УДК 004.04 DOI https://doi.org/10.35546/kntu2078-4481.2025.4.2.23

С. М. ШЕВЧЕНКО
кандидат педагогічних наук, доцент,
доцент кафедри інформаційної та кібернетичної безпеки
імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка
ORCID: 0000-0002-9736-8623

Ю. Д. ЖДАНОВА
кандидат фізико-математичних наук, доцент,
доцент кафедри інформаційної та кібернетичної безпеки
імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка
ORCID: 0000-0002-9277-4972

О. В. КОЧЕТКОВ
магістрант кафедри інженерії програмного забезпечення
Державний університет інформаційно-комунікаційних технологій
ORCID: 0009-0000-6678-6884

БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ ДЛЯ ВИБОРУ НАЙКРАЩОЇ КОНФІГУРАЦІЇ КОМП'ЮТЕРНИХ СИСТЕМ

Самостійна збірка комп'ютера є виправданою та вигідною стратегією для багатьох користувачів, дає значні переваги перед купівлею готової системи. Зокрема, це найкраща результативність витрат та економія, ідеальна адаптація до потреб користувача, можливість самому модернізувати, відремонтувати, здійснювати контроль якості комплектуючих. Допомогти самостійно зібрати комп'ютер можуть спеціалізовані рекомендаційні системи – онлайн-конфігуратори, які максимально дозволяють спростити та убезпечити процес підбору комплектуючих.

Дана стаття присвячена розробці гібридної рекомендаційної системи, алгоритм роботи якої поєднує методи обробки природної мови, евристичний генетичний пошук та генеративні можливості нейромережевої моделі.

На основі аналізу науково-технічної літератури формалізована задача підбору комплектуючих як задача багатокритеріальної оптимізації. Розв'язання даної задачі представлено за допомогою генетичного алгоритму. Завдяки генетичному алгоритму система здатна знайти близькі до оптимальних конфігурації із великого комбінаторного простору варіантів, а використання ChatGPT для генерації пояснень підвищує якість інтерпретації результатів для кінцевого користувача.

У дослідженні представлена архітектура запропонованої методики автоматизованого підбору комплектуючих для комп'ютерних систем, яка складається з декількох взаємодіючих компонентів, а саме, користувацький інтерфейс, модуль обробки запитів природною мовою, оптимізаційний модуль (генетичний алгоритм), модуль генерації відповіді (на базі ChatGPT) та реляційна база даних компонентів.

У статті подано основні кроки алгоритму конфігуратора, в кожному з яких описано початкові дані на вході і дані на виході.

Результати дослідження можуть бути використані закладами вищої освіти та навчальними центрами для модернізації навчальних програм та підвищення якості підготовки фахівців у сфері програмної інженерії.

Ключові слова: *генетичні алгоритми, багатокритеріальна оптимізація, штучний інтелект, конфігуратор, автоматизований підбір комплектуючих, бюджетні обмеження; сумісність.*

S. M. SHEVCHENKO
Candidate of Pedagogical Sciences, Associate Professor,
Associate Professor at the Department of Information and Cyber Security
named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University
ORCID: 0000-0002-9736-8623

YU. D. ZHDANOVA
Candidate of Physical and Mathematical Sciences, Associate Professor,
Associate Professor at the Department of Information and Cyber Security
named after Professor Volodymyr Buriachok
Borys Grinchenko Kyiv Metropolitan University
ORCID: 0000-0002-9277-4972

O. V. KOCHETKOV
Master's Student at the Department of Software Engineering
State University of Information and Communication Technologies
ORCID: 0009-0000-6678-6884

MULTI-CRITERIA OPTIMIZATION FOR SELECTING THE BEST CONFIGURATION OF COMPUTER SYSTEMS

Self-assembly of a computer is a justified and profitable strategy for many users, it provides significant advantages over purchasing a ready-made system. In particular, it is the best cost-effectiveness and savings, perfect adaptation to the user's needs, the ability to upgrade, repair, and control the quality of components yourself. Specialized recommendation systems – online configurators, which allow you to simplify and secure the process of selecting components as much as possible, can help you assemble a computer yourself.

This article is devoted to the development of a hybrid recommendation system, the algorithm of which combines natural language processing methods, heuristic genetic search, and generative capabilities of a neural network model.

Based on the analysis of scientific and technical literature, the problem of component selection is formalized as a multi-criteria optimization problem. The solution of this problem is presented using a genetic algorithm. Thanks to the genetic algorithm, the system is able to find close to optimal configurations from a large combinatorial space of options, and the use of ChatGPT to generate explanations improves the quality of interpretation of the results for the end user.

The study presents the architecture of the proposed method for automated selection of components for computer systems, which consists of several interacting components, namely, a user interface, a natural language query processing module, an optimization module (genetic algorithm), a response generation module (based on ChatGPT), and a relational database of components.

The article presents the main steps of the configurator algorithm, each of which describes the initial input data and output data.

The results of the study can be used by higher education institutions and training centers to modernize curricula and improve the quality of training of specialists in the field of software engineering.

Key words: *genetic algorithms, multi-criteria optimization, artificial intelligence, configurator, automated selection of components, budget constraints; compatibility.*

Постановка проблеми

Для більшості користувачів персональний комп'ютер (ПК) є інструментом для виконання відповідних завдань: професійна діяльність, ігри, навчання тощо. Очевидно, що кожен ПК так чи інакше має бути підлаштованим під вимоги власника. Людина, яка не хоче перейматися вибором компонентів, купує готовий продукт, інші користувачі обирають системи, зібрані за їх індивідуальним замовленням, що є бюджетно накладним. Тому для тих, хто є професіоналом у цій сфері, найкращий спосіб отримати комп'ютер під свої високі вимоги з мінімальним бюджетом є самостійна збірка компонентів.

Проте сьогодні цей процес є набагато доступнішим завдяки існуванню спеціалізованих рекомендаційних систем, що дозволить недосвідченим користувачам у цій сфері отримати саме той ПК, який відповідає їхнім бажанням.

