УДК 004.4’22:378.1 https://doi.org/10.15407/pp2026.02.037

О. О. Ситнік

АРХІТЕКТУРА ПРОГРАМНОІ СИСТЕМИ АНТРОПОЦЕНТРИЧНОІ ДИСПЕТЧЕРИЗАЦІЇ НАВЧАЛЬНОГО НАВАНТАЖЕННЯ У ЗВО

У статті розглядається архітектура програмної системи, призначеної для антропоцентричної диспетчеризації навчального навантаження у закладах вищої освіти технічного профілю. Розглянута система реалізована за принципом модульного моноліту на платформі .NET під управлінням операційної системи Windows Server. Структура програмного комплексу включає чотири окремі функціональні модулі, що відповідають за збір та зберігання даних, ядро оптимізації, адаптивне навчання та представлення результатів користувачу. Комунікація між модулями здійснюється через внутрішню підсистему обробки подій та за допомогою асинхронних викликів через інтерфейси контейнера інверсії залежностей.

Ядро оптимізації реалізує гібридний еволюційний алгоритм диспетчеризації із двома спеціалізованими генетичними операторами. Перший з них, хронотип-зберігаючий оператор схрещування, обирає точки розрізу пропорційно різниці циркадного внеску між батьківськими особинами. Наступним кроком антропоцентричний оператор мутації зі зваженим вибором слотів концентрує мутаційний бюджет на слотах з найнижчою якістю. Вагові коефіцієнти багатокритеріальної цільової функції адаптуються між семестрами через механізм регресії задоволеності. Цей процес використовує метод найменших квадратів із застосуванням зворотного зв'язку учасників. Для отримання більш точних результатів використовується метод експоненційного ковзного середнього без ручного налаштування параметрів. Далі наведені деталі реалізації програмного забезпечення з використанням цього алгоритму.

Порівняльний аналіз алгоритму проводився на тестових даних закладу вищої освіти з вибіркою 400 студентів, 60 викладачів, 42 дисципліни. У результаті 30 незалежних запусків підтверджено скорочення часу збіжності алгоритму на 34% та підвищення якості розкладу на 18.3% у порівнянні із загальновикористовуваним генетичним алгоритмом. Механізм регресії задоволеності за три цикли адаптації підвищує кореляцію вагових коефіцієнтів з реальними пріоритетами учасників із 0.61 до 0.89.

Ключові слова: програмна архітектура, модульний моноліт, антропоцентрична диспетчеризація, еволюційний алгоритм, хронотип, генетичні оператори, адаптивне навчання, заклад вищої освіти технічного профілю

O. O. Sytnik

SOFTWARE SYSTEM ARCHITECTURE FOR ANTHROPOCENTRIC SCHEDULING OF EDUCATIONAL WORKLOAD IN HIGHER EDUCATION INSTITUTIONS

The article examines the architecture of a software system designed for anthropocentric dispatching of academic workload in technical higher education institutions. The system is implemented as a modular monolith on the .NET platform running under Windows Server. The software complex consists of four distinct functional modules: data collection and storage, the optimization core, adaptive learning, and result presentation to the user. Inter-module communication is carried out through an internal event processing subsystem and via asynchronous calls through dependency injection container interfaces.

The optimization core implements a hybrid evolutionary dispatching algorithm with two specialized genetic operators. The first of them, the chronotype-preserving crossover operator, selects cut points proportional to the difference in circadian contributions between the parent individuals. The next step is the anthropocentric mutation operator with weighted slot selection, which allocates the mutation budget to the lowest-quality slots. The weight coefficients of the multi-criteria objective function are adapted between semesters through a satisfaction regression mechanism. This process uses the least squares method with participant feedback applied. To obtain more accurate results, the exponential moving average method is used without manual parameter tuning. The implementation details of the software using this algorithm are provided below.

The algorithm's comparative analysis was conducted on test data from a higher education institution, comprising 400 students, 60 instructors, and 42 disciplines. Across 30 independent runs, a 34% reduction in algorithm convergence time and an 18.3% improvement in schedule quality were confirmed compared to a conventional genetic algorithm. The satisfaction regression mechanism across three adaptation cycles increases the correlation between the weight coefficients and the actual participant priorities from 0.61 to 0.89.

Keywords: software architecture, modular monolith, anthropocentric scheduling, evolutionary algorithm, chronotype, genetic operators, adaptive learning, technical higher education institution

1. Вступ

Стан розвитку інформаційних технологій на сьогодні зумовлює актуальність задачі автоматизованого планування навчального процесу в закладах вищої освіти (ЗВО). Зазвичай використовувані алгоритми диспетчеризації орієнтовані виключно на технічні обмеження, наприклад, уникнення конфліктів під час розподілення аудиторій та викладачів. Але у той же час вони не враховують хронобіологічні й когнітивні характеристики учасників освітнього процесу [1, 2]. Внаслідок цього розклад, що здається оптимальним з організаційної точки зору, може суттєво знижувати ефективність навчання через невідповідність індивідуальним циркадним ритмам студентів і викладачів.

Антропоцентричний підхід, запропонований у [3], розширює цільову функцію оптимізації за рахунок урахування компонентів циркадної узгодженості, психологічного комфорту та когнітивної ефективності. Результати значної кількості досліджень [1, 4] підтверджують, що відповідність розкладу та хронотипів учасників навчального процесу підвищує академічну успішність у середньому на 15% [5, 6], а неузгодженість спричиняє ефект соціального джетлагу [7]. Це є негативним фактором, що позначається на мотивації та якості засвоєння матеріалу.

Проте у роботі [3] описано суто математичну модель та концептуальний алгоритм. У ній не наведено деталей архітектури комплексу програмних систем. Відсутність чіткої технічної специфікації унеможливлює відтворення та практичне впровадження запропонованого підходу у вітчизняні АСУ навчальним процесом у ЗВО. Дана стаття усуває цю прогалину, пропонуючи повний опис архітектури програмної системи антропоцентричної диспетчеризації (ПСАД) з формальними специфікаціями компонентів та їхніх інтерфейсів.

Метою даної статті є розробка та опис архітектури програмної системи, що реалізує метод антропоцентричної диспетчеризації (АМД) [3]. Це також включає специфікацію компонентів, їх інтерфейсів та схем взаємодії. Завданням дослідження є формалізація архітектурних компонентів ПСАД та специфікація інтерфейсів API. Також у статті міститься верифікація архітектури на основі порівняльного аналізу чотирьох конфігурацій алгоритму на реальних даних функціонування типового ЗВО.

