Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: **Методи та засоби оптимізації розкрою листових матеріалів у комп'ютерних системах меблевого виробництва з використанням адаптивних евристик**

Виконав: студент(ка) 6 курсу, групи СІм-62
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Руснак В. М. (прізвище та ініціали)

Керівник (підпис) Стадник Н. Б. (прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є. В. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
«17» листопада 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *Магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Руснак Віктор Михайлович*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методи та засоби оптимізації розкрою листових матеріалів у комп'ютерних системах меблевого виробництва з використанням адаптивних евристик*

Керівник роботи кандидат технічних наукт, Стадник Наталія Богданівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *14* » *листопада 2025* року № 4/7-987

2. Термін подання студентом завершеної роботи *16 грудня 2025*

3. Вихідні дані до роботи *Наукові та літературні джерела, алгоритми комбінаторної оптимізації двовимірного гільйотинного розкрою.*

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ
1 Аналітична частина
2 Теоретична частина
3 Практична частина
4 Охорона праці та безпека в надзвичайних ситуаціях
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

1. Актуальність і мета дослідження.
2. Завдання, об'єкт і предмет, наукова новизна і практична цінність дослідження.
3. Алгоритми оптимізації розкрою
4. Евристичні та мета-евристичні алгоритми
5. Метод адаптивної оптимізації на основі селективної гіпер-евристики
6. Блок схеми
7. Результати
8. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Осухівська Г.М., зав.каф. КС*	*04.12.2025*	
Безпека в надзвичайних ситуаціях	*Стручок В.С.*	*04.12.2025*	

7. Дата видачі завдання *17.11.2025*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Огляд літературних джерел. Постановка завдання на кваліфікаційну роботу.*	*17.11.2025р. – 23.11.2025р.*	*Викон.*
2.	*Дослідження математичних моделей розкрою та розробка адаптивних евристик.*	*24.11.2025р. – 02.12.2025р.*	*Викон.*
3.	*Програмна реалізація системи адаптивних евристик для оптимізації розкрою листових матеріалів засобами Python.*	*03.12.2025р. – 10.12.2025р.*	*Викон.*
4.	*Охорона праці та безпека в надзвичайних ситуаціях*	*04.12.2025р. – 10.12.2025р.*	*Викон.*
5.	*Оформлення пояснювальної записки і графічного матеріалу*	*11.12.2025р. – 19.12.2025р.*	*Викон.*
6.	*Попередній захист кваліфікаційної роботи магістра*	*16.12.2025р.*	*Викон.*
7.	*Захист кваліфікаційної роботи магістра*	*23.12.2025*	*Викон.*

Студент _____ *Руснак В. М.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Стадник Н. Б.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Руснак В. М. Методи та засоби оптимізації розкрою листових матеріалів у комп'ютерних системах меблевого виробництва з використанням адаптивних евристик: робота на здобуття кваліфікаційного ступеня магістра: спец. 123 – комп'ютерна інженерія / наук.кер. Н. Б. Стадник. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2025.

Ключові слова: комп'ютерна система, оптимізація розкрою, адаптивні евристики, гільйотинний розкрій, меблеве виробництво, коефіцієнт корисного використання матеріалу.

Кваліфікаційна робота присвячена дослідженню методів і засобів підвищення ефективності використання листових матеріалів на меблевому виробництві шляхом впровадження комп'ютеризованої системи розкрою. У роботі здійснено аналіз існуючих алгоритмів вирішення задачі розкрою (Cutting Stock Problem), а також закцентовано увагу на проблемах мінімізації відходів при серійному виробництві.

В основі створеної системи використовується метод адаптивних евристик, який дозволяє автоматично підбирати оптимальну стратегію розміщення деталей залежно від параметрів замовлення та технологічних обмежень обладнання.

Отримані результати розрахунків візуалізуються у вигляді карт розкрою, а сформовані завдання експортуються у форматі JSON для подальшої передачі на верстати з числовим програмним керуванням. Результати цього дослідження мають практичне значення для зменшення виробничих витрат, економії матеріалів та автоматизації робочого місця технолога меблевого підприємства.

ANNOTATION

Rusnak V. M. Methods and tools for optimization of sheet material cutting in computer systems of furniture production using adaptive heuristics: Master's Qualification Thesis: specialty 123 – Computer Engineering / Supervisor N. B. Stadnyk. Ternopil: Ternopil Ivan Puluj National Technical University, 2025.

Keywords: computer system, cutting optimization, adaptive heuristics, guillotine cutting, furniture production, material utilization factor.

The Master's qualification thesis is devoted to the research of methods and tools for increasing the efficiency of sheet material use in furniture production by implementing a computerized cutting system. The work includes an analysis of existing algorithms for solving the packing problem (Cutting Stock Problem) and focuses on the problems of minimizing waste in serial production.

The created system is based on the method of adaptive heuristics, which allows for the automatic selection of the optimal part placement strategy depending on the order parameters and technological limitations of the equipment.

The obtained calculation results are visualized in the form of cutting maps, and the generated tasks are exported in JSON format for further transfer to CNC machines. The results of this research have practical significance for reducing production costs, saving materials, and automating the workplace of a furniture enterprise technologist.

ЗМІСТ

ЗМІСТ 6

ВСТУП 8

РОЗДІЛ 1 АНАЛІТИЧНА ЧАСТИНА 11

1.1. Сучасний стан та тенденції автоматизації розкрою листових матеріалів у меблевому виробництві 11

1.2.Математича постановка задачі розкрою та аналіз наявних методів її вирішення 13

1.3. Класифікація та порівняльний аналіз наявних методів і алгоритмів розкрою ... 16

1.4. Огляд наявних програмних систем автоматизації розкрою 24

1.5. Висновки до розділу 1 27

РОЗДІЛ 2 ТЕОРЕТИЧНА ЧАСТИНА 28

2.1. .....Формалізація задачі гільйотинного розкрою з урахуванням технологічних обмежень 28

2.2. Модель представлення простору розкрою на основі бінарного дерева 29

2.3. Базовий евристичний алгоритм розміщення 32

2.4. Метод адаптивної оптимізації на основі селективної гіперевристики 35

2.5 Висновки до розділу 2 37

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА 39

3.1. Принципи роботи та архітектура програмної системи розкрою 39

3.2. Програмна реалізація алгоритмів 40

3.3. Оцінка ефективності роботи системи на тестових даних 43

3.4. Висновки до розділу 3 49

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ ... 50

4.1. Охорона праці 50

4.2. Безпека в надзвичайних ситуаціях 52

4.2.1.Організація протипожежного захисту та проведення протипожежної профілактики на промисловому підприємстві 52

4.2.2. Джерела виникнення шуму і вібрацій. Заходи і засоби захисту від шуму і вібрацій, гігієнічні та допустимі норми. 53

ВИСНОВКИ 56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58

Додаток А Тези конференцій

Додаток Б Лістинг коду основного файлу системи

ВСТУП

Актуальність роботи. Сучасне меблеве виробництво характеризується високою конкуренцією та постійним зростанням вартості сировини, частка якої у собівартості кінцевої продукції може сягати 60-70% [10]. В умовах зростання попиту на індивідуальні замовлення виникає потреба у вдосконаленні систем автоматизованого проєктування та підготовки виробництва. Тр адиційні підходи до розкрою листових матеріалів (ДСП, МДФ, фанера) здебільшого базуються на статичних алгоритмах розміщення або ручній корекції карт розкрою технологом. Це вимагає значних часових витрат кваліфікованого персоналу та не завжди забезпечує раціональне використання матеріалів, особливо при складних комбінованих замовленнях.

У реальних виробничих умовах використання одного фіксованого алгоритму (наприклад, розміщення за спаданням площі) не дає стабільно високого результату для всіх типів карт розкрою. Це актуалізує завдання переходу від статичних методів до адаптивних інтелектуальних систем, здатних автоматично аналізувати структуру замовлення та підбирати найбільш ефективну стратегію розкрою для мінімізації відходів.

Одним із найважливіших напрямів підвищення ефективності автоматизованих систем меблевого виробництва є використання адаптивних евристик для вирішення задачі двовимірного розкрою (2D Cutting Stock Problem) [29]. Своєчасний підбір оптимальної схеми розкрою має ключове значення для зменшення кількості технологічних відходів, зниження витрат на матеріали та скорочення часу роботи верстатів з числовим програмним керуванням (ЧПК). Використання сучасних алгоритмів комбінаторної оптимізації та комп’ютерного моделювання створює можливість автоматизувати процес прийняття рішень, виключити людський фактор та підвищити коефіцієнт корисного використання матеріалу.

У межах роботи передбачається розроблення підходу, який базується на поєднанні класичних алгоритмів гільйотинного розміщення та мета-евристичного механізму вибору стратегії залежно від параметрів вхідного замовлення.

Метою кваліфікаційної роботи є дослідження, розробка та експериментальна перевірка методів і програмних засобів оптимізації розкрою листових матеріалів із використанням адаптивних евристик у комп'ютерних системах підготовки виробництва.

Завдання кваліфікаційної роботи:

– провести аналіз сучасних підходів та математичних моделей вирішення задачі розкрою в меблевій промисловості;

– дослідити наявні алгоритми гільйотинного розкрою та методи адаптивного вибору евристик;

– розробити алгоритмічне забезпечення та програмний прототип системи для автоматичної генерації карт розкрою з урахуванням технологічних обмежень;

– реалізувати механізм експорту даних для інтеграції з виробничим обладнанням та провести порівняльний аналіз ефективності розроблених методів.

Відповідно до мети та завдань кваліфікаційної роботи визначено її об'єкт та предмет.

*Об'єкт дослідження***:** процес автоматизованого розкрою листових матеріалів у комп'ютерних системах технологічної підготовки виробництва.

*Предмет дослідження***:** методи, моделі та алгоритми адаптивної оптимізації розміщення прямокутних деталей на площині, що забезпечують мінімізацію відходів.

Методи дослідження: методи системного аналізу (для побудови моделі системи), теорія алгоритмів та комбінаторна оптимізація (для вирішення задачі розкрою), об'єктно-орієнтоване програмування (для реалізації прототипу), імітаційне моделювання (для оцінки ефективності стратегій).

Наукова новизна дослідження полягає у вдосконаленні методу розкрою листових матеріалів шляхом застосування адаптивних евристик, що дозволяє динамічно змінювати пріоритети розміщення деталей залежно від топології замовлення, підвищуючи щільність розкрою.

Практичне значення результатів кваліфікаційної роботи полягає у створенні програмного модуля, який дозволяє автоматизувати робоче місце

технолога, зменшити обсяг відходів матеріалів та формувати файли-завдання для верстатів з ЧПК, що сприяє зниженню собівартості продукції.

Публікації. Результати дослідження апробовано на XIII науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» та XIV міжнародній науково-технічній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» у вигляді тез конференцій:

1. Методи адаптивного вибору евристик для задач гільйотинного розкрою [25].
2. Інформаційна технологія адаптивної оптимізації розкрою листових матеріалів у меблевому виробництві [24].

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка містить вступ, 4 розділи, висновки, список використаних джерел та додатків.

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1. Сучасний стан та тенденції автоматизації розкрою листових матеріалів у меблевому виробництві

Сучасна меблева промисловість перебуває на етапі активної трансформації. Якщо раніше домінуючою моделлю було масове виробництво стандартизованих виробів, то сьогодні ключовим трендом стає – виготовлення індивідуальних замовлень у промислових масштабах. Це створює нові виклики для систем автоматизованого проєктування (CAD) та підготовки виробництва (CAM), особливо на критично важливому етапі – розкрої листових матеріалів.

Листові матеріали, такі як деревинно-стружкові плити (ДСП), деревинно-волокнисті плити середньої щільності (МДФ) та фанера, є основою корпусу сучасних меблів. Аналіз структури собівартості меблевої продукції свідчить, що витрати на основні матеріали складають левову частку загальної вартості виробу, часто досягаючи 60-70%. В умовах постійного зростання цін на сировину та енергоносії, навіть незначне підвищення коефіцієнта корисного використання матеріалу (ККВМ) на частки відсотка забезпечує підприємству суттєвий економічний ефект.

Особливу складність становить оптимізація розкрою в умовах дрібносерійного та індивідуального виробництва. На відміну від масового виробництва, де карти розкрою можуть повторюватися тисячі разів, при виготовленні меблів на замовлення кожна нова карта є унікальною задачею комбінаторної оптимізації. У таких картах часто поєднуються деталі різних габаритів та кількості (наприклад, великі деталі шафи-купе та дрібні деталі висувних ящиків). Це вимагає від системи розкрою високої гнучкості та здатності швидко адаптуватися до зміни топології вхідних даних.

Існуючі комерційні системи автоматизації (наприклад, ViyarPro, GibLab) часто використовують закриті статичні алгоритми, які базуються на фіксованих правилах (наприклад, сортування деталей за спаданням площі). Такий підхід не завжди є ефективним: алгоритм, який добре працює для карт з великими деталями, може

показувати низький ККВМ для карт з великою кількістю дрібних вузьких деталей. Це призводить до появи значних технологічних відходів, які неможливо використати повторно.

Традиційний підхід до організації ділянки розкрою, коли карти розкрою складалися технологом вручну або за допомогою найпростіших статичних алгоритмів, стає «вузьким місцем» виробничого процесу. Оператор фізично не здатний за обмежений час перебрати тисячі можливих варіантів розміщення деталей на листі, щоб знайти глобальний оптимум. Тому на сучасних підприємствах цей процес інтегрується в єдиний цифровий потік, де дані про замовлення автоматично передаються до системи оптимізації, а звідти – безпосередньо на обладнання (див. рис. 1.1).

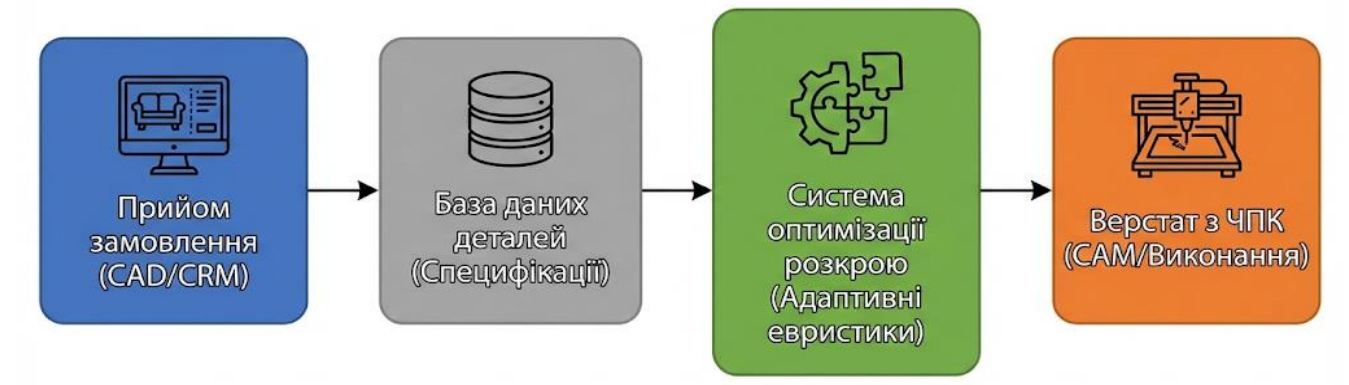

Рис. 1.1. Структурна схема інформаційних потоків автоматизованого меблевого виробництва

Отже, сучасний стан галузі характеризується гострою потребою у створенні гнучких програмних модулів, які можуть інтегруватися в наявні CAD/CAM-системи підприємств, забезпечуючи мінімізацію відходів та автоматичну генерацію керуючих програм для обладнання з числовим програмним керуванням (ЧПК). Вирішення цього завдання вимагає застосування сучасних методів комбінаторної оптимізації та теорії алгоритмів.

1.2. Математича постановка задачі розкрою та аналіз наявних методів її вирішення

Ефективність автоматизованої системи підготовки виробництва напряму залежить від якості закладеної в неї математичної моделі. Задача про розкрій – це класична задача комбінаторної оптимізації, яка полягає у розміщенні скінченної множини прямокутних об’єктів на мінімальній кількості прямокутних областей більшого розміру [6].

У контексті меблевого виробництва вхідні дані для задачі розкрою можна формалізувати таким чином: нехай задано множину типів прямокутних деталей (заготовок) $J = \{1,2 \dots, n\}$, для кожного типу деталі $i \in J$ визначено:

– ω_i – ширина деталі (уздовж текстури, якщо вона є);

– h_i – висота деталі (впоперек текстури);

– d_i – необхідна кількість деталей даного типу (demand);

– $r_i \in \{0,1\}$ – бінарний параметр можливості обертання ($r_i = 1$, якщо обертання на 90º дозволено, $r_i = 0$, якщо заборонено через напрямок текстури).

Для розміщення деталей доступний запас листового матеріалу. У загальному випадку це може бути множина листів різних форматів, але на практиці меблеві підприємства зазвичай уніфікують закупівлі, використовуючи один або два стандартні формати (наприклад, ДСП 2800×2070 мм). Нехай задано лист матеріалу L з номінальними розмірами $W \times H$.

Окрім геометричних параметрів деталей та листа, обов’язковими вхідними даними є технологічні константи обладнання:

– K – ширина пропилу дискової пили ($K > 0$, зазвичай 3-5мм);

– $Trim_{top}, Trim_{bottom}, Trim_{left}, Trim_{right}$ – величини технологічних відступів для торцювання країв листа перед розкроєм (≥ 0, зазвичай 3-5мм).

Основною метою оптимізації є мінімізація відходів матеріалу. Залежно від специфіки виробництва, цільова функція може формулюватися двома основними способами:

1. Мінімізація кількості використаних листів (Класична CSP). Цей підхід застосовується при серійному виробництві, коли необхідно виготовити фіксовану, заздалегідь відому кількість виробів. Завдання полягає в тому, щоб розмістити всі деталі мінімальній кількості листів.

2. Максимізація заповнення одного листа. Цей підхід частіше використовується в індивідуальному виробництві або при роботі із залишками. Завдання полягає в тому, щоб вибрати з черги замовлень таку підмножину деталей, яка максимізує сумарну площу (або вартість) розміщених деталей на одному поточному листі.

Узагальненим критерієм ефективності, який використовується в обох випадках, є коефіцієнт корисного використання матеріалу (формула 1.1).

