ДОДАТОК А

Висвітлення результатів кваліфікаційної роботи

Міністерство освіти і науки України

NURE

Харківський національний університет радіоелектроніки

ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2023

(Випуск 2)

[електронне видання]

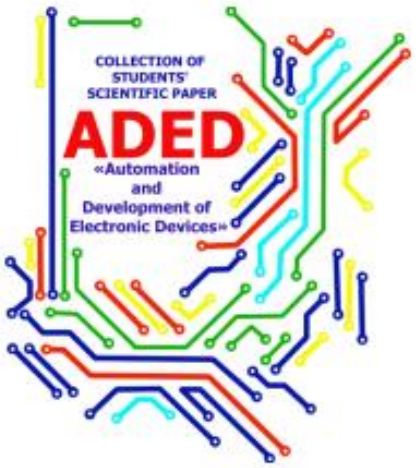
http://nure.ua/department/kafedrakomp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-tamehatroniki-kitam

http://itez.zntu.edu.ua/

http://kafea.kdu.edu.ua

Харків 2023

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)



ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2023

(Випуск 2)

[електронне видання]

Харків 2023

Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2023. – Вип. 2. – 408с.

Collection of Students' Scientific Paper «Automation and Development Of Electronic Devices» ADED-2023 Part 2 (Key infrastructure 2023) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2023. – 408p with.

Рекомендовано рішенням
Науково-технічної ради
Харківського національного
університету радіоелектроніки
протокол №6 від 29.11.2018

Рекомендовано рішенням Вченої ради
факультету Автоматики і комп'ютеризованих технологій
Харківського національного
університету радіоелектроніки
протокол № 4 від 30.11.2023

Збірник містить наукові статті здобувачів першого (бакалаврського), другого (магістерського) рівнів вищої освіти кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки, кафедри Інформаційних технологій електронних засобів (ІТЕД) Запорізького національного технічного університету та кафедри Електронних апаратів (ЕА) Кременчуцького національного університету ім. М. Остроградського які навчаються за спеціальностями: 151 Автоматизація та комп'ютерно-інтегровані технології, 172 Телекомунікації та радіотехніка, 171 Електроніка та 163 Біомедична інженерія. Статті надані в авторській редакції.

ЗМІСТ

УДК 37.014.00

АЛГОРИТМИ СТВОРЕННЯ ТА ОПТИМІЗАЦІЇ РОЗКЛАДУ ДЛЯ ЗАГАЛЬНООСВІТНІХ НАВЧАЛЬНИХ ЗАКЛАДІВ

М.А. Пісклов
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: mykola.pisklov@nure.ua

Анотація: Стаття присвячена дослідженню та застосуванню алгоритмів оптимізації розкладів для загальноосвітніх навчальних закладів. Існує багато алгоритмів, які можна використати для створення та оптимізації розкладу, наприклад: методи чисельного моделювання, локальний пошук, метод обмежень тощо. Результатами досліджень є визначені обмеження, набір змінних та опис цільової функції.

Ключові слова: оптимізація, алгоритм, метод обмежень, цільова функція, обмеження, змінні.

ALGORITHMS OF CREATION AND OPTIMIZATION OF SCHOOL-TIMETABLING

M. Pisklov
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: mykola.pisklov@nure.ua

Abstract: This article explores various algorithms for schedule optimization in schools. Methods such as simulated annealing, local search, and constraint programming are considered. The research results focus on identifying constraints, defining variables, and formulating the objective function.

Keywords: optimization, algorithm, constraint programming, total function, constraints, variables.

АКТУАЛЬНІСТЬ РОБОТИ. Використання комп'ютерних інформаційних технологій дозволяє практично повністю автоматизувати робочі процеси виробництва, організацій і підприємств [1-5]. Зазвичай інформаційні системи використовуються для обліку та зберігання важливих даних для організацій. Комп'ютерні технології також широко застосовуються у сфері освіти, зокрема у загальноосвітніх навчальних закладах. Особливо проблемним аспектом для загальноосвітніх навчальних закладів є складання та оптимізація розкладу. Цей процес є досить складним, якщо виконувати його вручну, та для більшості навчальних закладів є актуальною автоматизація цього завдання.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. У шкільному розкладі необхідно скласти графік проведення заданих зустрічей між учнями та вчителями так, щоб отримані розклади були прийнятними та відповідали вимогам усіх учасників навчального процесу [6]. Крім того, освітні системи відрізняються від країни до країни, і всередині конкретної освітньої системи освіта відрізняється на різних рівнях. У зв'язку з цим виникли різні погляди на характеризацію категорій шкільного розкладу, але в рамках цієї роботи буде розглянута система загальноосвітніх навчальних закладів України.

У типовому загальноосвітньому навчальному закладі складання розкладу базується на класах, які формуються шляхом групування учнів за спільною програмою (рис. 1).

Розклади не мають містити конфліктів, мають бути компактними та задовольняти різноманітні обмеження на розподіл і послідовність уроків. В абсолютній більшості закладів

України аудиторії для проведення занять розташовані в одному місці, тому час подорожей учнів під час складання розкладу враховуватись не буде.

	Mon	Tue	Wed	Thu	Fri
1					
2					
3					
4					
5					
6					

Рисунок 1 – Приклад сітки розкладу класу загальноосвітнього навчального закладу

Тож процес складання розкладу передбачає планування та розподіл ресурсів з метою створення розкладу для кожного класу. Як додатковий результат, отримуються розклади для кожної аудиторії та вчителя. Простір рішень лімітується наступними обмеженнями (рис. 2):

1. Обмеження доступу до ресурсів: ресурси (учні, вчителі та аудиторії) не мають бути подвійно забронюваними.
2. Розподіл класу на підгрупи.
3. Об'єм ресурсів: кількість аудиторій і вчителів, адже один вчитель може викладати кілька предметів.
4. Обмеження щоденного робочого навантаження: цей тип обмеження стосується учнів, вчителів, пар учень-вчитель, робіт і пар учнів та наборів робіт. Для кімнат таке обмеження не застосовується. Робоче навантаження вимірюється у часових слотах.
5. Обмеження кількості робочих днів.
6. Обмеження на наявність «вікон»: розклади можуть містити часові слоти, які не зайняті жодним заняттям. Такий часовий слот у середині дня називається вільним, прогалиною або «вікном», якщо перед і після нього заплановані зустрічі. Кількість вільних часових слотів, які виникають у розкладі однієї особи, називається загальним вільним часом цієї особи.

Рисунок 2 – Основа обмежень складання розкладу

Існує декілька алгоритмів, які можна використати для виконання задачі створення розкладу. Деякі з них включають методи оптимізації розподілу уроків, вчителів, предметів та аудиторій з урахуванням різних обмежень. Ось кілька прикладів алгоритмів оптимізації для шкільного розкладу:

1. Генетичні алгоритми (Genetic Algorithms) – генетичні алгоритми можуть використовуватися для пошуку оптимальних розкладів, застосовуючи принципи природного відбору та еволюції. Вони можуть ефективно працювати з великою кількістю змінних та обмежень.
2. Методи тваринного світу (Ant Colony Optimization) – цей алгоритм базується на поведінці мурах у процесі пошуку оптимальних маршрутів. Він може бути застосований до завдання побудови розкладу для оптимізації процесу.
3. Методи числового моделювання (Simulated Annealing) – можуть використовуватися для оптимізації розкладу шляхом імітації процесу відпалювання металу. Він досліджує простір рішень для пошуку оптимального розкладу.
4. Локальний пошук (Local Search) – локальні методи пошуку можуть бути використані для покращення поточного розкладу, вносячи невеликі зміни та оцінюючи їх вплив на цільову функцію.
5. Метод обмежень (Constraint Programming) – використовується для вирішення задач з обмеженнями. Він може бути ефективним для оптимізації розкладу, враховуючи різні обмеження, такі як доступність вчителів, аудиторій та інші фактори.
6. Методи штучного інтелекту (AI-Based Approaches) – штучний інтелект, включаючи методи машинного навчання, може використовуватися для оптимізації шкільних розкладів, враховуючи динамічні та складні фактори.