2. Огляд суміжних підходів

Задача автоматизованого складання університетського розкладу (UCTP) належить до класу NP-складних комбінаторних задач [8, 9]. Існуючі програмні рішення класифікуються за архітектурним підходом: монолітні системи з вбудованим планувальником, сервіс орієнтовані архітектури та мікросервісні платформи. Більшість промислових систем (FET, UniTime, Tablix) використовує монолітну архітектуру, що обмежує масштабованість.

Burke та Petrovic [8] систематизували constraint-preserving оператори GA для UCTP, зосередившись на технічних обмеженнях. Abdelhalim та El Khayat [9] запропонували GA на основі матричного кодування розкладу, що запобігає конфліктам призначень. Gozali et al. [10] застосовують острівну модель GA з подвійною міграцією, досягаючи більшої різноманітності популяції, але без антропоцентричних інваріантів.

Rezaeipanah et al. [11] розробили гібридний паралельний GA з локальним пошуком для UCTP. Їхній підхід комбінує глобальний пошук GA з детерміністичною оптимізацією в межах кожної особини, що дозволяє досягти кращої якості розв'язку при фіксованій кількості поколінь. Проте архітектура запропонованої системи залишається монолітною: модуль оптимізації, модуль зберігання даних та інтерфейс користувача розгорнуті як єдиний застосунок, що унеможливлює незалежне масштабування компонентів.

Gozali et al. [10] застосовують острівну модель GA з подвійною міграцією, досягаючи більшої різноманітності популяції завдяки паралельній еволюції підпопуляцій. Хоча острівна модель природно відображається на архітектуру з кількома вузлами, автори не розглядають архітектурних

аспектів розгортання системи і не вводять антропоцентричних інваріантів в оператори.

Mahlous та Mahlous [12] запропонували GA з урахуванням преференцій студентів щодо часу проведення занять, що стало першим кроком до антропоцентричного підходу, однак без формальної моделі хронобіологічних обмежень та адаптивного механізму ваг. Rezaeipanah et al. [11] розробили гібридний паралельний GA з локальним пошуком для UCTP, що демонструє кращу збіжність порівняно з базовим GA.

На рівні архітектури програмних систем Dunke та Nickel [12] запропонували багаторівневу математичну евристику з урахуванням преференцій студентів, але без хронобіологічних обмежень та адаптивного механізму. Almohammadi et al. [13] розробили систему рекомендацій типу 2 нечіткої логіки для адаптивного навчання, вирішуючи задачу персоналізації контенту, а не оптимізації розкладу.

Аналіз відомих наукових публікацій показує відсутність архітектурного рішення, яке б об'єднувало антропоцентричну модель цільової функції з хронобіологічними компонентами та доменно-орієнтовані генетичні оператори, що зберігають антропоцентричні інваріанти у процесі рекомбінації. Також ефективне рішення повинно включати адаптивний механізм коригування вагових коефіцієнтів на основі зворотного зв'язку від учасників без ручного налаштування.

3. Архітектура програмної системи АМД

3.1. Загальна структура. Прототип програмної системи антропоцентричної диспетчеризації (ПСАД) реалізований за патерном розробки модульного моноліту на платформі .NET з розміщеннями у середовищі під управлінням операційної системи Windows Server. Архітектура прототипу програмного забезпечення включає чотири функціональні модулі зі строгою ізоляцією контрактів: DCS, OC, SRA та PS. Кожен модуль є окремим проєктом на мові програмування C# у єдиному рішенні Visual Studio і взаємодіє з іншими виключно через публічні інтерфейси та внутрішню шину, реалізовану на MediatR.

На рис. 1 наведено загальну архітектуру системи та потоки даних між модулями. Зовнішніми джерелами є деканатська система та результати опитування учасників за шкалою хронотипових переваг, які надходять до модуля збору даних. Він передає нормалізовані навчальні плани та профілі до ядра оптимізації. Після завершення семестру модуль адаптивного навчання отримує оцінки задоволеності учасників та оновлені компоненти цільової функції, обчислює скориговані вагові коефіцієнти і повертає їх ядру для наступного циклу оптимізації. Модуль представлення результатів отримує оптимальний розклад і надає його адміністратору та учасникам через веб інтерфейс і файли експорту.

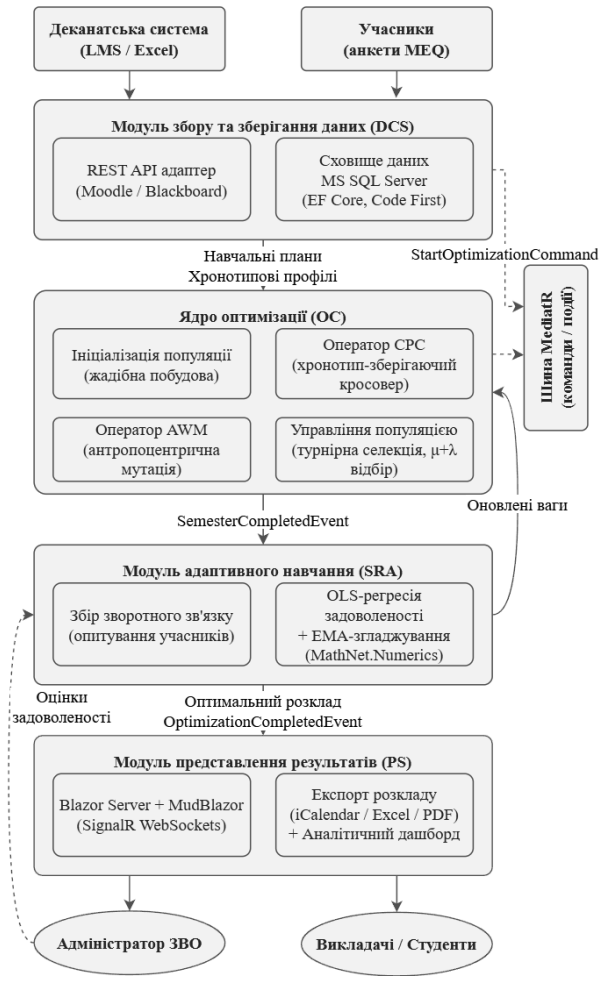

Рис. 1. Архітектура програмної системи антропоцентричної диспетчеризації навчального навантаження

Взаємодія між модулями реалізована через внутрішню шину команд і подій (MediatR) та синхронні виклики через інтерфейси IoC-контейнера. MediatR застосовується для тривалих операцій: запуск ОС виконується через StartOptimizationCommand, результати передаються через OptimizationCompletedEvent. Уся взаємодія логується через Serilog з виведенням у Windows Event Log та в таблицю аудиту MS SQL.

