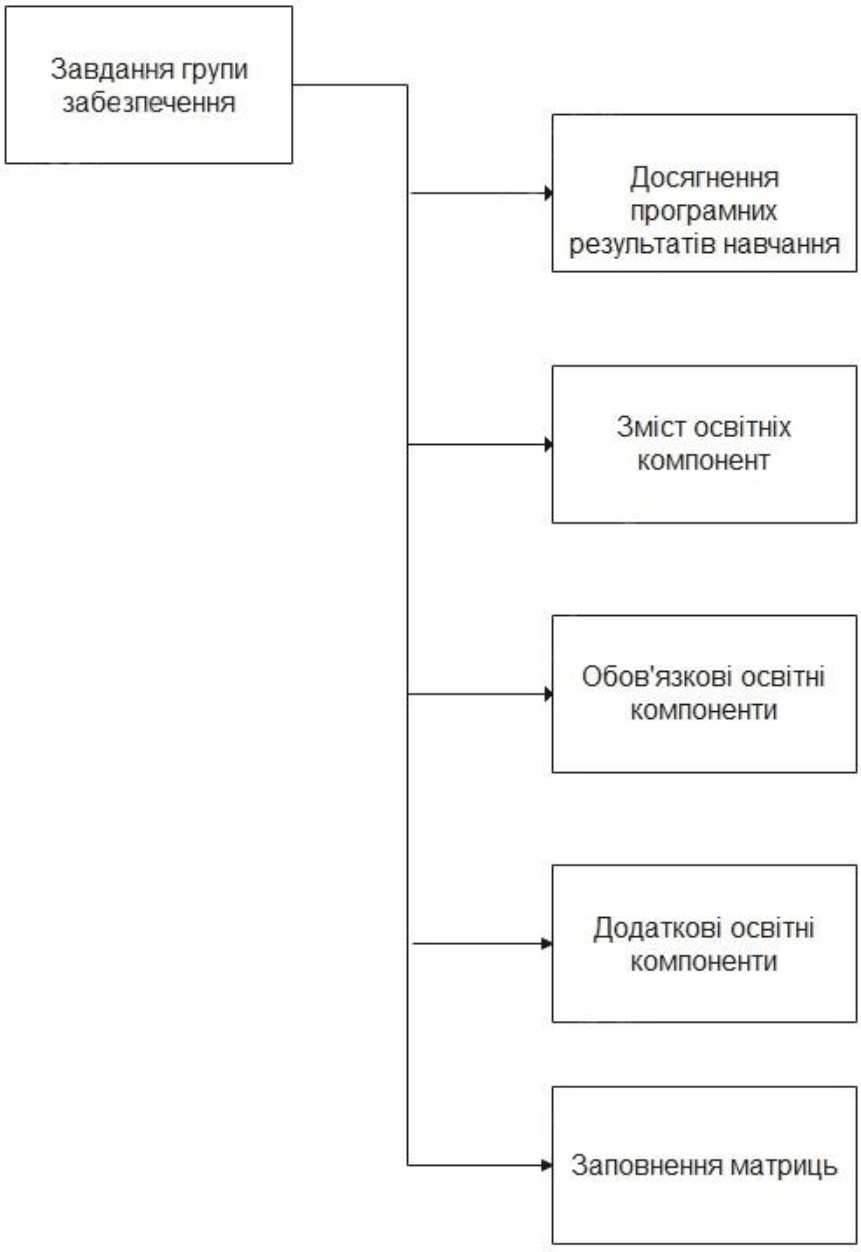


Рисунок 2.6 – Завдання групи забезпечення

4. Визначення форми атестації для здобувачів вищої освіти;

На четвертому етапі встановлюються форми атестації здобувачів освіти згідно з вимогами стандартів вищої освіти, що відповідає освітній програмі ЗВО.

5. Аналіз відповідності підрозділів, які забезпечують освітню діяльність у процесі підготовки студентів, які здобувають вищу освіту за освітніми програмами закладу вищої освіти;

На п'ятому етапі аналізуються у який спосіб буде досягнуто передбачених програмних результатів навчання та їх оцінка згідно з такими критеріями, як форми атестації, вимоги до кадрового, інформаційного забезпечення, матеріально-технічного та навчально-методичного освітнього процесу відповідно до конкретної спеціальності та вимог проведення освітнього процесу в закладах вищої освіти (рисунок 2.7).

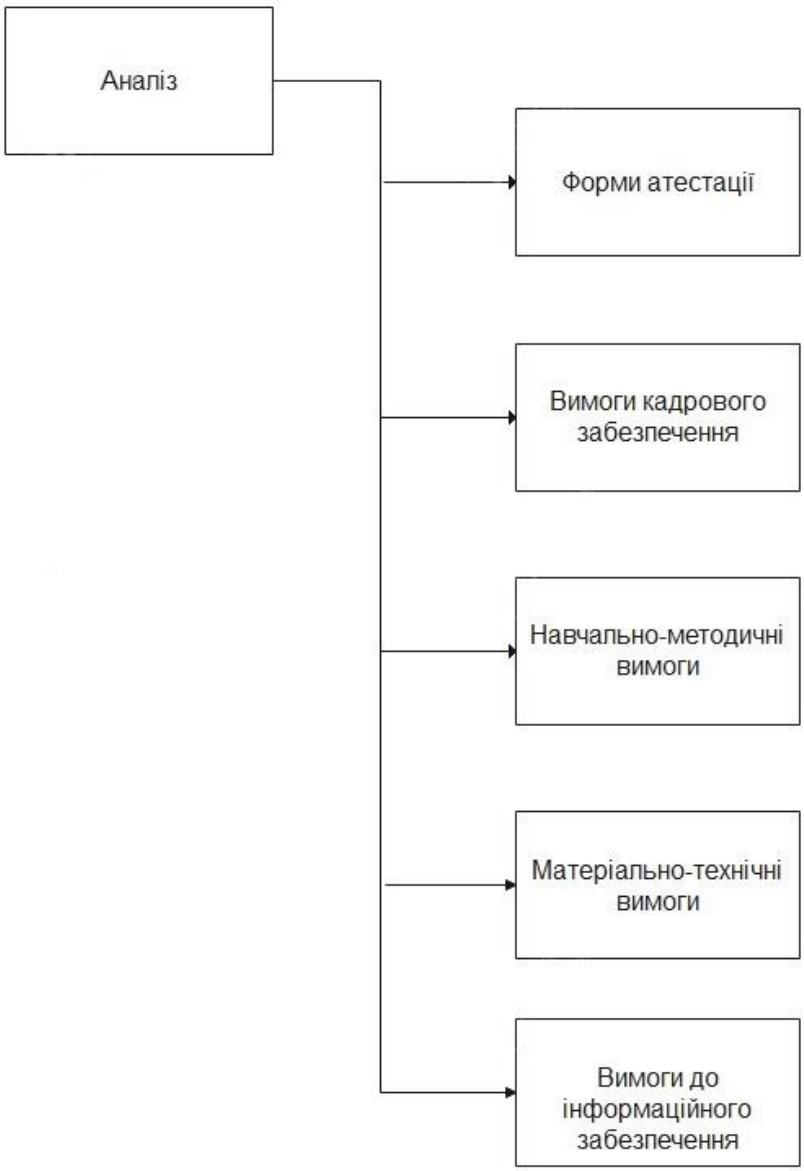


Рисунок 2.7 – Аналіз досягнення запланованих результатів навчання та їх оцінювання

6. Складання матриць взаємозв’язку між компонентами освітньої програми;

На шостому етапі здійснюється складання матриці, яка відображає взаємозв'язок між компетентностями та ОК, навчального плану, а також відношення між програмними результатами навчання, компетентностями та ОК. Ці матриці дозволяють команді виявити, які компетентності не повністю враховані освітніми компонентами, які програмні результати навчання повторюються в різних дисциплінах та інші відношення. На основі цих виявлень формується остаточний перелік і зміст освітніх компонентів освітньої програми.

Результат виконаної роботи узагальнюється у вигляді трьох матриць: компетентностей до освітніх компонент, відношення результатів навчання до компетентностей, та програмних результатів навчання до освітніх компонент.

7. Проведення перевірки обсягу кредитів ЄКТС та формування структурно-логічної схеми;

Визначення кількості кредитів повинно бути розумним відносно навчального навантаження студента. Це означає, що кількість кредитів повинна бути достатньою для досягнення програмних результатів навчання, враховуючи загальний обсяг освітньої програми в кредитах, що становить тридцять кредитів на семестр для бакалаврського та другого магістерського рівнів. Також обов'язковою вимогою є забезпечення резерву, який складає не менше двадцяти відсотків від загального обсягу освітньої програми в кредитах для вибіркових ОК.

У результаті проведена робота представляється у формі структурно-логічної схеми, яка включає перелік ОК та їх послідовність.

8. Визначення необхідних систем для забезпечення якості освіти в закладах вищої освіти, відповідно до вимог українського законодавства;

9. Визначення підходів до проведення навчального процесу, навчання та оцінювання;

На дев’ятому етапі обираються підходи і методи викладання та оцінювання результатів по завершенні навчання за відповідним освітнім компонентом, а також проводиться розподіл часу між різними видами

навчальної діяльності, такими як лекції, практичні та лабораторні заняття, семінари та самостійна робота. При цьому в робочій програмі окремої навчальної дисципліни детально описуються методи навчання та викладання, форми поточного та підсумкового контролю та оцінювання здобувачів вищої освіти.

10. Створення навчального плану;

На десятому етапі здійснюється остаточне оформлення навчального плану відповідно до вимог освітньої програми. Освітня програма визначає перелік і кількість освітніх компонентів, порядок їх вивчення, форми проведення навчальних дисциплін та їх кількість, графік освітнього процесу, а також форми поточної і підсумкової атестації контролю знань здобувачів вищої освіти.

11. Узгодження, затвердження та впровадження освітньої програми.

На останньому етапі узгоджується ОП, навчальний план з відповідними підрозділами ЗВО та набирає чинності шляхом видання наказу ректора.

2.3 Особливості використання генетичного алгоритму в задачі побудови освітньої програми

2.3.1 Загальна схема генетичного алгоритму

Вирішення поставленого завдання може включати використання різних типів алгоритмів. Найбільш поширеним підходом до вирішення даного типу задач на сьогодні є використання ГА.

Тому, ГА також може бути використаний для побудови оптимальної ОП, яка відповідатиме потребам ринку, студентів, вимогам педагогічного процесу і вимогам «Державного стандарту вищої освіти».

Генетичні алгоритми є новим напрямом. Вони здатні не лише вирішувати та спрощувати виконання великих задач, а й легко адаптуватися до зміни умов завдання. Одна ітерація генетичного алгоритму має вигляд восьми пов'язаних етапів. Під час кожної ітерації чисельність популяції збільшується вдвічі, та

повертається у вихідний стан, тобто пошук раціонального рішення проводиться серед більшої кількості популяції, ніж розміри самої популяції.

Загалом генетичний алгоритм складається з наступних кроків (рисунок 2.8):

1. формування початкової популяції (варіанти ОП);

Початкова популяція генетичних об'єктів (освітніх програм) генерується випадковим чином.

2. селекція особин;

Селекція визначає, які генетичні об'єкти будуть обрані для подальшого розмноження та створення нащадків.

3. схрещування;

Обрані генетичні об'єкти комбінуються між собою шляхом схрещування, що моделює процес спарювання та генетичного спадкування.

4. мутація;

Випадковим чином змінюються деякі характеристики генетичних об'єктів.

5. оцінка пристосованості;

Оцінка придатності здійснюється для нової популяції після застосування схрещування і мутації.

6. формування нової популяції;

Після застосування операторів селекції, схрещування, мутації та локального пошуку, потрібно створити нову популяцію генетичних об'єктів на основі отриманих результатів.

7. перевірка умови зупинки алгоритму;

Перевіряється умова завершення алгоритму. Якщо вона не виконується, повертаємося до другого етапу зі збільшеною чисельністю популяції.

8. вибір «найкращої» особини.

Після кількох ітерацій алгоритму, після застосування операторів селекції, схрещування, мутації та локального пошуку, можна визначити найкращу особину в популяції.

Збільшення чисельності популяції вдвічі під час кожної ітерації дозволяє розширити простір пошуку та збільшити шанси знайти раціональне рішення серед більшої кількості кандидатів. Це важливо для забезпечення різноманітності та ефективності пошуку оптимального рішення в контексті побудови освітньої програми.

Рисунок 2.8 – Схема роботи генетичного алгоритму

Ітераційний процес генетичного алгоритму триває до того часу, поки пройде задане число поколінь чи виконається якийсь інший критерій зупинки. Тобто, структура генетичного алгоритму є лінійним циклом.

2.3.2 Адаптація генетичного алгоритму для способу побудови освітньої програми

Адаптація генетичного алгоритму для способу побудови освітньої програми полягає в імплементації термінів предметної області вирішуваної задачі в загальну структуру генетичного алгоритму (рисунок 2.9).

Рисунок 2.9 – Імплементація термінів задачі в загальну структуру генетичного алгоритму

Кожна з освітніх компонент забезпечує набуття певних загальних та фахових компетентностей та досягнення певних програмних результатів навчання, що у гені відповідає «1». Якщо та чи інша загальна або фахова компетентність не здобувається чи певний програмний результат навчання не доясгається, тоді відповідний ген кодується «0».

Таким чином, структура генетичного алгоритму для способу побудови освітньої програми, має вигляд представлений на (рисунок 2.10)

Рисунок 2.10 – Схема роботи генетичного алгоритму для способу побудови освітньої програми

2.3.3 Основні етапи генетичного алгоритму для способу побудови освітньої програми

2.3.3.1 Формування початкового варіанту освітньої програми

Для відтворюваного генетичним алгоритмом еволюційного процесу початковий варіант освітньої програми (початкова популяція) служить початковим матеріалом Під час формування початкової ОП важливо враховувати розмірність простору параметрів (кількість компетентностей), значення кожного параметру (0 або 1).

Відбір способу формування початкової популяції (рисунок 2.11) залежить від конкретної задачі та її властивостей.

Рисунок 2.11 – Освітня програма

Початкова популяція повинна (освітня програма) бути достатньо великою, щоб забезпечити належне покриття простору параметрів, та відносно рівномірною, щоб уникнути утворення зони локального оптимуму. Також важливо пам'ятати, що початкова популяція не є остаточною і може бути покращена під час роботи ГА.

Початкова популяція може бути сформована з деякої базової освітньої програми, яку можна змінювати і оптимізувати. Кожна особина в популяції представлятиме певний набір освітніх компонент, які повинні бути включені до освітньої програми.

На першому етапі випадковим чином формується початкова освітня програма, що складається із заданого числа N освітніх компонент, кожна з яких забезпечує набуття певних загальних та фахових компетентностей та досягнення певних програмних результатів навчання.

Індивид складається з N хромосом.

Кожна хромосома особини (освітня програма) у свою чергу складається з генів (загальні та фахові компетентності, що здобуваються, та програмні результати навчання, які досягаються), кількість яких відповідає вимогам державних стандартів у сфері вищої освіти (рисунок 2.12).

Перелік атрибутивних ознак освітньої компоненти (навчальної дисципліни):

1. Назва курсу,
2. Програмні результати навчання;
3. Загальні та фахові компетентності;
4. Предметна область курсу;
5. Тип оцінювання – ген, що кодує тип оцінювання студентів, наприклад, «контрольні роботи», «заліки», «екзамени» тощо;

Побудова ОП для ЗВО часто супроводжується обмеженнями (жорсткими та м'якими) через різноманіття складових навчального процесу порівняно зі школами, де вимоги є дуже обмеженими.

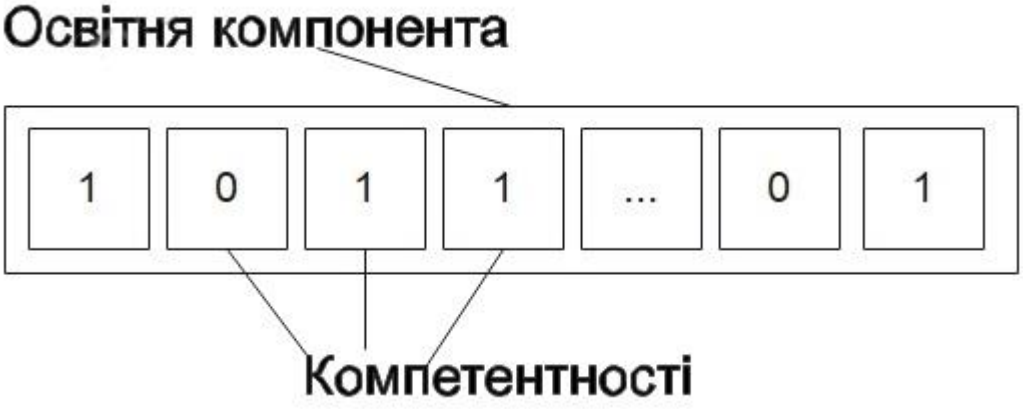


Рисунок 2.12 – Освітня компонента

Проблема побудови ОП для ЗВО базуватиметься на наявних професорах, аудиторіях, часових інтервалах проведення навчального процесу, групах студентів та кількості ЄКТС.