Ці методи можуть застосовуватися як окремо, так і в комбінації, залежно від конкретних вимог та обмежень задачі оптимізації шкільного розкладу.

Особливо слід звернути увагу на метод обмежень, оскільки його використання дозволяє отримати певні переваги під час складання й оптимізації розкладу загальноосвітнього навчального закладу.

По-перше, метод обмежень дозволяє гнучко визначати різноманітні обмеження, які характеризують задачу. Користувач легко включати обмеження, такі як доступність вчителів, аудиторій, обмеження на кількість уроків для учнів тощо.

По-друге, виразність обмежень робить легшим формалізацію вимог та умов користувача. Це особливо корисно у складних задачах оптимізації, де важко визначити точну математичну функцію цільового впливу.

По-третє, метод обмежень часто демонструє ефективність у вирішенні складних комбінаторних проблем, таких як задачі оптимізації розкладу, оскільки його структура дозволяє ефективно розглядати велику кількість можливих комбінацій.

Також перевагами методу обмежень є можливість врахування різноманітних умов; ітеративність (метод обмежень дозволяє ітеративно додавати або змінювати обмеження у процесі розробки розкладу, що може бути корисно в умовах непередбачуваних змін у вимогах чи умовах); зручність у використанні інструментів тощо. Саме тому метод обмежень може бути ефективним для складання розкладів, де важко визначити функцію цільового впливу та важко передбачити всі можливі умови.

Метод обмежень (Constraint Programming, CP) є парадигмою програмування та методом розв'язання задач оптимізації, який базується на явному визначенні обмежень, яким мають відповідати змінні рішення. Тобто користувач декларативно визначає обмеження на припустимі рішення для набору змінних прийняття рішень [7]. Цей метод широко

використовується для вирішення різних задач, включаючи задачі планування та оптимізації розкладів, таких як шкільні розклади.

Далі розглянемо основні принципи методу обмежень. Змінні рішення: задача формулюється через визначення змінних рішення, які представляють собою параметри, що підлягають оптимізації. У випадку шкільного розкладу ці змінні можуть включати час проведення уроків, вчителів, класи, аудиторії тощо.

Визначаються обмеження, яким мають задовольняти змінні рішення. Ці обмеження відображають умови, які мають бути виконані для того, щоб рішення було прийнято. Наприклад, обмеження може визначати, що два уроки в одному класі не можуть перетинатися у часі.

Метод обмежень включає цільову функцію, яку слід оптимізувати. Ця функція виражає важливість або прийнятність конкретних рішень. У випадку оптимізації шкільного розкладу цільова функція може, наприклад, враховувати мінімізацію перерв між уроками або позначати важливість відповідності певним умовам.

Для розв'язання задачі використовується інфраструктура обмежень, яка забезпечує засоби для визначення та вирішення обмежень. Це може включати в себе бібліотеки чи інші інструменти, які дозволяють легко висловлювати та розв'язувати обмеження.

У контексті шкільного розкладу, метод обмежень може бути використаний для вирішення задачі, враховуючи різноманітні обмеження, такі як доступність учителів, аудиторій, максимальна кількість уроків на день для учнів, врахування перерв між уроками і так далі. Під час визначення цих обмежень і створенні цільової функції можна використовувати метод обмежень для знаходження оптимального розкладу, який задовольняє визначеним умовам.

Щоб полегшити застосування технології обмежень до проблеми складання розкладу для загальноосвітнього навчального закладу, нам потрібно перетворити високорівневий опис проблеми (у термінах вчителів, учнів та аудиторій) на так звану задачу задоволення обмежень (Constraint satisfaction problem, CSP). Задача задоволення обмежень формулюється у термінах обмежень для змінних з областями значень, і для знаходження рішення потрібно вибрати значення з області значень кожної змінної так, щоб отримане значення задовольняло всі обмеження.

Зазвичай спосіб опису CSP полягає у наданні трійки (X, δ, C) із набором змінних X, набором обмежень C для X та цільовою функцією δ на X, яка пов'язує кожну змінну із її областю значень. Інший спосіб вказати CSP – це надання гіперграфа змінних у вигляді вузлів і обмежень у вигляді гіпердуг. Такий граф називається мережею обмежень.

Обмеження вважається зайвим відносно визначеної проблеми, якщо кожне рішення проблеми задовольняє обмеження, хоча проблема не визначає це обмеження явно.

У контексті оптимізаційних задач, цільова функція визначає той критерій, який необхідно оптимізувати чи мінімізувати. Вона відображає нашу основну мету.

Обмеження, з іншого боку, визначають умови, яким має відповідати оптимальне рішення. Ці обмеження обмежують простір можливих рішень і враховують фактори, які є необхідність врахувати у процесі оптимізації.

Під час оптимізації розкладу із використанням методу обмежень, шукаємо такі значення змінних, які задовольняють всі обмеження та мінімізують (або максимізують, залежно від завдання) значення цільової функції.

Головним критерієм оптимізації для цільової функції було обрано зведення до мінімуму кількості «вікон» між заняттями класів, а у випадку початкового та середнього рівня навчання взагалі відсутність так званих «простоїв» між заняттями чи на початку дня (наприклад, початок занять з другого або нульового уроку, до чого часто прибігають у складанні розкладу старших класів для запобігання конфлікту доступу до ресурсів: вчителів чи аудиторій). Треба

зазначити, що у випадку розглядання розкладу вчителя можна не приділяти багато уваги цьому аспекту, оскільки під час «вікна» вчитель може займатися перевіркою контрольних, самостійних чи домашніх робіт, а також підготовкою до майбутніх занять.

ВИСНОВКИ. У ході виконання дослідження проблеми складання та оптимізації розкладу занять для загальноосвітніх навчальних закладів, яке було представлено у цій статті, було розглянуто багато алгоритмів і методів складання розкладу. Було вирішено зупинитись на методі обмежень, оскільки він дозволить у повному обсязі охопити обмеження, які є специфічними для обраної предметної області, а саме: обмеження доступу до ресурсів, розподіл класу на підгрупи, обсяг ресурсів, обмеження щоденного робочого навантаження, обмеження кількості робочих днів, обмеження на наявність «вікон». Головним критерієм для цільової функції обрано мінімізацію наявності «вікон» у розкладі.

Також хотілося б розглянути перспективи розвитку проведеного дослідження. Визначені у ході дослідження обмеження, змінні та критерії для цільової функції можуть бути використані для створення інформаційної системи з інтерфейсом користувача для складання та оптимізації розкладу. Одним із варіантів може виступити платформа .NET та технологія WPF (Windows Presentation Foundation).

Платформа .NET Framework – це технологія, яка підтримує створення та виконання веб-сервісів і додатків Windows [8]. Була обрана саме операційна система Windows, оскільки більшість користувачів, які працюють у закладах загальноосвітніх закладах, використовують саме цю операційну систему. Саме тому технологія WPF підійде для розв'язання задачі створення застосунку для роботи з розкладом. WPF – це незалежна від роздільної здатності інтерфейсна платформа, що використовує векторний механізм рендеринга, який може скористатися перевагами сучасного графічного обладнання. WPF надає повний набір функцій розробки додатків, які включають мову XAML, елементи керування, прив'язку даних, макет, 2D і 3D графіку, анімацію, стилі, шаблони тощо [9].

ЛІТЕРАТУРА

1. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 : монографія / І. Ш. Невлюдов, В. В. Євсєєв, А. О. Андрусевич, С. С. Максимова ; – Oktan Print Prague. 2023. – 321 с.