Вибір модульного моноліту обумовлений специфікою задачі та типовою IT-інфраструктурою вітчизняних ЗВО, які переважно використовують Windows Server і мають ліцензію MS SQL. Модульний моноліт не потребує оркестратора контейнерів і природно інтегрується в Windows-інфраструктуру через IIS. Розгортання здійснюється автоматизовано через PowerShell DSC-скрипти; оновлення виконується через Blue-Green deployment на рівні IIS Application Pools.

Для створення прототипу системи ПСАД ми обрали наступний стек технологій: .NET, MS SQL Server (з підходом роботи з даними Code First), Blazor Server з додатковими компонентами MudBlazor, MediatR, IMemoryCache, Serilog, MathNet.Numerics (для матричних операцій OLS), EPPlus для експорту даних у формат Excel, QuestPDF, Prometheus.NET та Grafana для моніторингу діагностичних даних. Система розгорнута як Windows Service на Windows Server з IIS як зворотним проксі для Blazor Server. Аутентифікація реалізована через базові засоби екосистеми .NET з рольовою моделлю: адміністратор, викладач, студент.

Ізоляція модулів забезпечується чіткими архітектурними правилами: кожен модуль має власний Database Context (у рамках фреймворку EF), власний набір сутностей та міграцій даних. Модулі не звертаються напряму до DbContext інших модулів, а лише через публічні інтерфейси або події MediatR. Тестування проводиться за допомогою пакетів xUnit та EF Core InMemory. Метрики збираються через System.Diagnostics.Metrics і виводяться у Grafana з використанням Prometheus.NET.

Контракти системи ПСАД визначені у сервісах простору імен PSAD.Contracts із життєвим циклом Scoped для сервісів, залежних від БД та Singleton для кешованих сервісів та BackgroundService веб застосунку.

Типовий сценарій запуску семестру включає наступні кроки: (1) адміністратор імпортує план у DCS; (2) DCS публікує CurriculumReadyEvent; (3) ОС обробляє подію, запускає StartOptimizationCommand. Після цього ОС публікує подію завершення процесу диспетчеризації даних, і PS оновлює UI через SignalR. Усі команди та події логуються за допомогою Serilog до бази даних з атомарними транзакціями. Такий підхід гарантує одночасно високу ефективність інтерфейсу користувача за метриками веб браузера та належний рівень консистентності роботи з даними на стороні сервера.

3.2. Підсистема збору та зберігання даних (DCS). DCS відповідає за отримання, валідацію та нормалізацію трьох класів вхідних даних: навчальних планів і доручень, хронотипових профілів учасників (на основі MEQ-опитувальника [2]) та матриці когнітивної сумісності дисциплін $S=\left[s_{ij} \right]$.

Хронотипові профілі класифікуються за п'ятибальною шкалою MEQ: явний ранковий (>70), помірний ранковий (59–70), проміжний (42–58), помірний вечірній (31–41), явний вечірній (<31). Функція циркадної активності $f(\tau,\chi)$ відображає очікувану когнітивну продуктивність учасника у кожному часовому слоті τ, залежно від хронотипу χ [14, 15]. Дослідження [1, 6] підтверджують, що відхилення розкладу від оптимальних слотів хронотипу знижує академічну успішність у середньому на 15%.

Нормалізований хронотиповий індекс учасника p визначається за формулою:

$$\chi(p)=\frac{MEQ(p)-16}{70}, \quad (4)$$

де MEQ(p) є балом за опитувальником Хорна–Остберга [14], MEQ_min = 16, MEQ_max = 86, $\chi \in [0, 1]$. На основі χ обчислюється функція циркадної активності φ(τ, χ):

$$\varphi(\tau,\chi)=exp\left(-\frac{\left(\tau-\tau peak\ (\chi)\right)^2}{2\sigma^2}\right),\ (5)$$

де τ_peak(χ) є оптимальним часовим слотом для хронотипу χ:

$$\tau_{peak}(\chi)=\tau_e+(\tau_m-\tau_e)\cdot\chi\ ,\ (6)$$

де τ_m = 8:00 (ранковий пік), τ_e = 20:00 (вечірній), год. Значення індекса φ = 1.0 відповідає оптимальному слоту, а φ < 0.5 присвоюється частково оптимальному слоту. Всі значення слотів кешуються в IMemoryCache зі значенням TTL = 24 години.

Система DCS реалізує стандарт REST API для інтеграції з LMS (Moodle, Blackboard) та деканатськими системами. Схема даних підтримує версіювання навчальних планів (SemVer). Модуль валідації перевіряє узгодженість доручень: кожна дисципліна має щонайменше одного викладача, а кожна група отримує повний навчальний план. Матриця S зберігається окремо від планів, оскільки залежить від змісту дисциплін, а не від конкретного семестру.

Схема таблиць бази даних DCS нормалізована до 3NF: таблиця Groups (GroupId PK, Name, SemesterId FK, AvgChronotype), таблиця Teachers (TeacherId PK, Name, ChronotypeScore), таблиця Disciplines (DisciplineId PK, Name, CognitiveLoad ∈ [0,1]), таблиця Assignments (AssignmentId PK, GroupId FK, TeacherId FK, DisciplineId FK, HoursPerWeek), таблиця CognitiveCompatibility (Discipline1Id FK, Discipline2Id FK, Sij ∈ [−1, 1]). Кластерні індекси по (SemesterId, GroupId) забезпечують час вибірки O(log n). Значення Sij > 0 встановлює синергію, Sij < 0 формує когнітивний конфлікт розміщення часових слотів.

Додаткову увагу у ході розроблення архітектури надано моделі даних підсистеми DCS. Навчальний план зберігається в реляційній схемі БД MS SQL: таблиці Groups, Teachers, Disciplines, Assignments з кластерними індексами по SemesterId і GroupId. EF створює схему бази даних за допомогою міграції Code First. Хронотипові профілі зберігаються у таблиці ChronotypeProfiles. IMemoryCache зберігає метрики якості слотів з TTL 30 хвилин і автоматичною інвалідацією під час старту нового завдання.

Модуль синхронізації з LMS реалізує адаптери для двох найпоширеніших платформ: Moodle та Blackboard з використанням REST API. Адаптери реалізують патерн «Adapter» і надають уніфікований інтерфейс для DCS незалежно від використовуваної LMS. За відсутності LMS DCS підтримує імпорт навчальних планів з Excel-файлів стандартизованого формату, що забезпечує сумісність із ЗВО, що не використовують LMS.