Кожному заняттю буде призначено часовий інтервал, професора, кімнату та групу студентів.

Для кожного заняття, запланованого цією ОП, будуть встановлені наступні жорсткі обмеження:

– Заняття можна проводити лише у вільних аудиторіях;

– Викладач може проводити лише одне заняття одночасно;

– Аудиторії повинні бути достатньо великими, щоб вмістити групу студентів.

ОП складається з ОК. Освітній компоненті відповідає три групи генів (рисунок 2.13):

Забезпечення здобуття загальних компетентностей (рисунок 2.14),

Забезпечення здобуття фахових компетентностей (рисунок 2.15),

Досягнення програмних результатів навчання (рисунок 2.16).

Формування кожної окремої освітньої компоненти початкової освітньої програми відбувається випадковим чином: по черзі для кожної окремої компетентності, що наприклад позначає загальну компетентність, визначається деяке значення (0 або 1). Після чого здійснюється перехід до наступної компетентності цієї освітньої компоненти. Коли алгоритм досягає останньої компетентності освітньої компоненти та визначає його значення, так само

заповнюються значення компетентностей другої освітньої компоненти. Аналогічно відбувається заповнення інших освітніх компонент.

Рисунок 2.13 – Освітні компоненти

Описаний процес продовжується до моменту досягнення граничної чисельності популяції, яка визначається на початку алгоритму.

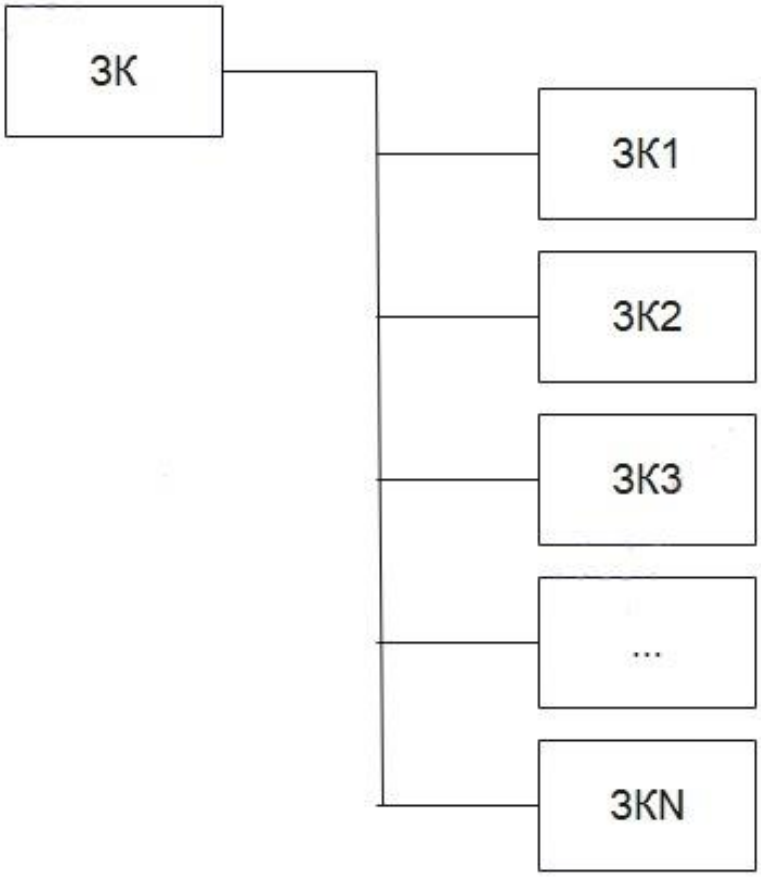


Рисунок 2.14 – Загальні компетентності

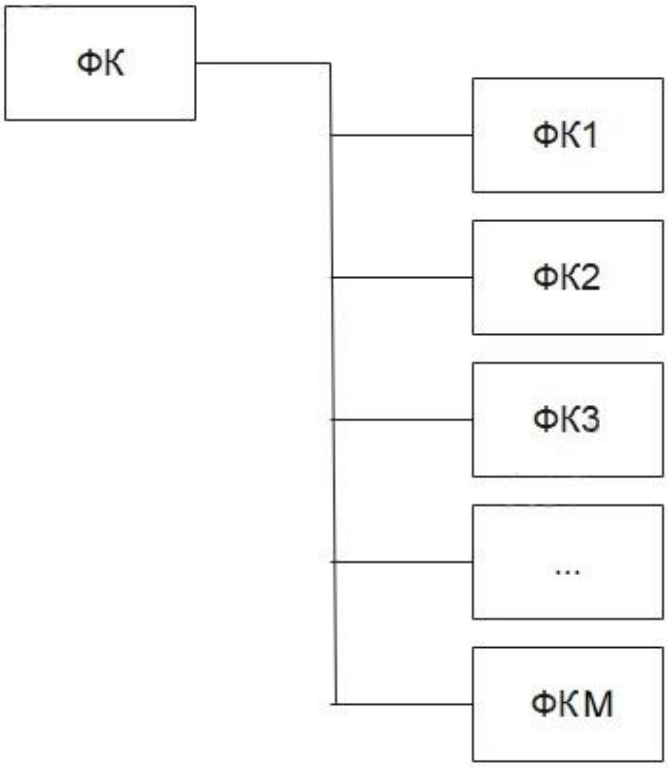


Рисунок 2.15- Фахові компетентності

Результатом навчання є ПРН (Рисунок 2.16).

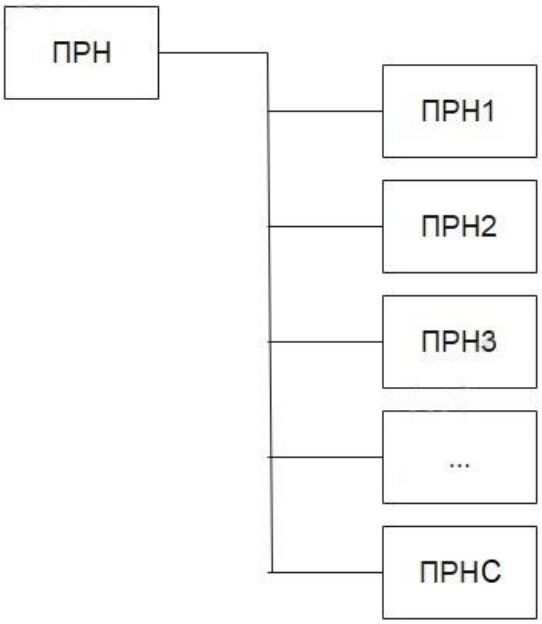


Рисунок 2.16 – Програмні результати навчання

Результатом цього етапу є початковий набір освітніх компонент.

2.3.3.2 Селекція освітніх компонент

Після формування початкової популяції, генетичний алгоритм переходить до процесу відбору, який полягає у визначенні тих особин, які будуть взяті для подальшої еволюції.

На цьому етапі відбувається відбір (селекція) найбільш пристосованих особин (освітні компоненти), що мають більш кращі значення функції придатності порівняно з іншими особинами.

Відбір може здійснюватись різними способами, наприклад:

– Рулетковий відбір – кожна особина отримує ймовірність бути вибраною, яка пропорційна її пристосованості. При цьому кращі особини мають більші шанси на вибір, але ймовірність вибору може бути й досить низькою для менш пристосованих особин;

– Елітарний відбір – дозволяє зберегти найкращі розв'язки у популяції і захистити їх від зникнення в процесі еволюції. Це сприяє збереженню цінної інформації та забезпечує продовження пошуку оптимального розв'язку. Крім того, елітарний відбір покращує швидкість збіжності алгоритму та може сприяти досягненню кращих результатів у задачах оптимізації.

– Відбір за рангом – особини впорядковуються за показником пристосованості, а потім вибирається фіксована кількість найкращих;

– Турнірний відбір – вибирається випадкова підмножина популяції, з якої обирається найкраща особина за показником пристосованості.

В даній роботі використовується елітарний відбір.

У контексті побудови ОП, елітарний відбір працює наступним чином:

1. Оцінка якості рішень;

Цей етап спрямований на перевірку ефективності та відповідності рішень встановленим цілям та вимогам.

Кожному рішенню з популяції ОП присвоюється показник якості, який відображає відповідність програми вимогам, цілям та обмеженням.

2. Відбір найкращих рішень;

Критерієм відбору є висока якість рішень. Найкращі варіанти ОП, що мають найвищі показники якості, вважаються елітарними і вибираються для перенесення до наступного покоління.

3. Збереження елітарних рішень;

Елітарні рішення передаються без змін до наступного покоління, щоб забезпечити їх збереження і уникнути втрати високоякісних рішень.

Елітарний відбір дозволяє зосередитись на найкращих рішеннях та підвищує шанси на знаходження ОП.

2.3.3.3 Схрещування освітніх компонент

Схрещування (crossover) є одним з ключових операторів ГА, який використовується в контексті побудови ОП для створення нових програм шляхом комбінування генетичного матеріалу двох батьківських програм. Цей оператор моделює процес спарювання та генетичного спадкування, що сприяє пошуку нових і покращених рішень у задачі побудови ОП.

Схрещування використовується для комбінування різних характеристик та особливостей батьківських програм з метою створення нащадків, які можуть мати кращу придатність або нові комбінації генетичного матеріалу. Цей процес може відбуватися на рівні генів, підпрограм, правил або інших структурних одиниць освітньої програми.

Схрещування моделює процес спарювання та генетичного спадкування, які є ключовими механізмами еволюції в природі. Під час схрещування обрані батьківські програми обмінюються деякою частиною свого генетичного матеріалу, що призводить до створення нащадків з комбінованими характеристиками обох батьків.

При виконання схрещування за допомогою ГА для побудови освітньої програми, батьківська освітні програма розбиваються на генетичні складники (освітні компоненти), які можуть бути різними елементами програми. Потім

обираються випадковим чином відповідні складники з двох освітніх компонент і комбінуються, щоб створити нову ОК-нащадок. Цей процес сприяє розширенню простору можливих рішень та пошуку оптимальних комбінацій освітніх компонент.

Вибір та спосіб комбінування освітніх компонент можуть варіюватись залежно від конкретного генетичного алгоритму та природи задачі побудови освітньої програми. Наприклад, можуть використовуватись одноточкове (рисунок 2.17) або багатоточкове схрещування, що впливає на кількість та місце перетину освітніх компонент. Також можуть бути використані різні стратегії відбору освітніх компонент та контролю процесу схрещування і досягнення кращих результатів.

Комбінування освітніх компонент через схрещування допомагає згенерувати різні комбінації ОК та знаходити оптимальні рішення для побудови освітньої програми.

У контексті побудови ОП, схрещування працює наступним чином (рисунок 2.17):

1. Вибір освітніх компонет;

З популяції освітніх компонет, вибираються дві програми, які виступають у ролі батьків для створення нової ОП.

2. Створення нащадка;

Генетичний матеріал (гени) обох батьківських програм комбінується для створення нащадка (нової ОП).

3. Розмноження нащадка.

Нова програма (нащадок) додається до популяції і може бути піддана подальшій еволюції через ітерації ГА.

Схрещування дозволяє комбінувати корисні ознаки та властивості батьківських ОК, створюючи нащадків з новими комбінаціями компетентностей. Цей процес сприяє рознообразністю в ОП та пошуку оптимальних рішень.

У результаті схрещування відбувається обмін генетичним матеріалом між батьківськими програмами, що призводить до створення нових програм, що поєднують характеристики обох батьків. Цей процес дозволяє розширити

простір пошуку та знайти оптимальніші рішення у задачі побудови освітньої програми шляхом комбінації варіантів та експериментування з генетичним матеріалом.

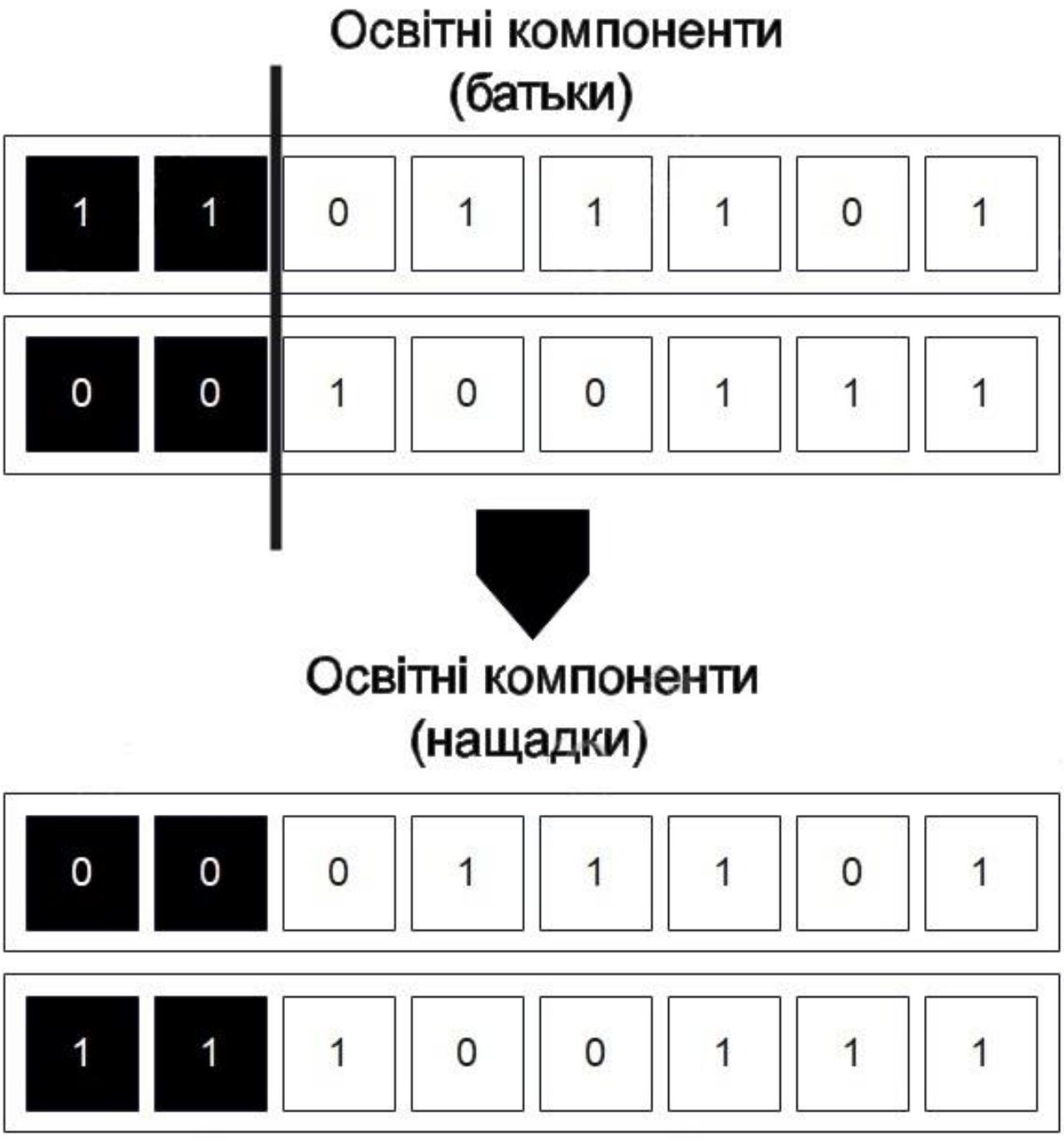


Рисунок 2.17 – Схрещування

В контексті побудови освітньої програми схрещування допомагає вдосконалювати освітні програми, шляхом комбінування та поєднання їх освітніх компонент.

2.3.3.4 Мутація

Після проведення схрещування настає етап мутації, під час якого з деякою вірогідністю змінюються окремі компетентності в освітніх компонент ах (рисунок 2.18). Це дозволяє зберегти різноманітність освітньої програми та забезпечити можливість знаходження оптимального рішення в майбутньому.