2. Viktoriia Bortnikova, Vladyslav Yevsieiev, Iryna Botsman, Igor Nevliudov, Kostiantyn Kolesnyk, Nazariy Jaworski. Queries classification using machine learning for implementation in intelligent manufacturing // Chapter 6 in Monograph «Methods and tools in CAD – selected issues». – Białystok (Poland): Publishing House of Bialystok University of Technology. – 2021. – PP. 63-74.

3. V. Bortnikova, I. Nevliudov, I. Botsman and O. Chala, "Search Query Classification Using Machine Learning for Information Retrieval Systems in Intelligent Manufacturing," in CEUR Workshop Proceedings of the 15th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer (ICTERI'2019), June 12-15, 2019, Kherson, Ukraine.

4. Боцман І. В. Автоматизація процесу реконструювання параметрів технологічного процесу виготовлення гнучких структур // Матеріали конференції «Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку», Черкаси, 12-18 березня 2018 р. – С. 73-75.

5. Жарикова И. В. Системологический подход при исследовании параметров РЭС / И. В. Жарикова, В. В. Невлюдова // Технология приборостроения. –2014. –No2. –С. 40-43.

6. Michael Marte. Models and Algorithms for School Timetabling – A Constraint-Programming Approach, 2002.

7. F. Rossi, P. van Beek and T. Walsh. Handbook of Constraint Programming, 2006.
8. Overview of .NET framework. Microsoft : веб-сайт. URL: https://learn.microsoft. com/en-us/dotnet/framework/get-started/overview.
9. Desktop Guide (WPF .NET). Microsoft : веб-сайт. URL: https://learn.microsoft.com /en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0.

Науковий керівник: Жарікова Ірина Володимирівна, доцент, кандидат технічних наук, доцент кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки.