Аналіз останніх досліджень і публікацій

Сучасний світ характеризується інтенсивним розвитком інформаційних технологій, що дозволяє автоматизувати процеси у різних сферах людської діяльності. Тому питання оптимізації конфігурації комп'ютерних систем набуває важливого значення. Необхідність узгодження широкого спектра доступних апаратних компонентів із фінансовими лімітами вимагає розробки інтелектуальних рішень, які дозволяють модернізувати комп'ютерну систему до вимог користувача. Останній може бути і не професіоналом у цій сфері, що робить такі системи надзвичайно затребуваними.

У цьому руслі цікава розробка [1], автори якої представили систему рекомендацій для пошуку компонентів ПК. В основі цієї технології лежить генетичний алгоритм. Його було обрано через здатність ефективно досліджувати великий простір можливих конфігурацій за допомогою операторів кросоверу та мутації, генеруючи різноманітні альтернативи. Такий підхід переважає над простими жадібними алгоритмами, що можуть давати субоптимальні рішення, і є значно швидшим за повний перебір усіх комбінацій.

Слід відзначити дослідження [2], автори якого запропонували систему рекомендацій щодо конфігурації компонентів ПК у вигляді вебдодатку. У даній технології використовували колаборативну фільтрацію.

Нам імпонує практична розробка системи аналізу комп'ютерних компонентів для удосконалення проєктування ПК, їх повного аналізу та оптимізації інформаційно-технічної допомого користувачам на основі алгоритму гібридної рекомендаційної системи авторів [3]. Вони вважають, що для рекомендацій краще використовувати груповий підхід, а не індивідуальний, а для пошуку груп використали змішаний категоріально-числовий метод кластеризації та гібридний метод на основі коефіцієнта розрідженості.

Аналіз інформаційних джерел дозволив виділити як світові, так і локальні (українські) сервіси, які допомагають користувачам у збірці ПК. У таблиці 1 представлені деякі з них та їх характеристики.

Таблиця 1.

Порівняльний аналіз сучасних конфігураторів

Метод	**Класичний ручний підбір**	**Онлайн-конфігуратори інтернет-магазинів**	**Спеціалізовані калькулятори (типу be quiet! PSU Calculator) [4]**	**Пошук через прайс-агрегатори та фільтри**
Принцип роботи	Користувач самостійно підбирає комплектуючі за оглядами, форумами, порадами знайомих, порівнює характеристики та ціни на різних сайтах.	Користувач поетапно обирає компоненти з каталогу магазину; система частково відсікає несумісні варіанти та рахує загальну вартість збірки.	Користувач вводить вже сформовану або бажану конфігурацію, сервіс оцінює споживання енергії та підбирає відповідний блок живлення з певного модельного ряду.	Користувач задає діапазон цін і базові параметри (тип сокета, обсяг пам'яті тощо) в каталозі, а потім вручну комбінує вибрані деталі у збірку.
Ключові недоліки	Велика витрата часу; потрібні глибокі технічні знання; високий ризик помилок сумісності; відсутня автоматична оптимізація під бюджет та наявну конфігурацію.	Прив'язані до асортименту одного магазину; обмежена перевірка сумісності; зазвичай немає режиму «апгрейд існуючого ПК»; немає інтелектуальної оптимізації продуктивність/ціна, лише підрахунок суми.	Розв'язує лише вузьку задачу вибору блока живлення; не підбирає інші компоненти; не враховує бюджет та актуальні ціни; не пропонує альтернативних збалансованих конфігурацій.	Кожен компонент підбирається окремо; сумісність перевіряється вручну; складно отримати цілісний «оптимальний набір»; важко збалансувати продуктивність і вкластися в бюджет.

Актуальність даної проблеми, її доцільність є обґрунтуванням даного дослідження, присвяченого багатокритеріальній оптимізації для вибору найкращої конфігурації компонентів в персональному комп'ютері.

Формулювання мети дослідження

Окреслене визначило мету нашого дослідження – формалізувати задачу підбору конфігурації як задачу багатокритеріальної оптимізації та представити гібридний алгоритм такого конфігуратора з використанням методів NLP, що дозволить удосконалити процес збірки ПК самостійним чином.

Викладення основного матеріалу дослідження

1. Формалізація задачі підбору конфігурації

Формалізацію задачі підбору конфігурації компонентів в ПК розглянемо як задачу багатокритеріальної оптимізації. Багатокритеріальна конфігурація – це процес одночасної оптимізації двох або більше конфліктуючих цільових функцій в заданій області [5, 6]. Постановка задачі оптимізації конфігурації ПК виглядає наступним чином.

1. Визначення множин та змінних

C – множина типів компонентів, необхідних для збірки ПК. Наприклад,

$$C = \{\text{CPU}, \text{GPU}, ...\}$$

A_c – множина всіх доступних артикулів моделей для типу $c \in C$. Наприклад,

$$A_{\text{CPU}} = \{\text{Intel i5} - 14600K, ...\}$$

$x_{c,a} = \begin{cases} 1, \textit{якщо обрано модель } a \in A_c, \\ 0, \text{в протилежному випадку,} \end{cases}$ – бінарна змінна рішення.

Наприклад, $x_{\text{CPU,i5-14600}K}$.

2. Характеристики компонентів

$P_{c,a}$ – продуктивність моделі a типу c,

$V_{c,a}$ – вартість моделі a типу c,

$W_{c,a}$ – енергоспоживання (теплової енергії) моделі a типу c,

$S_{c,a}$ – характеристики суміжності

3. Цільова функція (фітнес-функція)

$$\max \sum_{c \in C} \sum_{a \in A_c} \omega_c \cdot P_{c,a} \cdot x_{c,a},$$

де ω_c – ваговий коефіцієнт, що відображає важливість компонента типу c для фінальної продуктивності, знайдене експертним методом.