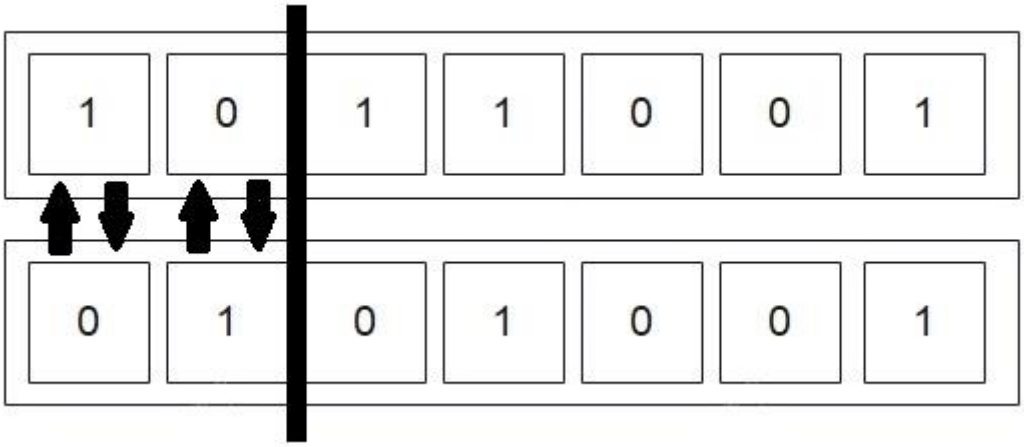


Рисунок 2.18 – Процес мутації

В ГА, зазвичай, використовують досить низьку ймовірність мутації (0.01 або менше), щоб уникнути занадто великих змін в генетичному матеріалі (компотнентностях) та зберегти корисну інформацію, яку вже накопичила популяція (освітня програма). Після проведення мутації отримуються нащадки, які переходять до наступної ітерації генетичного алгоритму. При цьому, новоутворені освітні компоненти можуть замінити менш ефективні освітні компоненти в популяції, що дозволяє зберегти тільки найкращі рішення. У ГА кількість ітерацій визначається цільовою функцією. Після досягнення цільовою функції алгоритм зупиняється. Результатом роботи ГА може бути найдієвіший або найдосяжніший результат, залежно від того, яку мету ми переслідуємо.

В контексті побудови ОП, мутація буде застосована до певних елементів програми, таких як навчальні модулі, заняття, оцінювання або компетентностей, які зкривають ті чи інші ОК, з метою внесення випадкових змін. Це дозволяє

відкривати нові можливості та експериментувати з варіантами програми, що може призвести до покращення якості освітнього процесу.

Процес мутації при побудові ОП включає в себе наступні кроки:

1. Вибір варіантів ОП (хромосом) для мутації – визначаються варіанти розв'язку задачі побудови ОП, які будуть піддані мутації;

2. Вибір елемента для мутації – визначаються елементи ОП (гени), які будуть піддані мутації;

3. Застосування мутаційного оператора – вибраний елемент змінюється за допомогою мутаційного оператора;

4. Оцінка результату мутації – після зміни елемента в ОП оцінюється якість нового рішення. Оцінка якості нового рішення задачі включає в себе, аналіз впливу мутації на цілі та вимоги програми, а також оцінку його відповідності освітнім стандартам та потребам студентів та викладачів.

Під час виконання мутації, також важливо враховувати баланс між мутацією та збереженням стабільних та ефективних елементів програми, а також використовувати певні критерії та обмеження для забезпечення відповідності програми стандартам і вимогам.

2.3.3.5 Оцінка пристосованості

Оцінка пристосованості є одним із найважливіших етапів роботи ГА, оскільки від неї залежить вибір тих особин, які будуть входити до наступного покоління. Кращі особини, які мають найвищі значення, мають більші шанси на виживання та більш ймовірно будуть вибрані для схрещування та мутації, що забезпечує збільшення якості наступного покоління.

Під час цього етапу кожній особині з популяції надається оцінка її пристосованості або фітнес (fitness). Функція пристосованості є цільовою функцією проблеми. Фітнес дасть значення, яке потім вкаже, яке рішення є найкращим рішенням чи ні. Ця оцінка придатності також забезпечить циклічний процесу генетичному алгоритмі бути зупиненим чи ні. Іншими словами, фітнес

буде контролювати та підтримувати процеспотік генетичного алгоритму. Варіантом наступного покоління буде контроль і обслуговування, а не буде йти далі від нього. Фітнес-функція має наступний вигляд:

$$f = \frac{1}{1+x}; \tag{2.1}$$

Де x – це подання конфліктів (clashes) або обмежень:

$$f = x + MT + C + R + D; \tag{2.2}$$

I є інструктором, інструктор присвоїть значення 0 у номері, якщо на зустрічі буде лише I інструктор час T, курс C в аудиторії R значення дорівнюватиме 1, якщо є будь-який конфлікт час зустрічі T, курс C, кімната R, або іншим викладачем. Якщо зіткнень немає, то значення дорівнює 0. Тому для значення x воно може дорівнювати 0, якщо конфлікту немає, тоді як воно призначатиме значення 1, 2, 3, 4 або 5, якщо виникає конфлікт, залежить від кількості конфліктів.

Фітнес може вважатися як міра успіху особини відносно розв'язання задачі, що вирішується за допомогою ГА. Оцінка пристосованості може використовувати різноманітні критерії, в залежності від конкретної задачі, що вирішується. Оцінка фітнесу зазвичай базується на цільовій функції або метриці, яка відображає якість рішення поставленої задачі. Процес визначення фітнесу включає обчислення відповідної метрики для кожної особини в популяції. Особини, що мають вищий фітнес, вважаються більш придатними або кращими в контексті задачі. Це дозволяє генетичному алгоритму орієнтуватися на особини з вищим фітнесом при здійсненні операцій селекції, схрещування та мутації, що сприяє покращенню якості рішення з плином поколінь. Наприклад, у задачі максимізації функції фітнес може бути прямо пропорційний значенню цієї

функції, а у задачі мінімізації функції, фітнес може бути обернено пропорційний значенню функції.

2.3.3.6 Формування нової популяції

На цьому етапі відбувається фільтрація, яка виділяє у складі популяції особини, що мають низьке значення функції придатності. Знайдені слабкі особини видаляються з популяції до того часу, поки чисельність досягає вихідної.

Нове покоління, сформоване внаслідок застосування операторів відбору, схрещування, мутації, так звана популяція нащадків, замінює батьківську популяцію. Цей етап є важливим в ГА, оскільки саме від якості нової популяції залежить ефективність алгоритму. Існує кілька способів заміни батьківської популяції новим поколінням нащадків, зокрема повна заміна, заміна лише певної кількості слабших особин або заміна з додаванням деякої кількості найкращих особин з попереднього покоління. Вибір конкретного способу залежить від постановки задачі і характеристик популяції. Зазвичай, кращим варіантом є заміна батьківської популяції новим поколінням нащадків з додаванням деякої кількості найкращих особин з попереднього покоління, оскільки це дозволяє зберегти найбільш корисну інформацію з попереднього покоління та забезпечує збільшення шансів знаходження найкращого розв'язку поставленої задачі.

2.3.3.7 Перевірка умови зупинки алгоритму

На цьому етапі відбувається перевірка умови припинення роботи алгоритму. Вона заснована на використанні збільшення функції придатності. Якщо протягом кількох поколінь особин збільшення значення функції придатності найбільш "найкращої" особин виявляється незначною, то робота

алгоритму завершується. Критерієм зупинки може бути також задана кількість ітерацій, або цільова функція.

2.3.3.8 Вибір «найкращої» особини

На цьому етапі серед отриманих особин вибирається найкраща особина, яка і буде рішенням завдання. Під кращою особистістю розуміється та, що має значення функції придатності є мінімальним для вирішення завдання.

2.4 Функціональна структура інформаційної системи

Функціональна структура є важливою складовою проектування ІС, оскільки вона дає змогу описати необхідні функцій та процесів, що дозволяє відобразити логіку взаємодії між компонентами системи. Це дозволяє забезпечити більш точне розуміння вимог до системи та сприяє підвищенню її ефективності та продуктивності.

Побудова функціональної структури ІС зазвичай включає наступні етапи (рисунок 2.19):

1. Аналіз вимог до системи;

На цьому етапі здійснюється вивчення вимог до інформаційної системи, що включає в себе визначення функціональних вимог, вимог до продуктивності, надійності та безпеки.

2. Визначення функцій системи;

На цьому етапі визначаються всі необхідні функції, які повинна виконувати інформаційна система для задоволення вимог.

3. Побудова блок-схем;

Після визначення функцій системи створюють блок-схеми, які відображають послідовність виконання функцій та взаємозв'язок між ними.

4. Визначення вхідних та вихідних даних;

На цьому етапі визначаються всі вхідні та вихідні дані для кожної функції системи.

5. Опис процесів обробки даних;

На цьому етапі визначаються всі процеси обробки даних, що відбуваються в інформаційній системі;

6. Побудова функціональної схеми;

На цьому етапі всі блок-схеми та описи функцій системи об'єднуються в одну функціональну схему, яка відображає взаємозв'язок між всіма компонентами системи.

7. Валідація функціональної структури;

На цьому етапі проводиться перевірка того, чи відповідає функціональна структура вимогам, які були визначені на першому етапі.

Рисунок 2.19 – Етапи побудови функціональної структури ІС

У цьому розділі було розглянуто важливі аспекти створення функціональної структури ІС.

2.5 Проектування структури інформаційної системи

Проектування структури ІС складається з наступних рівнів (рисунок 2.20):

1. Концептуальний рівень;

На концептуальному рівні визначається загальна концепція ІС для побудови ОП, визначаються потреби та вимоги до функціональності системи, а також об’єкти, що будуть взаємодіяти з системою. На цьому етапі розробляється бізнес-модель, яка відображає принципи діяльності системи і її мету.

Основними завданнями на концептуальному рівні є:

– аналіз бізнес-процесів та вимог до системи;

– формулювання вимог до функціональної та нефункціональної частин ІС;

– розробка концепції архітектури системи, включаючи визначення компонентів та їх взаємодії.

Результатом цього етапу є документ з концепцією ІС, який слугує основою для подальшої розробки системи на наступних рівнях проектування.

2. Логічний рівень;

Це рівень проектування ІС, на якому конкретизуються вхідні дані та вихідні результати, описуються процеси обробки даних та вибір алгоритмів, а також здійснюється вибір технологій інформаційної системи.

На логічному рівні створюється логічна модель ІС, яка описує процеси обробки даних та способи їх зберігання та передачі. На цьому рівні проектування формуються такі елементи, як сутності, атрибути та зв'язки між ними, які описують структуру даних в системі.

визначається архітектура системи, тобто обрані технології та апаратне забезпечення, що будуть використовуватись для реалізації системи. В результаті

роботи на логічному рівні формується документація проекту, що містить детальний опис логічної моделі системи та вимог до реалізації.

Також на цьому етапі визначаються логічні схеми БД та розробляються інтерфейси користувача, що дозволяють здійснювати взаємодію з інформаційною системою.

На логічному рівні також проводять аналіз вимог до системи та визначаються функціональні та нефункціональні вимоги. Функціональні вимоги описують бізнес-процеси, які повинна виконувати система, тоді як нефункціональні вимоги включають технічні та надійнісні вимоги до системи, такі як продуктивність, безпека та доступність.

При проектуванні на логічному рівні враховуються принципи модульності, абстракції та розширюваності системи. Модулі системи розробляються таким чином, щоб бути самостійними, змінюваними та готовими до використання у майбутньому компонентами. Логічна структура повинна бути гнучкою, щоб забезпечити можливість майбутнього розширення та модифікації системи.

Усе це допомагає розробити логічну модель ІС, яка є проміжним кроком між концептуальним та фізичним рівнем.

3. Фізичний рівень;

Фізичний рівень інформаційної системи описує апаратне та програмне забезпечення, на якому функціонує система, тобто конкретні комп'ютерні пристрої та їхню організацію. Цей рівень включає в себе опис архітектури та конфігурації апаратних засобів, таких як процесори, пам'ять, пристрої зберігання даних, мережеві інтерфейси та інші компоненти.

Основним завданням на фізичному рівні проектування інформаційної системи є забезпечення ефективної та надійної роботи системи в реальному середовищі. Для цього важливо розглядати технічні обмеження та можливості реальних апаратних та програмних рішень, які можуть бути використані в проекті.

На фізичному рівні також вирішуються питання щодо безпеки та забезпечення доступу до ресурсів системи, таких як автентифікація

користувачів, керування дозволами, шифрування даних та інші заходи забезпечення безпеки.

4. Реалізація та тестування;

Після проектування ІС, наступним кроком є її реалізація та тестування.

Етап реалізації, передбачає програмування та налагодження програмного забезпечення, яке включає в себе різноманітні компоненти, такі як бази даних, сервери, клієнтські додатки та інше.

Після реалізації системи необхідно провести тестування, щоб переконатися, що вона відповідає вимогам та специфікації.

Після успішного тестування система може бути введена в експлуатацію, але розробник повинен продовжувати виконувати обслуговування та підтримку системи, виправляти помилки та додавати нові функції, які вимагаються користувачами.

5. Експлуатація та підтримка.

Цей етап охоплює процеси, пов'язані з запуском системи в роботу, підтримкою її функціонування та вирішенням проблем, які можуть виникати в процесі експлуатації.

На етапі експлуатації системи необхідно забезпечити постійне функціонування системи та моніторинг її роботи. Для цього можуть бути використані різні інструменти, наприклад, моніторингові програми, які дозволяють відслідковувати роботу системи та виявляти можливі проблеми.

Рисунок 2.20 – Етапи проектування структури ІС

Архітектура інформаційної системи для побудови освітньої програми включає компоненти та модулі, що співпрацюють між собою для забезпечення ефективного процесу розробки, впровадження і управління освітньою програмою. Основні складові інформаційної системи (рисунок 2.21):

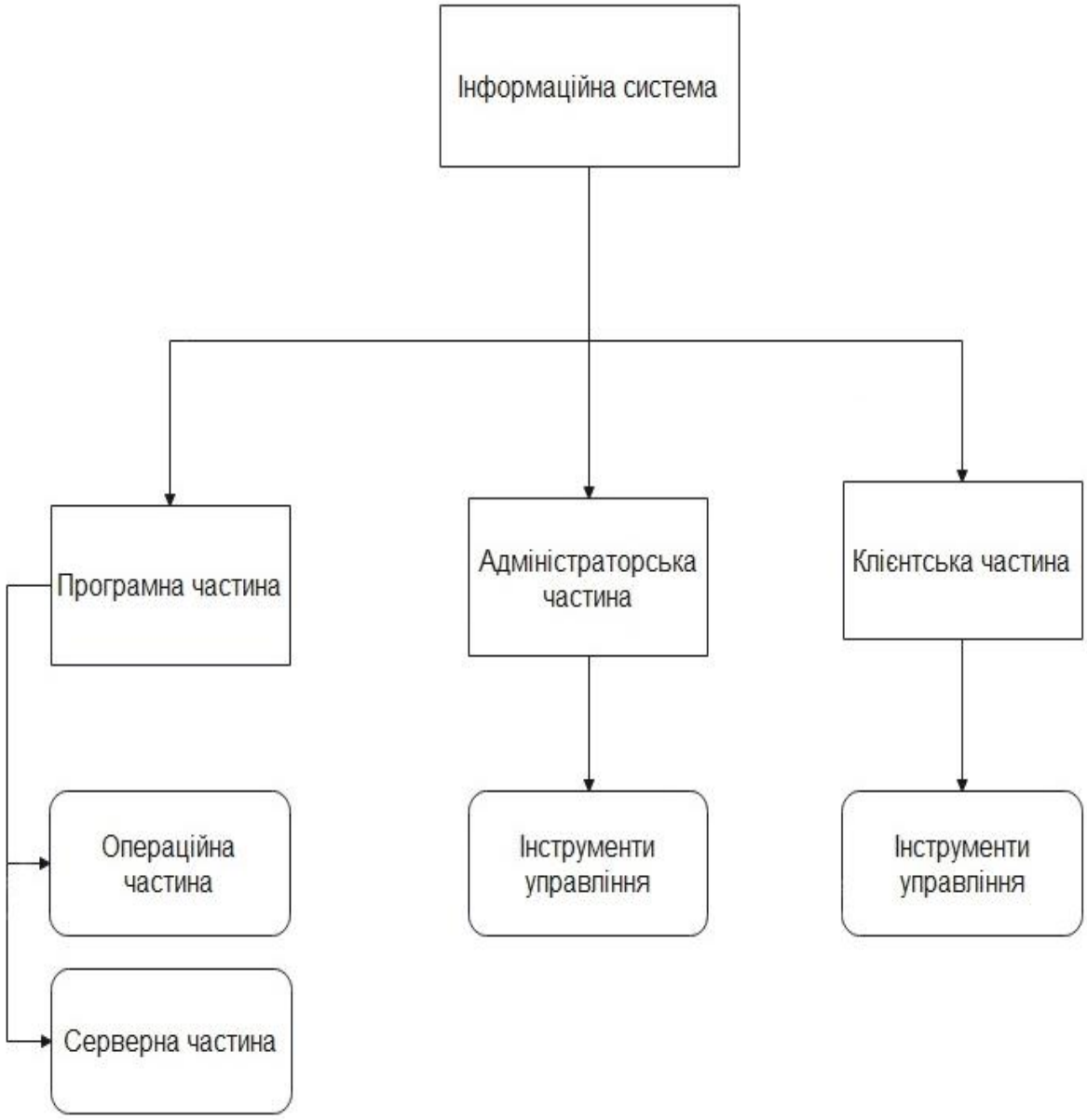


Рисунок 2.21 – Складові інформаційної системи

Узагальнюючи, етап експлуатації та підтримки інформаційної системи є важливим для забезпечення її стабільної роботи та вирішення проблем, які можуть виникати в процесі її використання.