ДОДАТОК Б

Лістинг коду програми

```
namespace Schedule.ViewModels
{
    public class ScheduleViewModel : ViewModelBase
    {

        public ObservableCollection<ScheduleJoin>
SchedulesInfo { get; set; } = new
ObservableCollection<ScheduleJoin>();
        private List<Teacher> Teachers { get; set; } = new
List<Teacher>();
        //Collections for schedule
        public List<int> DaysIds { get; set; } = new
List<int>();
        public List<int> TimeIds { get; set; } = new
List<int>();
        private Dictionary<(int day, int time, int clas),
bool> ClassSlotsAvailability { get; set; }
            = new Dictionary<(int day, int time, int clas),
bool>();
        private Dictionary<(int day, int time, int teacher),
bool> TeacherSlotsAvailability { get; set; }
            = new Dictionary<(int day, int time, int teacher),
bool>();
        private List<TakenSlotInfo> TakenSlots { get; set; } =
new List<TakenSlotInfo>();
        private List<ScheduleSubjectToClass>
YearSubjectsToClass = new List<ScheduleSubjectToClass>();
        public List<SlotInfo> SlotsForTheView { get; set; } =
new List<SlotInfo>();
        private List<AvgHoursForClass> AvgHoursForClass { get;
set; } = new List<AvgHoursForClass>();
        public int ClassesCount { get; set; }
        private readonly IScheduleDataProvider _dataProvider;
        private readonly ISubjectToClassDataProvider
_subjectToClassDataProvider;
        private readonly ISlotDataProvider _slotDataProvider;
        private readonly ITeacherDataProvider
_teacherDataProvider;
```

```
        public DelegateCommand
OpenWindowForAssigningSubjectsCommand { get; }
        public DelegateCommand CreateScheduleCommand { get; }

        public ScheduleViewModel(IScheduleDataProvider
scheduleDataProvider, ISubjectToClassDataProvider scdp,
ISlotDataProvider slotDataProvider,
            ITeacherDataProvider tdp)
        {
            _dataProvider = scheduleDataProvider;
            _subjectToClassDataProvider = scdp;
            _slotDataProvider = slotDataProvider;
            _teacherDataProvider = tdp;
            OpenWindowForAssigningSubjectsCommand = new
DelegateCommand(OpenWindowForAssigningSubjects);
            CreateScheduleCommand = new
DelegateCommand(CreateSchedule);
        }
        public async override Task LoadAsync()
        {
            var schedules = await
_dataProvider.GetSchedulesForAYearAsync();
            if(schedules is not null)
            {
                foreach(var schedule in schedules)
                {
                    SchedulesInfo.Add(schedule);
                }
            }
            var teachers = await
_teacherDataProvider.GetAllTeachersAsync();
            if(teachers is not null)
            {
                foreach (var teacher in teachers)
                {
                    Teachers.Add(teacher);
                }
            }
            ClassesCount = SchedulesInfo.Count;
            SetDaysIds();
            SetTimeIds();
            SetEmptySlots();
        }
```

```
        public async void
OpenWindowForAssigningSubjects(object? obj)
        {
            var info = obj as ScheduleJoin;
            if (info is not null)
            {
                SubjectToClassView newWindow = new
SubjectToClassView();
                var viewModel = new
SubjectToClassViewModel(info, new
TeacherSubjectDataProvider(), new
SubjectToClassDataProvider());
                newWindow.DataContext = viewModel;
                await viewModel.LoadAsync();
                newWindow.Show();

            }
        }
        private async void SetDaysIds()
        {
            DaysIds = await _dataProvider.GetDaysId();
        }
        private async void SetTimeIds()
        {
            TimeIds = await _dataProvider.GetTimeId();
        }
        private void SetEmptySlots()
        {
            if(DaysIds.Any() && TimeIds.Any() &&
SchedulesInfo.Any())
            {
                foreach (var schedule in SchedulesInfo)
                {
                    foreach (var day in DaysIds)
                    {
                        foreach (var time in TimeIds)
                        {
                            ClassSlotsAvailability.Add((day,
time, schedule.ScheduleId), true);
                        }
                    }
                }
                foreach (var teacher in Teachers)
```

```
                {
                    foreach (var day in DaysIds)
                    {
                        foreach (var time in TimeIds)
                        {
                            TeacherSlotsAvailability.Add((day,
time, teacher.Id), true);
                        }
                    }
                }

            }
        }
        public async void CreateSchedule(object? obj)
        {
            YearSubjectsToClass = await
_subjectToClassDataProvider.GetAllSubjectToClassForAYear();
            bool isScheduleCreationSuccessFull = true;
            var list = YearSubjectsToClass.ToList();
            CalculateAvgHoursForClasses(list);
            var lessons = CreateLessons(list);
            if (IsSolutionPossible(lessons.Count))
            {
                var difficulLessons = lessons.Where(x =>
x.DifficultCoefficient >= 1.7 && x.Hour != 0.5).ToList();
                var otherLessons = lessons.Where(x =>
x.DifficultCoefficient < 1.7 && x.Hour != 0.5).ToList();
                var partialLessons = lessons.Where(x => x.Hour
== 0.5).ToList();
                var restOfTHeLessons = new List<Lesson>();

                var random = new Random();
                difficulLessons = difficulLessons.OrderBy(x =>
random.Next()).ToList();
                //DIFFICULT LESSONS
                int i = 0;
                while (i < difficulLessons.Count)
                {