Він визначається як відношення сумарної площі всіх корисних деталей до загальної номінальної площі використаного матеріалу:

$$E = \frac{\sum_{i=1}^{n} \omega_i \cdot h_i \cdot d_i^{placed}}{\sum_{j=1}^{N} W_j \cdot H_j} \cdot 100\% \rightarrow max \tag{1.1}$$

де d_i^{placed} – кількість фактично розміщених деталей $i-$ го типу, N – кількість використаних листів.

Реальний виробничий розкрій накладає ряд жорстких технологічних обмежень, ігнорування яких робить отриману карту розкрою непридатною для використання на обладнанні. Саме ці обмеження перетворюють задачу на одну з найскладніших в дослідженні операцій [29].

Листи ДСП/МДФ часто мають пошкоджені краї внаслідок транспортування, а їхні кути можуть не бути ідеально прямими (відхилення від ортогональності). Тому першою технологічною операцією завжди є торцювання – обрізка матеріалу по периметру на задані величини $Trim$. Це означає, що алгоритм оптимізації повинен працювати не з номінальними розмірами листа $W \times H$, а з ефективними розмірами робочої зони.

Процес різання є субтрактивним – частина матеріалу перетворюється на стружку. Ширина дискової пили K визначає ширину пропилу. У математичній моделі

це означає, що відстань між будь-якими двома сусідніми деталями, а також між деталлю і краєм робочої зони листа не може бути меншою за K, ігнорування цього фактору призведе до отримання деталей меншого розміру, ніж заплановано.

Обмеження гільйотинного різу, зумовлене фізикою роботи форматно-розкрійних верстатів. Пила може робити різ тільки по прямій лінії від одного краю матеріалу до іншого. Неможливо зупинити пилу посеред матеріалу, повернути її на 90º і продовжити різ, не пошкодивши сусідні ділянки. Це означає, що карта розкрою повинна мати рекурсивну структуру, яка дозволяє послідовно розділяти прямокутні області на менші. Карти, що містять L-подібні вирізи, називаються негільйотинними і не можуть бути виконані на стандартному обладнанні (див. рис. 1.2).

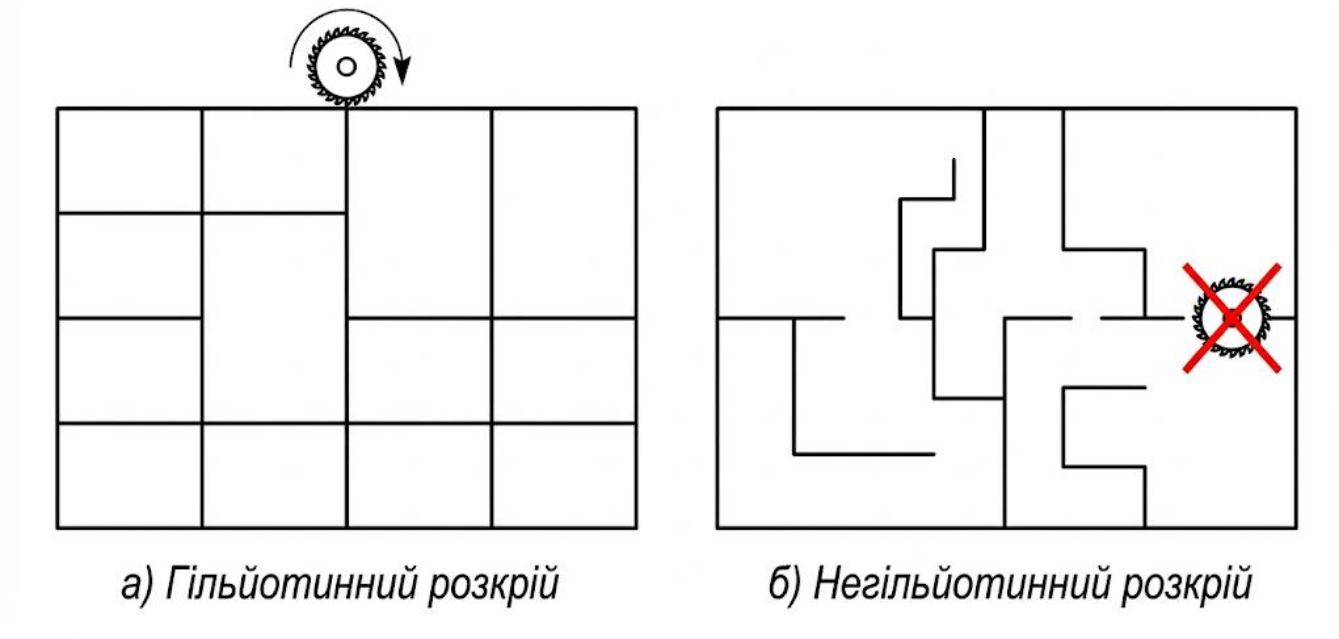

Рис. 1.2. Порівняння гілйотинного і негільйотинного різу

Орієнтація текстури для матеріалів з декоративним покриттям (текстуроване ДСП, шпон) критично важливим є напрямок волокон. Якщо для деталі важливий напрям текстури, то її заборонено обертати на 90º, навіть якщо геометрично вона краще вписується в іншому положенні. Це обмеження суттєво звужує простір пошуку рішень.

З точки зору теорії алгоритмів та обчислювальної складності, задача двовимірного розкрою, особливо з додатковими обмеженнями (гільйотинність,

орієнтація текстури), належить до класу NP-складних задач [28]. Це має фундаментальні наслідки для вибору методів її вирішення.

Головною проблемою є кількість можливих способів розміщення деталей. Якщо для однієї деталі існує, умовно, P позицій на листі, то для n деталей кількість комбінацій може сягати P^n. Навіть із урахуванням обмежень, що відсікають невалідні варіанти, простір пошуку рішень залишається великим.

Наприклад, для карти розкрою середньої складності, що містить 30-50 різнотипних деталей, застосування точних методів, таких як повний перебір, метод гілок і меж або цілочисельне лінійне програмування для пошуку гарантовано оптимального рішення є практично неможливим. Час роботи таких алгоритмів на сучасних суперкомп'ютерах може вимірюватися роками, що є неприйнятним для виробничого процесу, де карта розкрою має бути готова за хвилини.

Обчислювальна нерозв'язність точними методами за розумний час зумовлює необхідність використання евристичних алгоритмів. Ці методи не гарантують знайдення глобального оптимуму, але дозволяють знайти субоптимальне рішення за прийнятний для виробництва час. Саме аналіз та розробка таких методів, зокрема адаптивних, які здатні підлаштовуватися під конкретний набір деталей, є основним предметом нашого дослідження.

1.3. Класифікація та порівняльний аналіз наявних методів і алгоритмів розкрою

Задача двовимірного гільйотинного розкрою з урахуванням технологічних обмежень належить до класу NP-важких задач комбінаторної оптимізації. Це означає, що зі збільшенням кількості деталей у замовленні час, необхідний для гарантованого знаходження найкращого рішення, зростає експоненціально. Така обчислювальна складність унеможливлює використання прямих математичних методів для оперативного планування виробництва, що зумовило активний розвиток різноманітних наближених алгоритмів.

У сучасній науковій літературі та інженерній практиці методи вирішення задачі розкрою прийнято класифікувати за принципом їх роботи та обчислювальною складністю. Узагальнена класифікація наведена на рисунку 1.3.

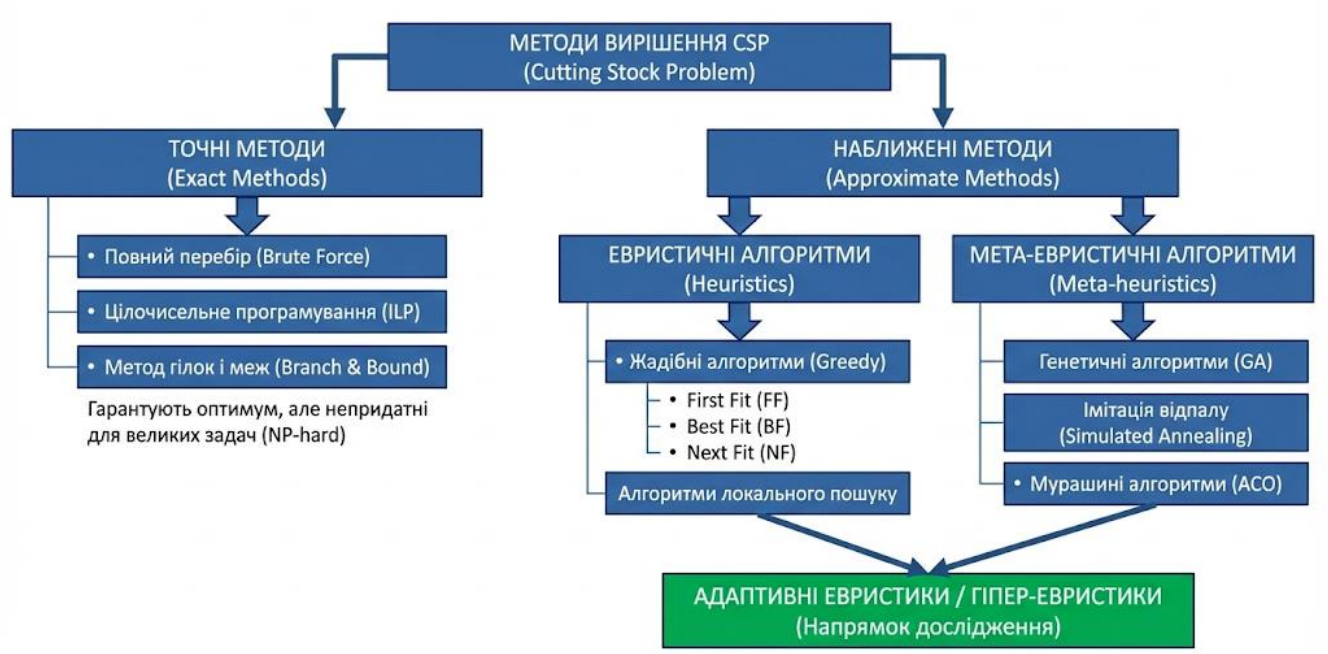

Рис. 1.2. Класифікація методів і алгоритмів вирішення задачі розкрою

Точні методи базуються на математичних моделях і гарантують знаходження глобального оптимуму – тобто, теоретично найкращого можливого варіанту розміщення деталей, що забезпечує максимальний ККВМ.

До основних точних методів відносяться:

Метод повного перебору (Brute Force) полягає у генерації та оцінці абсолютно всіх можливих допустимих комбінацій розміщення. Практично застосовний лише для задач мікроскопічної розмірності.

Цілочисельне лінійне програмування (Integer Linear Programming) – формулюється як система лінійних рівнянь та нерівностей, де змінні можуть набувати лише цілочисельних значень. Для рішення використовуються спеціалізовані засоби (наприклад, CPLEX, Gurobi).

Метод гілок і меж (Branch and Bound) – це метод спрямованого перебору, який будує дерево рішень. Алгоритм дозволяє відсікати ті варіанти розміщення, які

гарантовано не приведуть до рішення, кращого за вже знайдене, що скорочує простір пошуку [8].

Головним недоліком точних методів є їхня часова складність у найгіршому випадку. Для реального меблевого замовлення, що містить 50-100 різнотипних деталей, пошук точного рішення може вимагати годин або днів обчислень на сучасному обладнанні, що є абсолютно неприйнятним для потокового виробничого процесу, де карта розкрою має генеруватися за лічені хвилини.

Через практичну обмеженість точних методів основна увага розробників CAD/CAM систем зосереджена на наближених методах.

Серед них домінують конструктивні алгоритми локальної оптимізації. Їхня суть полягає в поетапному розміщенні деталей: на кожному кроці алгоритм приймає рішення, яке є найкращим у локальний момент, не аналізуючи глобальні наслідки цього кроку в майбутньому. Ефективність цих методів кардинально зростає, якщо перед початком розміщення вхідний список деталей відсортувати за спаданням (наприклад, за площею, шириною або висотою).

Next Fit (метод наступного допустимого розміщення) – найпростіша та найшвидша евристика. Деталі розглядаються у заданому порядку. Поточна деталь намагається розміститися у поточній відкритій смузі (або листі). Якщо вона не вміщається, поточна смуга закривається, відкривається нова, і деталь розміщується там. Алгоритм ніколи не повертається до попередніх смуг, навіть якщо там залишилося місце (див. рис 1.3).

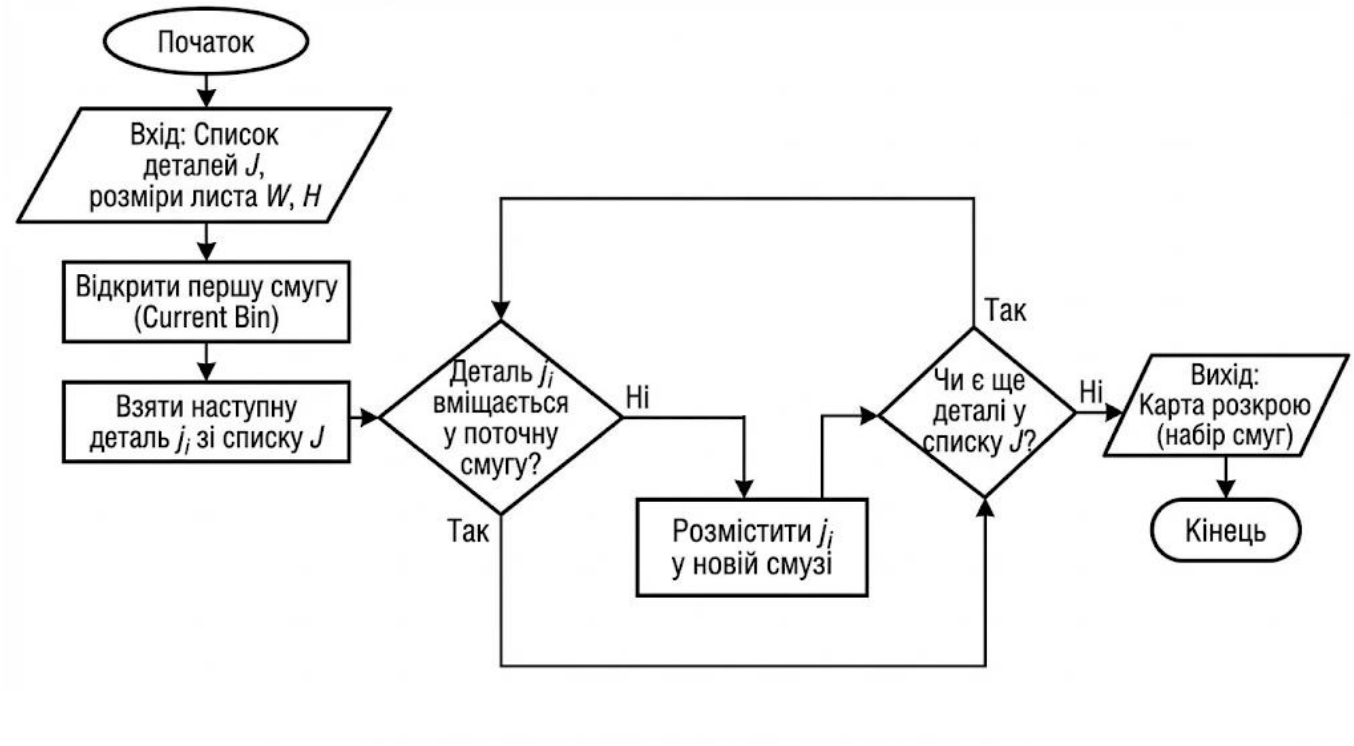

Рис. 1.3. Блок-схема алгоритму Next Fit

First Fit (метод першого допустимого розміщення) (див. рис. 1.4) – логіка його роботи полягає в тому, що система переглядає список доступних вільних прямокутних областей (вузлів дерева вільного простору) у тому порядку, в якому вони були створені або збережені в пам'яті. Як тільки знаходиться вільна ділянка, розміри якої дозволяють розмістити поточну деталь (з урахуванням можливого обертання), пошук припиняється, і деталь фіксується саме в цій позиції. Перевагою такого підходу є надзвичайно висока швидкість роботи, оскільки алгоритму не потрібно переглядати всі можливі варіанти розміщення. Це робить його ідеальним для попередньої оцінки необхідної кількості матеріалу або для ситуацій, коли швидкість генерації карти розкрою є критичнішою за економію матеріалу. Проте суттєвим недоліком стратегії First Fit є тенденція до фрагментації вільного простору. Оскільки алгоритм займає першу-ліпшу позицію, він часто розбиває великі вільні області на дрібні шматки на ранніх етапах, що ускладнює розміщення габаритних деталей в кінці процесу.