2.6 Висновок до другого розділу

Було створено опис способу побудови освітньої програми за генетичним алгоритмом включно із загальними положеннями щодо побудови освітньої програми.

Розроблено загальну послідовність побудови освітньої програми із визначення стадій та етапів.

Генетичний алгоритм адаптовано до вирішуваної задачі із визначенням сутностей «популяція», «особина», «хромосома», «ген» та формування послідовності використання генетичного алгоритму.

Спроектовано структуру інформаційної системи.

Розділ 3 Програмна реалізація способу побудови освітньої програми за генетичним алгоритмом

3.1 Структура модулів системи, їх взаємозв’язок

Структура модулів системи визначає організацію компонентів та їх взаємозв'язок з метою досягнення функціональності та цілей системи. Кожен модуль виконує певну функцію або набір функцій і може бути незалежним від інших модулів, або залежати від них.

Взаємозв'язок між модулями визначає, як вони співпрацюють і взаємодіють між собою. Це може бути передача даних, обмін повідомленнями, виклик функцій чи подій між модулями. Взаємозв'язок дозволяє забезпечити взаємодію та обмін інформацією між різними частинами ІС, що необхідно для її функціонування та досягнення поставлених цілей.

Розглянемо діаграму компонентів ІС для побудови ОП (рисунок 3.1).

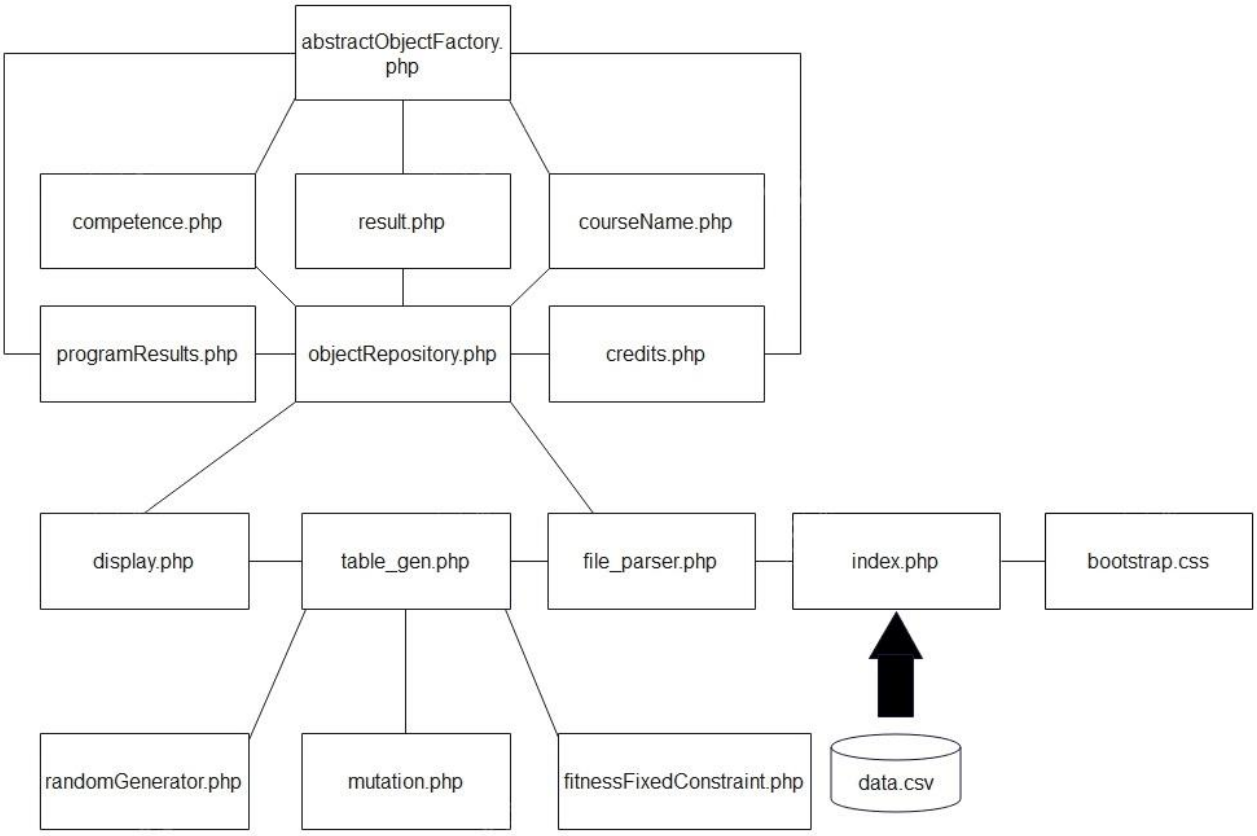


Рисунок 3.1 – Діаграма компонентів ІС

На діаграмі компонентів представлені наступні компоненти ІС:

1. Папка «GA»;

– index.php ;

– table_gen.

2. Папка «Classes» в папці «GA»;

– display.php;

– file_parser.php;

– fitnessFixedConstraint.php;

– mutation.php;

– objectRespository.php;

– randomGenerator.php.

3. Папка «Objects» в папці «Classes»;

– competence.php;

– result.php;

– courseName;

– programResults.php;

– credits.php.

4. Папка «Inheritables» в папці «Objects»;

– abstractObjectFactory.

5. Папка «css».

– Bootstrap.css.

Проектування структури модулів та їх взаємозв'язку має велике значення для успішної реалізації інформаційної системи. Вона дозволяє розподілити функціональність системи на логічно пов'язані компоненти, що полегшує розробку, тестування та підтримку системи. Крім того, правильно визначений взаємозв'язок між модулями забезпечує гнучкість та можливість розширення системи у майбутньому.

У даному розділі було розглянуто структуру модулів системи та їх взаємозв'язок, що є важливим аспектом проектування ІС.

3.2 Особливості реалізації способу побудови освітньої програми за генетичним алгоритмом

Реалізація побудови освітньої програми за допомогою генетичного алгоритму має свої особливості, які варто враховувати:

1. Кодування варіантів рішень;

Основним елементом ГА є представлення рішень у вигляді хромосом. В контексті побудови ОП, хромосома може містити гени, які представляють різні атрибути курсу, такі як назва, кількість годин, форма проведення тощо.

2. Функція пристосованості;

Важливим елементом є визначення функції пристосованості, яка оцінює якість кожної особини (освітньої програми) в популяції. Ця функція може враховувати такі фактори, як актуальність курсу, відповідність освітніх потреб студентів, компетентність викладачів тощо.

3. Операції ГА;

У процесі еволюції популяції освітніх програм, використовуються стандартні операції генетичного алгоритму, такі як схрещування, мутація, відбір найкращих особин. Під час схрещування можуть поєднуватися різні атрибути курсів, а мутація дозволяє вносити випадкові зміни в гени.

4. Параметри алгоритму;

Важливо правильно налаштувати параметри генетичного алгоритму, такі як розмір популяції, ймовірність схрещування і мутації, кількість ітерацій тощо. Ці параметри впливають на швидкість збіжності алгоритму та якість отриманих рішень.

5. Експертні знання;

Для ефективної побудови ОП за допомогою ГА важливо використовувати експертні знання. Експерти можуть надавати оцінки атрибутів курсів, рекомендації щодо зв’язків між ними та інші корисні вказівки для покращення процесу побудови ОП.

При побудові ОП, ІС враховує наступні критерії: аудиторії, професори, часові інтервали, курси, групи студентів та кількість ЄКТС.

Інформаційна система для побудови освітньої програми складається з наступних параметрів та їх значень:

– Розмір популяції: 1000;

Частота кросинговеру: 0.01.

3.3 Опис функціональних можливостей інформаційної системи

Розглянемо головну сторінку ІС (рисунок 3.2).

Головна сторінка ІС для побудови ОП є ключовим елементом системи, який надає користувачам доступ до основних функціональних можливостей та інформації. Її основна мета полягає в забезпеченні зручного та ефективного способу навігації по системі та надання необхідних інструментів для створення та роботи з ОП. Головна сторінка має чітку та логічну структуру, щоб користувачі могли швидко зорієнтуватися і знайти потрібну інформацію.

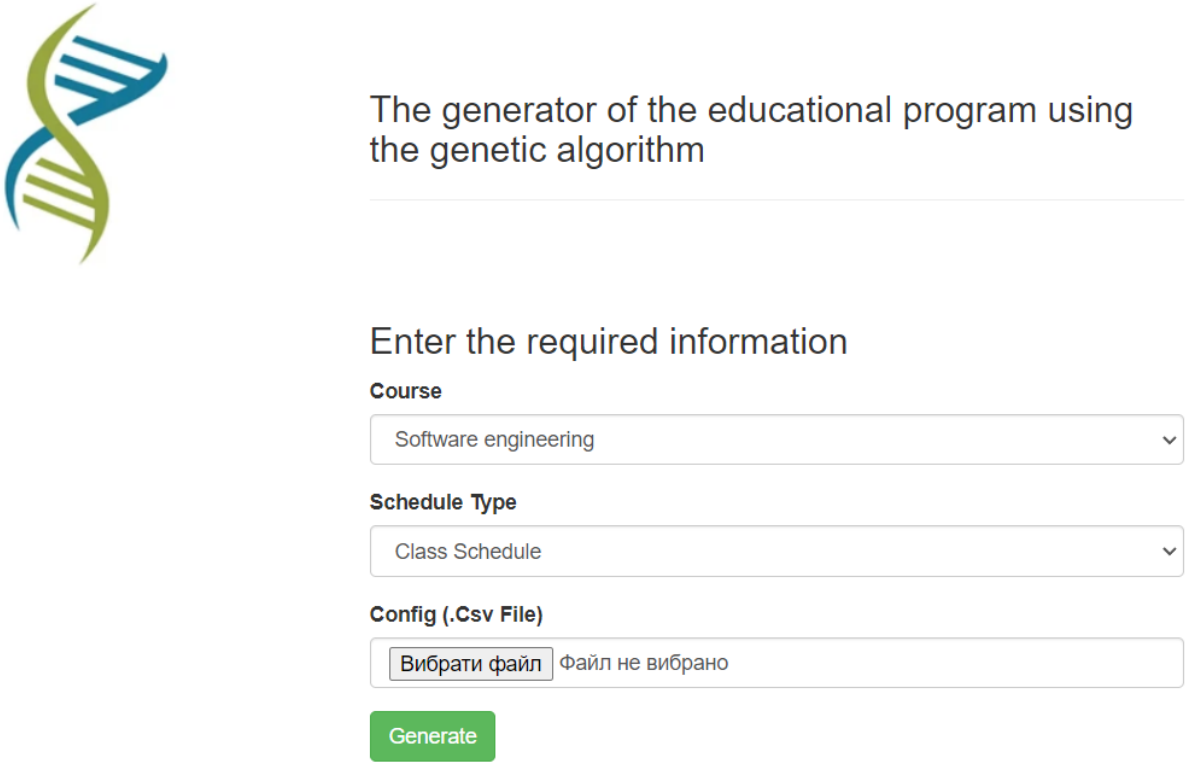


Рисунок 3.2 – Головна сторінка ІС

В результаті роботи інформаційної системи для побудови освітньої програми, ми отримуємо освітню програми зображену на (рисунок 3.3).

Software engineering

Fitness Statistics

Clashing course 21
Fitness Score = 0.79047619047619
Elite Chromosomes: 0

Number of Generations: 1000

Number of Crossovers: 349

Number of Mutations: 308

	Mon	Tue	Wed	Thu	Fri
8 - 9				**Назва:** Дискретна математика **ЗК,ФК:** ФК9 **ПРН:** ПРН1 **Іспит:** Yes **ЄКТС:** 5	
9 - 10					
10 - 11					
11 - 12					
12 - 1					
1 - 2					
2 - 3				**Назва:** Дискретна математика **ЗК,ФК:** ФК2 **ПРН:** ПРН1 **Іспит:** No **ЄКТС:** 10	**Назва:** Фізика **ЗК,ФК:** ФК5 **ПРН:** ПРН1 **Іспит:** Yes **ЄКТС:** 6

Рисунок 3.3 – Результат роботи інформаційної системи для побудови освітньої програми

3.4 Апробація способу

Для демонстрації способу було сформовано тестовий набір освітніх компонент із відповідними загальними та фаховими компетентностями та програмними результатами навчання (рисунок 3.4).

Освітні компоненти у цьому наборі включають основні елементи освітньої програми, такі як назва дисципліни, ЄКТС, результат (іспит або залік), загальні та фахові компетентності і програмні результати навчання.

Кожен освітній компонент включає наступну інформацію:

– Назва дисципліни: Ідентифікує конкретну дисципліну або курс, який вивчається студентами;

– ЄКТС (Європейська кредитно трансферна-накопичувальна система): Визначає обсяг роботи, необхідний для успішного завершення дисципліни. Використовується для визначення навантаження студента та прогнозування тривалості навчання;

– Результат (іспит або залік): Вказує на спосіб оцінювання студентів для даної дисципліни, чи це буде іспит, залік або інша форма оцінювання;

– Загальні компетентності: Визначають навички, знання та особистісні якості, які студенти мають набути під час проходження дисципліни;

– Фахові компетентності: Визначають специфічні навички і знання, які студенти мають розвинути в рамках даної дисципліни;

– Програмні результати навчання: Вказують на очікувані досягнення студентів після успішного закінчення дисципліни.

Навчальні дисципліни	**ЄКТС**	**Результ.**	**ЗК, ФК**	**ПРН**
Вища математика	15	іспит	**ФК 1**	**ПРН 2**
Дискретна математика	5	іспит	**ФК 1** **ФК 3**	**ПРН 2**
Фізика	10	залік	***ФК 1***	
Англійська мова за професійним спрямуванням	5	залік	**ЗК05**	
Алгоритмізація та програмування	6	іспит	***ФК 8***	***ПРН 9***
Теорія ймовірності та математична статистика	7	іспит	***ФК 1*** ***ФК 2***	**ПРН 3**
Безпека життєдіяльності, охорона праці, цивільний захист та екологічна безпека	8	іспит		
Теорія алгоритмів	6	іспит	***ФК 1.*** **ФК 3**	**ПРН 5**
Дослідження операцій та основи теорії прийняття рішень	5	залік	***ФК 1. ФК 2. ФК 3.*** **ФК 5**	***ПРН 2.*** **ПРН 7**
Філософія	5	залік	**ЗК01** **ЗК10** **ЗК11** ***ЗК15***	**ПРН 1**

Рисунок 3.4 – Тестовий набір освітніх компонент

Інформаційна система була протестована для того,щоб переконатися, що всі отримані рішення є оптимальними.

Кількість поколінь, необхідних, щоб досягти оптимального рішення зображено на (рисунок 3.5). Збільшення кількості циклів свідчить про постійне зменшення числа поколінь, які необхідні для досягнення оптимального рішення.