                    int startingI = i;
                    var slotInfo = difficulLessons[i];
                    var availableSlots =
ClassSlotsAvailability.Where(x => x.Value == true &&
x.Key.clas == slotInfo.FkSchedule
```

```
                        && x.Key.day >= 2 && x.Key.day <=
3).ToList();
                    var avgHours = AvgHoursForClass.First(x =>
x.FkSchedule == slotInfo.FkSchedule);
                    foreach (var slot in availableSlots)
                    {

                        if
(IsClassHaveTwoSameSubjectsInDay(slotInfo, slot.Key))
                        {
                            continue;
                        }
                        if (IsTeacherFree(slotInfo, slot.Key)
&& IsPreviousSlotEmpty(slot.Key))
                        {
                            if
(IsDayHoursNormal(avgHours.AvgHours, slot.Key))
                            {

ClassSlotsAvailability[slot.Key] = false;
                                var teacherSlotKey =
(slot.Key.day, slot.Key.time, slotInfo.FkTeacher);

TeacherSlotsAvailability[teacherSlotKey] = false;
                                AddSlotToTakenSlots(slotInfo,
(slot.Key.day, slot.Key.time));
                                i++;
                                break;
                            }
                        }

                    }
                    if (i == startingI)
                    {
                        otherLessons.Add(slotInfo);
                        i++;
                    }
                }
                i = 0;
                //OTHER LESSONS
                otherLessons = otherLessons.OrderBy(x =>
random.Next()).ToList();
                i = 0;
```

```
                while (i < otherLessons.Count)
                {
                    int startingI = i;
                    var slotInfo = otherLessons[i];
                    var availableSlots =
ClassSlotsAvailability.Where(x => x.Value == true
                        && x.Key.clas ==
slotInfo.FkSchedule).ToList();
                    var avgHours = AvgHoursForClass.First(x =>
x.FkSchedule == slotInfo.FkSchedule);
                    var schedule = SchedulesInfo.First(x =>
x.ScheduleId == slotInfo.FkSchedule);
                    foreach (var slot in availableSlots)
                    {

                        if
(IsClassHaveTwoSameSubjectsInDay(slotInfo, slot.Key) ||
avgHours.MaxLessonADay <= slot.Key.time || slot.Key.time > 4)
                        {
                            continue;
                        }
                        if (IsTeacherFree(slotInfo, slot.Key)
&& IsPreviousSlotEmpty(slot.Key))
                        {
                            if
(IsDayHoursNormal(avgHours.AvgHours, slot.Key))
                            {

ClassSlotsAvailability[slot.Key] = false;
                                var teacherSlotKey =
(slot.Key.day, slot.Key.time, slotInfo.FkTeacher);

TeacherSlotsAvailability[teacherSlotKey] = false;
                                AddSlotToTakenSlots(slotInfo,
(slot.Key.day, slot.Key.time));
                                i++;
                                break;
                            }
                        }

                    }
                    if (i == startingI)
                    {
```

```
                        restOfTHeLessons.Add(slotInfo);
                        i++;
                    }
                }
                i = 0;
                var finalLessons = new List<Lesson>();
                while (i < restOfTHeLessons.Count)
                {
                    int startingI = i;
                    var slotInfo = restOfTHeLessons[i];
                    var availableSlots =
ClassSlotsAvailability.Where(x => x.Value == true
                    && x.Key.clas ==
slotInfo.FkSchedule).ToList();
                    var avgHours = AvgHoursForClass.First(x =>
x.FkSchedule == slotInfo.FkSchedule);
                    foreach (var slot in availableSlots)
                    {

                        if
(IsClassHaveTwoSameSubjectsInDay(slotInfo, slot.Key) ||
avgHours.MaxLessonADay <= slot.Key.time)
                        {
                            continue;
                        }
                        if (IsTeacherFree(slotInfo, slot.Key)
&& IsPreviousSlotEmpty(slot.Key))
                        {

                            ClassSlotsAvailability[slot.Key] =
false;
                            var teacherSlotKey =
(slot.Key.day, slot.Key.time, slotInfo.FkTeacher);

TeacherSlotsAvailability[teacherSlotKey] = false;
                            AddSlotToTakenSlots(slotInfo,
(slot.Key.day, slot.Key.time));
                            i++;
                            break;

                        }

                    }
```

```
                    if (i == startingI)
                    {
                        finalLessons.Add(slotInfo);
                        i++;
                    }
                }
                i = 0;
                //FINAL LOOP
                while (i < finalLessons.Count)
                {
                    int startingI = i;
                    var slotInfo = finalLessons[i];
                    var availableSlots =
ClassSlotsAvailability.Where(x => x.Value == true
                        && x.Key.clas ==
slotInfo.FkSchedule).ToList();
                    var avgHours = AvgHoursForClass.First(x =>
x.FkSchedule == slotInfo.FkSchedule);
                    foreach (var slot in availableSlots)
                    {
                        if
(IsClassHaveTwoSameSubjectsInDay(slotInfo, slot.Key))
                        {
                            continue;
                        }
                        if (IsTeacherFree(slotInfo, slot.Key)
&& IsPreviousSlotEmpty(slot.Key))
                        {