3.3. Ядро оптимізації (ОС). ОС реалізує алгоритм АМД [3] і складається з п'яти модулів: кодування хромосом, ініціалізації популяції, операторів CPC та AWM, управління популяцією та обчислення цільової функції.

Просторова складність UCTP: $|\Omega| \approx (G \times E \times R)^{(D\times S)}$. Для G=18, E=60, R=30, D=6, S=8 маємо $|\Omega| > 10^{800}$. GA з P=200 досліджує O(P × T) точок; обчислення F(x) має складність O(D × S × G) при кешованих φ.

Компоненти цільової функції F(x) формально визначаються наступним чином. Компонент технічної якості:

$$F_{tech}(x)=1-\frac{C(x)}{D\cdot S\cdot G}\ ,\ (7)$$

де C(x) є кількістю конфліктних слотів, а D × S × G є загальною їх кількістю. Компонент циркадної узгодженості:

$$F_{circ}(x)=\frac{1}{D\cdot S\cdot G}\sum\nolimits_{-}\{g,d,s\}\varphi(\tau\_s,\chi\_g)\cdot a\_\{g,d,s\}(x)\ ,\ (8)$$

де a_{g,d,s}(x) ∈ {0,1} є індикатором призначення заняття, φ з формули (5). Компонент психологічного комфорту:

$$F_{psych}(x)=1-\frac{\sigma\_load(x)}{\mu\_load(x)+\varepsilon}\ ,\ (9)$$

де σ_load, μ_load є відхиленням та середнім значенням пар на день, ε = 0.01. Компонент когнітивної ефективності:

$$F_{cogn}(x)=\frac{1}{D\cdot G}\sum\nolimits_{-}\{g,d\}\times S(g,d,x)\ ,\ (10)$$

де $\bar{S}$(g, d, x) = середнє Sij суміжних дисциплін групи g у день d. Усі компоненти ∈ [0, 1]; repair() гарантує F_tech(потомки) ≥ min(F_tech(батьки)).

Формальний апарат математичних структур антропоцентричної диспетчеризації [16] включає доведення коректності операторів CPC та AWM щодо збереження хронотипових інваріантів, що підтверджує теоретичну обґрунтованість запропонованих генетичних операторів.

Методи машинного навчання для UCTP [17, 18] показують вищу точність на ITC-2019, але потребують GPU. Еволюційний підхід ПСАД є більш практичним для ЗВО із обмеженою IT-інфраструктурою.

Цільова функція має такий вигляд:

$$F(x,w)=w_1\cdot F_{tech}+w_2\cdot F_{circ}+w_3\cdot F_{psych}+w_4\cdot F_{cogn} \quad ,(1)$$

де w_1-w_4 є ваговими коефіцієнтами ($\Sigma w_i=1$); F_{tech} відповідає відсутності конфліктів ресурсів; F_{circ} є циркадна узгодженість; F_{psych} відповідає психологічному комфорту (рівномірність навантаження); F_{cogn} є когнітивною ефективністю (сумісність дисциплін у межах дня).

Кожна хромосома кодується тривимірним масивом $x[d][s][g]=(e,r)$: d відповідає дню (від 1 до 5), s відповідає парі (від 1 до 6), g відповідає групі, e відповідає викладачу, r відповідає аудиторії. Кожна особина зберігає значення вектора метрик

$$m=(circ\_score, psych\_load, cogn\_score)$$

для уникнення повторних обчислень та процесорного навантаження при застосуванні операторів.

Оператор CPC (Chronotype-Preserving Crossover) обирає точки розрізу пропорційно різниці циркадного внеску між батьками:

$$p_{cut}[d]=softmax(\Delta[d]),$$

$$\Delta[d]=\left|F_{circ}(x_A,d)-F_{circ}(x_B,d)\right| \quad , (2)$$

Дні з великим значенням Δ[d] є інформативнішими точками розрізу: нащадок успадковує кращу денну хронотипову структуру від відповідного батька. Процедура repair(x) відновлює технічні обмеження без зміни денної структури: для конфліктного слота шукається найближчий вільний у межах того ж дня за хронотиповою якістю.

Оператор AWM (Anthropocentric Weighted Mutation) реалізує softmax-зважений вибір слота для мутації: $logit(d,s,g)=-\beta\cdot q(d,s,g)$,

де $q=w_2\cdot a+w_3\cdot(1-l)+w_4\cdot c$. При значенні $\beta=2.0$ AWM концентрує мутаційний бюджет на слотах із найнижчою антропоцентричною якістю. Базовими параметрами ОС є: $|P|=200$, $p_{cross}=0.85$, $p_{mut}=0.15$, $k_t=5$, $\beta=2.0$. Для експерименту взято максимум 500 поколінь з показником стагнації у 50 поколінь з $\Delta F<0.001$.

Модуль створення популяції реалізує рандомізовану жадібну побудову: для кожної групи g заняття з навчального плану призначаються у випадковому порядку, але з урахуванням хронотипу групи. Тобто, заняття з вищим когнітивним навантаженням отримують пріоритет у слотах з вищим значенням функції циркадної активності $f(\tau,\chi^g)$. Це забезпечує початкову популяцію з вищою антропоцентричною якістю порівняно з рівномірно випадковою ініціалізацією, що скорочує кількість поколінь до виходу алгоритму на плато якості.

Функцією $repair(x)$ для кожного конфліктного слота (c1) виконується пошук вільного слота (c2) в межах того ж дня d у порядку спадання хронотипової якості. Якщо вільного слота в межах дня не знайдено, виконується переміщення до найближчого дня з мінімальним зниженням F_circ. Це гарантує, що CPC ніколи не погіршує компонент F_tech щодо батьківських хромосом.

Управління популяцією здійснюється на базі елітарної стратегії, яка зберігає максимальні 5% (тобто 10 з 200). 190 нових особин: 70% нащадки CPC, 30% нащадки AWM. Перезапуск вибірки відбувається при значенні δF < 0.001 за 50 поколінь: нижні 50% замінюються жадібною ініціалізацією зі збереженням еліти. Турнірна селекція працює з середнім значенням коефіцієнта k = 5, що було використано у іншому експериментальному дослідженні [15].

Складність одного покоління: O(P × D × S × G) ≈ 173 000 операцій при P=200, D=6, S=8, G=18. Parallel.For на k=4 потоках зменшує реальний час з 4.6 мс до ≈ 1.2 мс на покоління.