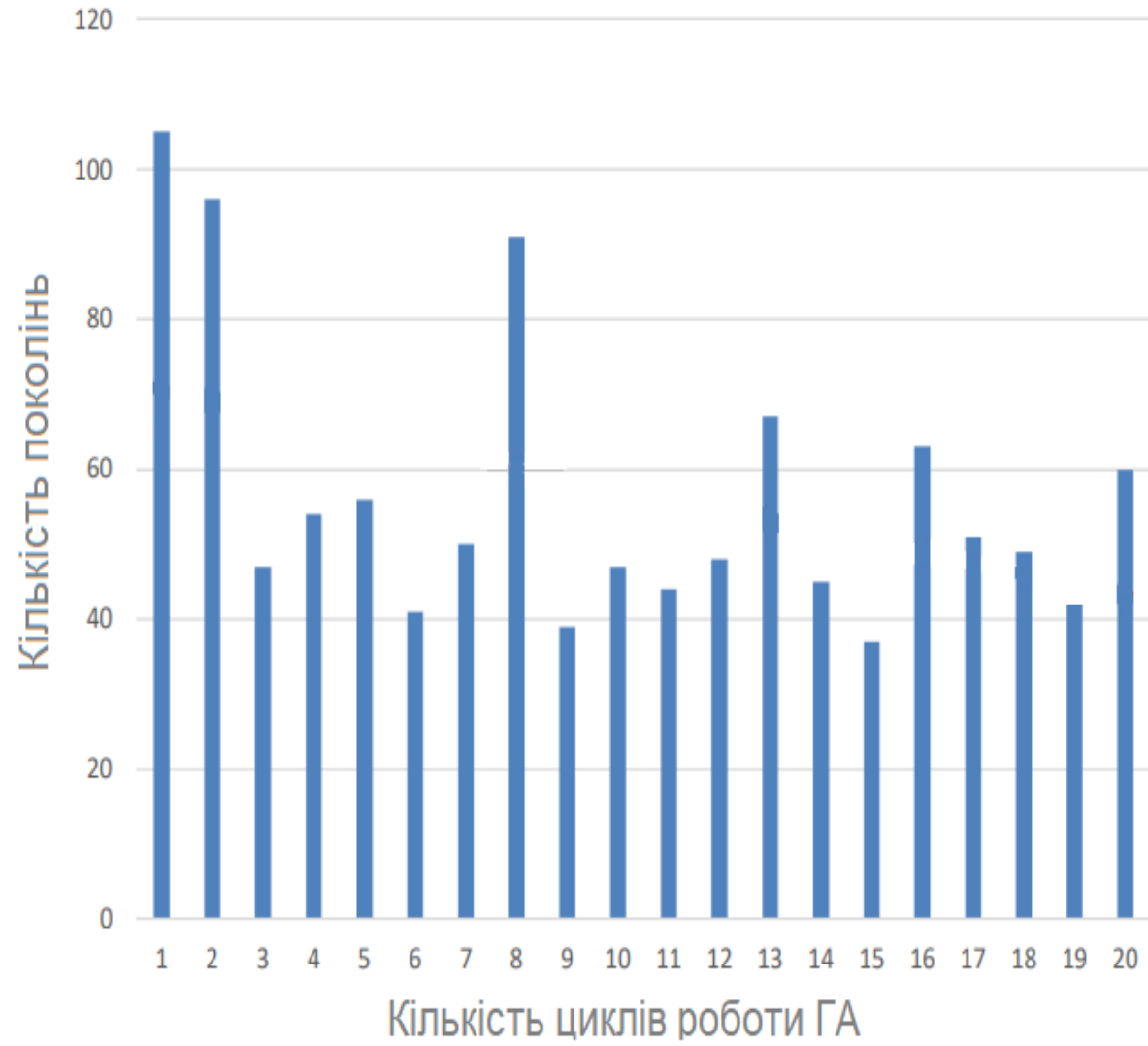


Рисунок 3.5 – Кількість поколінь для досягнення оптимального рішення

Збільшення чисельності популяції свідчить про постійне збільшення числа поколінь, які потрібні для досягнення оптимального рішення (рисунок 3.6).

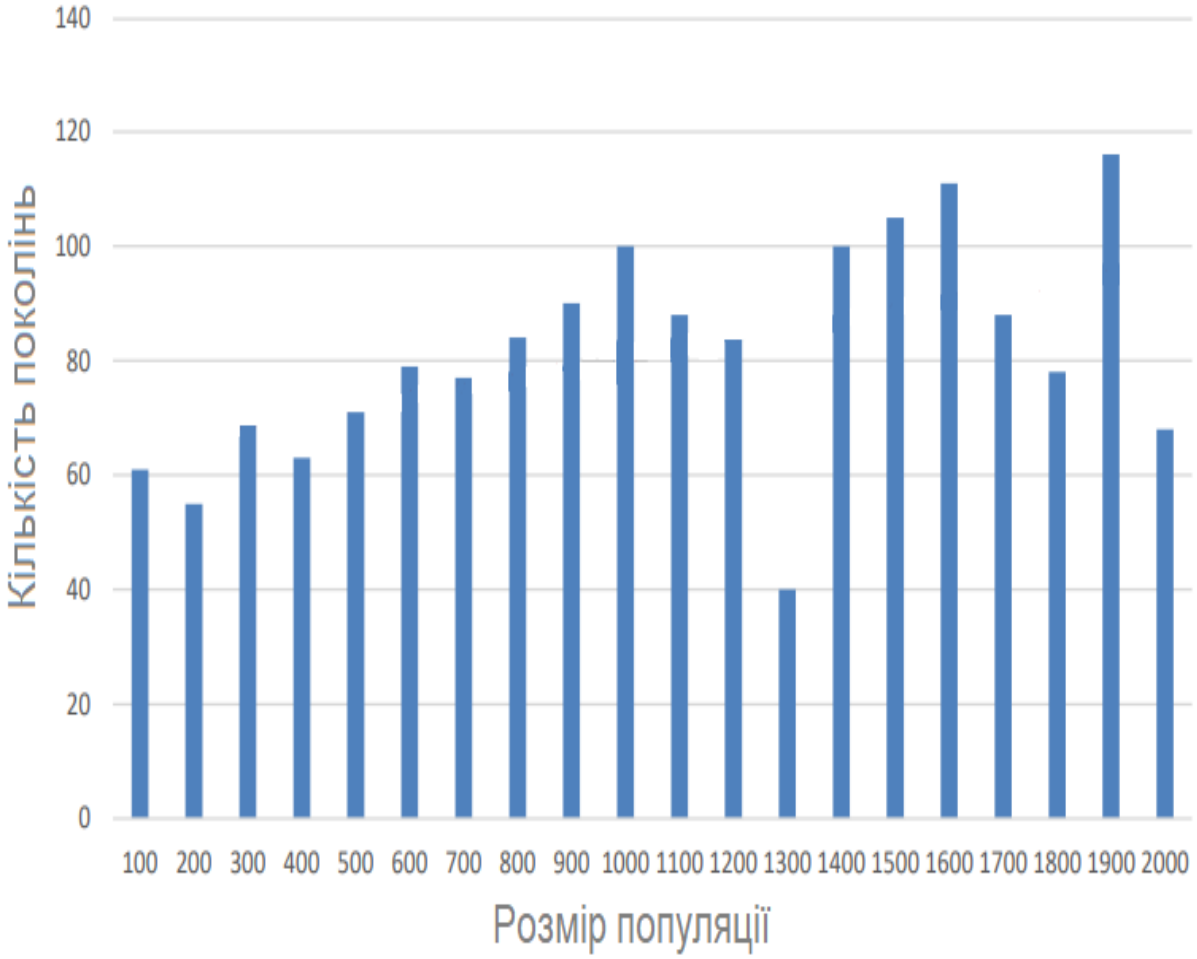


Рисунок 3.6 – Залежність кількості поколінь від розміру популяції

Частота мутацій відіграє важливу роль у генетичних алгоритмах.

Вона визначає ймовірність зміни освітнього компоненту під час створення та формування нащадків в освітній програмі. В результаті проведення цього тесту було визначено кількість зіткнень (clashes) (рисунок 3.7). та значення придатності (fitness) (рисунок 3.8). в залежності від значення частоти мутацій. Кількість поколінь було обмежено 1000, діапазон частоти мутацій був від 0,01 до 0,21.

Аналізуючи ці графіки, можна зробити висновки про оптимальне значення частоти мутацій, яке дозволяє досягти найкращих результатів у генетичному алгоритмі для даної задачі.

Рисунок 3.7 – Залежність кількості зіткнень від частоти мутацій

Як видно з графіка вище, збільшення частоти мутацій збільшує кількість зіткнень. У результаті швидкість мутації є оптимальною на рівні 0,01.

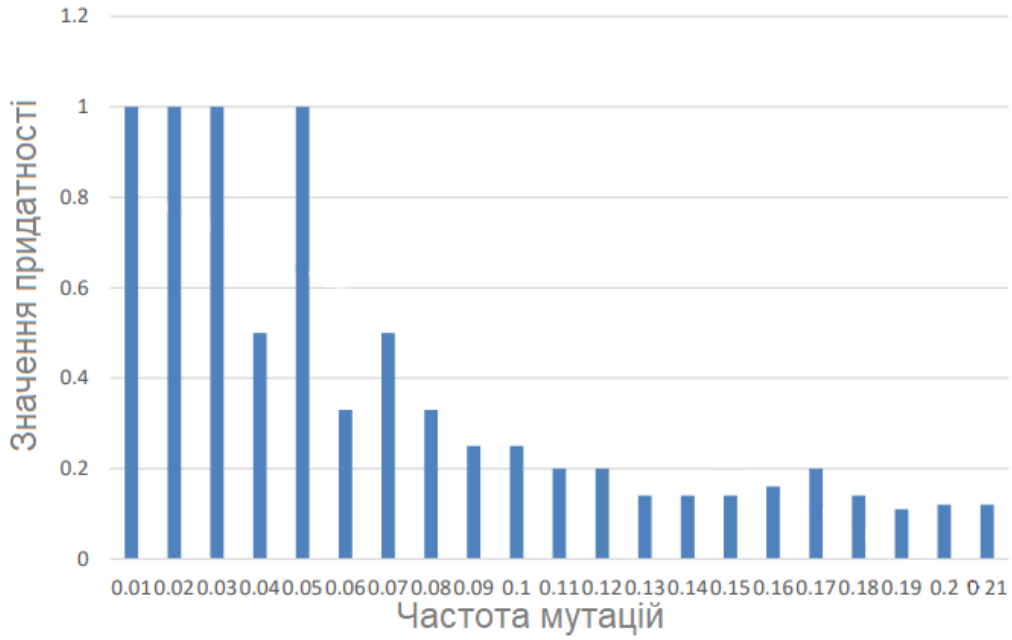


Рисунок 3.8 – Залежність значення придатності від частоти мутацій

3.5 Висновки до третього розділу

Виконано програмну реалізацію способу побудови освітньої програми за генетичним алгоритмом із визначенням структури модулів системи та їх взаємозв’язку.

Програмну реалізацію способу побудови освітньої траєкторії виконано з урахуванням особливостей імплементації генетичного алгоритму.

Визначено функціональні можливості реалізованої інформаційної системи.

Проведено апробацію способу побудови освітньої програми за генетичним алгоритмом на прикладі освітніх компонент та їх характеристик для спеціальності «Комп’ютерні науки».

Висновки

В кваліфікаційній роботі реалізовано спосіб побудови освітньої програми за генетичним алгоритмом

Досягнута мета кваліфікаційної роботи, яка полягає в розробці способу побудови освітньої програми за генетичним алгоритмом.

Для досягнення поставленої мети було вирішено такі задачі – проведено аналіз методів побудови освітньої програми, її цілей і завдань, а також проаналізовано можливості, переваги та недоліки генетичного алгоритму; визначено послідовність застосування способу побудови освітньої програми за генетичним алгоритмом; реалізовано інформаційну технологію способу побудови освітньої програми за генетичним алгоритмом; проведено експериментальне тестування інформаційної системи.

Реалізовано спосіб побудови освітньої програми за генетичним алгоритмом включно із загальними положеннями щодо побудови освітньої програми. Розроблено загальну послідовність побудови освітньої програми із визначення стадій та етапів.

Генетичний алгоритм адаптовано до вирішуваної задачі із визначенням сутностей «популяція», «особина», «хромосома», «ген» та формування послідовності використання генетичного алгоритму. Спроектовано структуру інформаційної системи.

Виконано програмну реалізацію способу побудови освітньої програми за генетичним алгоритмом із визначенням структури модулів системи та їх взаємозв’язку. Програмну реалізацію способу побудови освітньої траєкторії виконано з урахуванням особливостей імплементації генетичного алгоритму. Визначено функціональні можливості реалізованої інформаційної системи.

Проведено апробацію способу побудови освітньої програми за генетичним алгоритмом на прикладі освітніх компонент та їх характеристик для спеціальності «Комп’ютерні науки».

Перелік посилань

1. Про освіту. URL: https://zakon.rada.gov.ua/laws/show/2145-19#Text

2. Національний авіаційний університет. URL: https://nau.edu.ua/download/Quality%20Assurance_ukr/Systema_QA/Documentacija_QA/14_05_2020/2020_05_12_Pologenja_pro_osvitni_programi_NAU_end.pdf

3. Вибіркова складова ОПП. URL: https://ftbrp.chdtu.edu.ua/vybirkova-skladova-opp/

4. Основні форми контролю знань студентів. URL: https://osvita.ua/vnz/reports/pedagog/14679/

5. Створення внутрішньої системи забезпечення якості вищої освіти. URL: https://naqa.gov.ua/wp-content/uploads/2021/07/Recommendations-UA.pdf

6. Підручник за модульно-рейтинговою системою навчання для студентів магістратури. URL: https://shron1.chtyvo.org.ua/Vitvytska_Svitlana/Osnovy_pedahohiky_vyschoi_shkoly.pdf

7. Про формування і реалізацію освітніх програм ЗВО. URL: https://osvita.ua/vnz/reform/62689/

8. Компетентності і результати навчання у нових стандартах вищої освіти. URL: http://education-ua.org/ru/articles/702-kompetentnosti-i-rezultati-navchannya-u-novikh-standartakh-vishchoji-osviti

9. Компетентності і результати навчання у нових стандартах вищої освіти. URL: http://education-ua.org/ru/articles/702-kompetentnosti-i-rezultati-navchannya-u-novikh-standartakh-vishchoji-osviti

10. Нормативний інструментарій внутрішнього забезпечення якості освітньої діяльності в Чернівецькому національному університеті імені Юрія Федьковича. URL: https://fmi.chnu.edu.ua/media/agqpgvmc/zbirnyk-normatyvnykh-dokumentiv-chnu_2021.pdf

11. Ключові освітні компетентності. URL: https://osvita.ua/school/method/2340/

12. Методологія Tuning. URL: https://www.unideusto.org/tuningeu/images/stories/documents/General_Brochure_Ukrainian_version.pdf

13. Методичні підходи до формування варіативної складової освітніх програм в Харківському національному економічному університеті імені Семена Кузнеця. URL: https://www.hneu.edu.ua/wp-content/uploads/2018/11/Metodychni-pidhody-do-formuvannya-variatyvnoyi-skladovoyi-osvitnih-prohram-1.pdf

14. Інженерно-технічна освіта. URL: http://vstup.kpi.kharkov.ua/admission/engineering/

15. Індивідуалізація і диференціація викладання. URL: http://posibnyk.nus.org.ua/theme-5-individualization-and-differentiation-teaching-as-taking-into-account/

16. Інтегративний підхід: актуальність, сутність, особливості. URL: https://znayshov.com/FR/6748/235.pdf

17. Що таке компетентнісний підхід у навчанні. URL: https://nus.org.ua/questions/zo-take-kompetentnisnyj-pidhid-u-navchanni-vidpovidaye-derzhavna-sluzhba-yakosti-osvity/

18. Технологічний підхід як засіб підвищення ефективності навчально-виховного процесу. URL: https://phm.cuspu.edu.ua/nauka/konferentsii/fizyka-tekhnolohii-navchannia/82-2016/teoriia-ta-metodyka-tekhnolohichnoi-osvity/556-tekhnolohichnyy-pidkhid-yak-zasib-pidvyshchennya-efektyvnosti-navchalno-vykhovnoho-protsesu.html

19. Інтердисциплінарний підхід до викладання предметів. URL: https://svitppt.com.ua/pedagogika/interdisciplinarniy-pidhid-do-vikladannya-predmetiv-prirodnichomatemat.html

20. Київський національний університет імені Тараса Шевченка. URL: http://www.univ.kiev.ua/

21. Національний технічний університет «Харківський політехнічний інститут». URL: https://www.kpi.kharkov.ua/ukr/

22. Львівська національна музична академія імені Миколи Лисенка. URL: https://lnma.edu.ua/

23. Освітні програми, що реалізуються в закладі освіти. URL: http://antonivska-zosh21.ks.sch.in.ua/publichna_informaciya/osvitni_programi_scho_realizuyutjsya_v_zakladi_osviti/

24. Генетичні алгоритми як інструмент розв’язання нелінійних крайових задач. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/122839/02-Vakal.pdf?sequence=1

25. Освітні програми, що реалізуються в закладі освіти. URL: http://antonivska-zosh21.ks.sch.in.ua/publichna_informaciya/osvitni_programi_scho_realizuyutjsya_v_zakladi_osviti/

26. Wikipedia Генетичний алгоритм. URL: https://uk.wikipedia.org/wiki/Генетичний_алгоритм

27. aSc Timetables. URL: https://www.asctimetables.com/

28. Moodle. URL: https://moodle.org/?lang=uk

ДОДАТКИ

Додаток А

Програмні коди

Лістинг index.php:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Educational program Generator</title>
<link href='css/bootstrap.min.css'rel="stylesheet">
</head>
<body>

<div class="container row">
    <div class="col-xs-2">
    </div>
    <div class="col-xs-3">
        <img src="imgs/logo.png" alt="logo" height="200"/>
    </div>
    <div class="col-xs-6">
        <br/>   <br/> <br/>
        <h3>The generator of the educational program using the genetic algorithm</h3>
        <hr/>

    <br/><br/>

    <h3>Enter the required information</h3>

        <form id="form1" name="form1" enctype="multipart/form-data" method="post"
action="tTable_gen.php">

            <div class="form-group">
                <label>Course</label>
                <select class="form-control" name="semester">
                    <option>Software engineering</option>
                     <option>Computer Science</option>