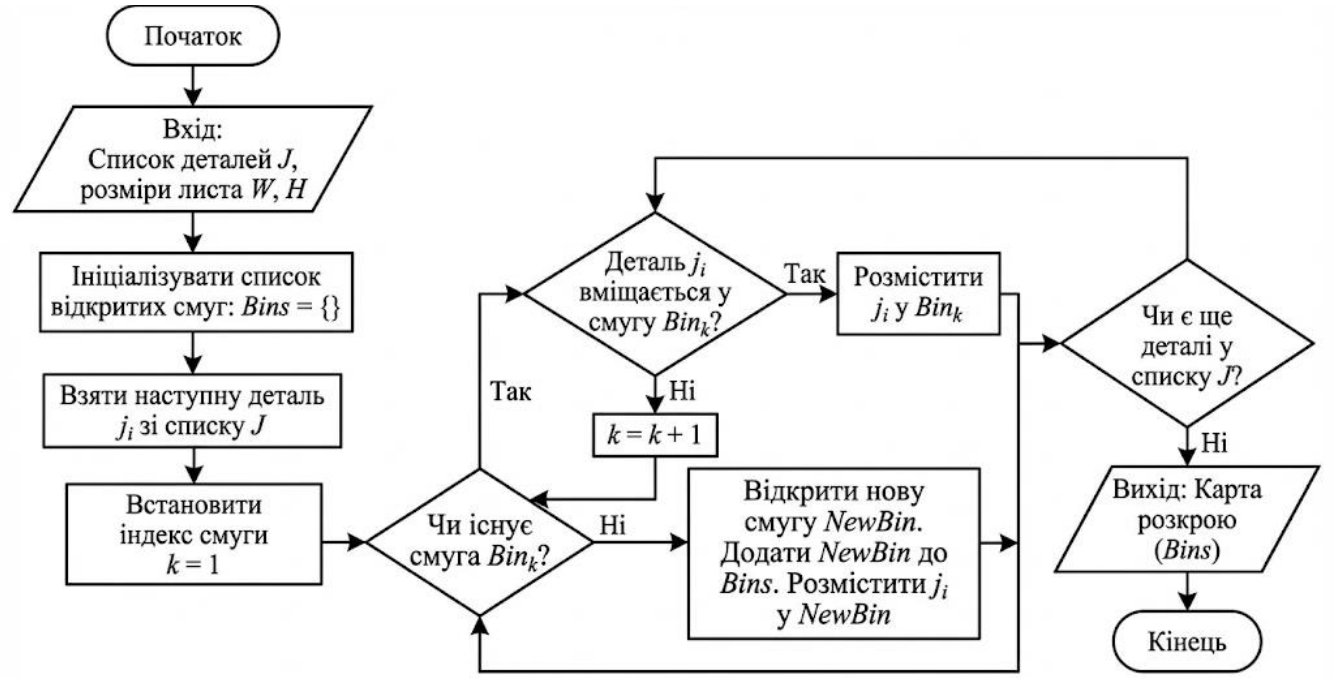

Рис. 1.4. Блок-схема алгоритму First Fit

Best Fit (метод найкращої відповідності) (див. рис. 1.5). На відміну від попереднього методу, цей алгоритм переглядає абсолютно всі доступні вільні області на листі, перш ніж прийняти рішення. Для кожного потенційного місця розміщення обчислюється певна оцінка якості, і деталь поміщається туди, де ця оцінка є найкращою. Поняття "найкращого" може трактуватися по-різному, що породжує кілька варіацій цього алгоритму.

Найпоширенішою метрикою є мінімізація площі залишку. Алгоритм шукає таку вільну область, площа якої є мінімально можливою серед тих, що вміщують дану деталь. Іншими словами, система намагається знайти "найтісніше" місце для деталі, щоб залишити великі вільні зони незайманими для майбутніх великих деталей. Ця стратегія демонструє значно вищу щільність укладання, ніж First Fit, оскільки вона зберігає цілісність великих шматків матеріалу якомога довше. Інша варіація Best Fit фокусується на мінімізації короткої сторони залишку. У цьому випадку алгоритм намагається розмістити деталь так, щоб залишок мав мінімальну ширину або висоту. Це сприяє утворенню довгих вузьких смуг, які можуть бути корисними для розміщення довгих деталей (наприклад, царг або планок), що є характерним для меблевого виробництва.

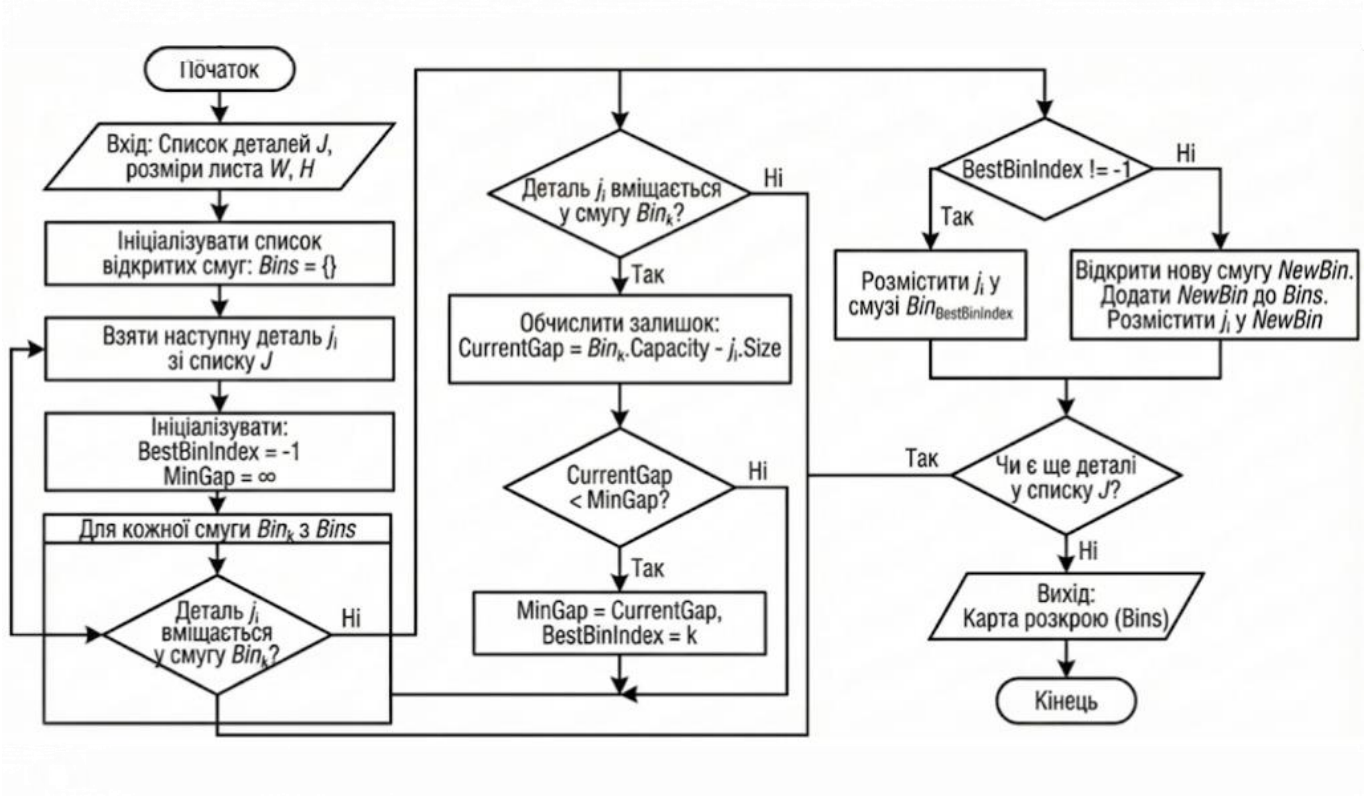

Рис. 1.5. Блок-схема алгоритму Best Fit

Головними перевагами є висока швидкість роботи та простота реалізації. Проте через свій характер вони часто приймають рішення, які блокують можливість ефективного розміщення інших деталей на пізніших етапах, що призводить до субоптимального глобального результату. Крім того, вони є статичними: одна й та сама стратегія застосовується до будь-якого набору даних, що не завжди ефективно.

Для подолання недоліків простих евристик використовуються мета-евристики – це алгоритми вищого рівня, які керують процесом пошуку в просторі рішень, комбінуючи різні базові евристики та використовуючи механізми виходу з пасток локальних оптимумів. Вони часто надихаються природними або фізичними процесами.

Генетичні алгоритми (Genetic Algorithms) (див. рис 1.6.) базуються на принципах еволюційної біології, працюють не з одним рішенням, а з цілою популяцією рішень (карт розкрою). До популяції ітеративно застосовуються оператори селекції (вибір найкращих батьків), кросинговеру (обмін частинами карт

між батьками) та мутації (випадкові зміни в картах). У процесі еволюції виживають рішення з найвищим коефіцієнтом корисно використаного матеріалу [4, 17].

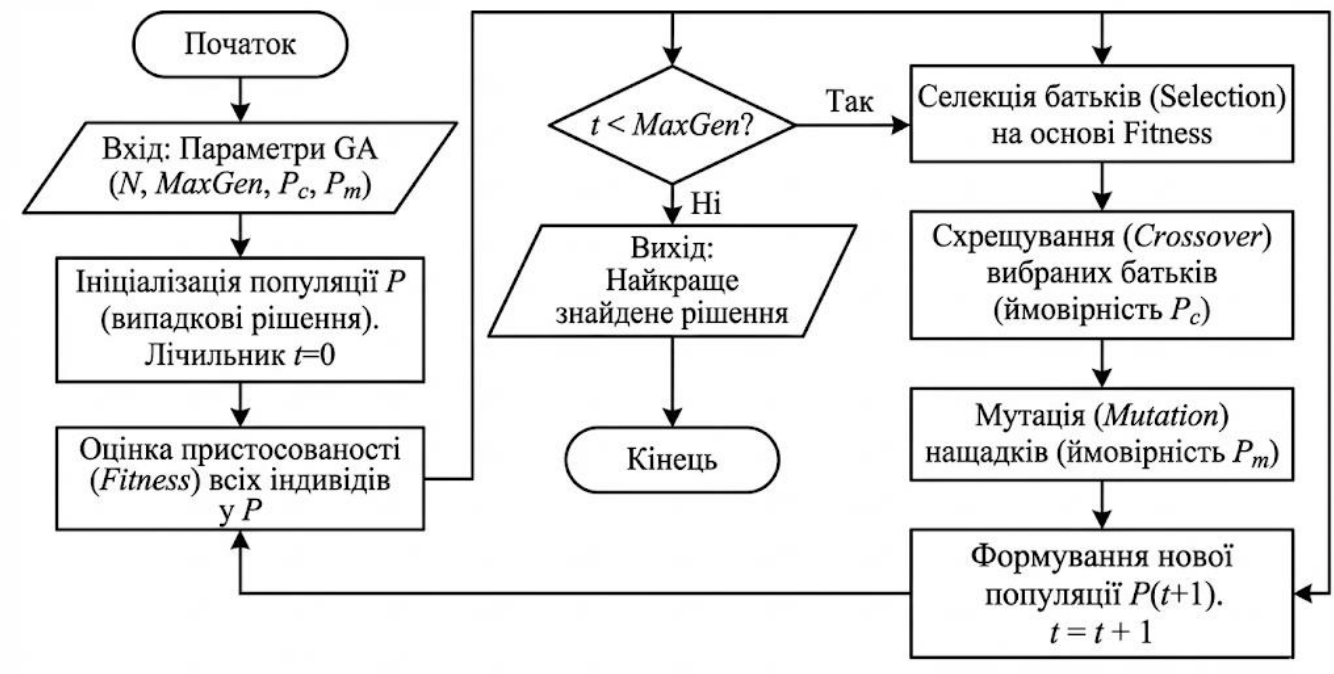

Рис. 1.6. Блок-схема генетичного алгоритму (GA) для задачі розкрою

Імітація відпалу (Simulated Annealing) (див. рис. 1.7.) – метод локального пошуку, натхненний процесом термодинамічного відпалу металів [1]. Головна особливість алгоритму – здатність з певною ймовірністю приймати рішення, які погіршують поточний результат (знижують ККВМ). Такий механізм дозволяє алгоритму виходити з локальних оптимумів у пошуках глобального.

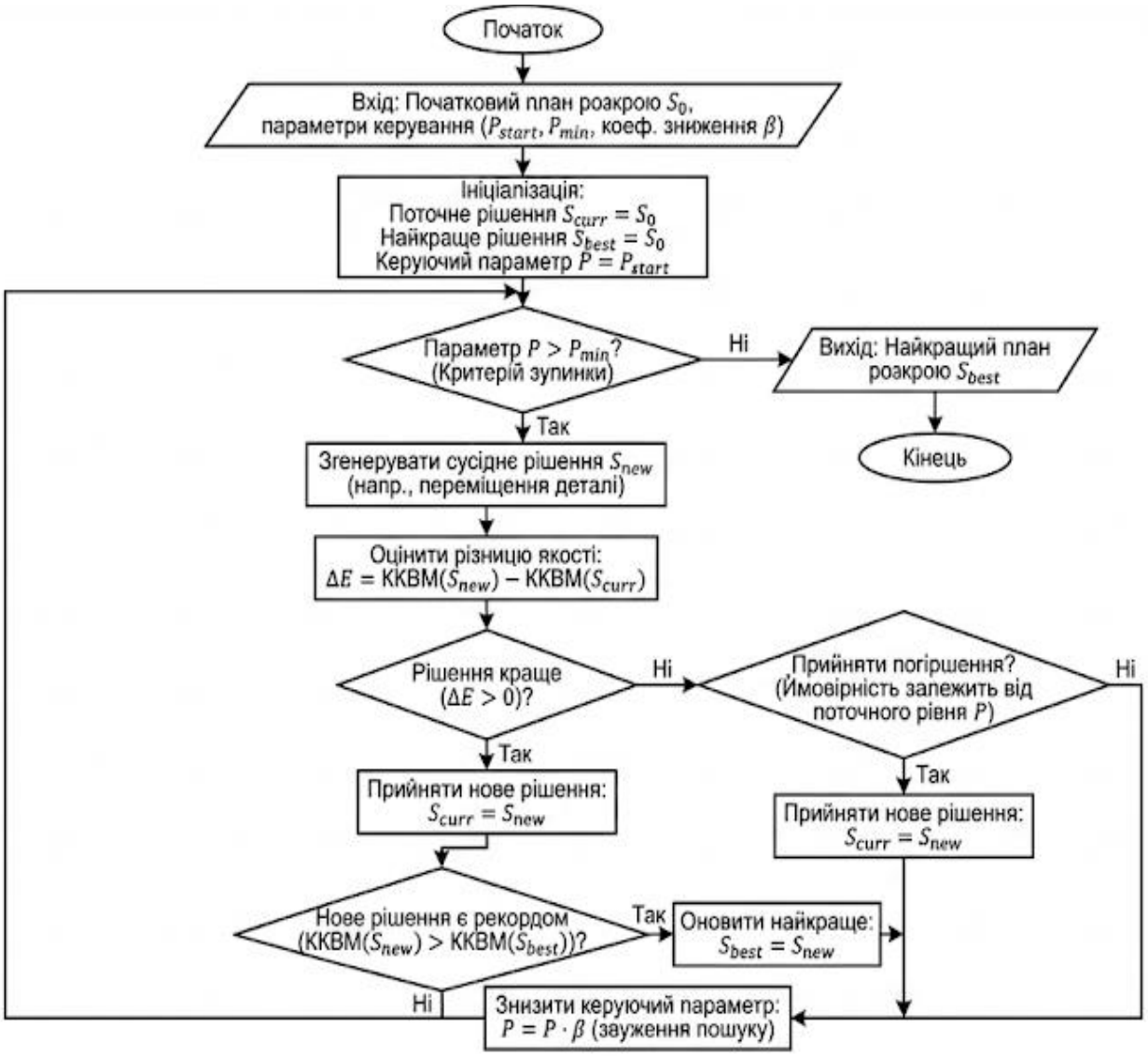

Рис. 1.7. Блок-схема алгоритму Імітації відпалу (SA)

Ці методи дозволяють отримувати значно якісніші рішення, ніж прості жадібні алгоритми, часто наближаючись до результатів точних методів. Однак вони є значно складнішими в реалізації, вимагають ретельного налаштування багатьох параметрів (розмір популяції, ймовірності мутації, графік охолодження) і працюють суттєво повільніше за методи локальної оптимізації, що може бути критичним для інтерактивних систем.

Проведений порівняльний аналіз демонструє суттєвий розрив між наявними класами методів. З одного боку, існують швидкі, але статичні та локально-орієнтовані евристики, які не завжди забезпечують високий ККВМ. З іншого боку – потужні, але повільні та складні у налаштуванні мета-евристики та точні методи.

Це обґрунтовує актуальність наукового завдання зі створення гібридного підходу, який поєднував би обчислювальну ефективність простих методів із гнучкістю складних. Перспективним напрямком є розробка адаптивних евристичних систем, які здатні аналізувати характеристики вхідного замовлення і динамічно обирати найбільш відповідну стратегію розкрою [13]. Саме цей підхід покладено в основу кваліфікаційної роботи.

1.4. Огляд наявних програмних систем автоматизації розкрою

Сучасний ринок програмного забезпечення для автоматизації меблевого виробництва пройшов довгий шлях еволюції від простих калькуляторів площі до складних інтелектуальних систем управління виробничими процесами. На початкових етапах розвитку галузі інженери використовували звичайні електронні таблиці для розрахунку необхідної кількості матеріалів, що неминуче призводило до значних похибок та перевитрат сировини. З появою перших спеціалізованих програм ситуація змінилася, проте ранні версії таких продуктів базувалися на статичних алгоритмах перебору, які не враховували специфіки технологічного обладнання.

Варто детальніше розглянути програмний комплекс Astra R-Nesting, який тривалий час був одним із лідерів на ринку країн пострадянського простору. Ця система пропонує користувачам можливість автоматичного та ручного розміщення деталей, а також функцію імпорту замовлень з інших конструкторських програм. Основною перевагою даного програмного продукту є його відносна простота та невисока вартість порівняно із західними аналогами. Проте суттєвим недоліком залишається закритість алгоритмічного ядра, що унеможливлює адаптацію стратегії розкрою під нестандартні виробничі задачі. Користувач змушений покладатися на вбудовані евристики, які не завжди ефективно справляються зі складними картами розкрою, де присутні деталі великих габаритів разом із великою кількістю дрібних елементів.

Іншим популярним рішенням є програма Cutting 3, яка здобула популярність завдяки своїй невибагливості до апаратних ресурсів комп’ютера. Вона дозволяє

виконувати розкрій не лише листових, а й лінійних матеріалів, таких як труби чи профілі. Особливістю цієї програми є можливість задавати пріоритетність розкрою, обираючи між швидкістю обчислень та мінімізацією відходів. Тим не менш, інтерфейс програми є морально застарілим і не відповідає сучасним стандартам ергономіки програмного забезпечення. Крім того, відсутність підтримки хмарних технологій та можливості інтеграції з сучасними CRM-системами робить її менш привабливою для великих меблевих фабрик, які прагнуть до повної цифровізації виробничого циклу.

Окремої уваги заслуговують хмарні сервіси, такі як ViyarPro, які інтегровані безпосередньо з виробничими потужностями постачальників матеріалів. Такий підхід дозволяє конструктору меблів не лише створити карту розкрою, а й миттєво оформити замовлення на порізку, кромкування та свердління отворів. Це значно скорочує час на комунікацію між клієнтом та виробником, зменшує ймовірність помилок при передачі даних та дозволяє відстежувати статус виконання замовлення в режимі реального часу. Однак використання таких систем прив’язує виробника меблів до конкретного постачальника послуг, що може бути економічно невигідним у разі зміни ринкової кон’юнктури або цін на матеріали.