4. Обмеження

Обмеження вибору: для кожного типу $c \in C$ можна вибрати лише одну модель.

$$\sum_{a \in A_c} x_{c,a} = 1, \forall c \in C$$

Обмеження бюджету: загальна вартість компонентів не повинна перевищувати заданий грошовий поріг.

$$\sum_{c \in C} \sum_{a \in A_c} V_{c,a} \cdot x_{c,a} \le B_{\max}.$$

Обмеження сумісності: усі обрані S моделі мають бути сумісні. Використовується бінарна деривація $S_c(x)$, яка дорівнює 1 якщо обмеження порушено, 0 – у протилежному випадку.

Обмеження потужності: сумарне енергоспоживання обраних ключових компонентів не повинно перевищувати потужність обраного блоку живлення з певним запасом Δ Δ.

$$\sum_{a \in A_{\text{Core}}} W_{c,a} \cdot x_{c,a} + \sum_{b \in A_{\text{Periph}}} W_{b,a} \cdot x_{b,a} \le \sum_{d \in A_{\text{PSU}}} (W_{\text{PSU},d} - \Delta) \cdot x_{\text{PSU},d}.$$

2. Генетичні алгоритми

Розв’язання задачі багатокритеріальної оптимізації з використанням генетичних алгоритмів вперше запропонував Річард Розенберг [7, 8].

Генетичні алгоритми були відкриті Джоном Голландом у 1960-х роках [9]. Науковець ставив за мету формалізувати феномен адаптації у природі та імпортувати ці механізми природної адаптації в комп'ютерні системи. Генетичний алгоритм працює шляхом ініціалізації популяції випадкових рішень, які потім ітераційно оцінюються за пристосованістю, а найкращі рішення відбираються та комбінуються за допомогою кросовера та мутації для створення кращого наступного покоління [1,7 – 14].

Генетичний алгоритм – ітераційний метод, узагальнена блок-схема якого представлена на рис. 1 [11, 13].

Завдяки ефективній паралелізації еволюційних алгоритмів на сучасних обчислювальних платформах, генетичні алгоритми з великою кількістю індивідів та епох навчання можуть швидко знаходити якісні рішення для широкого спектру задач, включаючи виявлення шаблонів, обробку сигналів та навчання нейронних мереж тощо [14].

Застосування генетичного алгоритму у даному дослідженні представлено нижче.

3. Логічна структура системи

Архітектура запропонованої методики автоматизованого підбору комплектуючих для комп’ютерних систем складається з декількох взаємодіючих компонентів. Основними модулями є: користувацький інтерфейс, модуль обробки запитів природною мовою, оптимізаційний модуль (генетичний алгоритм), модуль генерації відповіді (на базі ChatGPT) та реляційна база даних компонентів. Кожен з цих компонентів відіграє чітко визначену роль у загальній системі, забезпечуючи узгоджену роботу від моменту введення запиту користувача до формування рекомендації. Розгляньмо сутнісні характеристики кожного компонента більш детально.

Користувацький інтерфейс (веб-застосунок) – реалізований на основі Flask (Python) сервер, забезпечує взаємодію із користувачем. Він приймає запит у вигляді природно-мовного тексту (наприклад, користувач вводить: «Я хочу зібрати ПК для ігор з бюджетом до 30000 грн»), передає його в систему та відображає отриману відповідь. Інтерфейс є точкою доступу до функціоналу конфігуратора, надаючи зручну форму введення вимог та отримання результатів.

Модуль обробки запитів – відповідає за Natural Language Processing (NLP) запиту, тобто розбір тексту користувача і виділення ключових параметрів. Цей модуль аналізує вхідне речення, визначаючи тип бажаного використання системи (ігровий, офісний, для дизайну тощо), бюджетні обмеження (сума в гривнях) та особливі умови (які компоненти потрібно залишити або замінити). Для цього можуть застосовуватися як прості лінгвістичні правила (пошук ключових слів: «ігор», «офісний», розпізнавання числових даних і валюти), так і спеціалізовані моделі обробки мови. В результаті модуль формує структуру запиту: наприклад, із фрази «Мені потрібен офісний комп’ютер, залишаючи SSD і БЖ зі старої збірки» буде отримано, що користувач потребує конфігурацію для офісної роботи, явного бюджету не зазначено, але слід залишити наявний SSD-накопичувач та блок живлення (БЖ) з попередньої системи. Ці виділені параметри передаються на наступні етапи системи.