                            ClassSlotsAvailability[slot.Key] =
false;
                            var teacherSlotKey =
(slot.Key.day, slot.Key.time, slotInfo.FkTeacher);

TeacherSlotsAvailability[teacherSlotKey] = false;
                            AddSlotToTakenSlots(slotInfo,
(slot.Key.day, slot.Key.time));
                            i++;
                            break;

                        }

                    }
                    if (i == startingI)
```

```
                    {
                        i++;
                        isScheduleCreationSuccessFull = false;
                        break;
                    }
                }
                //PartialLessons
                i = 0;
                while (i < partialLessons.Count)
                {
                    int startingI = i;
                    var slotInfo = partialLessons[i];
                    var availableSlots =
ClassSlotsAvailability.Where(x => x.Value == true
                        && x.Key.clas ==
slotInfo.FkSchedule).ToList();
                    foreach (var slot in availableSlots)
                    {
                        if (IsTeacherFree(slotInfo, slot.Key)
&& IsPreviousSlotEmpty(slot.Key) &&
IsThereSpaceInPartialSlot(slot.Key))
                        {
                            ClassSlotsAvailability[slot.Key] =
false;
                            var teacherSlotKey =
(slot.Key.day, slot.Key.time, slotInfo.FkTeacher);

TeacherSlotsAvailability[teacherSlotKey] = false;
                            AddSlotToTakenSlots(slotInfo,
(slot.Key.day, slot.Key.time));
                            i++;
                            break;

                        }
                    }
                    if (i == startingI)
                    {
                        i++;
                        isScheduleCreationSuccessFull = false;
                        break;
                    }
                }
                if (isScheduleCreationSuccessFull)
```

```
                {
                    OnSchedulingCompleted();
                }
                else
                {
                    TakenSlots.Clear();
                    TeacherSlotsAvailability.Clear();
                    ClassSlotsAvailability.Clear();
                    SetEmptySlots();
                    CreateSchedule(obj);
                }
            }
            else
            {
                MessageBox.Show("Рішення не має. Забагато
уроків");
            }

        }

        private bool IsClassHaveTwoSameSubjectsInDay(Lesson
sc, (int day, int time, int clas) clasKey)
        {
            var subjectCount =
TakenSlots.Where(x=>x.SubjectToClassId == sc.SubjectToClassId
&& x.DayId == clasKey.day).Count();
            if(subjectCount < 2)
            {
                return false;
            }
            else
            {
                return true;
            }
        }

        private void
CalculateAvgHoursForClasses(List<ScheduleSubjectToClass>
subjectsToClass)
        {
            foreach (var schedule in SchedulesInfo)
            {
```

```
                double sumOfHours = subjectsToClass.Where(x =>
x.FkSchedule == schedule.ScheduleId).Sum(x => x.Hours);
                double hoursForClass;
                double divideRest = 0;
                if (sumOfHours <= 5)
                {
                    hoursForClass = sumOfHours;
                }
                else
                {
                    hoursForClass = Math.Round(sumOfHours /
5);
                    divideRest = sumOfHours % 5;
                }
                AvgHoursForClass avg = new AvgHoursForClass
                {
                    FkSchedule = schedule.ScheduleId,
                    AvgHours = hoursForClass,
                    DivideRest = divideRest,
                    MaxLessonADay = (int)hoursForClass + 1
                };
                AvgHoursForClass.Add(avg);
            }
        }
        private bool IsDayHoursNormal(double avgHours, (int,
int, int) clasKey)
        {
            var classTakenSkots = TakenSlots.Where(x=>x.DayId
== clasKey.Item1
                && x.ScheduleId == clasKey.Item3).ToList();
            var countHoursInDay = classTakenSkots.Count();
            if (countHoursInDay < avgHours)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        private bool IsThereSpaceInPartialSlot((int day, int
time, int clas) classKey)
        {
```

```
            var takenPartialSlots = TakenSlots.Where(x =>
x.DayId == classKey.day && x.TimeId == classKey.time
                && x.ScheduleId == classKey.clas &&
x.IsPartial == 1);
            if(takenPartialSlots.Count() > 1)
            {
                return false;
            }
            else
            {
                return true;
            }

        }
        private bool IsTeacherFree(Lesson lesson, (int, int,
int) clasKey)
        {
            var teacherKey = (clasKey.Item1, clasKey.Item2,
lesson.FkTeacher);
            bool isteacherFree =
TeacherSlotsAvailability[teacherKey];
            return isteacherFree;
        }
        private bool IsPreviousSlotEmpty((int day, int time,
int clas) clasKey)
        {
            if (clasKey.time == 1)
                return true;
            var previousLesson =
ClassSlotsAvailability.Where(x => x.Key.day == clasKey.day &&
x.Key.clas == clasKey.clas
                && x.Value == false).OrderByDescending(x =>
x.Key.time).FirstOrDefault();
            int sub = clasKey.time - previousLesson.Key.time;
            if(sub > 1)
            {
                return false;
            }
            else
            {
                return true;
            }
        }
```

```
        public async void OnSchedulingCompleted()
        {
            //TESTING