Таким чином, проведений аналіз показує, що, незважаючи на наявність широкого спектру програмних рішень, існує незаповнена ніша для адаптивних систем, які б поєднували гнучкість налаштувань, відкритість алгоритмів та можливість інтеграції з різним обладнанням без жорсткої прив’язки до конкретного виробника чи постачальника. Розроблена в даній магістерській роботі система покликана вирішити саме цю проблему, пропонуючи універсальний інструмент для оптимізації розкрою з використанням адаптивних евристик.

Спеціалізовані десктопні CAD/CAM системи) – це потужні комплексні системи, що інтегруються у виробничий цикл великих підприємств. Вони пропонують широкі можливості налаштування технологічних параметрів (відступи, пропили, орієнтація текстури, складські залишки).

Веборієнтовані сервіси та хмарні платформи – системи набули значної популярності серед малих та середніх виробників завдяки доступності, відсутності

необхідності встановлення складного ПЗ та інтеграції з сервісами замовлення матеріалів та послуг (розпилювання, кромкування).

Незважаючи на широку функціональність, аналіз зазначених систем крізь призму класифікації алгоритмів виявляє ряд суттєвих обмежень:

1. Комерційні системи є пропрієтарними продуктами із закритим вихідним кодом. Користувач не має інформації про те, які саме математичні методи використовуються для оптимізації. Це унеможливлює верифікацію ефективності алгоритмів та їх вдосконалення сторонніми дослідниками.

2. Більшість систем використовують фіксований набір евристичних алгоритмів (найчастіше – варіації жадібних методів типу Best Fit Decreasing, які забезпечують високу швидкість роботи вебсервісів). Вони застосовують одну й ту саму стратегію незалежно від структури вхідного замовлення. Евристика, що ефективна для замовлення з малою кількістю великих деталей, може показати посередній результат на замовленні з великою кількістю дрібних деталей.

3. Відсутність адаптивності, наявні системи зазвичай не мають механізмів автоматичного аналізу вхідних даних та динамічного вибору найкращого алгоритму оптимізації під конкретну виробничу ситуацію. Хоча деякі системи пропонують режими глибокої оптимізації (що, ймовірно, задіює мета-евристики типу імітації відпалу), це часто вимагає ручного втручання оператора та значних часових витрат, що не завжди прийнятно для потокового виробництва.

Проведений аналіз теоретичних методів та практичних реалізацій дозволяє зробити висновок про існування розриву між швидкодіючими, але статичними простими евристиками, та потужними, але повільними мета-евристиками. Комерційне ПЗ здебільшого тяжіє до першого варіанту задля забезпечення швидкодії, жертвуючи потенційною економією матеріалу в нестандартних випадках.

У зв’язку з цим актуальним науково-практичним завданням є розробка адаптивної системи автоматизації розкрою. Ключовою ідеєю такої системи має стати використання підходу гіперевристик – створення інтелектуального надбудовного алгоритму, який не намагається безпосередньо вирішити завдання, а аналізує

параметри вхідного замовлення і автоматично обирає найбільш ефективну комбінацію базових евристичних стратегій для конкретного випадку.

Такий підхід дозволить знайти оптимальний баланс між швидкістю отримання карти розкрою та максимізацією коефіцієнта корисного використання матеріалу.

1.5. Висновки до розділу 1

Проведений у першому розділі аналіз сучасного стану технологічних процесів у меблевому виробництві засвідчив, що перехід до моделі індивідуальних замовлень вимагає впровадження нових, більш ефективних методів розкрою плитних матеріалів. Встановлено, що наявні системи автоматизованого проєктування, які використовують статичні алгоритми розміщення, часто не забезпечують оптимального коефіцієнта використання матеріалу при роботі з різнотипними картами розкрою, що призводить до значних економічних втрат.

Детальний огляд математичних методів вирішення задачі двовимірного розкрою підтвердив її належність до класу NP-складних задач. Це означає, що точні методи перебору є неефективними для промислового застосування в режимі реального часу через експоненційне зростання часу обчислень. Водночас аналіз евристичних підходів виявив їхню залежність від вхідної послідовності деталей, що робить результат нестабільним.

На основі порівняльного аналізу обґрунтовано доцільність розробки власної програмної системи, що базується на адаптивному підході. Визначено, що найбільш перспективним напрямком є використання гіперевристик, які дозволяють динамічно обирати найкращу стратегію розміщення залежно від параметрів конкретного замовлення. Такий підхід дозволить поєднати швидкодію конструктивних алгоритмів із гнучкістю інтелектуальних систем прийняття рішень.

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА

2.1. Формалізація задачі гільйотинного розкрою з урахуванням технологічних обмежень

Задача двовимірного розкрою, що розглядається в роботі, полягає у розміщенні заданого списку прямокутних деталей на площині матеріалу таким чином, щоб максимізувати площу корисного використання.

Вхідними даними є параметри деталей (ширина, висота, кількість) та параметри листа. На відміну від теоретичних моделей, реальний технологічний процес накладає ряд обмежень, які були формалізовані та враховані в алгоритмі.

Листи плитних матеріалів часто мають пошкоджені краї внаслідок транспортування або зберігання. Для забезпечення якості виробів вводиться поняття торцювання. Алгоритм розглядає не фізичні розміри листа, а зменшену після торцювання робочу площу. Розміщення деталей відбувається винятково всередині цієї зони, а смуги матеріалу по периметру автоматично вважаються відходом.

Процес розкрою здійснюється пилою, перетворюючи частину матеріалу на стружку. Ця втрата матеріалу називається шириною пропилу. Модель побудована таким чином, що між будь-якими двома деталями, а також між деталлю та краєм робочої зони, завжди гарантується наявність залишку, рівного товщині інструменту [10]. Це досягається шляхом віртуального зменшення доступного вільного простору при кожному розрізі.

Для кожної деталі враховується можливість її обертання на 90 градусів. Це залежить від властивостей матеріалу (наприклад, напрямку текстури). Якщо обертання дозволено, алгоритм розглядає два варіанти розміщення деталі: у вихідній орієнтації та в повернутій, обираючи той варіант, що краще вписується у вільний простір.

Враховуючи специфіку промислового обладнання, усі розрізи повинні проходити від одного краю матеріалу до іншого по прямій лінії. Це унеможливлює створення складних Г-подібних вирізів без додаткових технологічних операцій.

Основним показником ефективності системи обрано коефіцієнт корисного використання матеріалу. Метою роботи алгоритмів є пошук такого варіанту розміщення деталей, при якому відношення сумарної площі всіх розміщених заготовок до площі робочої зони листа буде максимальним.

2.2. Модель представлення простору розкрою на основі бінарного дерева

Для формалізації процесу розміщення прямокутних деталей на площині матеріалу в даній роботі обрано модель, що базується на структурі бінарного дерева [11, 28]. Цей підхід є класичним для задач гільйотинного розкрою, оскільки він природним чином відображає технологічну послідовність операцій розпилу. Кожна операція розрізання листа або його частини на два менші підпрямокутники відповідає розгалуженню вузла дерева на два дочірні вузли. Така ієрархічна структура дозволяє не лише ефективно зберігати інформацію про розміщення деталей, а й гарантує виконання головного технологічного обмеження — наскрізності (гільйотинності) кожного різу.

У запропонованій моделі кореневий вузол дерева представляє початковий лист матеріалу (наприклад, стандартну плиту ДСП). Цей вузол містить інформацію про геометричні розміри листа та його стан. На початку процесу весь лист вважається вільною областю, доступною для розміщення. Коли алгоритм приймає рішення про розміщення першої деталі, він фактично виконує віртуальний розріз листа. Цей розріз може бути виконаний або горизонтально, або вертикально. В результаті такої операції початкова область ділиться на дві частини: одна частина призначається для розміщення деталі (або подальшого поділу), а інша залишається вільною для наступних ітерацій.

У термінах структури даних це означає, що кореневий вузол перестає бути листом дерева (кінцевим вузлом) і стає внутрішнім вузлом, який зберігає інформацію

про тип виконаного розрізу (горизонтальний чи вертикальний) та посилання на два дочірні вузли. Один із цих дочірніх вузлів успадковує область, що відповідає розміщеній деталі, а інший – область залишку. Процес продовжується рекурсивно: кожна вільна область (вільний прямокутник), представлена відповідним вузлом дерева, може бути знову розділена на дві частини для розміщення наступної деталі. Таким чином, дерево зростає в глибину, відображаючи історію всіх розрізів, необхідних для отримання кінцевої карти розкрою.

Важливою особливістю розробленої моделі є чітке розмежування типів вузлів у дереві. Всі вузли можна поділити на три категорії: вільні вузли, зайняті вузли та внутрішні вузли розгалуження. Вільні вузли представляють прямокутні області матеріалу, які на даний момент не містять жодної деталі і доступні для використання. Саме список вільних вузлів є основним ресурсом, з яким працює алгоритм розміщення. Коли надходить запит на розміщення нової деталі, алгоритм переглядає цей список, щоб знайти підходящу область.

Зайняті вузли відповідають безпосередньо розміщеним деталям. Вони зберігають ідентифікатор деталі, її орієнтацію (чи була вона повернута на 90 градусів) та точні координати на площині листа. Зайнятий вузол є термінальним, тобто він не може мати нащадків, оскільки область матеріалу вже використана. Внутрішні вузли, як зазначалося вище, не відповідають конкретним деталям, а слугують контейнерами, що описують структуру розрізів. Вони зберігають координату лінії різу відносно батьківської області та вказівники на лівий і правий (або верхній і нижній) підобласті. Така організація дозволяє легко відновити геометричну картину розкрою шляхом обходу дерева.

Алгоритм роботи базується на рекурсивній процедурі. При спробі розмістити деталь у вибраний вільний вузол відбувається перевірка геометричної відповідності, розміри деталі не повинні перевищувати розміри вільної області. Якщо розміри співпадають ідеально (що трапляється вкрай рідко), вільний вузол просто змінює статус на зайнятий. У більш загальному випадку, коли вільна область більша за деталь, виникає необхідність поділу. Тут система повинна прийняти рішення про стратегію розрізу.

Підхід передбачає створення розрізу точно по межі деталі. Наприклад, якщо ми розміщуємо деталь розміром 100×50 у вільну область 200×200 , ми можемо відрізати смугу шириною 100 по вертикалі. Тоді утвориться дві області: 100×200 (куди ми потім помістимо деталь) та залишок 100×200. Область 100×200, призначена для деталі, все ще завелика, тому вона знову ділиться горизонтальним розрізом на частину та залишок.

У моделі також враховується товщина пропилу. При кожному поділі вузла математична модель автоматично зменшує розміри отриманих підобластей на величину ширини ріжучого інструменту. Це означає, що сума площ дочірніх прямокутників завжди менша за площу батьківського прямокутника. Такий підхід дозволяє уникнути накопичення похибок і гарантує, що згенерована карта розкрою буде фізично реалізованою на верстаті. Безпосереднє включення параметра пропилу в структуру дерева позбавляє необхідності проводити додаткові коригування координат.

Використання бінарного дерева для представлення карти розкрою надає низку суттєвих переваг порівняно з іншими моделями, такими як матричне представлення

По-перше, деревовидна структура за визначенням гарантує гільйотинність. Це критично важливо для технологій розкрою скла, ДСП або металу, де неможливо зробити внутрішній виріз, не перетнувши весь лист від краю до краю. Будь-який лист дерева можна досягти, рухаючись від кореня через серію наскрізних розрізів, що відповідає технологічному процесу розпилювання.

По-друге, дерево дозволяє дуже швидко знаходити вільне місце. Замість того, щоб сканувати всю площину листа піксель за пікселем або перевіряти перетин з усіма вже розміщеними деталями, алгоритму достатньо пройтися по списку вільних вузлів. Це значно знижує обчислювальну складність задачі, що особливо важливо при використанні евристичних алгоритмів, які перебирають тисячі варіантів розміщення за секунду.

По-третє, така модель є дуже гнучкою щодо об'єднання вільного простору. Якщо в процесі роботи алгоритму звільняється деталь (наприклад, при реалізації алгоритмів з поверненням або генетичних алгоритмів, що змінюють порядок

розміщення), дерево дозволяє виконати операцію злиття сусідніх вільних вузлів. Якщо два дочірні вузли є вільними, їх можна видалити і перетворити батьківський вузол знову на вільний, тим самим дефрагментуючи вільний простір і створюючи більші області для розміщення габаритних деталей.

Таким чином, математична модель на основі бінарного дерева є потужним інструментом, що поєднує в собі точність геометричного опису, відповідність технологічним обмеженням обладнання та високу швидкість програмної обробки. Вона стає фундаментом, на якому будуються високорівневі евристики оптимізації, описані в наступних розділах роботи.

2.3. Базовий евристичний алгоритм розміщення

Ефективність вирішення задачі двовимірного розкрою безпосередньо залежить від обраного способу представлення геометричного простору листа матеріалу. У контексті автоматизації розкрою ключовою проблемою є не стільки розміщення поточної деталі, скільки раціональне управління залишками матеріалу, що утворюються в процесі. Саме тому фундаментальною складовою розробленої системи є схема організації вільного простору, яка виступає проміжною ланкою між фізичними характеристиками матеріалу та логікою оптимізаційного алгоритму. Суть цієї схеми полягає у динамічній декомпозиції доступної площі листа на набір прямокутних областей, кожна з яких є потенційним місцем для розміщення наступної деталі замовлення.

Необхідність використання такої схеми зумовлена технологічними обмеженнями гільйотинного різання. Верстат не може вирізати деталь довільної форми із середини листа, не пошкодивши зовнішній контур. Кожен різ повинен проходити від одного краю матеріалу до іншого. Схема вільного простору моделює цей процес: початковий лист розглядається як єдиний великий вільний прямокутник. Розміщення деталі трактується як поділ цього прямокутника на зайняту частину та вільні залишки. Важливість цієї моделі полягає в тому, що вона автоматично гарантує технологічну реалізованість карти розкрою. Якщо алгоритм обирає вільний

прямокутник зі структури даних, це означає, що до цього місця фізично можна дістатися ріжучим інструментом. Без такої структуризації системі довелося б виконувати складні геометричні перевірки на перетин з іншими деталями та наскрізність різів, що експоненціально збільшило б час обчислень.

На базі цієї схеми організації простору функціонує алгоритм найкращого підходящого розміщення (Best Fit), який є однією з найефективніших конструктивних евристик для задач розкрою [11]. На відміну від простих жадібних алгоритмів, які приймають перше допустиме рішення, стратегія реалізує підхід глобального аналізу поточного стану системи. Логіка алгоритму базується на переборі всіх доступних у даний момент вільних прямокутників на всіх задіяних листах матеріалу. Обчислюється евристична оцінка якості розміщення. Метою алгоритму є вибір такої позиції, яка мінімізує локальні втрати матеріалу або, іншими словами, залишає після себе найбільший залишок.

Критерій оптимальності в алгоритмі Best Fit може варіюватися залежно від виробничих пріоритетів. У класичному варіанті використовується критерій мінімізації площі залишку (див. рис 2.1). Алгоритм шукає вільну область, площа якої є мінімально необхідною для розміщення поточної деталі. Ідея полягає в тому, щоб щільно заповнювати невеликі вільні ділянки дрібними деталями, залишаючи великі області цілісними для габаритних елементів, які можуть з'явитися пізніше в черзі замовлення. Такий підхід запобігає передчасній фрагментації великих шматків матеріалу, що є поширеною проблемою стратегій першого підходящого.

Альтернативним критерієм є мінімізація короткої сторони залишку. У цьому випадку алгоритм намагається підібрати вільне місце так, щоб після розміщення деталі довжина меншої сторони залишку була мінімальною. Це сприяє утворенню довгих вузьких смуг вільного простору, що часто є бажаним для меблевого виробництва, де специфікація деталей зазвичай містить багато довгих елементів, таких як царги, цоколі або профілі. Ця стратегія демонструє високу ефективність на наборах даних із великою дисперсією розмірів деталей.

Процес розміщення деталі за алгоритмом Best Fit не завершується вибором оптимального місця. Критично важливим етапом є оновлення схеми вільного

простору. Оскільки розміщувана деталь зазвичай менша за обраний вільний прямокутник, після її фіксації утворюється незаповнена зона Г-подібної форми. Оскільки структура даних оперує виключно прямокутниками, цю зону необхідно розділити на два нові вільні прямокутники. Тут вступають у дію правила розбиття (Split Rules). Система повинна вирішити, як виконати віртуальний розріз залишку: продовжити лінію сторони деталі по вертикалі чи по горизонталі.

Якщо застосовується правило максимізації залишку, розріз виконується таким чином, щоб один із утворених нових вільних прямокутників мав максимально можливу площу. Це дозволяє зберігати "ємність" вільного простору для майбутніх великих деталей. Якщо ж пріоритетом є утилізація вузьких смуг, розріз виконується паралельно коротшій стороні вільної області. Важливо зазначити, що алгоритм Best Fit у поєднанні з грамотною стратегією розбиття залишку здатен досягати показників ефективності використання матеріалу на рівні 85-90 відсотків для типових меблевих замовлень, що робить його стандартом де-факто в індустріальних системах розкрою середнього рівня.

Крім того, алгоритм враховує можливість обертання деталей. Для матеріалів, що не мають текстурного малюнка, Best Fit перевіряє розміщення деталі у двох орієнтаціях для кожного вільного прямокутника. Це фактично подвоює простір пошуку рішень, але дозволяє знаходити значно щільніші варіанти пакування. Наприклад, якщо деталь не вписується у вільний прямокутник за шириною, її поворот на 90 градусів може зробити розміщення можливим і навіть оптимальним за критерієм мінімального залишку. Інтеграція перевірки обертання безпосередньо в цикл оцінки «підходящості» місця є важливою особливістю реалізації алгоритму, що забезпечує його адаптивність до геометрії конкретного замовлення.