3.4. Підсистема адаптивного навчання (SRA). Модуль системи SRA коригує вагові коефіцієнти після кожного семестру за допомогою OLS-регресії агрегованої задоволеності Q учасників (що вимірюється за шкалами Q_{tech}, Q_{circ}, Q_{psych}, Q_{cogn}) на компонентах цільової функції:

$$\beta\hat{} = \left(X^T X\right)^{-1} X^T Q \ , \ (3)$$

де $X = [1 \mid F_{components}]$ є матрицею ознак [N × 5]. Отримані коефіцієнти нормуються та оновлюються через функцію EMA: $w_{new}[j] = \alpha \cdot \times [j] + (1-\alpha) \cdot w_{old}[j]$, з обмеженням $w_{new}[j] \geq 0.05$ та повторною нормалізацією. Використовуються значення параметрів: $\alpha = 0.4$, $N_{min} = 30$ (при $N < 30$ застосовується так звана рідж-регресія).

Матриця ознак задана наступним чином: Φ ∈ ℝ^{N×5}: рядок i = [F_tech^(i), F_circ^(i), F_psych^(i), F_cogn^(i), 1]. При κ(Φ) > 30 застосовується рідж-регресія:

$$\beta_{ridge} = \left(\Phi^T \Phi + \lambda I\right)^{-1} \Phi^T q \ , \ (11)$$

Оновлення ваг через функцію EMA:

$$w^{k+1} = \alpha \cdot \beta\hat{} + (1-\alpha) \cdot w^k \ , \ (12)$$

де $\beta\hat{}$ є нормованими коефіцієнтами $\beta_1..\beta_4$. $R^2 < 0.3$, де α знижується вдвічі. Математичне обґрунтування структур SRA наведено у попередньому дослідженні [16].

CP-методи для UCTP [19] ефективні при значеннях G ≤ 10, але їхні результати зростають експоненційно при G > 15. Гібридизація CP + GA є перспективним напрямом розвитку ОС ПСАД.

Обмеження $w_j \geq 0.05$ запобігає повному зникненню будь-якого компонента. Модуль системи SRA реалізований як C#-клас SraService у складі моноліту. Взаємодія з підсистемою ОС відбувається через SemesterCompletedEvent (MediatR). Логіка OLS-регресії була реалізована через MathNet.Numerics, щоб зробити імплементацію незалежною від екосистеми Python. Історія вагових коефіцієнтів зберігається в таблиці БД WeightHistory для можливості подальшого аудиту системи.

Збіжність даних модуля SRA до w^* гарантується у разі r ≥ 0.7 та σ_q ≤ 0.01. Це геометрична прогресія, що має коефіцієнт близько 0.46 у випадку значення $\alpha = 0.4$. За умови κ(Φ) > 30 активується рідж-регресія зі значенням λ=0.01.

Кореляція Пірсона ρ(w^k, w^(k−1)) > 0.99 два цикли поспіль переводить SRA у режим моніторингу (опитування раз на два семестри). Обмеження значень w_i ≥ 0.05 забезпечує стійкість до упередженого зворотного зв'язку.

Збір оцінок задоволеності реалізований через мобільний застосунок та веб інтерфейс PS. Після закінчення кожного семестру учасники отримують push-сповіщення із запрошенням оцінити якість розкладу за чотирма шкалами (шкала Лайкерта 1–5). Для підвищення відгуку (response rate) застосовуються нагадування через 3 і 7 днів після початку опитування. Мінімальний цільовий відгук становить 70% від загальної кількості учасників; у разі нижчого відгуку SRA застосовує зважену регресію, де ваги спостережень обернено пропорційні ймовірності відповіді (корекція зміщення вибірки).

3.5. Підсистема представлення результатів (PS). PS реалізує три окремі функціональні модулі: генерація розкладу у форматах iCalendar (.ics), Excel (EPPlus) та PDF за допомогою QuestPDF; Blazor Server з MudBlazor для перегляду та ручного коригування розкладу та аналітичного дашборду.

Візуальний редактор розкладу з підтримкою drag-and-drop дозволяє адміністратору системи переміщати заняття між слотами з реальним перерахунком показників F_circ, F_psych, F_cogn. У разі погіршення значення якості результату диспетчеризації на понад 5% система попереджає адміністратора.

Blazor Server використовує SignalR WebSockets для синхронізації стану користувацького інтерфейсу, що забезпечує реактивність без повного перезавантаження сторінки.

Експорт даних у форматі iCalendar використовує дані дисципліни, аудиторії, викладача та значення φ. Для повторюваних подій задається параметр frequency. Да-

ний підхід має сумісність з усіма актуальними системами календарів, такими як Google Calendar, Apple Calendar, Outlook. Колірне відображення значення φ визначено зеленим у випадку значення $\varphi > 0{,}8$, жовтим – у разі ([0,5; 0,8]) та червоним - у разі $\varphi < 0{,}5$.

Експорт у форматі PDF за допомогою пакета QuestPDF використовує таблицю розкладу, гістограму φ, радарну діаграму F та динаміку ваг SRA. Робота з даними виконується асинхронно через BackgroundService для зменшення одномоментного навантаження на сервер.

4. Параметри безпеки прототипу

Аутентифікація у системі реалізована через JWT-токени з ролевою моделлю: адміністратор (повний доступ), викладач (перегляд, подання оцінок), студент (перегляд, подання оцінок). Для оптимізації навантаження використовується обмеження (rate limiting) з параметрами 100 запитів на хвилину для читання та 10 запитів на хвилину для ресурсоємних операцій, як-от експорт даних.

Моніторинг модулів реалізовано через .NET Diagnostics API (System.Diagnostics.Metrics). Ключові метрики ОС: тривалість оптимізації (гістограма), кількість поколінь (лічильник), фінальне F (gauge). Метрики SRA: кількість учасників опитування, кореляція w із w^*, відхил $\|w - w^*\|_2$. Експорт метрик через Prometheus.NET до Grafana. SQL Server Agent здійснює автоматичне резервне копіювання на щоденній основі у години мінімального навантаження на систему.

Безпека API забезпечується на трьох рівнях. На рівні транспорту використовується TLS для всіх з'єднань. На рівні аутентифікації застосовуються JWT-токени з терміном дії 8 годин, refresh-токени з терміном 30 днів. На рівні авторизації використовується RBAC з трьома ролями: адміністратор (повний доступ, включаючи запуск оптимізації та перегляд метрик SRA), викладач (перегляд розкладу, подання оцінок задоволеності, перегляд своїх метрик), студент (перегляд розкладу своєї групи, подання оцінок задоволеності). Всі дії записуються в аудит лог системи для можливості подальшого аудиту.