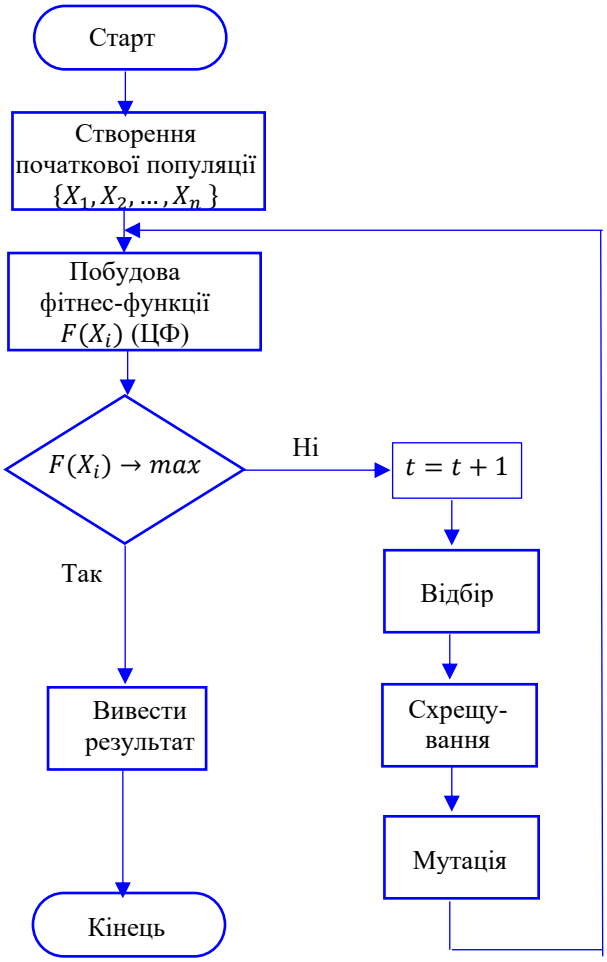

Рис. 1. Блок-схема генетичного алгоритму

База даних компонентів – реляційна база даних (SQLite) містить інформацію про апаратні комплектуючі. Дані організовані за категоріями компонентів: процесори, материнки, модулі пам'яті, графічні карти, накопичувачі, блоки живлення тощо. Для кожного компоненту зберігаються опис, технічні характеристики (наприклад, сокет CPU, чипсет материнської плати, тип і обсяг пам'яті, потужність БЖ), поточна ціна та відомості про сумісність. Сумісність може бути закодована через атрибути (наприклад, процесор має поле «Socket», яке повинно співпасти з полем «Socket» материнської плати; форм-фактор оперативної пам'яті DDR4/DDR5 повинен підтримуватися материнською платою тощо). База даних слугує знаннями для системи: оптимізаційний модуль звертається до неї, щоб отримати списки доступних компонентів та їх параметри, а модуль відповіді може використовувати описи при формуванні рекомендації.

Оптимізаційний модуль (генетичний алгоритм) – ядро системи, яке виконує пошук оптимальної конфігурації під задані критерії. Він отримує від модуля обробки запиту визначені вимоги (тип використання, бюджет, фіксовані чи виключені компоненти) та звертається до бази даних за переліком доступних компонентів. Цей модуль реалізує генетичний алгоритм (ГА) для комбінаторної оптимізації вибору компонентів. Оптимізаційний модуль враховує обмеження сумісності і бюджету при формуванні нових рішень: якщо користувач позначив деякі компоненти як такі, що треба залишити, алгоритм фіксує їх у конфігурації і не замінює. Решта компонентів добираються алгоритмом із урахуванням технічної сумісності (забороняються несумісні за сокетом або інтерфейсами поєднання) та таким чином, щоб сумарна вартість не перевищувала бюджет.

Модуль оцінки та вибору – підмодуль всередині оптимізаційного блока, який відповідає за обчислення функції пристосованості (fitness function) для кожної згенерованої конфігурації та вибір найкращих рішень. Для оцінки конфігурації можуть застосовуватися кількісні критерії, що відбивають продуктивність системи для заданого типу використання. Зокрема, кожному компоненту може бути приписаний показник продуктивності (наприклад, бал бенчмарку або відносна потужність), і на основі цього обчислюється сумарний рейтинг продуктивності системи. Щоб врахувати специфіку використання, вводяться вагові коефіцієнти: наприклад, для ігрового ПК внесок

потужності GPU (відеокарти) у рейтинг більший, ніж для офісного ПК, де важливіше швидкодія CPU та накопичувача. Нейромережева модель (за наявності) може бути інтегрована на цьому етапі для оцінки сумісності та продуктивності складних комбінацій компонентів. Зокрема, нейронна мережа може прогнозувати узгодженість роботи компонентів (виявлення "вузьких місць", коли, наприклад, надто потужна відеокарта не розкривається через слабкий CPU) та давати інтегральний показник ефективності системи. Враховуючи такі оцінки, генетичний алгоритм відбирає найбільш перспективні рішення за допомогою оператора відбору (наприклад, турнірного відбору) для подальшого схрещування.

Модуль генерації відповіді (з використанням ChatGPT) – завершує цикл роботи системи, формуючи кінцеву рекомендацію у вигляді текстового висновку для користувача. Після того, як оптимізаційний модуль визначив одну або кілька оптимальних конфігурацій, вибрана конфігурація передається в модуль генерації відповіді. Цей модуль використовує можливості великої мовної моделі ChatGPT для аналізу отриманої конфігурації та пояснення вибору компонентів. ChatGPT (Generative Pre-trained Transformer від OpenAI) – це сучасна нейромережева модель, здатна генерувати зв'язний і контекстно обґрунтований текст на основі вхідних даних [15]. У нашій системі вона слугує для забезпечення кращого пояснення та обґрунтування причин рекомендацій: модель аналізує, як підібрані комплектуючі задовольняють вимоги користувача, та формує роз'яснення зрозумілою мовою. Зокрема, генерується опис того, чому обрано саме ці компоненти, як вони забезпечують потрібний рівень продуктивності, чи не виходить за бюджет, які переваги надає збереження певних старих комплектуючих тощо. Результат роботи цього модуля – детальна текстова рекомендація, яка включає список підібраних компонентів та обґрунтування вибору – надсилається назад до користувацького інтерфейсу для відображення користувачу.

Взаємодія компонентів – усі зазначені модулі інтегровані в єдину систему через серверну логіку Flask. Запит від інтерфейсу послідовно проходить через модуль обробки мови до оптимізаційного модуля; останній здійснює пошук у просторі даних, що зберігаються в базі, і обирає оптимальне рішення; далі модуль генерації відповіді формує людинозрозумілий висновок. Така багаторівнева архітектура забезпечує гнучкість: можна оновлювати базу даних новими компонентами без зміни алгоритму, вдосконалювати NLP-модуль або підключати більш потужні методи оптимізації чи нейромережеві моделі без перегляду роботи інших компонентів. Система модульна, що полегшує її підтримку та подальший розвиток. На рис. 2 представлена архітектура системи та взаємодія її компонентів.