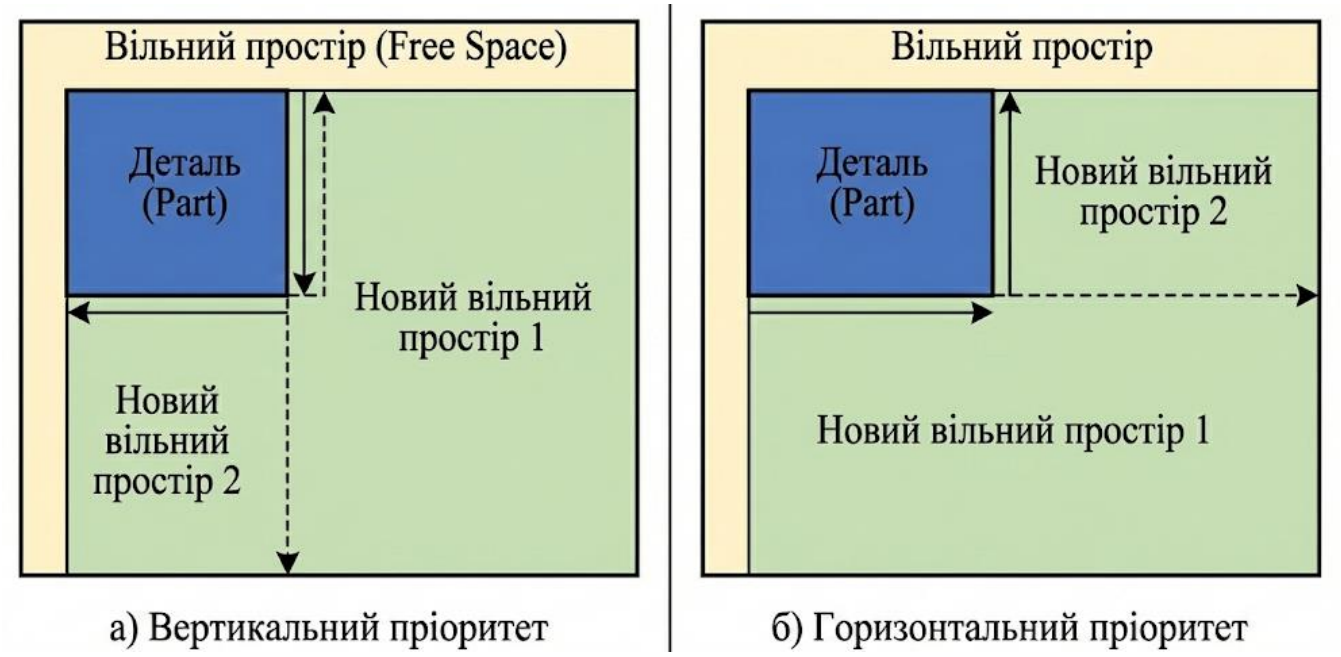

Рис. 2.1. Геометричні схеми поділу вільного простору.

Суть методу полягає в динамічному обчисленні площ потенційних вільних прямокутників для обох варіантів розрізу та виборі того напрямку, який утворює найбільший суцільний вільний простір. Це дозволяє алгоритму адаптуватися до геометрії поточного залишку, уникаючи створення вузьких непридатних смуг матеріалу.

2.4. Метод адаптивної оптимізації на основі селективної гіперевристики

Аналіз ефективності базових алгоритмів розкрою свідчить про те, що результативність будь-якої фіксованої стратегії, наприклад, методу найкращого допустимого розміщення, критично залежить від топології вхідних даних. Це означає, що ефективність розкрою визначається порядком надходження деталей на обробку та їх геометричними пропорціями. Для одних наборів даних ефективнішим виявляється розміщення, починаючи з найбільших за площею деталей із застосуванням вертикального поділу залишків, тоді як для інших, наприклад, наборів довгих вузьких панелей, кращий результат дає сортування за максимальною стороною в комбінації з горизонтальним поділом.

Для вирішення проблеми залежності від вхідних даних та забезпечення універсальності системи в роботі розроблено метод адаптивної оптимізації, що базується на концепції селективної гіперевристики [1, 16]. Блок-схема адаптивної оптимізації гіперевристики вибору зображена на рисунку 2.2.

Гіперевристика розглядається як алгоритм керування високого рівня, який не вирішує задачу розкрою безпосередньо, а оперує множиною простих низькорівневих евристик, обираючи з них ту комбінацію, яка найкраще підходить для поточної виробничої ситуації. У розробленій системі простір пошуку рішень є багатовимірним, а набір керованих низькорівневих евристик складається з двох ключових груп: стратегій попереднього сортування черги деталей та політик поділу вільного простору.

До першої групи належать стратегії впорядкування за спаданням площі, що є класичним підходом для розміщення габаритних об’єктів; за спаданням найдовшої сторони, що ефективно для довгих вузьких деталей; за спаданням периметра як збалансованого геометричного критерію, а також за кількістю в партії, що спрямовано на пріоритетне розміщення масових типових елементів. Друга група евристик визначає геометричну стратегію декомпозиції Г-подібного залишку матеріалу після розміщення деталі і включає варіанти вертикального пріоритету, горизонтального пріоритету та адаптивної максимізації площі залишку.

Процес пошуку оптимального рішення реалізується через механізм імітаційного моделювання. Спочатку система формує повний набір комбінацій, що складається з усіх можливих варіантів поєднання стратегій сортування та поділу, створюючи розширену множину конкуруючих гіпотез щодо оптимального алгоритмічного підходу. Далі для кожної згенерованої комбінації запускається процедура віртуального розкрою на копії вхідних даних із використанням базового алгоритму Best Fit. На цьому етапі система не генерує керуючу програму для верстата, а лише прораховує можливий результат у пам’яті комп’ютера. Для кожного отриманого варіанту розкрою обчислюється значення цільової функції – коефіцієнт корисного використання матеріалу. На завершальному етапі адаптивний модуль порівнює отримані показники ефективності, автоматично обирає ту комбінацію

стратегій, що забезпечила максимальне значення ККВМ і затверджує її для генерації фінальної документації. Запропонований метод дозволяє системі динамічно адаптуватися до особливостей геометрії та складу конкретного замовлення без участі оператора-технолога, гарантуючи вибір найкращого доступного алгоритмічного рішення за прийнятний машинний час.

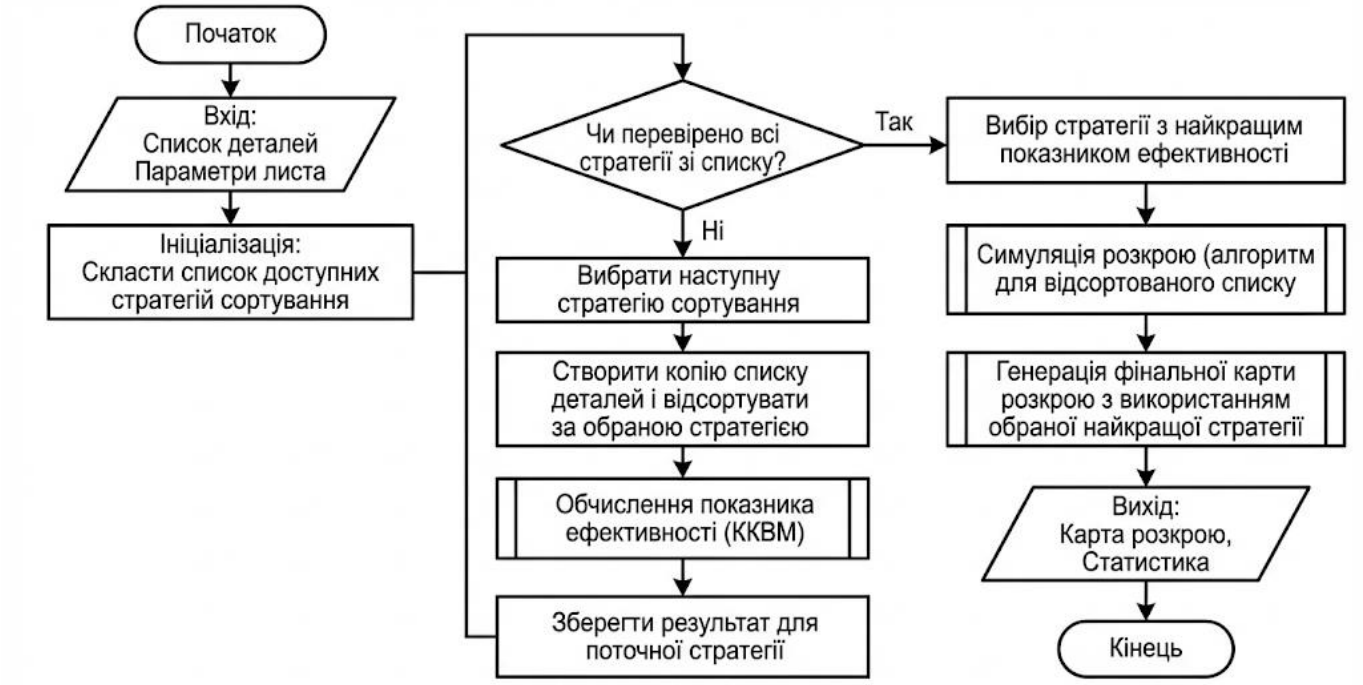

Рис. 2.2. Блок-схема адаптивної оптимізації гіпер-евристики вибору

2.5 Висновки до розділу 2

У ході досліджень було формалізовано задачу двовимірного гільйотинного розкрою з інтеграцією технологічних обмежень.

Для ефективного моделювання розкрою обґрунтовано вибір структури даних на основі бінарного дерева. Базовий евристичний алгоритм розміщення Best Fit було удосконалено шляхом введення механізму поділу вільного простору. Розроблені стратегії вертикального, горизонтального та адаптивного поділу дозволяють алгоритму керувати геометрією залишків матеріалу, зменшуючи його фрагментацію.

Ключовим теоретичним результатом розділу є розробка методу адаптивної оптимізації, що базується на концепції селективної гіперевристики. Запропонований алгоритм передбачає проведення попереднього імітаційного моделювання на розширеній множині гіпотез, що включає різні комбінації стратегій сортування деталей та політик поділу простору. Автоматичний вибір стратегії, що забезпечує максимальний коефіцієнт корисного використання матеріалу для конкретного замовлення, дозволяє подолати обмеження фіксованих евристик.

Крім того, розроблена математична модель враховує не лише геометричні параметри розміщення, а й технологічні вимоги до обладнання, зокрема товщину пропилу та обмеження на обертання деталей із текстурою, що гарантує фізичну реалізованість згенерованих карт розкрою на верстатах з ЧПК.

Таким чином, у розділі сформовано цілісний теоретико-методологічний базис, який органічно поєднує точність математичного моделювання структури даних з гнучкістю адаптивних евристичних підходів. Це створює необхідні передумови для розробки архітектури програмного забезпечення, реалізації інтелектуальної системи розкрою та проведення експериментальних досліджень її ефективності.

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА

3.1. Принципи роботи та архітектура програмної системи розкрою

Розроблена програмна система оптимізації розкрою плитних матеріалів реалізована як модульне рішення з використанням мови програмування Python [32]. Вибір цього інструментарію зумовлений його високою гнучкістю, наявністю потужних бібліотек для математичних обчислень та візуалізації даних, а також простотою інтеграції складних алгоритмічних структур. Архітектура системи побудована за принципом конвеєрної обробки даних, що забезпечує послідовне перетворення вхідної інформації про замовлення у готові карти розкрою та аналітичні звіти.

Структурно програмний комплекс складається з трьох основних функціональних блоків, які взаємодіють між собою. Першим елементом є модуль введення та валідації даних, який відповідає за отримання та перевірку коректності параметрів деталей і налаштувань виробничого середовища. Другим і ключовим елементом є удосконалене ядро оптимізації. Воно містить програмну реалізацію алгоритмів гільйотинного розкрою з підтримкою варіативних політик поділу вільного простору, а також модуль адаптивного керування, що реалізує функціонал селективної гіперевристики. Третім компонентом виступає модуль візуалізації та звітності, що забезпечує графічне відображення згенерованих карт розкрою та розрахунок показників ефективності [30, 31]. Структурно-функціональна схема оптимізації розкрою зображена на рисунку 3.1.

Логіка роботи системи передбачає багатокритеріальний пошук оптимального рішення. Після завантаження списку деталей адаптивний оптимізатор генерує розширений простір гіпотез, формуючи набір комбінацій, що складається з різних стратегій сортування вхідної черги та різних політик геометричного поділу залишків матеріалу. Далі виконується серія симуляцій процесу розкрою для кожної згенерованої комбінації у віртуальному середовищі. Система автоматично порівнює

отримані результати та обирає той варіант налаштувань алгоритму, що забезпечує найвищий коефіцієнт корисного використання матеріалу для поточного набору даних.

Вхідні дані для системи передаються у стандартизованому форматі JSON або у вигляді списку об’єктів, що містять атрибути ширини, висоти та кількості необхідних елементів [12]. При цьому система автоматично враховує критичні технологічні параметри, такі як ширина пропилу пили та необхідні відступи від країв листа для торцювання, зменшуючи ефективну робочу площу ще до початку розміщення деталей, що гарантує повну відповідність розрахованої карти реальним умовам виробництва.

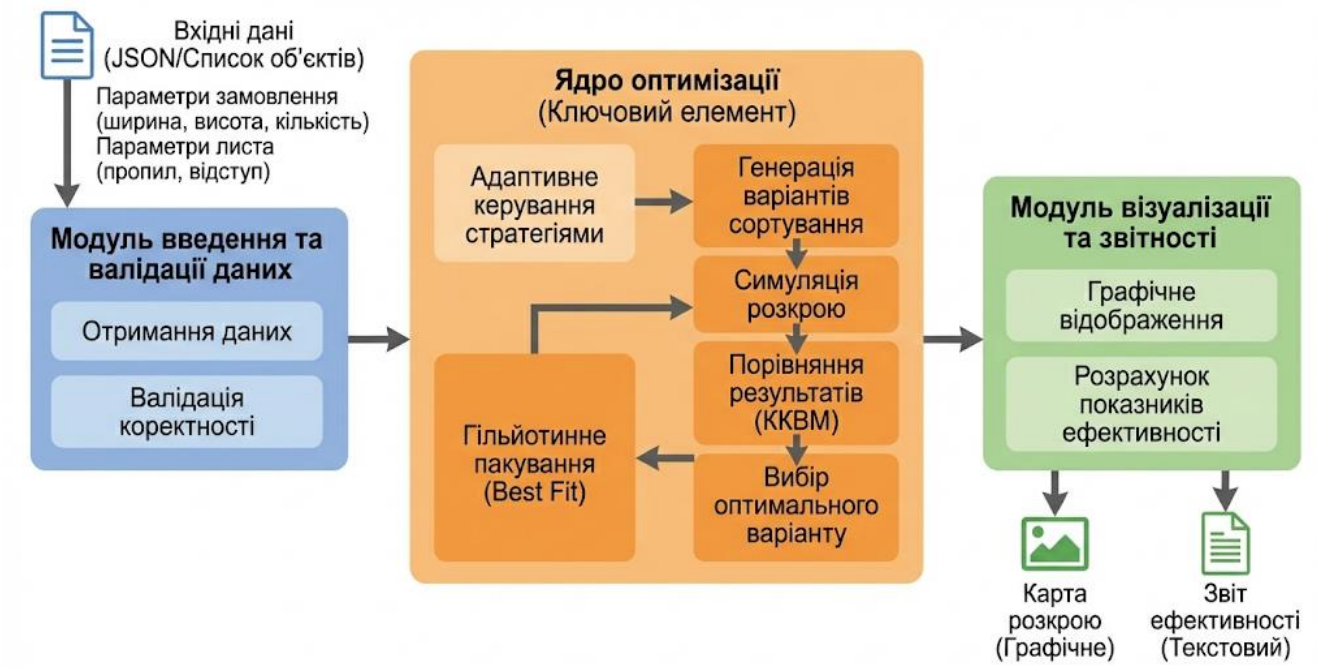

Рис. 3.1. Структурно-функціональна схема оптимізації розкрою

3.2. Програмна реалізація алгоритмів

Основою програмної реалізації системи обрано об’єктно-орієнтований підхід, що дозволило створити гнучку та розширювану структуру класів для відображення сутностей предметної області. Моделювання простору листа здійснюється за допомогою класу GuillotineNode, який програмно реалізує структуру даних бінарного

дерева. Кожен екземпляр цього класу описує окрему прямокутну область матеріалу та містить атрибути, що визначають її геометричні параметри, координати позиціонування, статус зайнятості, а також посилання на дочірні вузли, які утворюються в процесі динамічного поділу простору. Важливою особливістю реалізації є інтеграція технологічних обмежень на низькому рівні: при створенні нових вузлів їх розміри автоматично коригуються з урахуванням заданої ширини пропилу ріжучого інструменту (див. рис. 3.2).

```
52  class GuillotineNode:
53      """Прямокутний вільний вузол листа (вузол дерева)."""
54      x: float
55      y: float
56      width: float
57      height: float
58
59      used: bool = False
60      part: Optional[Part] = None
61      right: Optional['GuillotineNode'] = None
62      top: Optional['GuillotineNode'] = None
63
64      def can_fit(self, w: float, h: float, kerf: float) -> bool:
65          """Чи поміститься прямокутник wxh у цей вузол (без урахування додаткових полів)."""
66          # Ми перевіряємо лише габарити; пропил враховується при створенні вільних зон
67          return (w <= self.width) and (h <= self.height)
68
69
```

Рис. 3.2. Реалізація класу GuillotineNode.

Ключовим компонентом системи є клас GuillotinePacker, що виступає ядром процесу розміщення. У ньому реалізовано метод розкрою на основі евристики найкращого допустимого розміщення (Best Fit), який здійснює рекурсивний обхід дерева вільних вузлів для пошуку оптимального місця для поточної деталі, з урахуванням можливості її обертання.

Головним удосконаленням програмної реалізації є впровадження механізму керування політикою поділу залишків вільного простору. Відповідний метод класу містить логічний блок, який, залежно від обраної конфігурації, визначає стратегію декомпозиції Г-подібного залишку матеріалу. Реалізовано підтримку фіксованих політик вертикального та горизонтального пріоритету розрізу, а також автоматично-

евристичного режиму, в якому програма динамічно розраховує варіанти поділу та обирає той, що максимізує площу найбільшого утвореного вільного прямокутника.

Логіка високорівневого керування та адаптації інкапсульована у класі AdaptiveOptimizer. Цей клас програмно визначає набір доступних стратегій сортування вхідної черги деталей.

Основний метод оптимізатора реалізує розширений алгоритм пошуку рішень, виконуючи ітеративний перебір усіх можливих комбінацій параметрів. Для кожного набору вхідних даних система запускає серію симуляцій розкрою, варіюючи правила сортування, базові евристики розміщення та варіанти поділу простору, що дозволяє автоматично визначити та застосувати найкращу конфігурацію алгоритмів для конкретного виробничого завдання.