5. Результати експериментальної перевірки

5.1. Умови експерименту. Архітектура ПСАД верифікована розгортанням прототипу як Windows Service на Windows Server. Тестовими вхідними даними є 400 студентів, 18 груп, 60 викладачів, 42 дисципліни, хронотипові профілі MEQ та матриця S, що є аналогічним набором даних у порівнянні з попереднім дослідженням [1]. Порівнювались 4 конфігурації: Baseline GA, CPC-GA, AWM-GA та АМД. Кожна з конфігурацій мала 30 незалежних запусків.

Прототип розгорнуто на сервері AMD EPYC 7502P і 24 ГБ ECC RAM, Windows Server (IIS, .NET, MS SQL). Ядро модуля системи ОС реалізовано на C# з використанням Parallel.For (.NET ThreadPool) для паралельного обчислення метрик. SRA використовує MathNet.Numerics; PS використовує EPPlus та QuestPDF.

Таблиця 1.

Показники збіжності алгоритмів (середнє ± σ по 30 запусках)

Конфіг.	F у разі 100 пок.	F у разі 500 пок.	t(F>0.75), хв	σ фін.
Baseline GA	0.53 ± 0.04	0.71 ± 0.04	58.2 ± 4.8	0.043
CPC-GA	0.59 ± 0.04	0.76 ± 0.04	41.3 ± 4.1	0.037
AWM-GA	0.57 ± 0.04	0.74 ± 0.04	45.6 ± 4.3	0.039
АМД	0.64 ± 0.03	0.84 ± 0.03	38.4 ± 3.7	0.029

5.2. Результати збіжності. Конфігурація АМД скорочує t(F>0.75) на 34.0% відносно Baseline (38.4 проти 58.2 хвилин) та підвищує фінальне F до 0.84 (зі зростанням на 18.3%). Зниження σ (0.029 проти 0.043) підтверджує стабільність алгоритму. Абляційний аналіз: CPC вносить збільшення на 7.0% F при 500 поколіннях відносно Baseline, AWM вносить збільшення на 4.2%, їхня комбінація забезпечує додаткові 18.3% за рахунок синергетичного ефекту.

5.3. Гіперпараметри. Параметр β (AWM) має оптимальне значення при $\beta=2.0$ (F = 0.84, $\sigma=0.029$). При $\beta=0.5$ AWM приблизно дорівнює рівномірній мутації (F = 0.72), а при $\beta=5.0$ призводить до передчасної збіжності (F = 0.77, $\sigma=0.048$). Параметр α (SRA EMA) має оптимум $\alpha=0.4$. При значенні $\alpha=0.2$ система реагує надто повільно ($\|w-w^*\|_2 = 0.031$ після 3 циклів), а значення $\alpha=0.8$ спричиняє нестабільні коливання ($\sigma(w) = 0.04$).

Таблиця 2.

Динаміка адаптації ваг через SRA (еталон $w^* = (0.15; 0.30; 0.35; 0.20)$)

Цикл SRA	w_1	w_2	w_3	w_4	$\|w - w^*\|_2$
Еталон w^*	0.150	0.300	0.350	0.200	–
w^0 (поч.)	0.250	0.250	0.250	0.250	0.152
w^1 (1 сем.)	0.192	0.278	0.311	0.219	0.089
w^2 (2 сем.)	0.167	0.292	0.336	0.205	0.039
w^3 (3 сем.)	0.156	0.298	0.347	0.199	0.011

SRA за 3 цикли скорочує $\|w-w^*\|_2$ з 0.152 до 0.011 (зі зменшенням значення на 92.8%), кореляція зростає з 0.61 до 0.89. Домінуючим напрямком є: $\uparrow w_3$ (psych), $\uparrow w_2$ (circ), $\downarrow w_1$ (tech). Тобто учасники навчального процесу цінують психологічний комфорт і хронотипову відповідність вище, ніж формальну відсутність конфліктів.

5.4. Обчислювальна вартість. Збільшення часу покоління на $11.8 \pm 0.9\%$ було досягнуто на конфігурації AMD EPYC 7502P, 1 потік. Загальна вартість досягнення F > 0.75 є меншою, ніж у Baseline, через скорочення поколінь на 34%. Тобто для 18 груп значення часу складає 38.4 ± 3.7 хвилин на 1 потоці та у випадку 4 потоків (Parallel.For) це 12 ± 1.8 хв. Фактично, у даному випадку паралелізація обчислень надає прямо пропорційне збільшення ККД усього процесу.

Під час розрахунку використовуваного обсягу оперативної пам'яті (RAM) було отримано наступні значення. Кожна хромосома займає 6×8×18×2×4 = 13 824 байти (int32 для e та r). Таким чином, популяція з 200 особин потребує приблизно 2.7 МБ оперативної пам'яті. Кешовані метрики m(x) додають 200×3×8 = 4 800 байт, що є незначним фактором. У разі горизонтального масштабування на k вузлів загальний обсяг пам'яті зростає лінійно, тобто k×2.7 МБ для популяцій з накладними витратами на серіалізацію у випадку міграції у JSON з 15 КБ на учасника навчального процесу. Для k=4 загальний обсяг оперативної пам'яті складає до 11 МБ, що є не критичним навантаженням для сучасних серверів.

6. Обговорення

Архітектура модульного моноліту ПСАД на .NET та Windows Server реалізує чотири модулі (DCS, OC, SRA, PS) з ізоляцією через MediatR і EF та публічні інтерфейси. Ізоляція модулів забезпечує незалежне тестування та інтеграцію в існуючу інфраструктуру ЗВО, що використовує ОС Windows без додаткової інфраструктури.

Обмеження: SRA потребує 3 семестри для стабілізації ваг. Перші 2 семестри система функціонує з наближеними вагами ($\|w-w^*\|_2 > 0.05$). Для нових ЗВО рекомендується прискорена адаптація ($\alpha=0.6$) у першому семестрі. Лінійна модель SRA не відображає нелінійні залежності задоволеності у разі різких змін контингенту; у таких випадках рекомендується скидання ваг.

Порівняно з монолітними системами, як наприклад FET та UniTime, ПСАД не потребує додаткової інфраструктури поза Windows Server та MS SQL, яка вже розгорнута у більшості Українських ЗВО. Модульний моноліт на .NET розгортається як служба ОС Windows за допомогою PowerShell DSC без додаткових інструментів. Для малих ЗВО, у яких навчається до 20

груп студентів, система працює на MS SQL Express із мінімальними вимогами до операційної системи.