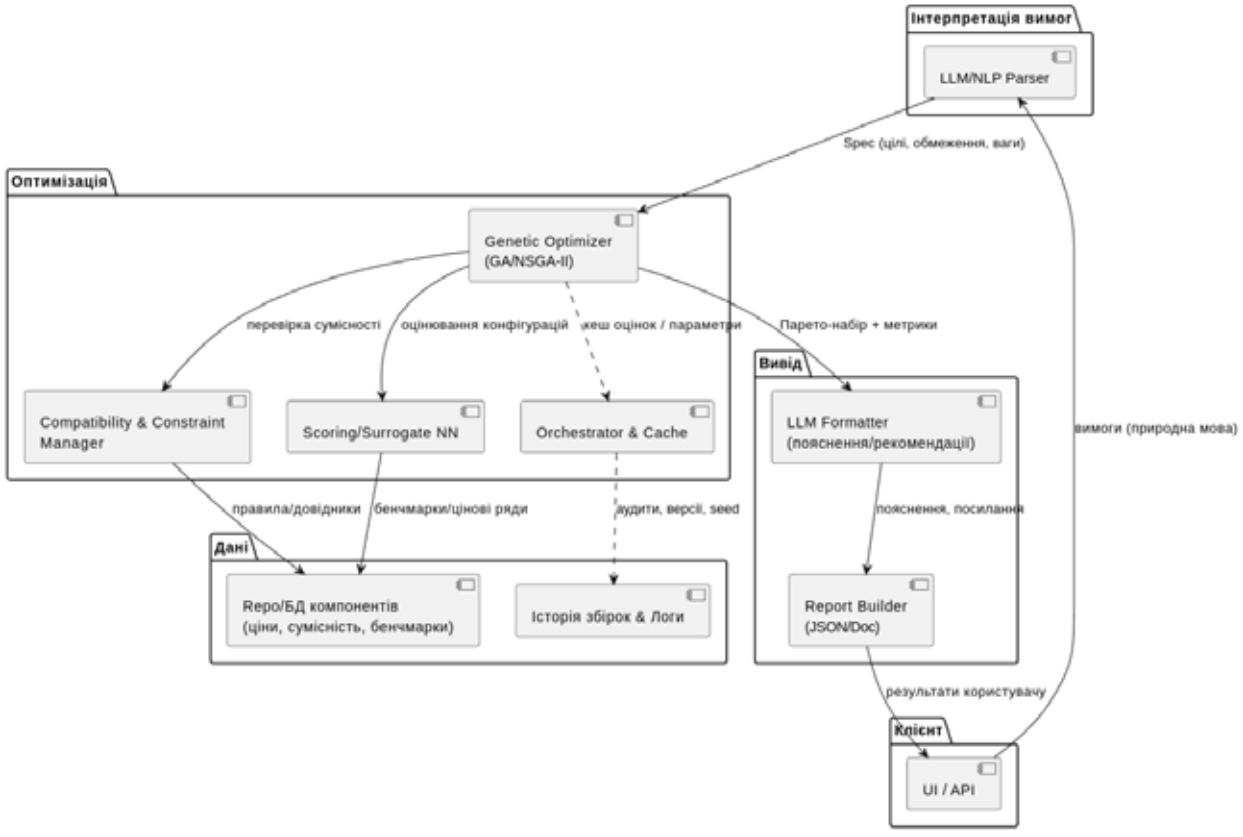

Рис. 2. Архітектура методики автоматизованого підбору комплектуючих для комп'ютерних систем

4. Алгоритм роботи конфігуратора

Алгоритм функціонування конфігуратора є послідовністю кроків, що трансформують початковий текстовий запит у конкретну рекомендацію з поясненнями. Нижче подано детальний покроковий опис цього процесу – від введення користувачем своїх вимог до отримання згенерованої відповіді:

1. Введення запиту користувача

Користувач формулює запит у довільній формі природною мовою через вебінтерфейс. Запит може містити бажаний сценарій використання, орієнтовний бюджет та додаткові умови щодо компонентів. Наприклад, запит типу «Я хочу зібрати ПК для ігор з бюджетом до 30000 грн» вказує на потребу в ігровому комп'ютері з обмеженням вартості ~30000 грн, тоді як фраза «Мені потрібен офісний комп'ютер, залишаючи SSD і БЖ зі старої збірки» задає офісний профіль системи, без прямого зазначення бюджету, але з вимогою повторно використати наявний SSD-накопичувач та блок живлення. Користувацький інтерфейс передає текст запиту до серверної частини для обробки.

2. Аналіз і розбір запиту (NLP)

На цьому етапі модуль обробки природної мови аналізує текст запиту. Застосовуються методи NLP для вилучення структурованих параметрів:

1) тип використання: класифікується на основі ключових слів (наприклад, «для ігор» → ігровий, «офісний комп'ютер» → офісний). Можливі категорії включають ігровий, офісний, графічний дизайн, серверний тощо;

2) бюджет: розпізнаються числові значення та валюта. Якщо бюджет явно не вказано, система може використовувати дефолтні значення або запитати уточнення. У випадку з прикладами вище, перший запит дає бюджет 30000 грн, а другий – бюджет не задано (система може припустити мінімально необхідний або середній бюджет для офісного ПК);

3) фіксовані/старі компоненти: виявляються згадки про існуючі комплектуючі, які слід залишити («SSD», «БЖ» тощо) чи замінити. Система ідентифікує типи цих компонентів (накопичувач, блок живлення) і помічає їх як такі, що не потребують підбору нових аналогів;

4) інші обмеження або побажання: наприклад, уподобання брендів чи технологій (типу «віддаю перевагу CPU Intel» або «хочу GPU з підтримкою RTX») – такі деталі також можуть бути враховані, якщо система їх розпізнає.

Результатом цього кроку є набір структурованих даних про запит: {тип: «офісний», бюджет: null, залишити: [«SSD», «БЖ»], …} для другого прикладу. Ці дані передаються далі в алгоритм.

3. Підготовка даних та початковий пошук компонентів

Оптимізаційний модуль отримує структурований запит і звертається до бази даних за списками доступних компонентів кожного типу. На цьому кроці можуть застосовуватися попередні фільтри:

1) виключаються категорії компонентів, які користувач планує залишити зі старої системи (наприклад, якщо вирішено залишити SSD і БЖ, то зі списків нових компонентів вилучаються всі накопичувачі та блоки живлення – оскільки їх купувати не потрібно);

2) можуть застосовуватися обмеження мінімальних вимог відповідно до типу використання: наприклад, для ігрового ПК система може відфільтрувати лише відеокарти не нижче середнього рівня та CPU з ≥4 ядрами, щоб прискорити пошук рішень;

3) якщо задано бюджет, можливо виключити компоненти, ціна яких явно не вписується (надто дорогі компоненти можуть відсіюватися, щоб не витрачати час на явно неприпустимі комбінації);

4) виконується перевірка базової сумісності: наприклад, якщо користувач залишає стару материнську плату, вже на цьому етапі можна обмежити вибір процесорів лише тими, що підтримують відповідний сокет, і навпаки.