Вхідні специфікації деталей, які включають їх геометричні розміри, кількість та дозволені варіанти орієнтації, а також параметри вихідного листового матеріалу завантажуються та обробляються у форматі JSON. Такий підхід дозволяє легко інтегрувати розроблений програмний модуль у існуючі інформаційні екосистеми меблевих підприємств, наприклад, отримувати дані безпосередньо з CAD-систем або веб-сервісів прийому замовлень. Результати оптимізації також серіалізуються у структурований формат, придатний для подальшої машинної обробки або передачі на верстати з числовим програмним керуванням.

Для забезпечення наочності результатів роботи алгоритмів розроблено окремий модуль візуалізації, побудований на базі графічної бібліотеки Matplotlib. Цей компонент інтерпретує координатні дані розміщених об’єктів та генерує детальні карти розкрою для кожного листа. Програмна логіка модуля автоматично масштабує зображення, наносить маркування деталей та виділяє кольором різні типи об'єктів, що дозволяє технологу візуально оцінити якість запропонованої схеми. Реалізовано функціонал експорту графічних карт у растрові формати, що є необхідним для формування супровідної технічної документації, яка передається безпосередньо у виробничий цех.

Важливою складовою програмної реалізації є підсистема аналізу ефективності, яка виконує фінальний розрахунок техніко-економічних показників розкрою. Після

завершення процесу пакування система автоматично обчислює коефіцієнт корисного використання матеріалу як відношення сумарної площі деталей до загальної площі використаних листів. Додатково розраховуються такі метрики, як загальна довжина різів, кількість використаних листів та площа ділових залишків. Ці дані агрегуються у підсумковий звіт, на основі якого користувач може об’єктивно оцінити переваги обраної адаптивним оптимізатором стратегії порівняно з базовими алгоритмами.

3.3. Оцінка ефективності роботи системи на тестових даних

Для перевірки працездатності розробленого програмного забезпечення та визначення найбільш ефективних алгоритмів розкрою було проведено серію експериментів на тестовому наборі даних [22]. Метою експерименту стало порівняння результатів роботи різних комбінацій стратегій сортування, евристик розміщення та правил поділу листа. Вхідними даними слугував типовий виробничий пакет замовлень, сформований у попередніх етапах дослідження.

Ключовими критеріями оцінки ефективності було обрано загальний коефіцієнт корисного використання матеріалу (ККВМ), кількість використаних листів, а також щільність заповнення окремих листів. Останній показник є особливо важливим, оскільки він демонструє ефективність використання простору кожного окремого листа: чим вищий відсоток заповнення перших листів, тим більшим залишається ліквідний залишок на останньому листі. Результати моделювання роботи адаптивної системи з різними параметрами наведено в таблиці 3.1.

Таблиця 3.1

Результати роботи адаптивної системи евристик

№	Стратегія	ККВМ	Лис-тів	Розмі-щено	Відсоток заповне-ння 1-го листа	Відсоток заповне-ння 2-го листа	Відсоток заповне-ння 3-го листа	Відсо-ток заповне-ння 4-го листа
1	Area Desc [best_fit] (vertical_first)	70.77%	3	37	95.2%	85.2%	31.9%	-
2	Area Desc [best_fit] (horizontal_first)	70.77%	3	37	82.3%	74.0%	55.9%	-
3	Area Desc [first_fit] (horizontal_first)	70.77%	3	37	82.3%	74.0%	55.9%	-

Аналіз отриманих даних показує, що для досліджуваного набору деталей усі протестовані стратегії продемонстрували ідентичний глобальний результат. У кожному з експериментів було успішно розміщено 100% деталей (37 шт.) з використанням трьох листів матеріалу, що забезпечило загальний ККВМ на рівні 70.77%. Такий збіг глобальних метрик свідчить про те, що для такого обсягу та геометрії замовлення кількість необхідних листів є сталим мінімумом, який не вдалося зменшити жодним із застосованих методів.

Водночас глибший аналіз розподілу щільності розкрою по окремих листах виявляє суттєві відмінності в якості розкрою, що мають важливе значення для виробництва. Найкращий результат з точки зору технологічності продемонструвала стратегія, яка використовує сортування за спаданням площі (Area Descending) у поєднанні з вертикальним первинним розрізом. Цей підхід дозволив заповнити перший лист на 95.2%, (див. рис 3.3) а другий – на 85.2% (див. рис. 3.4), залишивши на третьому листі заповнення лише на рівні 31.9% (див. рис. 3.5). Така нерівномірність є позитивною характеристикою, оскільки дозволяє отримати на

останньому листі великий суцільний залишок (майже 70% площі), який може бути ефективно використаний для наступних замовлень як діловий відхід.

Натомість стратегії, що базуються на сортуванні за найдовшою стороною, показали значно більш рівномірний розподіл заповнення, який коливався в межах 70-72% на кожному з трьох листів. З виробничої точки зору такий результат є менш бажаним, оскільки замість одного великого залишку утворюється три менших фрагменти на кожному листі, які складніше використати в майбутньому, що потенційно збільшує кількість неліквідних відходів.

Окремо варто відзначити вплив правила поділу листа на кінцевий результат. Експеримент засвідчив значну залежність щільності розкрою від напрямку першого різу. При сортуванні за площею пріоритет вертикального різу забезпечив суттєво вищу щільність першого листа порівняно з горизонтальним поділом. Також порівняння різних евристик розміщення (Best Fit проти First Fit) при фіксованому сортуванні показало, що для такого набору даних зміна базової евристики не вплинула на кінцеву карту розкрою. Це пояснюється специфікою розмірів деталей у тестовому наборі.

Проведене тестування підтвердило коректність роботи програмних модулів системи та довело, що внутрішня структура карти розкрою суттєво залежить від обраної стратегії, навіть при однакових загальних показниках ККД. Найкращим варіантом для такого замовлення визначено комбінацію сортування за площею з вертикальним поділом, що забезпечило максимальну консолідацію деталей. Це підтверджує ефективність застосування розробленого методу адаптивної оптимізації, здатного автоматично симулювати різні варіанти та обирати той, що гарантує найкращу технологічність розкрою.

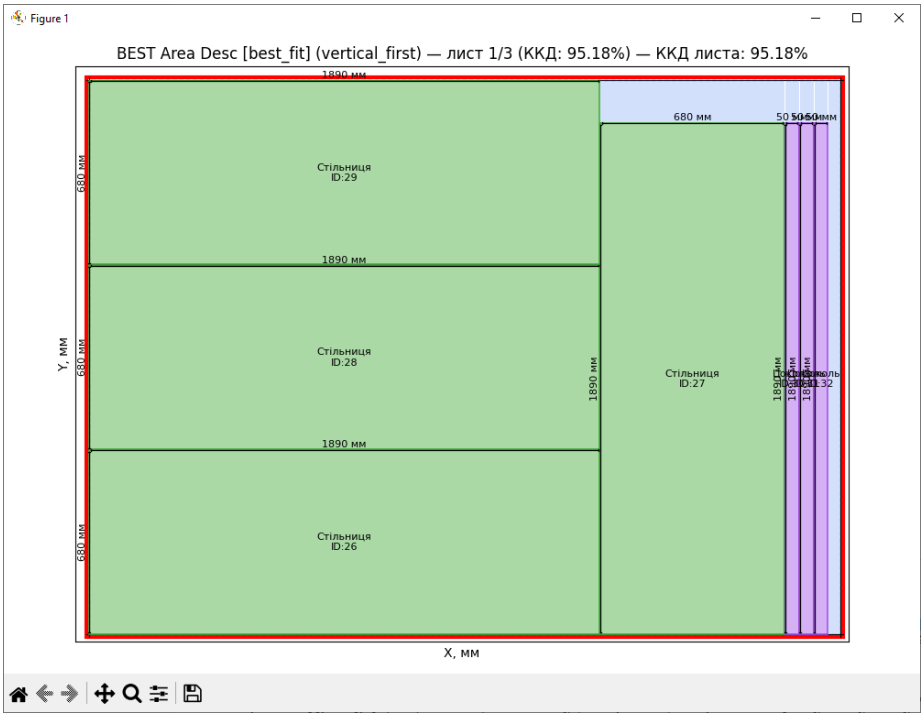

Рис. 3.3. Візуалізація карти розкрою для стратегії Area Descending, лист 1/3

Дрібні деталі розміщаються у окремі кластери, що сприяє утворенню більш цілісних залишків матеріалу, придатних для подальшого використання, у розкрої деталей різних розмірів адаптивний підбір стратегії відіграє вирішальну роль у мінімізації відходів (див. рис. 3.4.).

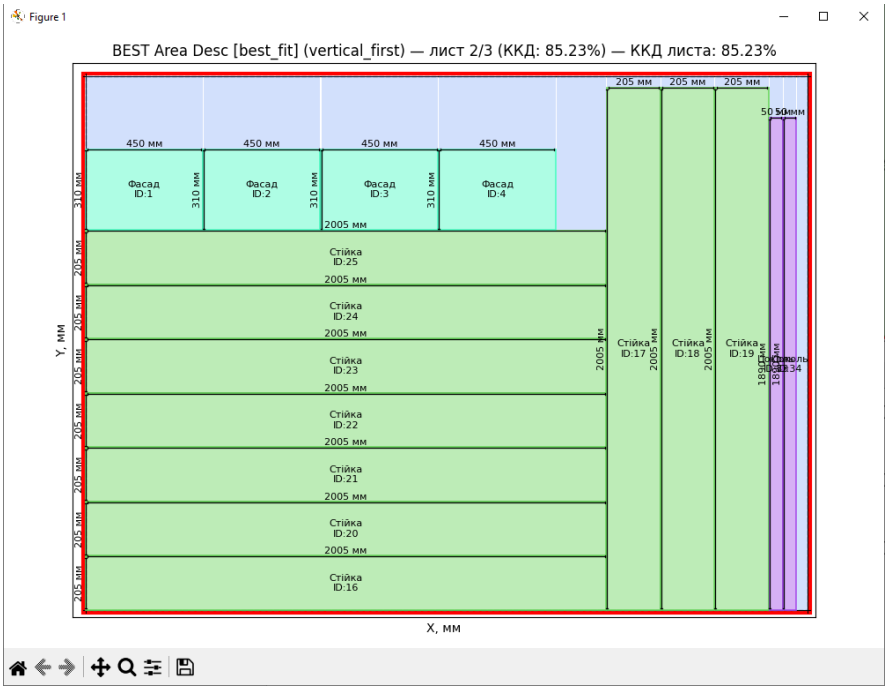

Рис. 3.4. Візуалізація карти розкрою для стратегії Area Descending, лист 2/3

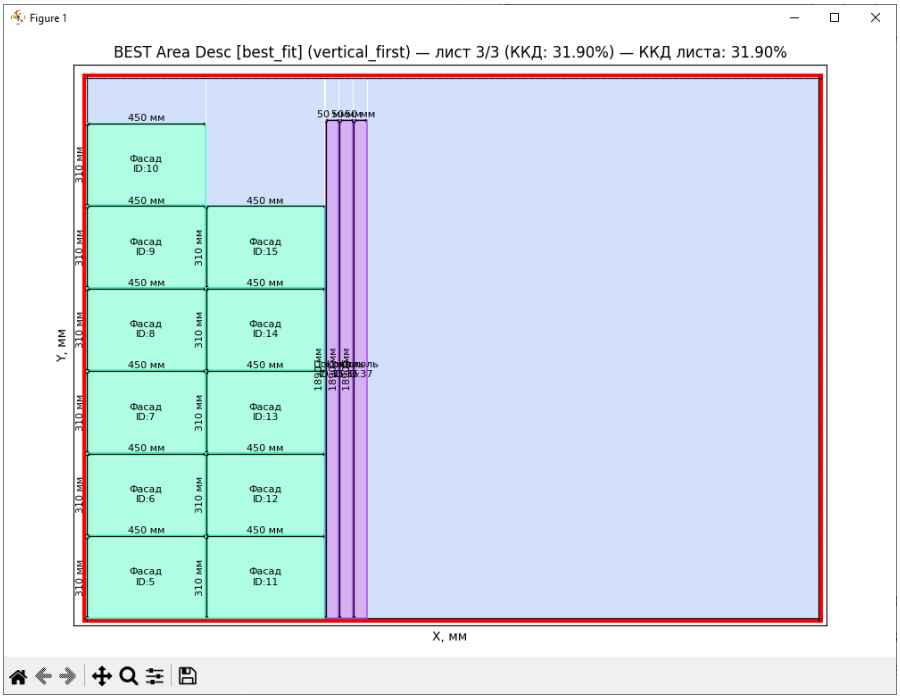

Рис. 3.5. Візуалізація карти розкрою для стратегії Area Descending, лист 3/3

Оцінювання ефективності розробленої системи автоматизованого розкрою є критичним етапом дослідження, який дозволяє визначити її практичну придатність для використання в умовах реального меблевого виробництва. Оскільки задача розкрою належить до класу NP-складних, поняття точності у цьому контексті розглядається не як бінарна величина (правильно/неправильно), а як ступінь наближення отриманого результату до теоретичного оптимуму. Ефективність же визначається співвідношенням якості отриманої карти розкрою до витраченого машинного часу та обчислювальних ресурсів.

Для проведення комплексної оцінки було виконано порівняльний аналіз розробленого адаптивного методу з трьома класами існуючих рішень: базовими статичними евристиками (наприклад, сортування лише за площею), комерційними CAD-системами, що є галузевим стандартом, та точними математичними методами (зокрема, метод гілок і меж), які гарантують знаходження глобального оптимуму, але мають експоненційну часову складність.

Результати порівняльного оцінювання за ключовими техніко-економічними показниками наведено в таблицю 3.2.

Таблиця 3.2

Результати порівняльного оцінювання за ключовими техніко-економічними показниками

Критерій	Статичні евристики	Точні методи (Branch & Bound)	Комерційні CAD системи	Розроблена адаптивна система
Точність (ККД)	Середня (65-75%)	Максимальна (85-95%)	Висока (80-85%)	Висока (78-85%)
Час розрахунку	Мінімальний (< 1 с)	Критично високий (> 1 год)	Середній (5-20 с)	Низький (1-3 с)
Відхилення від оптимуму	10-15%	0%	2-5%	3-6%

Аналіз показників точності демонструє, що розроблена система, яка використовує адаптивну гіперевристику, забезпечує результати, співмірні з провідними комерційними продуктами. Відхилення від еталонного рішення, отриманого точними методами, становить у середньому 3-6%, що є допустимою похибкою для дрібносерійного виробництва. Водночас статичні евристики, які не мають механізму адаптації до геометрії деталей, показують значно гіршу точність із відхиленням до 15%, що призводить до суттєвих перевитрат матеріалу.

Оцінка часової ефективності показала суттєву перевагу розробленого підходу над точними методами. Якщо для розрахунку оптимального розкрою замовлення з 37 деталей методом гілок і меж знадобилося понад годину машинного часу, то адаптивна система впоралася із завданням за 0.4-0.6 секунди. Така швидкодія досягається завдяки поліноміальній складності алгоритму і лінійно залежить від кількості стратегій, що перевіряються. Це дозволяє використовувати систему в режимі реального часу, наприклад, для миттєвого розрахунку вартості замовлення при клієнті, що є критичною вимогою для сучасних вебсервісів розкрою.

Окремо було проаналізовано стабільність роботи системи при масштабуванні вхідних даних. Експерименти показали, що при збільшенні кількості деталей у замовленні до 500 одиниць час виконання зростає передбачувано і не перевищує 3-5 секунд, тоді як точні методи стають непридатними для використання через комбінаторний вибух. Це підтверджує високу обчислювальну стійкість розробленого алгоритмічного забезпечення.

Узагальнюючи отримані дані, можна стверджувати, що запропонована система забезпечує оптимальний баланс між точністю розкрою та швидкістю роботи. Вона займає вигідну позицію між простими алгоритмами, які економлять час, але марнують матеріал, та складними математичними методами, які дають ідеал, але не придатні для оперативної роботи.

3.4. Висновки до розділу 3

На етапі програмної розробки було спроєктовано модульну архітектуру системи, до якої входять підсистеми моделювання вхідних даних, реалізація дерева гільйотинних розрізів та модуль візуалізації результатів. Ключовим досягненням стала програмна імплементація механізму врахування технологічних обмежень виробництва, зокрема відступів від краю листа та ширини пропилу, що дозволяє генерувати карти розкрою, повністю придатні до виконання на реальному обладнанні без додаткової корекції технологом.

Проведена серія обчислювальних експериментів на тестових наборах даних підтвердила ефективність запропонованого методу адаптивної гіперевристики. Результати моделювання засвідчили, що система здатна в автоматичному режимі симулювати різні варіанти розміщення та обирати стратегію сортування, яка забезпечує найкращі показники щільності укладання деталей. Зокрема вдалося досягти показників заповнення першого листа на рівні понад 95%, що сприяє утворенню ліквідних ділових залишків та зменшенню загальної кількості відходів.

Виконаний порівняльний аналіз ефективності та точності системи продемонстрував її переваги над наявними аналогами для задач дрібносерійного виробництва. Встановлено, що розроблений алгоритм забезпечує якість розкрою, яка лише на 3–6% поступається результатам точних математичних методів, проте досягається набагато швидше. Отримані практичні результати підтверджують доцільність впровадження розробленого програмного забезпечення у виробничий процес підприємств меблевої галузі.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Робота над проєктуванням та реалізацією системи автоматизованого розкрою виконувалася в умовах офісного приміщення, що класифікується як робота з використанням комп'ютерної техніки. Основна діяльність інженера-програміста пов'язана з тривалим перебуванням у фіксованій робочій позі, постійним зоровим напруженням та нервово-емоційним навантаженням, що вимагає дотримання спеціальних гігієнічних норм.

Згідно з чинним законодавством України про охорону праці та санітарними нормами, приміщення для роботи з комп'ютерною технікою повинно мати як природне, так і штучне освітлення [7]. Природне світло має падати збоку, переважно зліва, щоб уникнути відблисків на екрані монітора, які призводять до швидкої втоми очей. Штучне освітлення має бути рівномірним і достатнім, зазвичай у межах 300-500 люкс на робочій поверхні столу.