            //WINDOWS
            int windowCount = 0;
            var tempSlots =
ClassSlotsAvailability.Where(x=>x.Value == false).ToList();
            for(int i = 0; i < tempSlots.Count-1; i++)
            {
                if ((tempSlots[i].Key.time -
tempSlots[i+1].Key.time > 1)
                    && (tempSlots[i].Key.day == tempSlots[i +
1].Key.day)
                    && (tempSlots[i].Key.clas == tempSlots[i +
1].Key.clas))
                {
                    windowCount++;
                }
            }
            Console.WriteLine($"Window Count: {windowCount}");
            foreach(var clas in SchedulesInfo)
            {
                foreach(var day in DaysIds)
                {
                    var diffCount =
TakenSlots.Where(x=>x.ScheduleId == clas.ScheduleId && x.DayId
== day).Sum(x=>x.Diff);
                    Console.WriteLine($"Class:
{clas.ClassTitle}, Day: {day}, Diff: {diffCount}");
                }
                Console.WriteLine();
            }
            foreach (var slot in TakenSlots)
            {
                await _slotDataProvider.InsertSlotAsync(slot);
            }
            var slotInfos = await
_slotDataProvider.GetSlotsInfoForScheduleAsync();
            foreach(var slot in slotInfos)
            {
                SlotsForTheView.Add(slot);
            }
            Messanger.Instance.ScheduleDoneSend();
```

```
        }
        private List<Lesson>
CreateLessons(List<ScheduleSubjectToClass> list)
        {
            var lessons = new List<Lesson>();
            foreach (var subject in list)
            {
                while (subject.Hours >= 0.5)
                {
                    if (subject.Hours == 0.5)
                    {
                        var lesson = new Lesson()
                        {
                            SubjectToClassId = subject.Id,
                            FkTs = subject.FkTs,
                            FkSchedule = subject.FkSchedule,
                            FkTeacher = subject.FkTeacher,
                            Hour = 0.5,
                            DifficultCoefficient =
subject.DifficultCoefficient
                        };
                        lessons.Add(lesson);
                        subject.Hours -= 0.5;
                    }
                    else
                    {
                        var lesson = new Lesson()
                        {
                            SubjectToClassId = subject.Id,
                            FkTs = subject.FkTs,
                            FkSchedule = subject.FkSchedule,
                            FkTeacher = subject.FkTeacher,
                            Hour = 1,
                            DifficultCoefficient =
subject.DifficultCoefficient
                        };
                        lessons.Add(lesson);
                        subject.Hours -= 1;
                    }

                }
            }
            return lessons;
```

```
        }
        private void AddSlotToTakenSlots(Lesson slotInfo,
(int, int) dayTime)
        {
            TakenSlotInfo newSlot = new TakenSlotInfo()
            {
                SubjectToClassId = slotInfo.SubjectToClassId,
                ScheduleId = slotInfo.FkSchedule,
                TeacherSubjectId = slotInfo.FkTs,
                TeacherId = slotInfo.FkTeacher,
                ClassroomId = slotInfo.FkClassroom,
                DayId = dayTime.Item1,
                TimeId = dayTime.Item2,
                IsPartial = slotInfo.Hour == 0.5 ? 1 : 0,
                Diff = slotInfo.DifficultCoefficient
            };
            TakenSlots.Add(newSlot);
        }
        private bool IsSolutionPossible(int lessonsCount)
        {
            if (lessonsCount <= ClassSlotsAvailability.Count)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

ДОДАТОК В

Демонстраційний матеріал

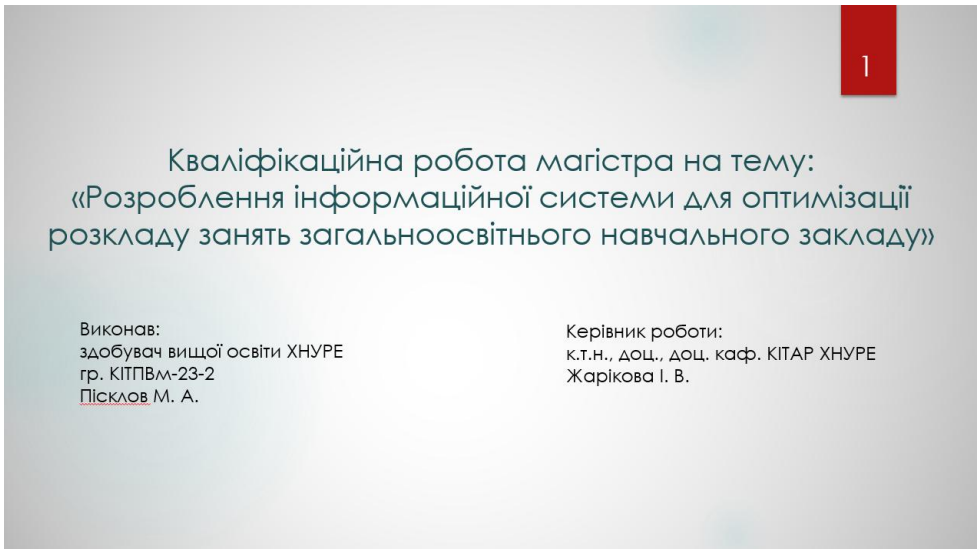


Рисунок В.1 – Слайд №1

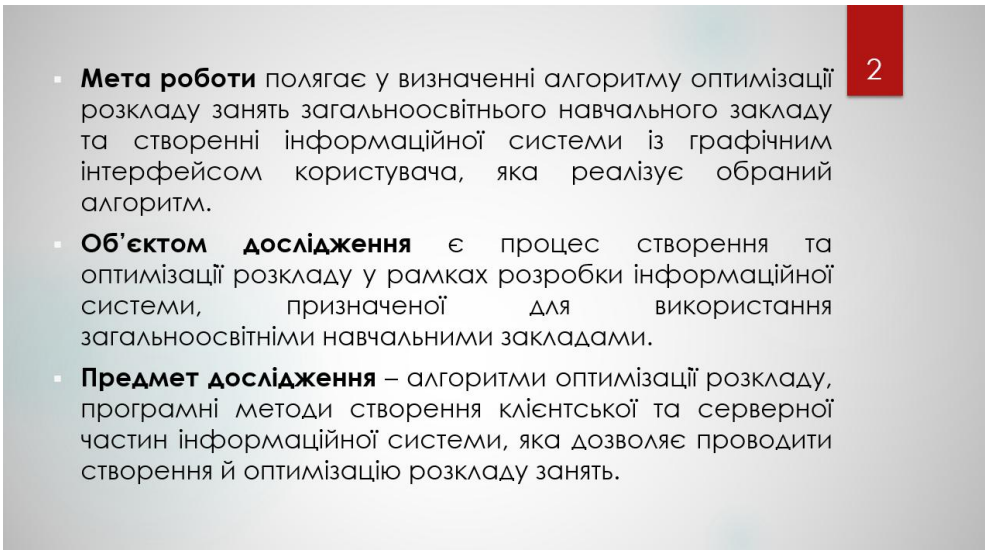


Рисунок В.2 – Слайд №2

Рисунок В.3 – Слайд №3

Актуальність теми

4

Загальні проблеми:

- велика кількість помилок і конфліктів за умови складання розкладу без використання комп'ютерних технологій;
- велика трудомісткість процесу складання розкладу;
- велика кількість обмежень і факторів, які треба враховувати;
- динамічність навчального процесу.

Рисунок В.4 – Слайд №4

Аналіз аналогічних систем

5

Проаналізовано наступні аналоги системи: EduPage, Free Timetabling Software та Skolaris. За результатами аналізу вже розроблених систем були виділені наступні недоліки:

- надмірна ускладненість інтерфейсу;
- більший акцент на загальне управління ЗНЗ;
- потреба у знаннях з налаштування параметрів для досягнення бажаних результатів.

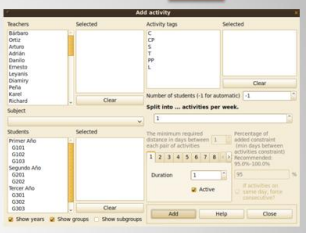


Рисунок В.5 – Слайд №5

Постановка задачі для розробки ПЗ

6

З цього випливають основні вимоги до ІС для оптимізації розкладу:

- **автоматизація складання розкладу.** Система має автоматично розподіляти предмети за часом на основі введених даних;
- **гнучке налаштування навантаження.** Користувачі повинні мати можливість легко керувати поведінкою програми, шляхом зміни навантаження на вчителів та класи;
- **графічний інтерфейс.** Система повинна мати інтуїтивний інтерфейс для введення даних, перегляду та редагування розкладу.

Рисунок В.6 – Слайд №6

Рисунок В.7 – Слайд №7

Рисунок В.8 – Слайд №8

Рисунок В.9 – Слайд №9

Рисунок В.10 – Слайд №10

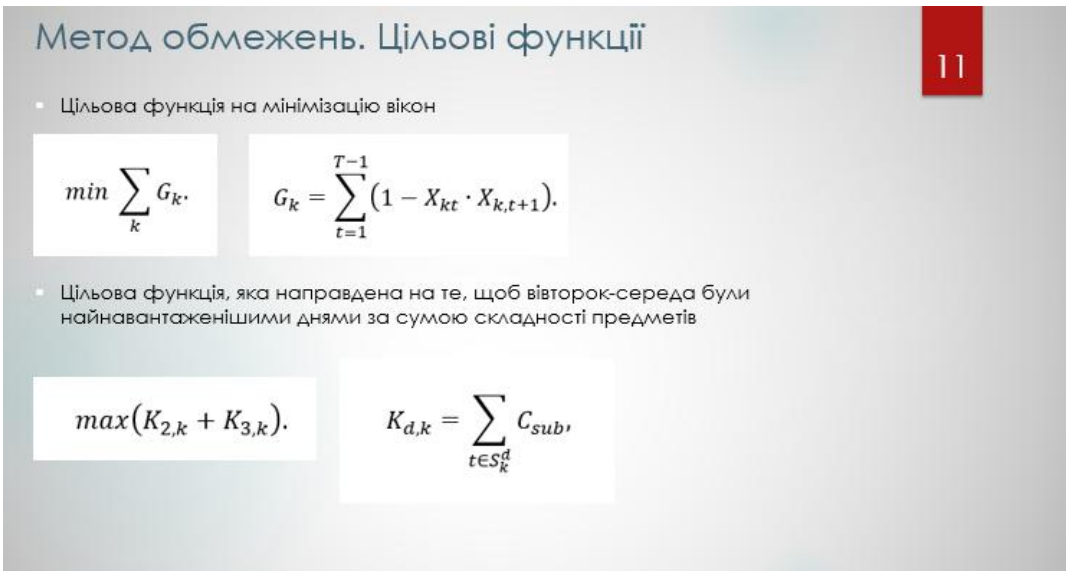


Рисунок В.11 – Слайд №11

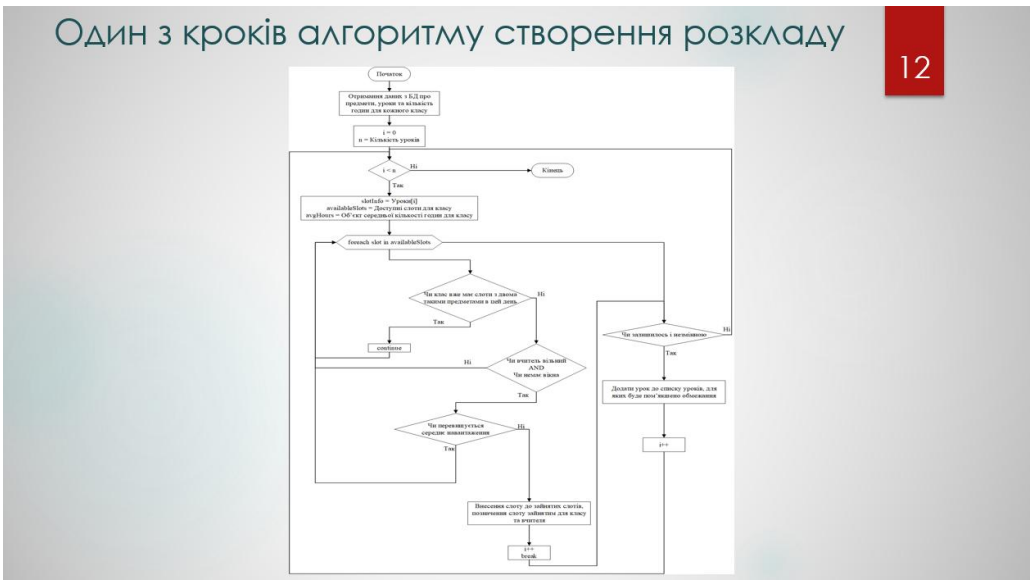


Рисунок В.12 – Слайд №12

Використані технології та архітектура

13

- SQLite;
- .NET 6.0;
- WPF;
- Dapper;

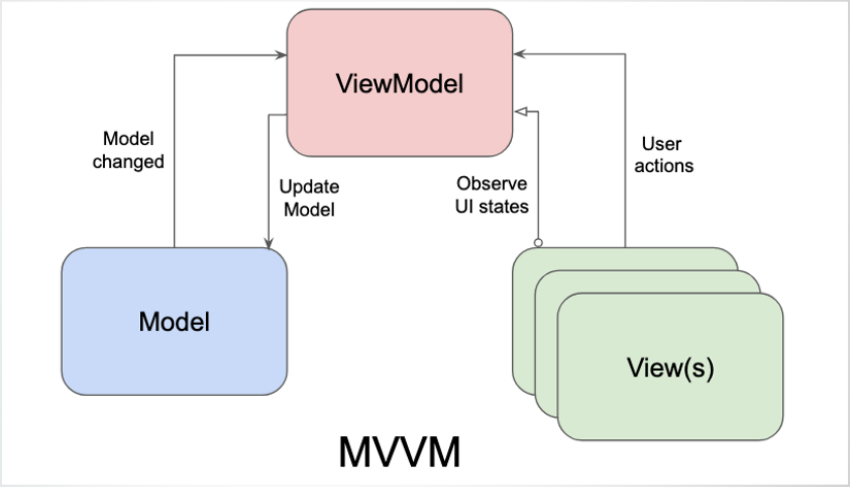


Рисунок В.13 – Слайд №13

Реалізація алгоритму створення розкладу. Розподіл «складних» уроків

14