Таким чином, формується початковий пул сумісних компонентів-кандидатів по кожній необхідній категорії (CPU, GPU, RAM, тощо), який буде використовувати генетичний алгоритм.

4. Ініціалізація генетичного алгоритму

Генетичний алгоритм починає роботу зі створення початкової популяції конфігурацій. Кожна потенційна конфігурація може бути подана у вигляді хромосоми – наприклад, як набір генів, що кодують вибір конкретних компонентів (ID процесора, ID материнської плати, ID GPU, тощо). Спочатку генерується декілька десятків або сотень випадкових конфігурацій, які відповідають базовим обмеженням:

1) кожна конфігурація містить по одному компоненту з кожної необхідної категорії (окрім тих, що фіксовані користувачем). Наприклад, якщо потрібно підібрати CPU, материнську плату, ОЗП, відеокарту і корпус (решту залишають старими), то ген в цих позиціях випадково обирається з відповідного списку доступних компонентів;

2) випадкове генерування відбувається з перевіркою сумісності: якщо згенерована комбінація містить несумісну пару (скажімо, CPU з сокетом AM5 та материнку під LGA1700), така конфігурація відбраковується або коригується. Таким чином, початкова популяція вже складається з фізично здійсненних варіантів збірки;

3) переконуємося, що жодна конфігурація не перевищує бюджет (якщо деякі випадкові комбінації виходять за межі бюджету, вони або відхиляються, або включаються з позначкою штрафу за перевищення);

4) оцінка пристосованості (fitness) конфігурацій: для кожної конфігурації в популяції обчислюється значення цільової функції – показник, який відображає, наскільки добре дана комбінація відповідає вимогам користувача. Цей показник формується на основі:

– продуктивності: агрегований бал продуктивності системи може розраховуватися з використанням технічних параметрів і вагових коефіцієнтів для кожного компонента, знайдених за допомогою експертного методу. Наприклад, для CPU – бали тестів (benchmark score), для GPU – продуктивність у 3D-графіці, для дисків – швидкість I/O, зведені до спільної шкали. Далі для ігрового ПК сумарний бал: 0.5CPU_score + 0.3GPU_score + 0.1RAM_score + 0.1Disk_score (умовно), тоді як для офісного може бути 0.4CPU + 0.1GPU + 0.3RAM + 0.2Disk. Ці ваги відображають важливість компонентів під конкретні завдання;

– відповідності бюджету: якщо конфігурація не перевищує бюджет, вона отримує повний бал за цим критерієм; якщо ж перевищує (у випадку відсутності явного відсіювання), може застосовуватися штраф або нульова придатність, щоб такі рішення не еволюціонували далі. У разі відсутності заданого бюджету, критерій може бути мінімізувати вартість або співвідносити продуктивність з ціною;

– сумісності та збалансованості: конфігурація отримує додаткові бали або штрафи за більш збалансований підбір компонентів. Наприклад, якщо всі компоненти сумісні (що обов'язково), а також добре узгоджені за класом – немає явних "перекосів" (типовий приклад перекосу: надпотужний GPU в парі зі слабким CPU, що призведе до недовикористання потенціалу GPU). Для кількісної оцінки такого балансу може використовуватися окрема модель. Нейронна мережа, навчена на даних про різні збірки, може передбачати коефіцієнт узгодженості системи, який враховується в fitness-функції. Така модель на вході отримує параметри вибраних CPU, GPU, ОЗП тощо, і видає оцінку, наскільки ефективно вони працюватимуть разом. Ця оцінка додається до загальної пристосованості конфігурації.

У підсумку, кожна конфігурація має чисельний рейтинг придатності. Конфігурації, що не відповідають критичним обмеженням (несумісні чи значно перевищують бюджет), отримують вкрай низький рейтинг або вилучаються з популяції.

5. Генетична еволюція рішень

Після оцінки поточної популяції застосовується власне ітеративний процес генетичного алгоритму. Відповідно до обчислених fitness-значень виконується відбір батьківських конфігурацій – кращі рішення мають вищі шанси бути обраними для породження нового покоління. Використовується, метод турнірного відбору чи рулетки. Далі здійснюються генетичні оператори:

1) кросовер (схрещування): випадковим чином вибираються одна або кілька позицій у хромосомі, і дві батьківські конфігурації обмінюються частиною компонентів. Це дає нащадків – нові конфігурації, що комбінують риси обох батьків. Наприклад, від одного батька успадковується CPU і материнська плата, від іншого – GPU і решта компонентів, утворюючи змішану конфігурацію. При цьому одразу перевіряється сумісність – якщо в результаті кросоверу отримано невалідну комбінацію, її можна або відкинути, або виправити (наприклад, заміною несумісного підкомпоненту випадковим сумісним);

2) мутація: до деяких нових конфігурацій застосовуються випадкові незначні зміни, наприклад, один з компонентів замінюється іншим випадковим із того ж класу. Мутація допомагає дослідити простір рішень більш широко і запобігти локальним екстремумам. Її ймовірність зазвичай низька (кілька відсотків);

3) формування нового покоління: із батьківських та нових рішень відбирається наступна популяція фіксованого розміру (наприклад, 50 найкращих конфігурацій з урахуванням як старих, так і новостворених). Генерація за генерацією, середній рівень пристосованості популяції зростає, конфігурації стають дедалі кращими з точки зору критеріїв.