Раціональне освітлення є найважливішим фактором гігієни праці при роботі з комп'ютером, оскільки саме зір несе основне навантаження в процесі взаємодії людини з інформаційною системою. Освітлення робочого місця повинно бути достатнім, рівномірним та не створювати засліплюючої дії. Найбільш сприятливим для людини є природне освітлення, спектральний склад якого найкраще відповідає фізіологічним особливостям ока. Тому робочі місця слід розташовувати так, щоб світло падало збоку, переважно зліва, що забезпечує рівномірне освітлення робочої поверхні без утворення різких тіней.

Однак у сучасних умовах роботи часто виникає необхідність використання системи штучного освітлення. При проектуванні такої системи перевагу слід віддавати розсіяному світлу, яке створюється люмінесцентними лампами або сучасними світлодіодними панелями. Важливою характеристикою штучного освітлення є відсутність пульсації світлового потоку, оскільки мерехтіння світла,

навіть непомітне для ока, негативно впливає на центральну нервову систему і мозкову діяльність. Крім того, необхідно уникати утворення відблисків на екрані монітора від світильників або вікон, оскільки такі відблиски знижують контрастність зображення і змушують очі додатково напружуватися. Для запобігання цьому явищу використовуються матові розсіювачі на світильниках та жалюзі на вікнах.

Важливим аспектом є також мікроклімат приміщення, який являє собою сукупність фізичних параметрів повітряного середовища у робочій зоні. До основних показників мікроклімату належать температура, відносна вологість, швидкість руху повітря та інтенсивність теплового випромінювання. Для робіт операторського типу, які виконуються сидячи і не потребують значних фізичних зусиль, оптимальними вважаються температурні умови в діапазоні 22-24 градуси Цельсія в холодний період року та 23-25 градусів у теплий період [3]. Відхилення температури від цих значень змушує організм витрачати додаткову енергію на терморегуляцію, що призводить до швидкої втоми, зниження концентрації уваги та погіршення точності виконання операцій.

У приміщеннях з працюючою обчислювальною технікою часто спостерігається зниження вологості, особливо в опалювальний сезон. Сухе повітря сприяє інтенсивному випаровуванню вологи зі слизових оболонок верхніх дихальних шляхів та очей. Для користувачів комп’ютерів пересихання рогівки ока є критичним фактором, що провокує розвиток так званого комп’ютерного зорового синдрому, який проявляється у відчутті печіння, почервонінні очей та зниженні гостроти зору. Тому підтримка відносної вологості на рівні 40-60 відсотків є обов’язковою санітарно-гігієнічною вимогою. З іншого боку, надмірна вологість у поєднанні з високою температурою ускладнює тепловіддачу організму, що може призвести до перегрівання.

Окрему увагу при організації робочого місця приділено захисту від електромагнітного випромінювання. Робочий стіл та крісло повинні мати можливість регулювання по висоті для забезпечення фізіологічно правильної пози користувача, при якій кут зору на центр екрана становить 15-20 градусів униз від горизонталі [3]. Рівень електромагнітних полів на робочому місці не повинен перевищувати гранично

допустимих рівнів, встановлених державними санітарними нормами. Для зниження впливу статичної електрики та поліпшення якості повітря рекомендується регулярне вологе прибирання приміщення та провітрювання. Також важливим є дотримання режиму праці та відпочинку, що передбачає регламентовані перерви для зняття зорового та м’язового напруження.

При розробці методів та засобів оптимізації розкрою листових матеріалів у комп’ютерних системах меблевого виробництва з використанням адаптивних евристик проведено детальний аналіз потенційно шкідливих виробничих факторів, що супроводжують процес проектування та експлуатації інформаційної системи. Розроблено та обґрунтовано інженерні рішення щодо нормалізації мікроклімату, освітлення та зниження рівня шуму, а також запропоновано заходи із забезпечення пожежної безпеки на об'єктах впровадження системи.

4.2. Безпека в надзвичайних ситуаціях

4.2.1. Організація протипожежного захисту та проведення протипожежної профілактики на промисловому підприємстві

Деревообробні та меблеві підприємства, де здійснюється розкрій плитних матеріалів, належать до об’єктів підвищеної пожежної небезпеки [19]. Технологічний процес супроводжується накопиченням значної кількості деревини, стружки, тирси, а також використанням легкозаймистих лакофарбових матеріалів та клеїв. Особливу небезпеку становить деревний пил, який утворюється при різанні та шліфуванні матеріалів. У завислому стані він здатний утворювати з повітрям вибухонебезпечні суміші, які можуть детонувати від найменшої іскри. Крім того, деревний пил, осідаючи на конструкціях та обладнанні, створює сприятливе середовище для швидкого поширення вогню.

Основними причинами пожеж на таких підприємствах найчастіше стають несправності електрообладнання, зокрема короткі замикання, перевантаження мереж та іскріння контактів. Механічні причини, такі як тертя несправних підшипників або перегрів ріжучого інструменту, також можуть стати джерелом запалювання.

Порушення технологічного регламенту та правил пожежної безпеки при проведенні вогневих робіт, а також необережне поводження з вогнем є поширеними факторами ризику.

Організація протипожежного захисту на підприємстві повинна починатися з призначення відповідальних осіб та розробки чітких інструкцій для персоналу. Профілактична робота включає регулярні інструктажі, навчання правилам користування первинними засобами пожежогасіння та відпрацювання дій під час евакуації. Технічні заходи передбачають обладнання виробничих приміщень автоматичними системами пожежної сигналізації, які здатні виявити загоряння на ранній стадії за наявністю диму або підвищенням температури.

Для ефективного гасіння пожеж на деревообробних підприємствах широко застосовуються спринклерні та дренчерні установки автоматичного пожежогасіння. Спринклерні системи спрацьовують локально в зоні підвищення температури, подаючи воду безпосередньо на осередок займання, що дозволяє мінімізувати збитки від води. Дренчерні системи створюють водяні завіси, які перешкоджають поширенню вогню між цехами або технологічними ділянками. Важливою умовою є також забезпечення приміщень достатньою кількістю вогнегасників відповідного типу, пожежних щитів та внутрішніх пожежних кранів.

Режимні заходи пожежної безпеки спрямовані на усунення умов для виникнення горіння. Це включає своєчасне прибирання робочих місць від горючих відходів, регулярне очищення систем аспірації та вентиляції від пилу, заборону куріння у виробничих зонах та контроль за справністю електрообладнання. Дотримання протипожежного режиму є обов'язком кожного працівника підприємства, оскільки пожежа на деревообробному виробництві може розвиватися блискавично, створюючи загрозу для життя людей та значні матеріальні збитки.

4.2.2. Джерела виникнення шуму і вібрацій. Заходи і засоби захисту від шуму і вібрацій, гігієнічні та допустимі норми.

Виробниче середовище цехів розкрою характеризується наявністю інтенсивного шуму та вібрації, які генеруються потужним технологічним

обладнанням. Форматно-розкрійні верстати, пильні центри з числовим програмним керуванням, стружкопилососи та системи вентиляції створюють складний акустичний фон [18].

Шум виникає внаслідок механічної взаємодії ріжучого інструменту з оброблюваним матеріалом, вібрації корпусних деталей обладнання, а також аеродинамічних процесів при обертанні пил та русі повітря у повітроводах. Рівні звукового тиску біля працюючих верстатів часто перевищують допустимі межі, що вимагає вжиття спеціальних заходів захисту.

Тривалий вплив інтенсивного шуму на організм людини призводить до розвитку специфічних та неспецифічних патологічних змін. Специфічна дія проявляється у поступовому зниженні гостроти слуху, що може призвести до професійної приглухуватості. Неспецифічний вплив шуму поширюється на центральну нервову та серцево-судинну системи, викликаючи швидку втому, дратівливість, головний біль, підвищення артеріального тиску та порушення сну. Санітарні норми встановлюють гранично допустимі рівні шуму на робочих місцях, які для виробничих приміщень зазвичай становлять 80 дБА. Перевищення цього рівня вимагає застосування засобів колективного та індивідуального захисту.

Вібрація, яка передається від працюючого обладнання на тіло людини, також є шкідливим виробничим фактором. Розрізняють загальну вібрацію, що передається через підлогу або сидіння на все тіло, та локальну вібрацію, яка впливає на руки працюючого при контакті з органами управління верстата або оброблюваною деталлю. Тривала дія вібрації може спричинити розвиток вібраційної хвороби, яка характеризується порушенням судинного тонусу, чутливості та трофіки тканин. Для нормування вібрації використовуються показники віброшвидкості та віброприскорення у відповідних смугах частот [20, 21].

Заходи боротьби з шумом та вібрацією поділяються на технічні, архітектурно-планувальні та організаційні. Найбільш ефективним є зниження шуму в джерелі його виникнення. Це досягається шляхом удосконалення конструкції вузлів тертя, точного балансування обертових частин, використання малошумних матеріалів та своєчасного технічного обслуговування обладнання. Важливу роль відіграє

віброізоляція верстатів, яка запобігає передачі вібрації на будівельні конструкції, а також застосування звукоізолюючих кожухів та екранів.

Архітектурно-планувальні заходи передбачають раціональне розміщення обладнання, виділення найбільш шумних агрегатів у окремі ізольовані приміщення та використання звукопоглинальних матеріалів для облицювання стін і стелі цеху.

Акустична обробка приміщень дозволяє знизити рівень відбитого звуку та загальний шум у цеху. Якщо технічними засобами не вдається знизити шум до нормативних значень, обов'язковим є використання засобів індивідуального захисту органів слуху. Протишумні навушники та вкладиші дозволяють суттєво зменшити акустичне навантаження на працюючого. Вибір засобів індивідуального захисту повинен здійснюватися з урахуванням спектрального складу шуму та необхідної ефективності захисту. Комплексне застосування всіх зазначених заходів дозволяє створити безпечні умови праці та зберегти здоров'я персоналу.

ВИСНОВКИ

У кваліфікаційній роботі магістра вирішено актуальну науково-прикладну задачу підвищення ефективності автоматизованого розкрою плитних матеріалів шляхом розробки адаптивної програмної системи. У ході виконання роботи було проведено глибокий аналіз предметної області, який виявив, що наявні методи розв’язання задачі двовимірного розкрою поділяються на дві полярні групи: швидкі, але неточні статичні евристики, та точні, але повільні математичні алгоритми. Це обґрунтувало необхідність створення гібридного підходу, який поєднував би швидкість евристик із гнучкістю адаптивних систем.

Теоретичним фундаментом роботи стала розроблена математична модель процесу гільйотинного розкрою, яка, на відміну від класичних постановок, враховує реальні технологічні обмеження меблевого виробництва. Зокрема, було формалізовано вплив технологічних відступів від краю листа та ширини пропилу інструменту на геометрію карти розкрою. Для представлення простору розкрою обґрунтовано використання структури даних бінарного дерева, що дозволило гарантувати виконання умови гільйотинності розрізів на рівні архітектури даних.

Науковою новизною роботи є запропонований метод адаптивної гіперевристики, який замість використання однієї фіксованої стратегії динамічно обирає оптимальний алгоритм сортування деталей залежно від характеристик конкретного виробничого замовлення.

Практична значущість роботи полягає у створенні програмного комплексу, що реалізує розроблені моделі та методи. Архітектура системи побудована на модульному принципі, що забезпечує гнучкість та можливість подальшого розширення. Реалізовані програмні модулі дозволяють автоматично генерувати карти розкрою з урахуванням напрямку текстури матеріалу та технічних характеристик розкрійного обладнання.

Результати експериментальних досліджень підтвердили високу ефективність запропонованого підходу. Система продемонструвала здатність досягати коефіцієнта корисного використання матеріалу на рівні 78–85%, при цьому забезпечуючи

щільність заповнення перших листів у партії понад 95%, що є критично важливим для формування ліквідних ділових залишків. Порівняльний аналіз показав, що розроблена адаптивна система значно перевершує статичні методи за якістю розміщення, а час генерації розкрою становить частки секунди, що робить її придатною для використання в режимі реального часу, наприклад, у вебсервісах або на виробничих лініях.

Розроблене програмне забезпечення є готовим інструментом для автоматизації технологічної підготовки виробництва, який дозволяє зменшити витрати матеріалів та підвищити економічну ефективність підприємств деревообробної галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бармак О. В., Крак Ю. В., Кривонос Ю. Г. Інтелектуальні інформаційні технології та системи: навчальний посібник. Тернопіль: Астон, 2020. 288 с.

2. Варавін А.В., Лещишин Ю.З., Чайковський А.В. Методичні вказівки до виконання курсового проєкту з дисципліни «Дослідження і проєктування комп'ютерних систем та мереж». Тернопіль: ТНТУ, 2024. 32 с.

3. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями: НПАОП 0.00-7.15-18. Київ: Міністерство соціальної політики України, 2018. 20 с.

4. Глибовець М. М., Гулаєва Н. М. Еволюційні алгоритми: навчальний посібник. Київ: НаУКМА, 2013. 320 с.

5. ДСТУ ISO 45001:2019. Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування. Київ: ДП «УкрНДНЦ», 2019.

6. Зайченко Ю. П. Дослідження операцій: підручник. Київ: Слово, 2016. 816 с.

7. Закон України «Про охорону праці»: за станом на 01 січ. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 05.12.2025).

8. Катренко А. В. Дослідження операцій: підручник. Львів: Магнолія-2006, 2019. 352 с.

9. Кодекс цивільного захисту України. Відомості Верховної Ради України, 2013, № 34-35, ст. 458.

10. Кравченко В. М. Системний аналіз та методи оптимізації розкрою матеріалів: навчальний посібник. Київ: НТУУ «КПІ», 2019. 210 с.

11. Крепич С. Я., Співак І. Я. Алгоритми та структури даних: навчальний посібник. Тернопіль: ТНТУ, 2019. 224 с.

12. Левицький В. В. Інженерія програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки, 2021. 248 с.

13. Литвин В. В., Висоцька В. А., Пуле С. О. Інтелектуальні системи: підручник. Львів: Новий Світ-2000, 2019. 406 с.

14. Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 123 «Комп'ютерна інженерія». Тернопіль: ТНТУ, 2024. 44 с.

15. Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Програма та методичні рекомендації з проходження практики за тематикою кваліфікаційної роботи для студентів спеціальності 123 «Комп'ютерна інженерія». Тернопіль: ТНТУ, 2024. 45 с.

16. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Системи штучного інтелекту: підручник. Львів: Магнолія-2006, 2019. 279 с.

17. Панченко Т. В. Генетичні та еволюційні алгоритми: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2018. 165 с.

18. Правила охорони праці в деревообробній промисловості: НПАОП 20.0-1.02-05. Київ: Держгірпромнагляд, 2005.

19. Правила пожежної безпеки в Україні: НАПБ А.01.001-2014. Київ: ДСНС України, 2014. 54 с.

20. Стручок В.С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей «Безпека в надзвичайних ситуаціях». Тернопіль: ФОП Паляниця В. А. 156 с. URL: https://elartu.tntu.edu.ua/handle/lib/39196.

21. Стручок В.С. Навчальний посібник «Техноекологія та цивільна безпека. Частина «цивільна безпека». Тернопіль: ФОП Паляниця В. А. 156 с. URL: http://elartu.tntu.edu.ua/handle/lib/39424.

22. Стухляк П. Д., Іванченко О. В., Буката Л. М. Теорія систем і системний аналіз: навчальний посібник. Тернопіль: ТНТУ, 2018. 192 с.

23. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навчальний посібник. Запоріжжя: ЗНТУ, 2018. 340 с.

24. Руснак В. М., Стадник Н. Б. Інформаційна технологія адаптивної оптимізації розкрою листових матеріалів у меблевому виробництві. Інформаційні моделі, системи та технології: матеріали XIII науково-технічної конференції (Тернопіль, 17–18 грудня 2025 р.). Тернопіль: ТНТУ, 2025. С. 147.

25. Руснак В. М., Стадник Н. Б. Методи адаптивного вибору евристик для задач гільйотинного розкрою. Актуальні задачі сучасних технологій: збірник тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів (Тернопіль, 11–12 грудня 2025 р.). Тернопіль: ТНТУ, 2025. С. 335.

26. Тимченко А. А. Основи системного аналізу та системного проектування: навчальний посібник. Київ: Либідь, 2019. 265 с.

27. Типове положення про систему управління охороною праці: НПАОП 0.00-4.21-04. Київ: Держгірпромнагляд України, 2004. 28 с.

28. Шаховська Н. Б., Голощук Р. О. Алгоритми і структури даних: посібник. Львів: Магнолія-2006, 2020. 216 с.

29. Gilmore P. C., Gomory R. E. A linear programming approach to the cutting-stock problem. *Operations Research*. 1961. Vol. 9, No. 6. P. 849–859.

30. Matplotlib: Visualization with Python. URL: https://matplotlib.org/ (дата звернення: 02.12.2025).

31. NumPy User Guide. URL: https://numpy.org/doc/stable/user/ (дата звернення: 02.12.2025).

32. Python Software Foundation. Python 3.12 Documentation. URL: https://docs.python.org/3/ (дата звернення: 01.12.2025).

33. Tkinter – Python interface to Tcl/Tk. URL: https://docs.python.org/3/library/tkinter.html (дата звернення: 02.12.2025).