```
while (i < difficulLessons.Count)
    {
        int startingI = i;
        var slotInfo = difficulLessons[i];
        var availableSlots = ClassSlotsAvailability.Where(x => x.Value == true && x.Key.clas == slotInfo.FkSchedule
            && x.Key.day >= 2 && x.Key.day <= 3).ToList();
        var avgHours = AvgHoursForClass.First(x => x.FkSchedule == slotInfo.FkSchedule);
        foreach (var slot in availableSlots)
        {

            if (IsClassHaveTwoSameSubjectsInDay(slotInfo, slot.Key))
            {
                continue;
            }
            if (IsTeacherFree(slotInfo, slot.Key) && IsPreviousSlotTaken(slot.Key))
            {
                if (IsDayHoursNormal(avgHours.AvgHours, slot.Key))
                {
                    ClassSlotsAvailability[slot.Key] = false;
                    var teacherSlotKey = (slot.Key.day, slot.Key.time, slotInfo.FkTeacher);
                    TeacherSlotsAvailability[teacherSlotKey] = false;
                    AddSlotToTakenSlots(slotInfo, (slot.Key.day, slot.Key.time));
                    i++;
                    break;
                }
            }

        }
        if (i == startingI)
        {
            otherLessons.Add(slotInfo);
            i++;
        }
    }
```

Рисунок В.14 – Слайд №14

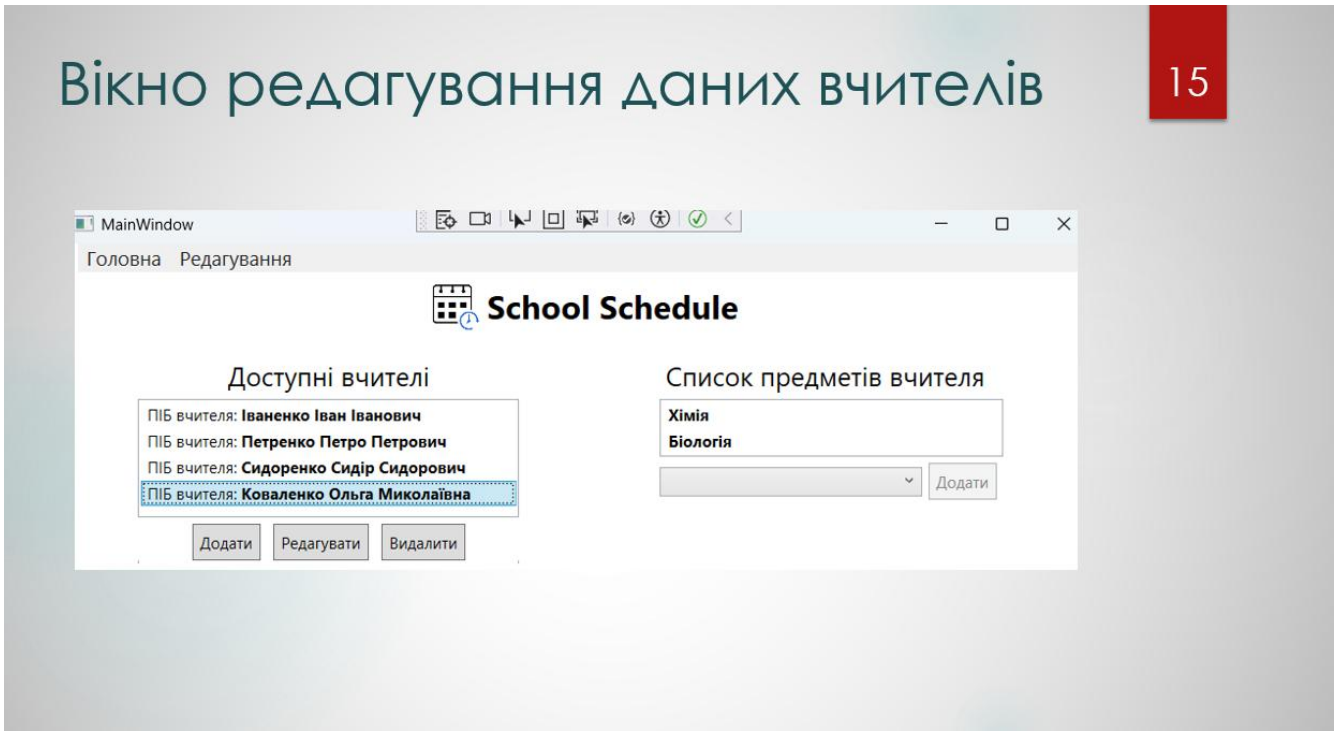


Рисунок В.15 – Слайд №15

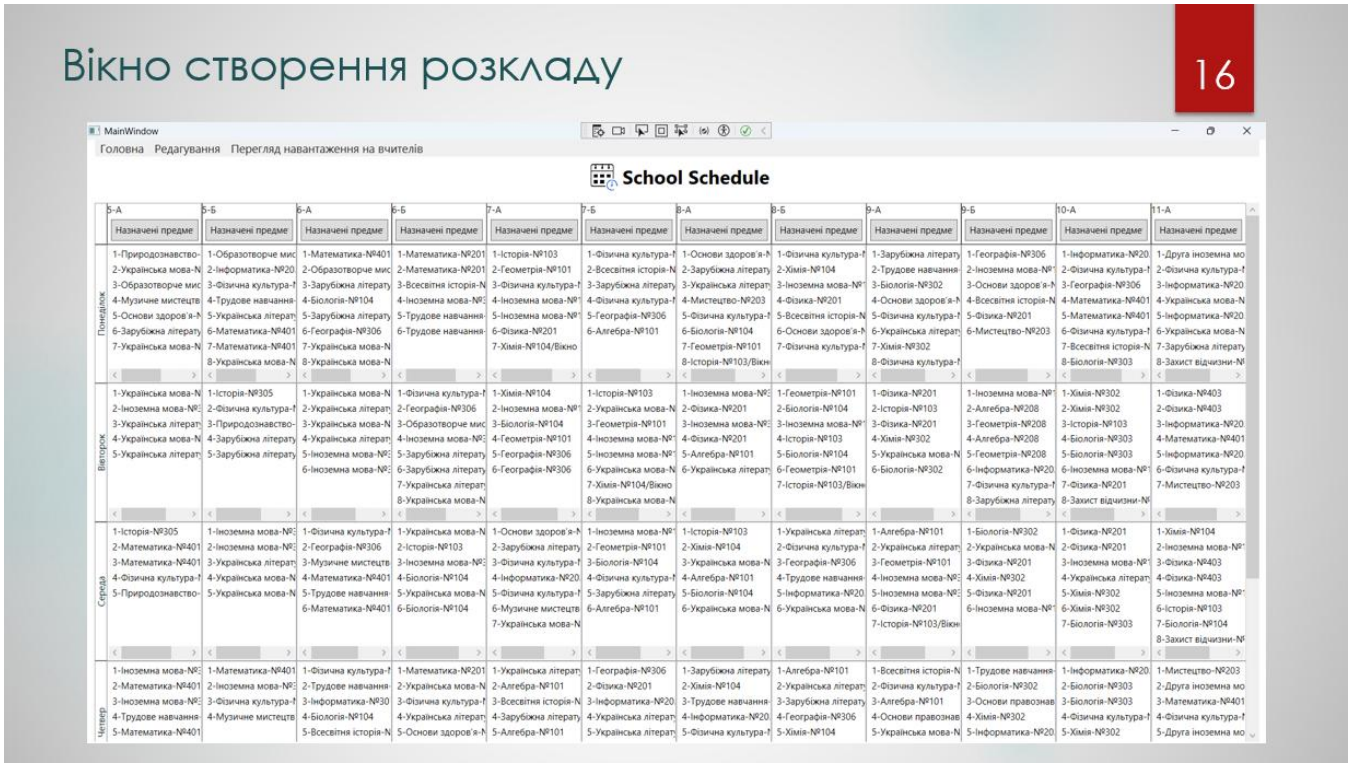


Рисунок В.16 – Слайд №16

17

Висновки

У рамках кваліфікаційної роботи обрано алгоритм оптимізації розкладу для загальноосвітнього навчального закладу, а також здійснено процес проєктування та розробки інформаційної системи.

Протягом виконання роботи було проведено аналіз предметної області, розробка вимог до системи, а також реалізовано всі функції застосунку, які було визначено в рамках постановки задачі на розробку ПЗ. Реалізовано користувацький інтерфейс у вигляді XAML-форм для взаємодії користувача з БД.

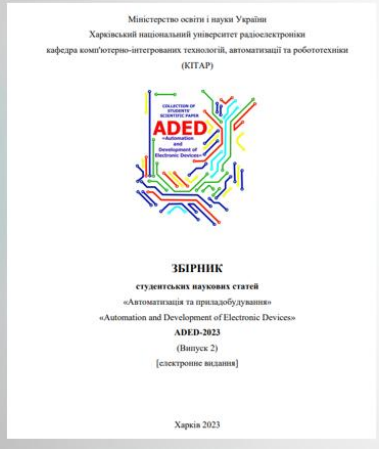
За результатами проведення експерименту оптимізації розкладу загальноосвітнього навчального розкладу можна зробити наступні висновки: розроблений програмний засіб успішно складає розклад без наявності вікон і забезпечує рівномірне навантаження протягом тижня. Але частина програми, яка стосується розподілу складних предметів, може бути вдосконалена, хоча абсолютна більшість розкладів класів відповідає цьому критерію, шляхом оптимізації алгоритму під велике навантаження вчителів.

Рисунок В.17 – Слайд №17

18

Дякую за увагу!

Пісклов М. А. Алгоритми створення та оптимізації розкладу для загальноосвітніх навчальних закладів. Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс]: збірник студентських наукових статей: ХНУРЕ, 2023. – Вип. 2. – С. 275-280.

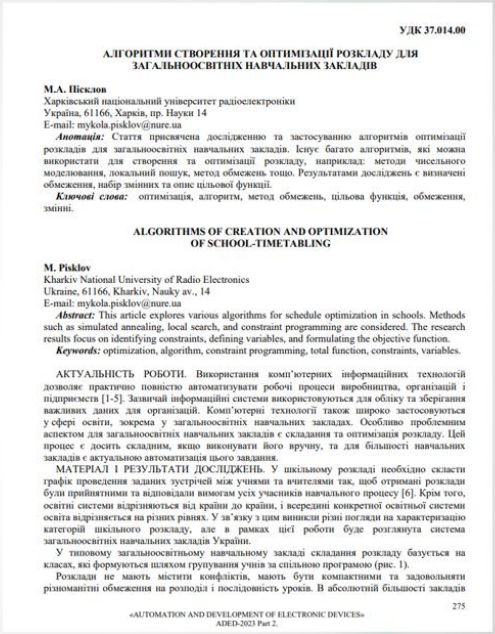


Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)

ADED

ЗБІРНИК
студентських наукових статей
«Автоматизація та приладобудування»
«Automation and Development of Electronic Devices»
ADED-2023
(Випуск 2)
[електронне видання]

Харків 2023



УДК 37.014.00

АЛГОРИТМИ СТВОРЕННЯ ТА ОПТИМІЗАЦІЇ РОЗКЛАДУ ДЛЯ ЗАГАЛЬНООСВІТНІХ НАВЧАЛЬНИХ ЗАКЛАДІВ

М.А. Пісклов
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: mykola.pisklov@nure.ua

Анотація: Стаття присвячена дослідженню та застосуванню алгоритмів оптимізації розкладів для загальноосвітніх навчальних закладів. Існує багато алгоритмів, які можна використати для створення та оптимізації розкладу, наприклад: методи чисельного моделювання, локальний пошук, метод обмежень тощо. Результатами досліджень є визначені обмеження, набір змінних та опис цільової функції.

Ключові слова: оптимізація, алгоритм, метод обмежень, цільова функція, обмеження, змінні.

ALGORITHMS OF CREATION AND OPTIMIZATION OF SCHOOL-TIMETABLING

M. Pisklov
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: mykola.pisklov@nure.ua

Abstract: This article explores various algorithms for schedule optimization in schools. Methods such as simulated annealing, local search, and constraint programming are considered. The research results focus on identifying constraints, defining variables, and formulating the objective function.

Keywords: optimization, algorithm, constraint programming, total function, constraints, variables.

АКТУАЛЬНІСТЬ РОБОТИ. Використання комп'ютерних інформаційних технологій дозволяє практично повністю автоматизувати робочі процеси виробництва, організацій і підприємств [1-5]. Зазвичай інформаційні системи використовуються для обліку та зберігання важливих даних для організацій. Комп'ютерні технології також широко застосовуються у сфері освіти, зокрема у загальноосвітніх навчальних закладах. Особливо проблемним аспектом для загальноосвітніх навчальних закладів є складання та оптимізація розкладу. Цей процес є досить складним, якщо виконувати його вручну, та для більшості навчальних закладів є актуальною автоматизація цього завдання.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. У шкільному розкладі необхідно скласти графік проведення заданих зустрічей між учнями та вчителями так, щоб отримані розклади були прийнятними та відповідали вимогам усіх учасників навчального процесу [6]. Крім того, освітні системи відрізняються від країни до країни, і всередині конкретної освітньої системи освіта відрізняється на різних рівнях. У зв'язку з цим виникли різні погляди на характеризацію категорій шкільного розкладу, але в рамках цієї роботи буде розглянута система загальноосвітніх навчальних закладів України.

У типовому загальноосвітньому навчальному закладі складання розкладу базується на класах, які формуються шляхом групування учнів за спільною програмою (рис. 1).

Розклади не мають містити конфліктів, мають бути компактними та задовольняти різноманітні обмеження на розподіл і послідовність уроків. В абсолютній більшості закладів

Рисунок В.18 – Слайд №18

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК.504390.029 ПЗ	Пояснювальна записка	84 с., ф. А4
		Графічні документи	
		Презентація	у форматі
			*.ppt

					ГЮИК.504390.029 ВД
Змін.	Арк.	№ документа	Підп.	Дата	
Розроб.		Пісклов М. А.			**Розроблення інформаційної системи для оптимізації розкладу занять загальноосвітнього навчального закладу** Відомість кваліфікаційної роботи
Перевір.		Жарікова І. В.			
Н. контр.		Демська Н. П.			
Затвер.		Невлюдов І. Ш.			

Літ.	Аркуш	Аркушів

ХНУРЕ

Кафедра КІТАР