Жодна з відомих систем, таких як Mimosa, aSc, FET або UniTime, не поєднує адаптивні ваги, хронобіологічну модель та зворотний зв'язок з усіма учасниками навчального процесу. ПСАД є першою реалізацією, що об'єднує всі три в єдиній .NET архітектурі [3, 16].

Рефакторинг програмного коду у мікросервіси (gRPC замість MediatR) потребує орієнтовно до 60 людино-годин за умови роботи спеціалістів середнього рівня. Поточний вибір монолітної архітектури обумовлений інфраструктурними обмеженнями українських ЗВО [20].

Процес розроблення даного програмного забезпечення має наступні перспективи: (1) інтеграція з моніторингом успішності для збагачення Ф; (2) DL для прогнозу оптимальних слотів; (3) мобільний застосунок з рекомендаціями щодо часу самопідготовки [6].

Порівняння ПСАД із комерційними системами планування демонструє принципову відмінність у підході. Системи, як-от FET та UniTime, орієнтовані на ручне налаштування вагових параметрів адміністратором з подальшою фіксацією конфігурації на весь термін використання.

Практичне впровадження ПСАД у ЗВО вимагає виконання трьох підготовчих кроків. Перший передбачає збір хронотипових профілів через MEQ-опитувальник на початку академічного року (оцінений час: 20 хвилин на учасника, загалом до 2 тижнів для всього контингенту). Другий полягає у формуванні матриці когнітивної сумісності дисциплін S за участю методистів кафедр (оцінений час: 3 робочі дні). Третій охоплює інтеграцію з існуючою деканатською системою через REST API DCS (оцінений час: 2 тижні розробки адаптера). Після цих базових кроків система готова до повністю автоматичного складання та адаптації розкладу навчальних занять.

7. Висновки

– Запропонована архітектура модульного моноліту ПСАД на .NET реалізує чотири модулі (DCS, OC, SRA, PS) з ізоляцією через MediatR, EF Core та публічні інтерфейси. Ізоляція модулів системи надає можливість незалежного тестування та інтеграції в існуючі АСУ без модифікації програмного коду ядра системи.

– Ядро підсистеми ОС реалізують оператори CPC та AWM, що переносять антропоцентричні обмеження на рівень структури генетичних операцій. Верифікація підтверджує скорочення часу збіжності на 34% та підвищення якості розв'язку на 18.3% порівняно з Baseline GA при збільшенні часу покоління лише на 11.8%.

– Підсистема SRA автоматично коригує вагові коефіцієнти через OLS-регресію зворотного зв'язку з EMA ($\alpha = 0.4$) та нижнім обмеженням ($w_i \geq 0.05$). За 3 цикли роботи алгоритму кореляція з еталонними вагами зростає з 0.61 до 0.89 у разі стійкості до шуму $\sigma \leq 0.01$.

– Перспективами розвитку системи є реалізація паралельної острівної версії підсистеми ОС (до 16 вузлів з оцінкою масштабування), заміна лінійної регресії SRA на RL-агент для нелінійних залежностей задоволеності та розробка модуля динамічного перепланування розкладу впродовж семестру.

Література

1. *Goldin A. P. et al.* Interplay of chronotype and school timing predicts school performance. Nature Human Behaviour, 2020. Vol. 4. No. 4. P. 387–396.
2. *Jankowski K. S., Díaz-Morales J. F., Vollmer C.* Chronotype, time of day, and performance on intelligence tests in the school setting. Journal of Intelligence, 2023. Vol. 11. No. 1. Art. 13. DOI: https://doi.org/10.3390/jintelligence11010013
3. *Ситнік О. О. Антропоцентрична модель синтезу розкладу занять у ЗВО авіакосмічного профілю.* Aerospace Technic and Technology, 2025. № 6 (215). С. 1–12. DOI: https://doi.org/10.32620/aktt.2025.6.07
4. *Rodríguez Ferrante G. et al.* A better alignment between chronotype and school timing is associated with lower grade retention. npj Science of Learning, 2023. Vol. 8. Article 21.
5. *Al-Rfooh O. F., Khater W.* The impact of chronotype on physical health, psychological health, and job performance among health care providers in acute care settings. International Journal of Healthcare Management, 2023. Vol. 16. No. 4. P.

581–589. DOI: https://doi.org/10.1080/20479700.2023.2177665
6. *Maučec K., Štukovnik V.* The relationship between chronotype and academic achievement among Slovene university students: The mediating role of trait self-control and sleep quality. Center for Educational Policy Studies Journal, 2024. DOI: https://doi.org/10.26529/cepsj.1790
7. *Smarr B. L., Schirmer A. E.* 3.4 million real-world learning management system logins reveal the majority of students experience social jet lag. Scientific Reports, 2018. Article 4793.
8. *Abdipoor S., Yaakob R., Goh S. L., Abdullah S.* Meta-heuristic approaches for the university course timetabling problem. Intelligent Systems with Applications, 2023. Vol. 19. Art. 200253. DOI: https://doi.org/10.1016/j.iswa.2023.200253
9. *Bashab A., Ibrahim A. O., Tarigo Hashem I. A. et al.* Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues. Computers, Materials & Continua, 2023. Vol. 74. No. 3. P. 6461–6484. DOI: https://doi.org/10.32604/cmc.2023.034051
10. *Rezaeipanah A. et al.* A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Applied Intelligence, 2021. Vol. 51. P. 467–492. DOI: https://doi.org/10.1007/s10489-020-01833-x
11. *Mahlous A. R., Mahlous H.* Student timetabling genetic algorithm accounting for student preferences. PeerJ Computer Science, 2023. Vol. 9. Article e1200. DOI: https://doi.org/10.7717/peerj-cs.1200
12. *Davison M., Kheiri A., Zografos K. G.* Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling, 2024. Vol. 28. P. 195–215. DOI: https://doi.org/10.1007/s10951-024-00817-w
13. *Zanevych O., Kukharskyy V.* Overview of machine learning methods for academic scheduling. Electronics and Information Technologies, 2024. Vol. 27. DOI: https://doi.org/10.30970/eli.27.8
14. *Horne J. A., Ostberg O.* A self-assessment questionnaire to determine morningness-eveningness in human circadian rhythms. International Journal of Chronobiology, 1976. Vol. 4. No. 2. P. 97–110.
15. Eiben A. E., Smith J. E. *Introduction to Evolutionary Computing*, 2nd ed. Springer, Berlin, 2015. 302 p.
16. *Ситнік О., Вдовітченко О. Математичні структури та засоби антропоцентричної диспетчеризації у закладах вищої освіти.* Open Information and Computer Integrated Technologies, 2025. № 105. С. 212–226. DOI: https://doi.org/10.32620/oikit.2025.105.17
17. *Chen M. C. et al.* A survey of university course timetabling problem: Perspectives, trends and opportunities. IEEE Access, 2021. Vol. 9. P. 106515–106529. DOI: https://doi.org/10.1109/ACCESS.2021.3100613
18. *Gu X., Krish M., Sohail S. et al.* From integer programming to machine learning: A technical review on solving university timetabling problems. Computation, 2025. Vol. 13. No. 1. Art. 10. DOI: https://doi.org/10.3390/computation13010010
19. *Zanevych O. B., Kukharskyy V. M.* Solving university timetabling problems using constraint programming with adaptive local search and elite solution memory. Visnyk of the Lviv University. Series Applied Mathematics and Computer Science, 2025. Vol. 34. DOI: https://doi.org/10.30970/vam.2025.34.13638
20. *Носиков О., Ситнік О. Формування єдиного інформаційного простору в закладах вищої освіти: виклики, рішення та перспективи розвитку.* Open Information and Computer Integrated Technologies, 2025. № 104. С. 200–213. DOI: https://doi.org/10.32620/oikit.2025.104.13