                </select>
            </div>
```

```
            <div class="form-group">
                <label>Schedule Type</label>
                <select class="form-control" name="s_type">
                    <option>Class Schedule</option>
                     <option>Examination Schedule</option>

                </select>
            </div>
            <div class="form-group">
                <label>Config (.Csv File)</label>
                <input type="file" name="userfile" id="fileField" class="form-control"/>
            </div>
            <div class="form-group">
                  <input name="generate" type="submit" value="Generate" class="btn btn-
success"/>
            </div>
    </form>

    </div>
    <div class="col-xs-2">
    <hr/><hr/>
    </div>

</div>
<br/><br/><br/><br/>
<div class="row">
    <div class="col-xs-2">
    </div>

    <div class="col-xs-8">

    </div>
</div>

</body>
</html>
```

Лістинг table_gen.php:

```
<?php
define('CLASS_DIR','classes\\');
define('OBJECT_DIR','classes\\Objects\\');
$cls_files = array(CLASS_DIR.'randomGenerator',
                    OBJECT_DIR.'cpompetence',
```

```
                    OBJECT_DIR.'result',
                    OBJECT_DIR.'Students',
                    OBJECT_DIR.'programResult',
                    CLASS_DIR.'objectRespository',
                    CLASS_DIR.'fitnessFixedConstraint',
                    CLASS_DIR.'mutation',
                    CLASS_DIR.'display',
                    OBJECT_DIR.'courseName');

 function file_includer($fl){
    for($i =0;$i<count($fl);$i++){
        include_once($fl[$i].'.php');
    }
}
file_includer($cls_files);
?>


<?php
use classes\randomGenerator,
    classes\Objects\competence,
    classes\Objects\result,
    classes\Objects\programresults,
    classes\Objects\course,
    classes\objectRespository,
    classes\fitnessFixedConstraint,
    classes\mutation,
    classes\display,
    classes\Objects\courseName;

class index {

    const DAYS_OF_WEEK = 5;
    const HOURS_PER_DAY = 11;

    const ROOM_1 = 0;
    const ROOM_2 = 1;
    const ROOM_3 = 2;

    private $file_location;
    private $ob;
    private $schedule_type;
```

```
private $semester;

public function __construct($csvFile,$st,$sm){
    $this->file_location = $csvFile;
    $this->schedule_type = $st;
    $this->semester = $sm;
}

public function rndGen($min_x,$max_x,$qty){
    $rnds = randomGenerator::generate($min_x,$max_x,$qty);
    return $rnds;
}

private function brkLine(){
    echo '<br/>';
}

private function echoBrkLine($a){
    echo $a.$this->brkLine();
}

private function map_class_to_hours($cls_hours){
    $sched = array( self::ROOM_1 => array(),
                    self::ROOM_2 => array(),
                    self::ROOM_3 => array());

    for($i=0; $i < count($cls_hours); $i++){
        for($j = 0;$j<count($cls_hours[$i]); $j++){
            if(is_object($cls_hours[$i][$j])){
                $sched[$i][$j] = $cls_hours[$i][$j];
                $j+=$sched[$i][$j]->getDuration()-1;
            }
        }
    }
    return $sched;
}

public function initialise_schedule(){//'C:\wamp\www\Ga_time_table\config.csv'
    $t_space = self::DAYS_OF_WEEK * self::HOURS_PER_DAY;
    $this->ob = new objectRespository($this->file_location);

    $clsRoom = array(self::ROOM_1 => array(),
```

```
                        self::ROOM_2 => array(),
                        self::ROOM_3 => array());

        $schedule = $clsRoom; //DRY concept (don't repeat yourself)
        $randNums = $clsRoom;

          $randNums[self::ROOM_1] = $this->rndGen(0,$t_space,21); //Generate 21 random
numbers
        $randNums[self::ROOM_2] = $this->rndGen(0,$t_space,21);
        $randNums[self::ROOM_3] = $this->rndGen(0,$t_space,21);

        $clsRoom[self::ROOM_1] = array_fill(0,$t_space,0);  //Fill each room with 0's
        $clsRoom[self::ROOM_2] = array_fill(0,$t_space,0);
        $clsRoom[self::ROOM_3] = array_fill(0,$t_space,0);

        $course_class = $this->ob->getCourse_Classes(); //Get the lectures

        for($i = 0; $i < count($course_class); $i++){
            $buff = array();
            $test = array();

            $dur = $course_class[$i]->getDuration();
            $cls = rand()%3;
            $time = abs($randNums[$cls][$i]);

            while(is_object($clsRoom[$cls][$time])){
                $time = rand()%60;
                if(!is_object($clsRoom[$cls][$time])){
                    break;
                }
            }

            if( $this->schedule_type == 'Examination Schedule' ){
                $rand_lect = $this->rndGen(1,10,2);
                $lects = '';
                foreach($rand_lect as $k => $p){
                    $lects.=$p .'-';
                }
                $lects = substr_replace($lects, '', strlen($lects)-1);
                $course_class[$i]->setProf($lects);
            }
```

```
            $buff = $clsRoom[$cls];
            $test = array_fill(0,$dur,$course_class[$i]);
            array_splice($buff,$time,0,$test);
            $clsRoom[$cls] = $buff;
        }
        $schedule = $this->map_class_to_hours($clsRoom);

        $fit = $this->calculateFitness($schedule,$clsRoom,count($course_class));
        $dr = new mutation();
        $generations = 1;
        $crossovers = 0;
        $elite = array();
        $mutations = 0;
        $new_fit = 0;

        while(($fit >= 1) || ($generations < 1000)){
            $dr->set_class_hours($clsRoom);
            $dr->set_schedule($schedule);

            if(rand()%3 == 2){
                $dr->do_crossover();
                $copy_cls_hrs = $dr->get_class_hours();
                $copy_cls_sch = $dr->get_schedule();
                                                        $new_fit = $this-
>calculateFitness($copy_cls_sch,$copy_cls_hrs,count($course_class));
                $crossovers++;
            }

            if(rand()%3 == 2){
                $dr->mutation();
                $mutations++;
            }

            if($new_fit > $fit){
                if($fit >= 0.95){
                    $elite[] = $clsRoom;
                }
                $schedule = $copy_cls_sch;
                $clsRoom = $copy_cls_hrs;
                $fit = $new_fit;
            }
            $generations++;
```

```
        }
        echo '<h1 style="font-align:center">'. $this->semester. '</h1>';
        echo '<h2 style="font-align:center"><u> Fitness Statistics </u></h2>';
        $fit = $this->calculateFitness($schedule,$clsRoom,count($course_class),true);
        //$this->brkLine();
        $this->echoBrkLine('Elite Chromosomes: '.count($elite));
        $this->brkLine();
        $this->echoBrkLine('Number of Generations: '. $generations);
        $this->brkLine();
        $this->echoBrkLine('Number of Crossovers: '. $crossovers);
        $this->brkLine();
        $this->echoBrkLine('Number of Mutations: '. $mutations);

        $ds = new display($schedule,$clsRoom,$this->ob);
        $ds->inputCourses();
    }

                                                private                          function
calculateFitness($schedule_array,$class_hours_array,$number_of_courses,$rpt=false){
        $schScore =0;
        $fit = new fitnessFixedConstraint($schedule_array,$number_of_courses,$this->ob);
        $schScore += $fit->classesNeedSpecialRoom();
        $schScore += $fit->classroomSize();
        $schScore += $fit->consecutiveHours($class_hours_array);
        $schScore += $fit->clashingClasses('#group');
        $schScore += $fit->clashingClasses('#prof');
        if($rpt == true){
            //$this->echoBrkLine('Need Special Room: '.$fit->classesNeedSpecialRoom());
            //$this->echoBrkLine('Classroom size: '.$fit->classroomSize());
                                //$this->echoBrkLine('Consecutive    Hours:    '.    $fit-
>consecutiveHours($class_hours_array));
            $this->echoBrkLine('Clashing course '.$fit->clashingClasses('#group'));
                            //$this->echoBrkLine('Clashing   course   lecturers:   '.$fit-
>clashingClasses('#prof'));
            //$this->echoBrkLine('Total Courses = '.$number_of_courses);
                $this->echoBrkLine('Fitness  Score  =  '  .$schScore/($number_of_courses  *
self::DAYS_OF_WEEK));
        }
        return ($schScore/($number_of_courses * self::DAYS_OF_WEEK));
    }

    private function get_lecturer_count(){
```

```
        $lecturers = $this->ob->getLecturers();
        $lect_count = 0;
        foreach($lecturers as $k => $p){
            $lect_count++;
        }
        return $lect_count;
    }

}

    $uploaddir = 'csv/';
    $uploadfile = $uploaddir . $_FILES['userfile']['name'];
    $schedule_type = $_POST['s_type'];
    $sem = $_POST['semester'];

    if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {

        $start_time = microtime(true);
        $start_memory = memory_get_peak_usage(false);

        $id = new index($uploadfile,$schedule_type,$sem);
        $id->initialise_schedule();

        $end_memory = memory_get_peak_usage(false);
        $end_time = microtime(true);

         echo '<h2>Speed Evaluation = '. number_format(($end_time - $start_time),4). '
Seconds </h2>';
            //echo '<h2>Memory Utilization = '. ($end_memory - $start_memory). '
Bytes</h2>';


    } else {
        echo "File Upload Error!\n";
    }
?>
```

Лістинг display.php:

```
<?php
    namespace classes;
    class display {
```

```
    private $_classes = array();
    private $_sched = array();
    private $_lecturer = array();
    private $_courseName = array();
    private $_room = array();
    private $_group = array();
    private $_lectures = array();

    public function __construct(array $s,array $c,$objectRepo){
        $this->_classes = $c;
        $this->_sched = $s;

        $ob = $objectRepo;

        $this->_lecturer = $ob->getLecturers();
        $this->_courseName = $ob->getCourseName();
        $this->_room = $ob->getRooms();
        $this->_group = $ob->getStudentGroups();
        $this->_lectures = $ob->getCourse_Classes();
        $this->create_css();
    }

    private function create_css(){
        $css = '<head>
                <style type="text/css">
                    body{margin:0px; padding:0px}
                    table.Ttable{border:solid thin; width:100%}
                    /*table.small-table{border:solid thin; width:2%; height:2%} */
                              td.courses{width:30px;  max-width:30px;  align:center;text-
align:center}
                    td.time{width:8px; height:8px; border:none; text-align:center}
                    td.day{border:none; text-align:center}
                    p.class-details{font-size:8px; width:5px;}
                              .Ttable>tbody>tr:nth-child(odd)>td,.timeTable>tbody>tr:nth-
child(odd)>th{background-color:#eaeaea}
                </style>
                <title>GA</title>
                </head>';
        echo $css;
    }

    private function startTable(){
```

```
        echo '<table border="1" class="Ttable">';
    }

    private function endTable(){
        echo '</table>';
    }

    public function inputCourses(){
        for($u =0;$u<1;$u++){
            //$cl_det = '<table align="center" border="1">';
            //$cl_det .= '<tr align="center"><td colspan="2">';
            //$cl_det .= '<h3>Name: '.$this->_room[$u]->getName().'</h3></td></tr>';
            //$cl_det .= '<tr align="center"><td>';
            //$cl_det .='<h3>Lab: '.$this->_room[$u]->getLab().'</h3></td>';
            //$cl_det .= '<td><h3>Seats: '.$this->_room[$u]->getRoomSize().'</h3></td>';
            //$cl_det .= '</table>';
            //echo $cl_det;

            $this->startTable();
            $cls_details = '<tr><td class="time"></td>';
            echo $cls_details;
                for($i=2;$i<=6;$i++){
                                                                        echo      '<td
class="day"><h4>'.date('D',gmmktime(0,0,0,1,$i)).'</h4></td>';
                }
            echo '</tr>';
            for($i=0;$i<12;$i++){
                $d = $i + 8;
                $b = '';
                    if($d<12){
                                $b  .=  '<tr><td  class="time"><h4>'.  $d  .'  -  '.  ($d+1)
.'</h4></td>';
                    }
                    elseif($d == 12){
                           $b  .= '<tr><td class="time"><h4>'. $d .' - '. (($d+1) - 12)
.'</h4></td>';
                    }
                    else{
                         $b .= '<tr><td class="time"><h4>'. ($d - 12) .' - '. (($d + 1) -
12) .'</h4></td>';
                    }
                echo $b;
```

```
                for($j=0;$j<5;$j++){
                    $a ='<td class="courses">';
                    $c = $i + ($j * 12);
                    if(is_object($this->_classes[$u][$c])){
                        $a .= $this->getCourseComponents($this->_classes[$u][($c)],0);
                    }
                    $a.='</td>';
                    echo $a;
                }
                echo '</tr>';
            }
            $this->endTable();
            echo $this->brkLine();
        }
    }

    private function brkLine(){
        return '<br/>';
    }

    private function getStudentGroupName($grps){
        $stdGroup = array();
        $grp_arr = array();
        $result = '';

        if(substr_count($grps,'-') != 0){
            $stdGroup = explode('-',$grps);
        }
        else{
            $stdGroup[] = $grps;
        }

        foreach($stdGroup as $g){
            $grp_arr[] = $this->_group[(int)$g-1]->getName();
        }
        return implode('|',$grp_arr);
    }

    private function getCourseComponents($course,$cls){
              $c   =   '<b>Назва:    </b>'.$this->_courseName[($course->getCourseName()-1)]-
>getName() . $this->brkLine();
        $c .= '<b>ЗК,ФК:  </b>'. $this->sortLecturers($course) . $this->brkLine();
```

```
        $c .= '<b>ПРН:  </b>'.$this->_room[$cls]->getName() . $this->brkLine();
         $c .= $course->getLab() == 'true' ? '<b>Іспит:</b>  Yes'. $this->brkLine() :
'<b>Іспит:  </b> No' . $this->brkLine();
        $c .= '<b>ЄКТС:  </b>'.$this->getStudentGroupName($course->getStudentGroup());
        return $c;
    }

    private function sortLecturers($c){
        $bunch = $c->getProf();

        if(substr_count($bunch,'-') != 0){
            $bunch = explode('-',$bunch);
        }
        else{
            $a = $bunch;
            $bunch = array();
            $bunch[] = $a;
        }
        $l = '';

        foreach($bunch as $k){
            $l .= $this->_lecturer[ $k-1 ]->getName() .',';

        }
        $l = substr_replace($l, '', strlen($l)-1);
        return $l;
    }

}
?>
```

Лістинг file_parser.php:

```
<?php
    namespace classes;

    class file_parser{
        private $_file_contents = '';

        public function __construct(array $f){
            $this->_file_contents = $f;
        }
```

```
        public function parse($tag){
            $tag_contents = array();

            for($i = 0; $i< count($this->_file_contents);$i++){
                $obj = explode(',',$this->_file_contents[$i]);
                    if($obj[0] == $tag){
                    $tag_contents[] = $this->_split_to_Assoc_Array($obj);
                }
            }
            return $tag_contents;
        }

        private function _split_to_Assoc_Array($reg_arr){
            $res = array();
            for($i = 1;$i < count($reg_arr); $i++){
                $k = trim(substr($reg_arr[$i],0,strpos($reg_arr[$i],'=')));
                $v = trim(substr($reg_arr[$i],strpos($reg_arr[$i],'=') + 1));
                $res[$k] = $v;
            }
            return $res;
        }
    }
?>
```

Лістинг fitnessFixedConstraint.php

```
<?php
    namespace classes;

    class fitnessFixedConstraint {

        private $_classes = array();
        private $_ob;
        private $rooms;
        private $_group;
        private $_num_of_classes;

        public function __construct(array $c,$cl_count,$objectRepo){
            $this->_classes = $c;
            $this->_ob = $objectRepo;
            $this->rooms = $this->_ob->getRooms();
            $this->_group = $this->_ob->getStudentGroups();
            $this->_num_of_classes = $cl_count;
```

```
        }

        public function classesNeedSpecialRoom(){
            $score_counter = 0;
            foreach($this->_classes as $k => $v){
                foreach($v as $p => $c){

                      if(($c->getLab() == 'false')||(($c->getLab() == 'true') && ($this-
>rooms[$k]->getLab() == 'true'))){
                          $score_counter +=1;
                    }
                }
            }
            return $score_counter;
        }

        public function classroomSize(){
            $score_counter = 0;
            $stdGroup = $this->_ob->getStudentGroups();
            foreach($this->_classes as $k => $v){
                foreach($v as $p => $c){
                    $t_students = 0;
                    $studs = $c->getStudentGroup();
                    if(substr_count($studs,'-') != 0){
                        $studs = explode('-',$c->getStudentGroup());
                        for($i=0;$i < count($studs); $i++){
                                        $t_students  +=  $stdGroup[($studs[$i]  -  1)]-
>getStudentGroupSize();
                        }
                    }
                    else{
                                           $t_students  +=  $stdGroup[($studs  -  1)]-
>getStudentGroupSize();
                    }
                    if($this->rooms[$k]->getRoomSize() >= $t_students){
                        $score_counter++;
                    }
                }
            }
                return $score_counter;
        }
```