4) критерії зупинки: алгоритм повторює кроки оцінки → відбору → кросоверу/мутації, поки не виконається умова зупинки. Такою умовою може бути досягнення визначеної кількості поколінь, відсутність приросту fitness у найкращих рішень протягом кількох ітерацій або часове обмеження. На практиці генетичний алгоритм здатний за помірне число поколінь знайти дуже наближене до оптимального рішення конфігурації ПК [1];

5) вибір фінальної конфігурації: після завершення еволюційного процесу алгоритм має набір останньої популяції, де конфігурації відсортовані за придатністю. Найкраща конфігурація вилучається як результат оптимізації. Це конкретний набір компонентів, що максимально відповідає вимогам, наприклад, для ігрового ПК за 30000 грн – певний CPU, відповідна материнка, потужна GPU, достатній обсяг ОЗП, тощо, причому сумарна ціна укладається в 30000 грн). В цю конфігурацію вже інтегровані всі компоненти, які користувач забажав залишити: наприклад, старий SSD і БЖ додаються до списку, і система перевіряє, чи вибрані нові частини сумісні з ними;

6. Генерація пояснення та рекомендацій (ChatGPT)

Обрану конфігурацію разом з інформацією про запит (тип використання, бюджет, залишені компоненти) передають у модуль генерації відповіді. На цьому кроці задіюється модель ChatGPT, яка формує розгорнутий рекомендаційний висновок. Задля цього в модель надсилається продуманий промпт: перелік компонентів з назвами, ключовими характеристиками та цінами, а також контекст запиту користувача. ChatGPT аналізує, як ця підібрана система відповідає потребам. Результатом є текст рекомендації, що включає:

1) персоналізоване привітання або звернення до користувача (на кшталт «Виходячи з ваших потреб, оптимальною є така конфігурація…»);

2) перелік нових компонентів, запропонованих до купівлі, із поясненням кожного вибору. Наприклад, «Процесор Intel Core i5-13400F обрано через його високу продуктивність в іграх за розумною ціною, він забезпечить стабільний FPS і не перевищує бюджет»;

3) згадку про залишені компоненти, наприклад, «Ваш існуючий SSD SATA на 500 ГБ буде використано для зберігання даних, що зекономить кошти, а блок живлення 600W достатній для нової конфігурації, тому його теж залишаємо»;

4) загальне обґрунтування відповідності бюджету, наприклад, «Загальна вартість усіх нових компонентів становить ~29800 грн, що не перевищує заданий бюджет»;

5) поради на майбутнє або альтернативи, наприклад, «Якщо ви захочете підвищити продуктивність у майбутньому, ця система підтримує до 64 ГБ ОЗП, тож ви зможете додати модулі пам'яті».

7. Виведення результату користувачу

Сформований текст рекомендації разом зі структурованим списком компонентів повертається через Flask-сервер на фронтенд і відображається користувачу. У підсумковому вигляді користувач бачить пропоновану конфігурацію ПК (перелік комплектуючих із зазначенням моделей та цін) і детальний коментар, чому саме ці комплектуючі було обрано і як вони задовольняють його запит. За потреби користувач може відредагувати свої вимоги і запустити цикл підбору знову (наприклад, змінити бюджет або вказати додаткові побажання), оскільки система побудована як інтерактивний конфігуратор.

Таким чином, алгоритм роботи конфігуратора поєднує методи обробки природної мови, евристичний генетичний пошук та генеративні можливості нейромережевої моделі. Кожен етап – від розуміння запиту до оптимізації конфігурації і пояснення результату – забезпечує врахування відповідних аспектів задачі. Завдяки генетичному алгоритму система здатна знайти близькі до оптимальних конфігурації із великого комбінаторного простору варіантів, а використання ChatGPT для генерації пояснень підвищує якість інтерпретації результатів для кінцевого користувача. Це робить розроблену методику ефективним інструментом для підбору комп'ютерних комплектуючих в умовах бюджетних обмежень та наявних апаратних вимог**.**

Висновки

Самостійна збірка – це спосіб отримати максимально потужний, збалансований та довговічний комп'ютер за мінімальну ціну, а використання онлайн-конфігураторів усуває головний ризик цього процесу – несумісність комплектуючих.

Варто відмітити, що для покращення умінь збирати комп'ютерні системи як навчальні технології пропонуються віртуальна реальність [16] та доповнена реальність [17].

Список використаної літератури

1. Winarno, M. (2018). Design and development of computer specification recommendation system based on user budget with genetic algorithm. *International Journal of New Media Technology, 5*(1), 25–29. https://doi.org/10.31937/ijnmt.v5i1.814

2. Mishra, S., Bane, S., Pandit, R., & Phadnis, N. (2021). PC configuration and component recommendation system. *International Journal of Computer Applications, 183*(1), 5–8. https://doi.org/10.5120/ijca2021921411

3. Veres, O., Ilchuk, P., & Kots, O. (2023). Information system for analysis of hardware computer components. In *Proceedings of the 2023 IEEE International Conference on Computer Science and Information Technologies (CSIT)* (pp. 1–4). IEEE. https://doi.org/10.1109/CSIT61576.2023.10324147

4. Cooler Master. (n.d.). *Power supply calculator*. https://www.coolermaster.com/en-global/power-supply-calculator/

5. Ehrgott, M. (2008). Multiobjective optimization. *AI Magazine, 29*(1), 47–57. https://doi.org/10.1007/978-0-387-76635-5_6

6. Березький, О. М., Теслюк, В. М., Дубчак, Л. О., Мельник, Г. М., & Батько, Ю. М. (2022). *Дослідження і проектування комп'ютерних систем та мереж: навчальний посібник*. Тернопіль: ЗУНУ.