References

1. Goldin, A. P. et al. (2020), Interplay of chronotype and school timing, *Nature Human Behaviour,* Vol. 4, No. 4, P. 387–396.
2. Jankowski, K. S., Díaz-Morales, J. F., Vollmer, C. (2023), Chronotype, time of day, and performance on intelligence tests in the school setting, *Journal of Intelligence,* Vol. 11, No. 1, Art. 13. DOI: https://doi.org/10.3390/jintelligence11010013
3. Sytnik, O. O. (2025), Anthropocentric model of class scheduling in HEI of aerospace profile, *Aerospace Technic and Technology,* No. 6 (215), P. 1–12. DOI: https://doi.org/10.32620/aktt.2025.6.07
4. Rodríguez Ferrante, G. et al. (2023), A better alignment between chronotype and school timing, *npj Science of Learning,* Vol. 8, Article 21.
5. Al-Rfooh, O. F., Khater, W. (2023), The impact of chronotype on physical health, psychological health, and job performance among health care providers, *International Journal of Healthcare Management,* Vol. 16, No. 4, P. 581–589. DOI: https://doi.org/10.1080/20479700.2023.2177665

6. Maučec, K., Štukovnik, V. (2024), The relationship between chronotype and academic achievement among Slovene university students, *Center for Educational Policy Studies Journal.* DOI: https://doi.org/10.26529/cepsj.1790
7. Smarr, B. L., Schirmer, A. E. (2018), 3.4 million real-world LMS logins reveal social jet lag, *Scientific Reports,* Article 4793.
8. Abdipoor, S., Yaakob, R., Goh, S. L., Abdullah, S. (2023), Meta-heuristic approaches for the university course timetabling problem, *Intelligent Systems with Applications,* Vol. 19, Art. 200253. DOI: https://doi.org/10.1016/j.iswa.2023.200253
9. Bashab, A., Ibrahim, A. O., Tarigo Hashem, I. A. et al. (2023), Optimization techniques in university timetabling problem, *Computers, Materials & Continua,* Vol. 74, No. 3, P. 6461–6484. DOI: https://doi.org/10.32604/cmc.2023.034051
10. Rezaeipanah, A. et al. (2021), A hybrid algorithm for the university course timetabling problem, *Applied Intelligence,* Vol. 51, P. 467–492. DOI: https://doi.org/10.1007/s10489-020-01833-x
11. Mahlous, A. R., Mahlous, H. (2023), Student timetabling genetic algorithm accounting for preferences, *PeerJ Computer Science,* Vol. 9, Article e1200. DOI: https://doi.org/10.7717/peerj-cs.1200
12. Davison, M., Kheiri, A., Zografos, K. G. (2024), Modelling and solving the university course timetabling problem with hybrid teaching considerations, *Journal of Scheduling,* Vol. 28, P. 195–215. DOI: https://doi.org/10.1007/s10951-024-00817-w
13. Zanevych, O., Kukharskyy, V. (2024), Overview of machine learning methods for academic scheduling, *Electronics and Information Technologies,* Vol. 27. DOI: https://doi.org/10.30970/eli.27.8
14. Horne, J. A., Ostberg, O. (1976), A self-assessment questionnaire to determine morningness-eveningness, *International Journal of Chronobiology,* Vol. 4, No. 2, P. 97–110.
15. Eiben, A. E., Smith, J. E. (2015), *Introduction to Evolutionary Computing,* 2nd ed., Springer, Berlin, 302 p.
16. Sytnik, O., Vdovitchenko, O. (2025), Mathematical structures and tools for anthropocentric dispatching in higher education institutions, *Open Information and Computer Integrated Technologies,* No. 105, P. 212–226. DOI: https://doi.org/10.32620/oikit.2025.105.17
17. Chen, M. C. et al. (2021), A survey of university course timetabling problem: Perspectives, trends and opportunities, *IEEE Access,* Vol. 9, P. 106515–106529. DOI: https://doi.org/10.1109/ACCESS.2021.3100613
18. Gu, X., Krish, M., Sohail, S. et al. (2025), From integer programming to machine learning: A technical review on solving university timetabling problems, *Computation,* Vol. 13, No. 1, Art. 10. DOI: https://doi.org/10.3390/computation13010010
19. Zanevych, O. B., Kukharskyy, V. M. (2025), Solving university timetabling problems using constraint programming with adaptive local search and elite solution memory, *Visnyk of the Lviv University. Series Applied Mathematics and Computer Science,* Vol. 34. DOI: https://doi.org/10.30970/vam.2025.34.13638
20. Nosikov, O., Sytnik, O. (2025), Formation of a unified information space in higher education institutions, *Open Information and Computer Integrated Technologies,* No. 104, P. 200–213. DOI: https://doi.org/10.32620/oikit.2025.104.13

Дата першого надходження до видання: 06.04.2026
Внутрішня рецензія отримана: 23.04.2026
Зовнішня рецензія отримана: 29.04.2026
Дата прийняття статті до друку: 05.06.2026
Дата публікації: 29.06.2026

Про авторів:

Ситнік Олег Олександрович,
аспірант кафедри інженерії
програмного забезпечення
Sytnik Oleg,
Post-graduate student
ORCID: 0009-0009-4504-3489.

Місце роботи авторів:

Національний аерокосмічний університет
«Харківський авіаційний інститут»,
National Aerospace University
"Kharkiv Aviation Institute"
вул. Чкалова, 17, Харків, Україна, 61070.
E-mail: o.sytnik@khai.edu