```
public function consecutiveHours($s){
    $cnt = 0;
    $clsDurationCount = 0;
    $score_counter = 0;

    foreach($this->_classes as $k => $v){
        foreach($v as $p => $c){
            $crses = array_slice($s[$k],$p,$c->getDuration());
            if(count($crses) == $c->getDuration()){
                $score_counter++;
            }
        }
    }
    return $score_counter;
}

public function clashingClasses($tag){
    $score_counter = $this->_num_of_classes;
    $totalClash = 0;
    $stdGroup = '';
    $buff = '';
    foreach($this->_classes as $k => $v){
        $clss = array();
        if($this->classToCompare($k) == null){
            break;
        }
        $clss = $this->classToCompare($k);
            for($i=0;$i<count($clss);$i++){
                $intersect = array_intersect_key($v,$this->_classes[$clss[$i]]);

                if($intersect != null){
                    foreach($intersect as $ke => $va){

                        if($tag == '#prof'){
                            $buff = $this->lecturerToCompare($clss[$i],$k,$ke);
                        }
                        elseif($tag == '#group'){
                            $buff = $this->studentToCompare($clss[$i],$k,$ke);
                        }
                        $st = $buff[0];
                        if(substr_count($buff[1],'-') != 0){
                            $stdGroup = explode('-',$buff[1]);
```

```
                        }
                        else{
                            $stdGroup[] = $buff[1];
                        }
                        foreach($stdGroup as $s){
                            if(substr_count($st,$s) != 0){
                                $totalClash++;
                                break;
                            }
                        }
                    }
                }
            }
        }
                    return $score_counter - $totalClash;

}

private function classToCompare($cur_class){
    $clss = array();
    if($cur_class == 0){
        $clss[] = 1;
        $clss[] = 2;
    }
    elseif($cur_class == 1){
        $clss[] = 2;
    }
    else{
        $clss = null;
    }
    return $clss;
}

private function studentToCompare($class_to_comp,$cur_class,$time){
    $buff = array();
    $st = $this->_classes[$class_to_comp][$time]->getStudentGroup();
     $stdGroup = $this->_classes[$cur_class][$time]->getStudentGroup();

    $buff[] = $st;
    $buff[] = $stdGroup;
    return $buff;
}
```

```
        private function lecturerToCompare($class_to_comp,$cur_class,$time){
            $buff = array();
            $lct = $this->_classes[$class_to_comp][$time]->getProf();
            $lctGroup = $this->_classes[$cur_class][$time]->getProf();
            $buff[] = $lct;
            $buff[] = $lctGroup;
            return $buff;
        }
    }
?>
```

Лістинг mutation.php:

```
<?php
namespace classes;

class mutation {
    private $_classes = array();
    private $_sched = array();

    public function __construct(){

    }

    public function set_class_hours($cls_hours){
        $this->_classes = $cls_hours;
    }

    public function set_schedule($sch){
        $this->_sched = $sch;
    }

    public function get_class_hours(){
        return $this->_classes;
    }

    public function get_schedule(){
        return $this->_sched;
    }

    private function randClass(){
        $rand = array();
```

```
    $rand[] = $this->genRnd();
    $rand[] = $this->genRnd();
    while($rand[0]==$rand[1]){
        $rand[0] = $this->genRnd();
        $rand[1] = $this->genRnd();
        if($rand[0] != $rand[1]){
            break;
        }
    }
    return $rand;
}

private function genRnd(){
    return rand()%3;
}

private function map_class_to_hours($cls_hours){
    $sched = array();
    for($i=0; $i < count($cls_hours); $i++){
        for($j = 0;$j<count($cls_hours[$i]); $j++){
            if(is_object($cls_hours[$i][$j])){
                $sched[$i][$j] = $cls_hours[$i][$j];
                $j+=$sched[$i][$j]->getDuration()-1;
            }
        }
    }
    return $sched;
}

private function swapClasses($c1,$c2){
    $cls = array();
    $proxy_classes = $this->_classes;
    $cls[] = $proxy_classes[$c1];
    $cls[] = $proxy_classes[$c2];
    $mediator = $cls[1];
    $cls[1] = $cls[0];
    $cls[0] = $mediator;
    return $cls;
}

public function class_reselect(){
    $rnds = $this->randClass();
```

```
        $new_clss_hrs = $this->swapClasses($rnds[0],$rnds[1]);

        $this->_classes[$rnds[0]] = $new_clss_hrs[0];
        $this->_classes[$rnds[1]] = $new_clss_hrs[1];

        $this->_sched = $this->map_class_to_hours($this->_classes);

    }

    private function fill_arr($time,$duration,$content){
        $keys = array();
        $t = $time;
        for($i=0;$i<$duration;$i++){
            $keys[] = $t++;
        }
        return array_fill_keys($keys,$content);
    }

    public function hour_reselect(){
        $rnds = $this->randClass();
        $cls = array(0,0);

        do{
            $cls[0] = array_rand($this->_sched[$rnds[0]],1);
            $cls[1] = array_rand($this->_sched[$rnds[1]],1);
                        if($this->_sched[$rnds[0]][$cls[0]]->getDuration()   ==   $this-
>_sched[$rnds[1]][$cls[1]]->getDuration()){
                break;
            }
                    }while($this->_sched[$rnds[0]][$cls[0]]->getDuration()    !=    $this-
>_sched[$rnds[1]][$cls[1]]->getDuration());

        for($i=0; $i<2; $i++){
            $sel_cls_time[] = $this->_classes[$rnds[$i]][$cls[$i]];
        }
        $cls_duration = $this->_classes[$rnds[0]][$cls[0]]->getDuration();
        $clss_swapperA = array();
        $clss_swapperB = array();
        $clss_swapperA = $this->fill_arr($cls[0],$cls_duration,$sel_cls_time[1]);
        $clss_swapperB = $this->fill_arr($cls[1],$cls_duration,$sel_cls_time[0]);

        $proxy_cls = array_replace($this->_classes[$rnds[0]],$clss_swapperA);
```

```
        $this->_classes[$rnds[0]] = $proxy_cls;
        $proxy_cls = array_replace($this->_classes[$rnds[1]],$clss_swapperB);
        $this->_classes[$rnds[1]] = $proxy_cls;

        $this->_sched = $this->map_class_to_hours($this->_classes);
    }


    public function single_classroom_reselect($sch=''){
        $rnds = $this->randClass();
        $cls = array(0,0);

        do{
                $cls[0] = array_rand($this->_sched[$rnds[0]],1);
                $cls[1] = array_rand($this->_sched[$rnds[0]],1);
                if($cls[0]!=$cls[1]){
                            if($this->_sched[$rnds[0]][$cls[0]]->getDuration()  ==  $this-
>_sched[$rnds[0]][$cls[1]]->getDuration()){
                        break;
                    }
                }
        }while(1);

        for($i=0; $i<2; $i++){
            $sel_cls_time[] = $this->_classes[$rnds[0]][$cls[$i]];
        }
        //$this->_classes[$rnds[0]][$cls[1]]->getDuration()
        $cls_duration = $this->_classes[$rnds[0]][$cls[0]]->getDuration();
        $clss_swapperA = array();
        $clss_swapperB = array();

        $clss_swapperA = $this->fill_arr($cls[0],$cls_duration,$sel_cls_time[1]);
        $clss_swapperB = $this->fill_arr($cls[1],$cls_duration,$sel_cls_time[0]);

        $proxy_cls = array_replace($this->_classes[$rnds[0]],$clss_swapperA);
        $this->_classes[$rnds[0]] = $proxy_cls;
        $proxy_cls = array_replace($this->_classes[$rnds[0]],$clss_swapperB);
        $this->_classes[$rnds[0]] = $proxy_cls;

        $this->_sched = $this->map_class_to_hours($this->_classes);
    }
```

```
    public function mutation(){
        $rnds = $this->randClass();
        $cls = array(0,0);

        do{
                $cls[0] = array_rand($this->_sched[$rnds[0]],1);
                $cls[1] = array_rand($this->_sched[$rnds[1]],1);
                if($cls[0]!=$cls[1]){
                            if($this->_sched[$rnds[0]][$cls[0]]->getDuration()  ==  $this-
>_sched[$rnds[1]][$cls[1]]->getDuration()){
                        break;
                    }
                }
        }while(1);

        $dur = $this->_sched[$rnds[0]][$cls[0]]->getDuration();
        $new_time = array(0,0);
        $sel_cls_time = array();
        for($i=0;$i<2;$i++){
            $time = 0;
            do{
                $time = rand()%60;
                if(!is_object($this->_classes[$rnds[$i]][$time])){
                    break;
                }
            }while(is_object($this->_classes[$rnds[$i]][$time]));
            $new_time[$i] = $time;
        }

        for($i=0; $i<2; $i++){
            $sel_cls_time[] = $this->_classes[$rnds[$i]][$cls[$i]];
        }

        for($i=0;$i<2;$i++){
            $buff = $this->_classes[$rnds[$i]];
            $test = array_fill(0,$dur,$sel_cls_time[abs($i-1)]);
            array_splice($buff,$new_time[$i],0,$test);
            $this->_classes[$rnds[$i]] = $buff;
        }
        for($i=0;$i<2;$i++){
            $sel_cls_time[$i] = $this->fill_arr($cls[$i],$dur,0);
```

```
            array_replace($this->_classes[$rnds[$i]],$sel_cls_time[$i]);
        }
        $this->_sched = $this->map_class_to_hours($this->_classes);
    }

    public function do_crossover(){
        $this->class_reselect();
        $this->hour_reselect();
        $this->single_classroom_reselect();
    }
}
?>
```

Лістинг ObjectRepository.php:

```
<?php
    namespace classes;
    use classes\file_parser as fpass,
        classes\Objects\result,
        classes\Objects\programResults,
        classes\Objects\competence,
        classes\Objects\credits,
        classes\Objects\courseName;

    include_once('classes\file_parser.php');

    class objectRespository {
        private $rows;
        private $user_file;
        private $course_class;
        private $rooms;
        private $groups;
        private $prof;
        private $coursename;

        public function __construct($file_location){
            $this->user_file = $file_location;
            $this->course_class = $this->createObjects('#course');
            $this->rooms = $this->createObjects('#room');
            $this->groups = $this->createObjects('#group');
            $this->prof = $this->createObjects('#prof');
            $this->coursename = $this->createObjects('#coursename');
        }
```

```
public function getNameById(array $haystack,$id){
    for($i = 0; $i < count($haystack); $i++){
        if($haystack[$i]['id'] == $id){
            return $haystack[$i]['name'];
        }
    }
}


public function getAll($tag){
    if(file_exists($this->user_file)){
        $this->rows = file($this->user_file);
        $fp = new fpass($this->rows);
        return $fp->parse($tag);
    }
}

public function getCourse_Classes(){
    return $this->course_class;
}

public function getRooms(){
    return $this->rooms;
}

public function getStudentGroups(){
    return $this->groups;
}

public function getLecturers(){
    return $this->prof;
}

public function getCourseName(){
    return $this->coursename;
}

private function createObjects($tag){
    $anArr = $this->getAll($tag);
    $aObj = array();
    for($i = 0;$i <count($anArr); $i++){
```

```
                switch($tag){
                    case '#prof':
                        $aObj[] = new lecturer($anArr[$i]);
                        break;
                    case '#room':
                        $aObj[] = new classroom($anArr[$i]);
                        break;
                    case '#group':
                        $aObj[] = new students($anArr[$i]);
                        break;
                    case '#course':
                        $aObj[] = new course($anArr[$i]);
                        break;
                    case '#coursename':
                        $aObj[] = new courseName($anArr[$i]);
                        break;
                }
            }
            return $aObj;
        }

    }
?>
```

Лістинг randomGenerator.php:

```
<?php
namespace classes;

class randomGenerator{

    public static function generate($min,$max,$quantity){
        $numbers = range($min,$max);
        shuffle($numbers);
        return array_slice($numbers,0,$quantity);
    }
}
?>
```

Лістинг result.php:

```
<?php
    namespace classes\Objects;
    use classes\Objects\Inheritables\abstractObjectFactory;
```

```
    include_once('Inheritables\abstractObjectFactory.php');

    class classroom extends abstractObjectFactory{
        private $_lab;
        private $_roomSize;

        public function __construct(array $f){
            parent::__construct($f['id'],$f['name']);
            $this->_lab = $f['lab'];
            $this->_roomSize = (int)$f['size'];
        }

        public function getRoomSize(){
            return $this->_roomSize;
        }

        public  function getLab(){
            return $this->_lab;
        }

        public function toString(){
           // printf("Room Id: %s <br/> Room Name: %s <br/> Lab Equipment: %s <br/> Room
//Size: %s",$this->_Id,$this->_Name,$this->_lab,$this->_roomSize);
        }
    }
?>
<?php  ?>
```

Лістинг competence.php:

```
<?php
    namespace classes\Objects;
    use classes\Objects\Inheritables\abstractObjectFactory;

    include_once('Inheritables\abstractObjectFactory.php');

    class course {

        private $_prof;
        private $_duration;
        private $_student_group;
        private $_course_name;
```

```
private $_lab;
private $_classRoom;

public function __construct(array $f){
    //parent::__construct($f['id'],$f['name']);
    $this->_prof = $f['professor'];
    $this->_course_name = $f['coursename'];
    $this->_duration = $f['duration'];
    $this->_student_group = $f['group'];
    $this->_lab = $f['lab'];
}

public function getProf(){
    return $this->_prof;
}

public function setProf($p){
    $this->_prof = $p;
}

public function getCourseName(){
    return $this->_course_name;
}

public function getDuration(){
    return (int)$this->_duration;
}

public function getStudentGroup(){
    return $this->_student_group;
}

public function getLab(){
    return $this->_lab;
}

public function setClassRoom($r){
    $this->_classRoom = $r;
}

public function getClassRoom(){
    return $this->_classRoom;
```

```
        }

        public function toString(){
             printf("Prof: %s <br/> Course Name: %s <br/> Class Duration: %s Hours <br/>
 Student Group: %s",
                                                    $this->_prof,$this->_course_name,$this-
>_duration,$this->_student_group);
        }
    }
?>
```

Лістинг courseName.php:

```
<?php
    namespace classes\Objects;
    use classes\Objects\Inheritables\abstractObjectFactory;

    include_once('Inheritables\abstractObjectFactory.php');

    class courseName extends abstractObjectFactory{

        public function __construct(array $f){
            parent::__construct($f['id'],$f['name']);
        }

        public function toString(){
             printf("Course Id: %s <br/> Course Name: %s <br/>",$this->_Id,$this->_Name);

        }
    }
?>
```

Лістинг programResults.php:

```
<?php
    namespace classes\Objects;
    use classes\Objects\Inheritables\abstractObjectFactory;

    include_once('Inheritables\abstractObjectFactory.php');

    class lecturer extends abstractObjectFactory{

        public function __construct(array $f){
            parent::__construct($f['id'],$f['name']);
```

```
        }

        public function toString(){
              printf("Lecturer Id: %s <br/> Lecturer Name: %s <br/>",$this->_Id,$this->_Name);
        }
    }

?>
```

Лістинг credits.php:

```
<?php
    namespace classes\Objects;
    use classes\Objects\Inheritables\abstractObjectFactory;

    include_once('Inheritables\abstractObjectFactory.php');

    class students extends abstractObjectFactory{

        private $_size;

        public function __construct(array $f){
            parent::__construct($f['id'],$f['name']);
            $this->_size = $f['size'];
        }

        public function getStudentGroupSize(){
            return $this->_size;
        }

        public function toString(){
              printf("Id: %s <br/> Name: %s <br/>Student Group Size: %s <br/>",$this->_Id,$this->_Name,$this->_size);
        }
    }
?>
```

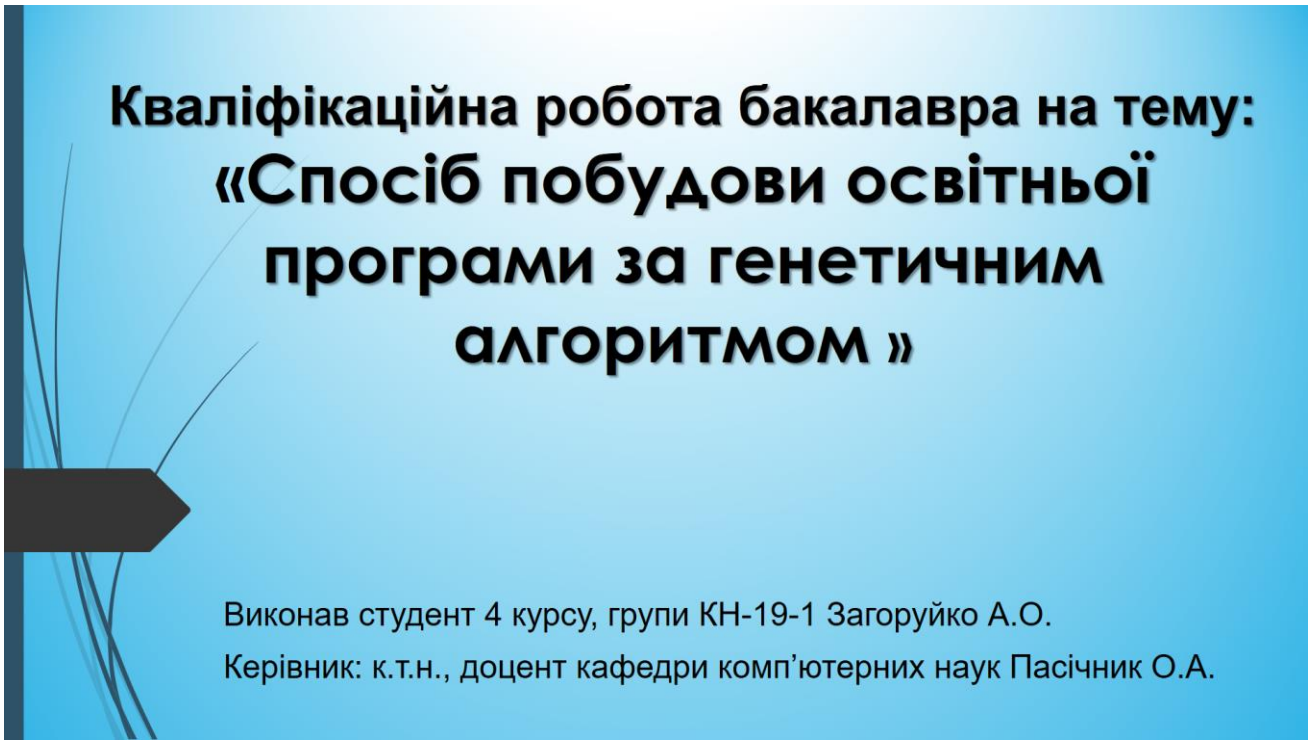
Лістинг abstractObjectFactory.php:

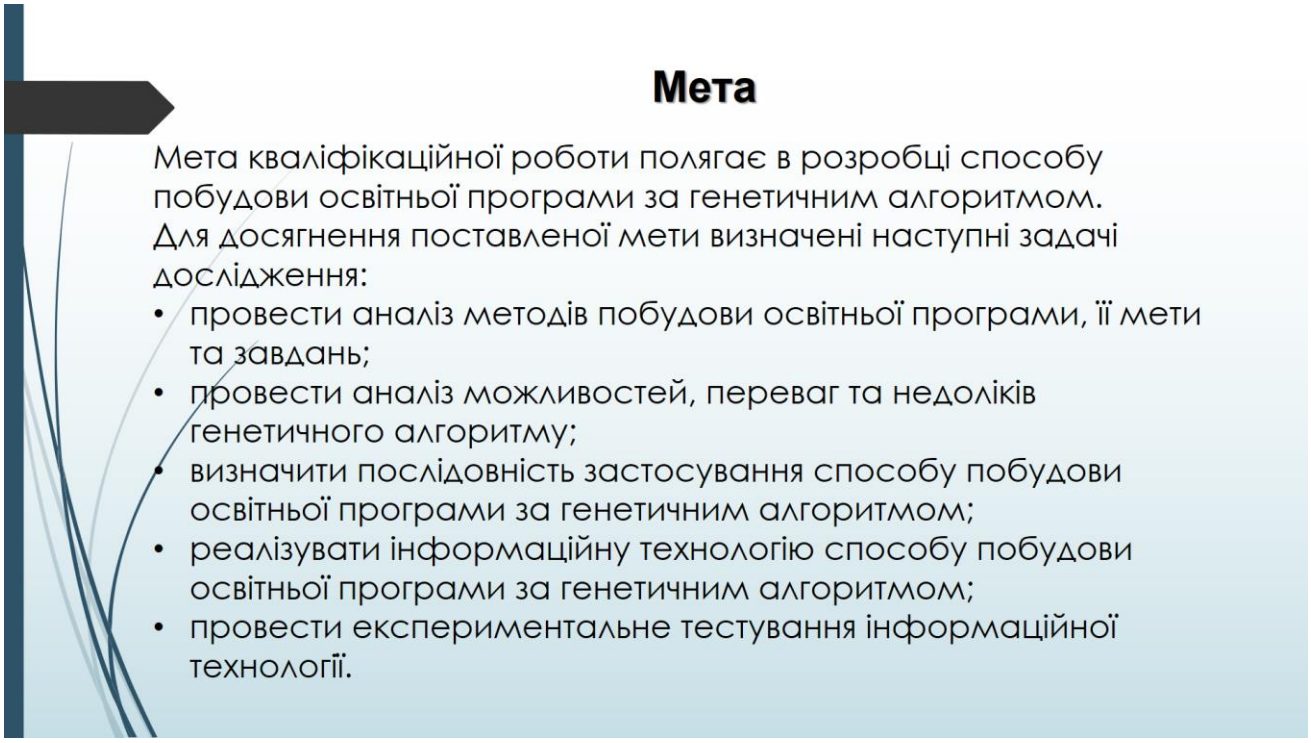
```
<?php
    namespace classes\Objects\Inheritables;

    abstract class abstractObjectFactory {
```

```
        protected $_Id;
        protected $_Name;

        public function __construct($id,$nm){
            $this->_Id = $id;
            $this->_Name = $nm;
        }

        public function getId(){
            return $this->_Id;
        }

        public function getName(){
            return $this->_Name;
        }
    }
?>
```

Додаток Б

Презентаційний матеріал





Етапи побудови освітньої програми в закладах вищої освіти

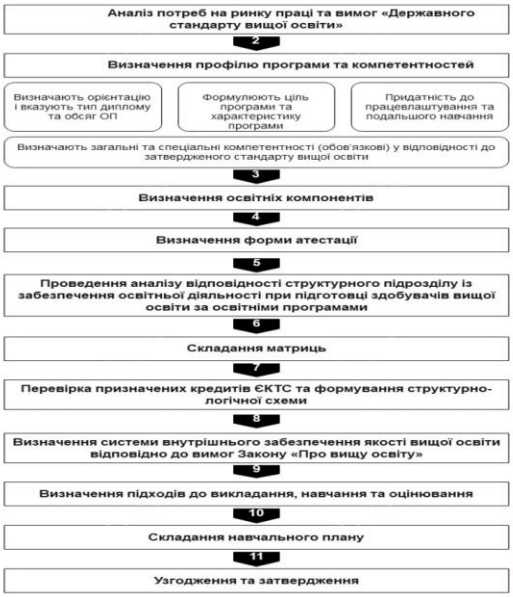
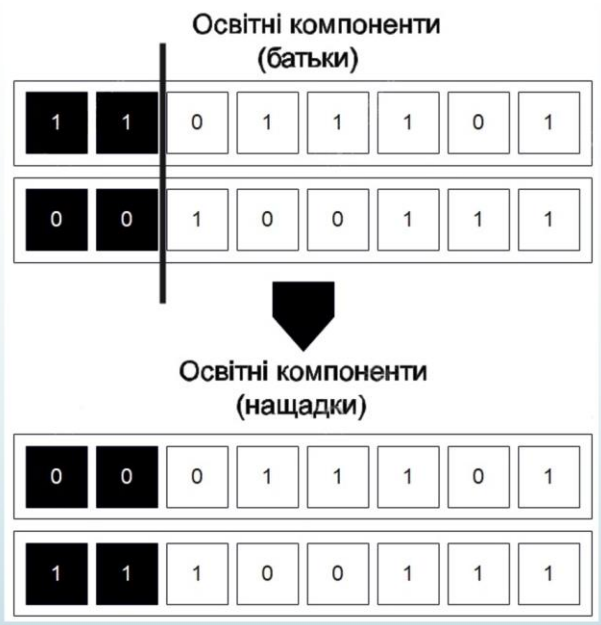


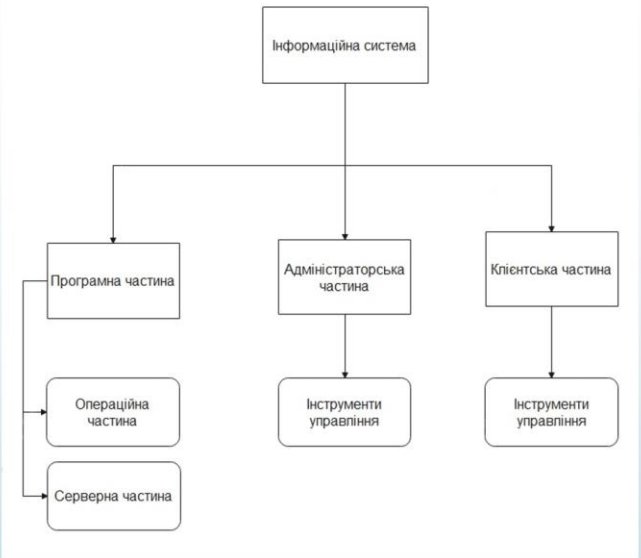
Схема роботи генетичного алгоритму для способу побудови освітньої програми


Освітня програма
1 0 1 1 ... 0 1
1 1 1 1 ... 0 1
0 0 1 1 ... 1 0
...
1 1 1 1 ... 1 0

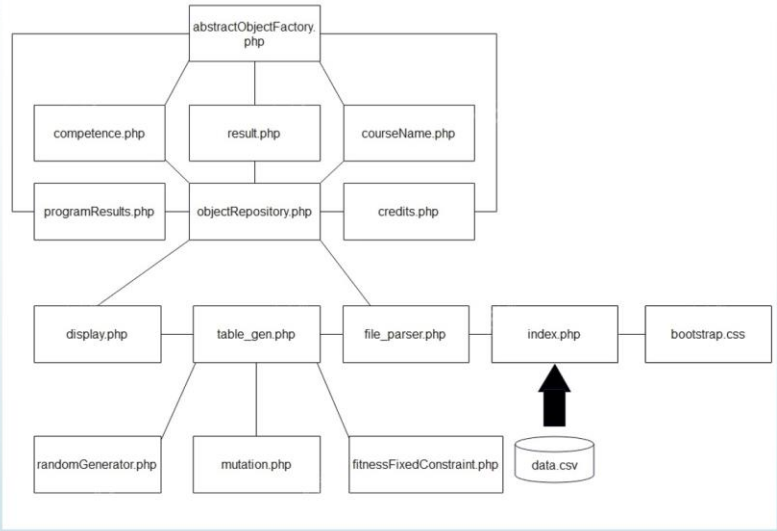

Освітні компоненти
...
ФК1 ФК2 ФК3 ... ФКN
ЗК
ПРН



Складові інформаційної системи



Діаграма компонентів



Дякую за увагу

Sun Jun 04 17:20:07 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 74.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 6%**

ID: 114646 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-04 Автора: А.О. Загоруйко Керівники: О.А. Пасічник Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	63721	957	47110 (74%)	728 (76%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
114504	Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-01 Автора: А.О. Загоруйко Керівники: О.А. Пасічник Консультанти: Опоненти:	46930 (74.0%)	719 (75.0%)

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015414477

Дата перевірки:
04.06.2023 18:13:02 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2023 18:17:35 EEST

ID користувача:
100005671

Назва документа: **КН-19-1 Загоруйко 02**

Кількість сторінок: **64** Кількість слів: **9312** Кількість символів: **74602** Розмір файлу: **1.75 MB** ID файлу: **1015077408**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

6.02%
Схожість

Найбільша схожість: **1.59%** з джерелом з Бібліотеки (ID файлу: **1014976252**)

5.69% Джерела з Інтернету	596	Сторінка 66
2.13% Джерела з Бібліотеки	101	Сторінка 68

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

73.5%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

Немає вилучених Інтернет-джерел

73.5% Вилученого тексту з Бібліотеки	1	Сторінка 68

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	16 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Спосіб побудови освітньої програми за генетичним алгоритмом

Автор: Загоруйко Артур Олегович

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.т.н., доцент Пасічник Олександр Анатолійович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 74 %, схожість виявлена з попереднім варіантом кваліфікаційної роботи бакалавра.

2) за програмою UNICHECK виявлені 6.02%.; Найбільша схожість: 1.59% з джерелом з бібліотеки (ID файлу: 1014976252), яке містить матеріали огляду предметної області; інші схожості є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.

збігів/ідентичності/схожості, складає 74 % і 6.02% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи Олександр ПАСІЧНИК

Гарант ОП Олександр МАЗУРЕЦЬ

Завідувач кафедри КН Олександр БАРМАК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Загоруйка Артура Олеговича*

за темою *Спосіб побудови освітньої програми за генетичним алгоритмом*

1. Актуальність теми

Сучасна Україна швидко та наполегливо інтегрується у європейській та світовий освітній простір. Загальносистемними рисами сучасної вищої освіти є, з одного боку, широка автономія закладів освіти, а, з іншого, зростаючі вимоги здобувачів щодо результатів навчання з точки зору конкурентоспроможності на ринку праці усіх рівнів локалізації. Інтегруючим результатом можливостей закладу вищої освіти в частині надання освітніх послуг задля задоволення очікувань та потреб здобувачів є освітня програма. Ряд обставин, обмежень та обтяжень роблять задачу гаранта та групи забезпечення по формуванню освітньої програми нетривіальним завданням. Сучасні інтелектуальні інформаційні технології надають широкий інструментарій технологій, методів та засобів, зокрема генетичні алгоритми, які можуть бути застосовані для ефективного та оперативного вирішення задачі формування освітньої програми, зроблять роботу гаранта освітньої програми та групи забезпечення об'єктивно ефективною та якісною. Розробка такого способу є актуальною задачею комп'ютерних наук.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є розробка способу побудови освітньої програми за генетичним алгоритмом. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, а саме генетичний алгоритм, що доцільно використовувати при розробці інформаційних технологій. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Загоруйко Артур Олегович виявив себе достатньо кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені етапи дослідження. В процесі написання пояснювальної записки, та при розробці прикладного програмного забезпечення проявив достатні для одержання успішного результату загальні та фахові компетентності та продемонстрував досягнення усіх програмних результати навчання за напрямком «Комп'ютерні науки».

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз актуальності та відомих досліджень в межах обраної теми, поставлені завдання, які у роботі виконані, та розроблено програмне забезпечення для валідації та верифікації запропонованого метода.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі способу побудови освітньої програми за генетичним алгоритмом та його програмна реалізація може бути використана гарантами освітніх програм, членами груп забезпечення та адміністраціями закладів вищої освіти при формування освітніх програм в умовах постійної модернізації та сталого покращення.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «задовільно» .

Керівник _____ *к.т.н., доцент Олександр ПАСІЧНИК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1Загоруйка Артура Олеговича*
за темою: *Спосіб побудови освітньої програми за генетичним алгоритмом*

1. Актуальність обраної теми

Сучасна Україна швидко та наполегливо інтегрується у європейській та світовий освітній простір. Загальносистемними рисами сучасної вищої освіти є, з одного боку, широка автономія закладів освіти, а, з іншого, зростаючі вимоги здобувачів щодо результатів навчання. Результатом можливостей закладу вищої освіти в частині надання освітніх послуг задля задоволення очікувань та потреб здобувачів є освітня програма. Умови праці роблять задачу гаранта та групи забезпечення по формуванню освітньої програми нетривіальним завданням.

2. Повнота розкриття мети та завдань роботи

Мета роботи розкрита повністю. Завдання роботи виконані.

3. Зміст кожного розділу роботи

В першому розділі виконано аналіз сучасних підходів до побудови освітньої програми; аналіз можливостей, переваг, недоліків та області застосування генетичного алгоритму та аналіз існуючих рішень для систем побудови освітньої програми. Визначено мету та завдання роботи.
В другому розділі розроблено спосіб побудови освітньої програми за генетичним алгоритмом із загальною послідовністю побудови освітньої програми та особливостями використання генетичного алгоритму в задачі побудови освітньої програми. Розроблено функціональна структура інформаційної системи. Виконано проектування структури інформаційної системи.
В третьому розділі виконано програмна реалізація способу побудови освітньої програми за генетичним алгоритмом та її експериментальне тестування.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблений у роботі способу побудови освітньої програми за генетичним алгоритмом та його програмна реалізація може бути використана у ЗВО.

5. Якість оформлення кваліфікаційної роботи бакалавра

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

6. Недоліки кваліфікаційної роботи бакалавра

З роботи не зрозуміло, чи враховується послідовність опанування освітніх компонент протягом всього періоду отримання вищої освіти за відповідним освітнім рівнем

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

*Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка **«задовільно»**.*

Рецензент Доц. каф ТМІТ Олег ПІВОВАР