7. Rosenberg, R. S. (1970). Stimulation of genetic populations with biochemical properties: I. The model. *Mathematical Biosciences, 7*(3–4), 223–257. https://doi.org/10.1016/0025-5564(70)90126-4

8. Rosenberg, R. S. (1970). Simulation of genetic populations with biochemical properties: II. Selection of crossover probabilities. *Mathematical Biosciences, 8*(1–2), 1–37. https://doi.org/10.1016/0025-5564(70)90140-9

9. Mitchell, M. (1999). *An introduction to genetic algorithms* (5th printing). Cambridge, MA: MIT Press.

10. Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications, 80*(5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

11. Пиріг, Я., Климаш, М., Пиріг, Ю., & Лаврів, О. (2023). Генетичний алгоритм як інструмент для вирішення задач оптимізації. *Інформаційно-комунікаційні технології, електронна інженерія, 3*(2), 95–107. https://doi.org/10.23939/ictee2023.02.095

12. Wang, Q. J. (1997). Using genetic algorithms to optimise model parameters. *Environmental Modelling & Software, 12*(1), 27–34. https://doi.org/10.1016/S1364-8152(96)00030-8

13. Shevchenko, S., Zhdanova, Y., & Bilous, M. (2025). Automation of class scheduling as a necessary condition for optimization of the functioning of an educational institution. In *Technical, agricultural and mathematical sciences: Scientific trends, problems and ways of their development: Collective monograph* (pp. 296–309). Boston: Primedia eLaunch. https://doi.org/10.46299/ISG.2025.MONO.TECH.2

14. Sineglazov, V., Ryazanovskiy, K., & Chumachenko, O. (2020). Multicriteria conditional optimization based on genetic algorithms. *System Research and Information Technologies, 3*(3), 89–104. https://doi.org/10.20535/SRIT.2308-8893.2020.3.07

15. OpenAI. (n.d.). *Introducing ChatGPT*. https://openai.com/index/chatgpt/

16. Резніченко, І. В., Негоденко, О. В., & Шевченко, С. М. (2024). Дослідження програмних та технічних засобів для реалізації технології віртуальної реальності. *Зв'язок, 1*(1). https://doi.org/10.31673/2412-9070.2024.014147

17. Westerfield, G., Mitrovic, A., & Billinghurst, M. (2015). Intelligent augmented reality training for motherboard assembly. *International Journal of Artificial Intelligence in Education,* 25(2), 157–172. https://doi.org/10.1007/s40593-014-0032-x

References

1. Winarno, M. (2018). Design and development of computer specification recommendation system based on user budget with genetic algorithm. *International Journal of New Media Technology, 5*(1), 25–29. https://doi.org/10.31937/ijnmt.v5i1.814

2. Mishra, S., Bane, S., Pandit, R., & Phadnis, N. (2021). PC configuration and component recommendation system. *International Journal of Computer Applications, 183*(1), 5–8. https://doi.org/10.5120/ijca2021921411

3. Veres, O., Ilchuk, P., & Kots, O. (2023). Information system for analysis of hardware computer components. In *Proceedings of the 2023 IEEE International Conference on Computer Science and Information Technologies (CSIT)* (pp. 1–4). IEEE. https://doi.org/10.1109/CSIT61576.2023.10324147

4. Cooler Master. (n.d.). *Power supply calculator*. https://www.coolermaster.com/en-global/power-supply-calculator/

5. Ehrgott, M. (2008). Multiobjective optimization. *AI Magazine, 29*(1), 47–57. https://doi.org/10.1007/978-0-387-76635-5_6

6. Berezkyi O. M., Tesliuk V. M., Dubchak L. O., Melnyk H. M., & Batko Yu. M. (2022). *Doslidzhennia i proektuvannia kompiuternykh system ta merezh: navchalnyi posibnyk* [Research and design of computer systems and networks: Textbook]. Ternopil: ZUNU.

7. Rosenberg, R. S. (1970). Stimulation of genetic populations with biochemical properties: I. The model. *Mathematical Biosciences, 7*(3–4), 223–257. https://doi.org/10.1016/0025-5564(70)90126-4

8. Rosenberg, R. S. (1970). Simulation of genetic populations with biochemical properties: II. Selection of crossover probabilities. *Mathematical Biosciences, 8*(1–2), 1–37. https://doi.org/10.1016/0025-5564(70)90140-9

9. Mitchell, M. (1999). *An introduction to genetic algorithms* (5th printing). Cambridge, MA: MIT Press.

10. Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications, 80*(5), 8091–8126. https://doi.org/10.1007/s11042-020-10139-6

11. Pyrih Ya., Klymash M., Pyrih Yu., & Lavriv O. (2023). Henetychnyi alhorytm yak instrument dlia vyrishennia zadach optymizatsii [Genetic algorithm as a tool for solving optimization problems]. *Information and Communication Technologies, Electronic Engineering, 3*(2), 95–107. https://doi.org/10.23939/ictee2023.02.095

12. Wang, Q. J. (1997). Using genetic algorithms to optimise model parameters. *Environmental Modelling & Software, 12*(1), 27–34. https://doi.org/10.1016/S1364-8152(96)00030-8

13. Shevchenko, S., Zhdanova, Y., & Bilous, M. (2025). Automation of class scheduling as a necessary condition for optimization of the functioning of an educational institution. In *Technical, agricultural and mathematical sciences: Scientific trends, problems and ways of their development: Collective monograph* (pp. 296–309). Boston: Primedia eLaunch. https://doi.org/10.46299/ISG.2025.MONO.TECH.2

14. Sineglazov, V., Ryazanovskiy, K., & Chumachenko, O. (2020). Multicriteria conditional optimization based on genetic algorithms. *System Research and Information Technologies, 3*(3), 89–104. https://doi.org/10.20535/SRIT.2308-8893.2020.3.07

15. OpenAI. (n.d.). *Introducing ChatGPT*. https://openai.com/index/chatgpt/

16. Reznychenko I. V., Nehodenko O. V., & Shevchenko S. M. (2024). Doslidzhennia prohramnykh ta tekhnichnykh zasobiv dlia realizatsii tekhnolohii virtualnoi realnosti [Research of software and hardware tools for the implementation of virtual reality technology]. *Zviazok 1*(1), 41-47. https://doi.org/10.31673/2412-9070.2024.014147

17. Westerfield, G., Mitrovic, A., & Billinghurst, M. (2015). Intelligent augmented reality training for motherboard assembly. *International Journal of Artificial Intelligence in Education, 25*(2), 157–172. https://doi.org/10.1007/s40593-014-0032-x

Дата першого надходження рукопису до видання: 26.11.2025
Дата прийнятого до друку рукопису після рецензування: 23.12.2025
Дата публікації: 31.12.2025