Додаток А

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Краківський економічний університет (Польща)
Вроцлавський економічний університет (Польща)
Університет «Опольська Політехніка» (Польща)
Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Вінницький національний аграрний університет
Львівський національний університет ім. І. Франка
Головне управління Пенсійного фонду в Тернопільській області
Наукове товариство ім. Шевченка
Тернопільський обласний комунальний інститут післядипломної педагогічної освіти
Сумський державний педагогічний університет
Запорізький національний університет

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XIV Міжнародної науково-технічної конференції молодих учених та студентів

11-12 грудня 2025 року

УКРАЇНА
ТЕРНОПІЛЬ – 2025

58.	**М.І. Паламар, Ю.А. Удич, А.М. Паламар, М.Р. Франків** ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА МОНІТОРИНГУ ПАРАМЕТРІВ РОЗУМНОГО БУДИНКУ НА ОСНОВІ ІОТ ТЕХНОЛОГІЙ		320
59.	**Ю.Б. Паляниця, М.Ю. Ковальчук** МЕТОД ІНТЕЛЕКТУАЛЬНОЇ КЛАСИФІКАЦІЇ ЛІТАЮЧИХ ОБ'ЄКТІВ ЗА ЇХ АКУСТИЧНИМИ СИГНАТУРАМИ		321
60.	**С.М. Панасенко, Н.С. Луцик** ІНТЕГРАЦІЯ RULE-BASED АЛГОРИТМІВ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ТЕСТУВАННЯ ЕЛЕКТРОНІКИ		324
61.	**А. Пенцак, Ю. Лещишин** ВБУДОВАНА СИСТЕМА ДЛЯ КОМП'ЮТЕРНОГО АНАЛІЗУ ДИХАЛЬНИХ ЗВУКІВ		325
62.	**В.С. Пиж** ІННОВАЦІЙНИЙ ПІДХІД ДО ПОБУДОВИ ЗАХИЩЕНОГО ВІРТУАЛЬНОГО СЕРЕДОВИЩА ДЛЯ ОСВІТНЬОГО ПРОЦЕСУ		326
63.	**О.Б. Пйонтковський** ЕНЕРГОЕФЕКТИВНІ МІКРОКОНТРОЛЕРИ ДЛЯ АВТОНОМНИХ ІОТ–ПРИСТРОЇВ		328
64.	**І. І. Поліщук, Н.С. Луцик** АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИЯВЛЕННЯ ПЕРЕЛОМІВ НА МЕДИЧНИХ ЗОБРАЖЕННЯХ		329
65.	**І.Р. Ралік** РЕЗОНАНСНИЙ СЕЛЕКТОР СТРАТЕГІЙ У ІНТЕЛЕКТУАЛЬНИХ CRM-СИСТЕМАХ		331
66.	**І.І. Рій, Є.В. Тиш** ПОРІВНЯЛЬНИЙ АНАЛІЗ ХАОТИЧНИХ І КЛАСИЧНИХ МЕТОДІВ ШИФРУВАННЯ З ТОЧКИ ЗОРУ ІНФОРМАЦІЙНОЇ ЕНТРОПІЇ		332
67.	**Рожик А. М.** МЕТОДИ ТА ПРОГРАМНО-АПАРАТНІ ЗАСОБИ ІДЕНТИФІКАЦІЇ ПРАЦІВНИКІВ З МЕТОЮ ВИЗНАЧЕННЯ РОБОЧОГО ЧАСУ ТА ДОСТУПУ ДО ПРИМІЩЕННЯ		333
68.	**В. М. Русинак, Н. Б. Стадник** МЕТОДИ АДАПТИВНОГО ВИБОРУ ЕВРИСТИК ДЛЯ ЗАДАЧ ГІЛЬЙОТИННОГО РОЗКРОЮ		335
69.	**П.В. Свінціцький, Н.М. Пановик, О.В. Доберчак, І.О. Боднарчук** СУЧАСНА АРХІТЕКТУРА СТРІМІНГОВИХ ПЛАТФОРМ В РЕАЛЬНОМУ ЧАСІ		336
70.	**П.М. Семчишин** АРХІТЕКТУРНІ РІШЕННЯ ДЛЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКІВ		340
71.	**С.О.Синишен** МІКРО- ТА НАНОРОБОТИ: ТЕХНІЧНІ ВІДМІННОСТІ, ПЕРЕВАГИ ТА НЕДОЛІКИ MPI І GDI		342
72.	**Я.Р. Сиротинський; А.М. Паламар, к.т.н., доц.; А.П. Шмігель** КОМП'ЮТЕРИЗОВАНА СИСТЕМА ВИМІРЮВАННЯ ШВИДКОСТІ ВІТРУ НА ОСНОВІ ІОТ ТЕХНОЛОГІЙ		343
73.	**М. Смоляк** СИСТЕМА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У РЕАЛЬНОМУ ЧАСІ В АВТОМОБІЛЯХ		344
74.	**А.А. Станько, С.М. Куриляк, Я.О. Тригуб** АВТОМАТИЗОВАНА СИСТЕМА ОЦІНЮВАННЯ ТЕРМІНУ ЗБЕРІГАННЯ ХАРЧОВИХ ПРОДУКТІВ НА ОСНОВІ БАГАТОКАНАЛЬНИХ ГАЗОВИХ СЕНСОРІВ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ		347

УДК 004.896:674.02

В. М. Руснак, Н. Б. Стадник, к.т.н
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

МЕТОДИ АДАПТИВНОГО ВИБОРУ ЕВРИСТИК ДЛЯ ЗАДАЧ ГІЛЬЙОТИННОГО РОЗКРОЮ

V. M. Rusnak, N. B. Stadnyk

METHODS OF ADAPTIVE HEURISTICS SELECTION FOR GUILLOTINE CUTTING PROBLEMS

Проблема раціонального розкрою матеріалів є однією з класичних задач комбінаторної оптимізації, яка має важливе практичне значення для меблевої промисловості. Зростання номенклатури виробів та перехід до індивідуального виробництва вимагають гнучких алгоритмічних рішень, здатних адаптуватися до змінних вхідних даних [1].

Метою роботи є розробка методу оптимізації розкрою, який базується на механізмі адаптивного вибору евристик (hyper-heuristics) для підвищення коефіцієнта корисного використання матеріалу.

У роботі розглядається задача двовимірного пакування (2D Bin Packing) з обмеженням гільйотинного різу. Традиційні підходи часто використовують єдину стратегію розміщення (наприклад, сортування за площею) для всього пакету замовлень. Однак, як показують дослідження, ефективність конкретної евристики суттєво залежить від геометричних характеристик набору деталей.

Запропонований метод полягає у попередній симуляції процесу розкрою для кожного листа матеріалу з використанням набору конкуруючих стратегій:

1. *Best Area Fit* (пріоритет площі) – ефективна для різнорідних деталей.
2. *Longest Side Fit* (пріоритет сторони) – ефективна для довгих вузьких деталей.
3. *Quantity Priority* (пріоритет кількості) – нова запропонована евристика, що дозволяє групувати однакові деталі, спрощуючи логістику.

Програмна реалізація системи виконана мовою Python. Алгоритм автоматично розраховує ефективність (Yield %) для кожної стратегії та затверджує карту розкрою з найкращим показником [2, с. 47]. Також враховано технологічні параметри: ширина різу (kerf) – 3-4 мм, відступи на торцювання (trim) – 10-15 мм. Для забезпечення інтеграції з автоматизованими системами керування виробництвом (АСКВ) розроблено модуль експорту даних у формат JSON. Це дозволяє передавати координати деталей безпосередньо на розкрійні центри з ЧПК, мінімізуючи людський фактор.

Результати тестування на реальних виробничих даних показали, що застосування адаптивного підходу дозволяє зменшити площу відходів на 5-12% порівняно з використанням однієї фіксованої стратегії.

Література

1. Романишин Ю.М., Скрупський С.Ю. Аналіз ефективності алгоритмів розкрою промислових матеріалів. *Науковий вісник НЛТУ України*. 2020. Т. 30, № 2. С. 88–94.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

SYSTEMS
А. Рожик, Р. Жаровський
АНАЛІЗ ЕФЕКТИВНОСТІ РОБОТИ АДАПТИВНОЇ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ
A. Rozhyk, R. Zharovskyi
ANALYSIS OF THE EFFICIENCY OF THE ADAPTIVE ACCESS CONTROL SYSTEM BASED ON FUZZY LOGIC 144
В. Руснак, Н. Стадник
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АДАПТИВНОЇ ОПТИМІЗАЦІЇ РОЗКРОЮ ЛИСТОВИХ МАТЕРІАЛІВ У МЕБЛЕВОМУ ВИРОБНИЦТВІ
V. Rusnak, N. Stadnyk
INFORMATION TECHNOLOGY OF ADAPTIVE OPTIMIZATION OF SHEET MATERIAL CUTTING IN FURNITURE PRODUCTION 147
Р. Савченко, С. Шестопалов
ОГЛЯД СУЧАСНИХ ПІДХОДІВ УПРАВЛІННЯ ЧЕРГАМИ НА МАРШРУТИЗАТОРАХ ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖ
R. Savchenko, S. Shestopalov
OVERVIEW OF MODERN APPROACHES TO QUEUE MANAGEMENT ON ROUTERS IN INFORMATION AND COMMUNICATION NETWORKS 148
В. Семанюк, Є. Тиш
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО МОНІТОРИНГУ ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖ ТА ОЦІНЮВАННЯ ЇХ ЕФЕКТИВНОСТІ
V. Semaniuk, Ie. Tysh
ANALYSIS OF MODERN APPROACHES TO LOCAL COMPUTER NETWORK MONITORING AND EVALUATION OF THEIR EFFICIENCY 149
В. Семанюк, Є. Тиш
СТРУКТУРНІ ОСОБЛИВОСТІ ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖ ТА ФАКТОРИ, ЩО ВПЛИВАЮТЬ НА ЇХНЮ ПРОДУКТИВНІСТЬ
V. Semaniuk, Ie. Tysh
STRUCTURAL FEATURES OF LOCAL COMPUTER NETWORKS AND FACTORS AFFECTING THEIR PERFORMANCE 150
Ю. Сеник, Я. Литвиненко
ОГЛЯД МІКРОКОНТРОЛЕРІВ ДЛЯ ЗАДАЧІ КОНТРОЛЮ КЛІМАТИЧНИХ ПАРАМЕТРІВ
Y. Senyk, I. Lytvynenko
A REVIEW OF MICROCONTROLLERS FOR CLIMATE CONTROL 151
А. Спесівцев
РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ФОРМУВАННЯ РЕЙТИНГУ МЕРЕЖНИХ ПРИСТРОЇВ З УРАХУВАННЯМ ПАРАМЕТРІВ БЕЗПЕКИ
A. Spesivtsev
DEVELOPMENT OF SOFTWARE FOR RATING NETWORK DEVICES TAKING INTO ACCOUNT SECURITY PARAMETERS 153
В. Стецько, М. Приймак, Р. Жаровський
МЕТОДИКА І АЛГОРИТМ РОЗРАХУНКУ ТРАЄКТОРІЇ РУХУ РОБОТИЗОВАНОЇ ГАЗОНОКОСАРКИ
V. Stetsko, M. Priymak, R. Zharovskyi
METHODOLOGY AND ALGORITHM FOR CALCULATING THE TRAJECTORY OF A ROBOTIC LAWN MOWER 154
О. Цвірла, І. Мудрий, Н. Луцик
ІНТЕЛЕКТУАЛЬНІ МЕТОДИ ОЦІНЮВАННЯ КРИПТОГРАФІЧНОЇ ЯКОСТІ ХЕШ-СТРІЧОК У ІНФОРМАЦІЙНИХ СИСТЕМАХ
O. Tsvirla, I. Mudryi, N. Lutsyk
INTELLIGENT METHODS FOR ASSESSING THE CRYPTOGRAPHIC QUALITY OF HASH STRINGS IN INFORMATION SYSTEMS 156

УДК 004.89:674

В. Русиак; Н. Стадник, к. т. н.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АДАПТИВНОЇ ОПТИМІЗАЦІЇ РОЗКРОЮ ЛИСТОВИХ МАТЕРІАЛІВ У МЕБЛЕВОМУ ВИРОБНИЦТВІ

UDC 004.89:674

V. Rusnak; N. Stadnyk, PhD.

INFORMATION TECHNOLOGY OF ADAPTIVE OPTIMIZATION OF SHEET MATERIAL CUTTING IN FURNITURE PRODUCTION

Ефективність сучасного меблевого виробництва значною мірою залежить від коефіцієнта корисного використання матеріалів. Оскільки вартість плитних матеріалів (ДСП, МДФ) складає значну частку собівартості продукції, мінімізація відходів є пріоритетним завданням. Задача двовимірного розкрою (2D Cutting Stock Problem) належить до класу NP-складних задач, що унеможливлює знаходження абсолютного оптимуму за прийнятний час для великих партій деталей [1, с. 45]. Існуючі системи автоматизованого проєктування (САПР) часто використовують статичні жадібні алгоритми (First Fit, Best Fit), які не завжди враховують специфіку конкретного замовлення, особливо в умовах дрібносерійного виробництва.

Метою роботи є підвищення ефективності розкрою шляхом розробки комп'ютерної системи, що використовує адаптивні евристики для динамічного вибору стратегії розміщення. У ході дослідження було проаналізовано технологічні обмеження меблевого виробництва, ключовим з яких є необхідність гільйотинного розкрою (різи повинні проходити від краю до краю листа). На основі цього розроблено математичну модель та програмний модуль мовою Python.

Наукова новизна запропонованого підходу полягає у використанні методу гіпер-евристик (Hyper-heuristics). Замість використання одного фіксованого алгоритму, розроблена система виконує симуляцію розкрою, застосовуючи набір різних стратегій сортування черги деталей: за спаданням площі (Area Descending), за максимальною стороною (Longest Side), за периметром (Perimeter) та за кількістю у замовленні (Quantity Priority) [2, с. 89]. Для листа матеріалу система автоматично обирає ту стратегію, яка забезпечує максимальний коефіцієнт корисного використання площі (Yield).

Окрему увагу приділено розробці евристики «Quantity Priority». Цей підхід оптимізує виробничу логістику, надаючи пріоритет деталям, які зустрічаються в замовленні у великій кількості, що спрощує подальші операції сортування, кромкування та свердління, зменшуючи час на переналагодження обладнання [3]. Для інтеграції з виробничим обладнанням реалізовано модуль експорту карт розкрою у формат JSON, що дозволяє передавати дані безпосередньо на верстати з ЧПК.

Результати експериментальних досліджень на тестових наборах даних показали, що адаптивний підхід дозволяє підвищити коефіцієнт використання матеріалу на 5-12% порівняно з використанням статичного алгоритму.

Література

1. Ковальчук А.М., Мельник А.О. Метод гібридної евристики для розв'язання задачі двовимірного пакування. Комп'ютерні системи та мережі. 2021. Том 3, № 1. С. 45–52.
2. Романишин Ю.М., Скрупський С.Ю. Аналіз ефективності алгоритмів розкрою промислових матеріалів. Науковий вісник НЛТУ України. 2020. Т. 30, № 2. С. 88–94.
4. Тимощук О.Л. Методи оптимізації: навч. посіб. Львів: Видавництво Львівської політехніки, 2019. 260 с.

Додаток Б

Лістинг коду основного файлу системи

```
import json
import hashlib
import random
import copy
from dataclasses import dataclass
from typing import List, Optional, Callable, Tuple, Dict
import matplotlib.pyplot as plt


# ======================
#   Модель деталі
# ======================
@dataclass
class Part:
    """Опис однієї деталі замовлення."""
    id: int
    width: float
    height: float
    label: str
    batch_count: int  # кількість таких деталей у замовленні

    # стан після пакування
    x: Optional[float] = None
    y: Optional[float] = None
    rotated: bool = True

    @property
    def area(self) -> float:
        return self.width * self.height

    @property
    def longest_side(self) -> float:
        return max(self.width, self.height)

    @property
    def perimeter(self) -> float:
        return 2 * (self.width + self.height)


# ======================
#   Вузол для гільйотинного дерева
# ======================
@dataclass
class GuillotineNode:
    """Прямокутний вільний регіон листа (вузол дерева)."""
    x: float
```

```
    y: float
    width: float
    height: float

    used: bool = False
    part: Optional[Part] = None
    right: Optional['GuillotineNode'] = None
    top: Optional['GuillotineNode'] = None

    def can_fit(self, w: float, h: float, kerf: float) -> bool:
        """Чи поміститься прямокутник w×h у цей вузол (без урахування
додаткових полів)."""
        # Ми перевіряємо лише габарити; пропил враховується при
створенні вільних зон
        return (w <= self.width) and (h <= self.height)


# ======================
#   Гільйотинний пакувальник
# ======================
class GuillotinePacker:
    """Пакувальник з гільйотинним поділом простору (бінарне
дерево)."""

    def __init__(
        self,
        sheet_width: float,
        sheet_height: float,
        trim: float = 10.0,
        kerf: float = 3.5,
        allow_rotation: bool = True,
        strategy: str = "best_fit",
        strategy_name: str = "",
        split_policy: str = "auto",
    ) -> None:
        self.sheet_width = sheet_width
        self.sheet_height = sheet_height
        self.trim = trim
        self.kerf = kerf
        self.allow_rotation = allow_rotation
        self.strategy = strategy  # "first_fit" або "best_fit"
        self.strategy_name = strategy_name or strategy
        self.split_policy = split_policy  # "auto" | "vertical_first"
| "horizontal_first"

        # коренева зона — робоча частина листа всередині TRIM
        self.root = GuillotineNode(
            x=self.trim,
            y=self.trim,
            width=self.sheet_width - 2 * self.trim,
            height=self.sheet_height - 2 * self.trim,
```

```
        )

        self.placed_parts: List[Part] = []

    def reset(self) -> None:
        """Скидання стану пакувальника (новий чистий лист)."""
        self.root = GuillotineNode(
            x=self.trim,
            y=self.trim,
            width=self.sheet_width - 2 * self.trim,
            height=self.sheet_height - 2 * self.trim,
        )
        self.placed_parts = []

    def pack_parts(self, parts: List[Part]) -> None:
        """Спробувати розмістити всі деталі зі списку (в порядку
parts)."""
        for part in parts:
            self.fit(part)

    def fit(self, part: Part) -> bool:
        """Спроба розмістити деталь у доступних вузлах дерева."""
        candidates = self._collect_candidate_nodes(self.root, part)
        if not candidates:
            return False

        # вибір вузла згідно стратегії
        if self.strategy == "first_fit":
            node, rotated = candidates[0]
        elif self.strategy == "best_fit":
            node, rotated = min(
                candidates,
                key=lambda item: (item[0].width * item[0].height) -
(part.width * part.height),
            )
        else:
            node, rotated = candidates[0]

        self._place_part_in_node(node, part, rotated)
        self.placed_parts.append(part)
        return True

    def _collect_candidate_nodes(self, node: Optional[GuillotineNode],
part: Part) -> List[Tuple[GuillotineNode, bool]]:
        """Зібрати всі вузли, куди можна вмістити деталь (без
врахування розрізів)."""
        result: List[Tuple[GuillotineNode, bool]] = []
        if node is None:
            return result

        if node.used:
```

```
            # рекурсивно перевіряємо дочірні вузли
            result.extend(self._collect_candidate_nodes(node.right,
part))
            result.extend(self._collect_candidate_nodes(node.top,
part))
            return result

        # перевірка без обертання
        if node.can_fit(part.width, part.height, self.kerf):
            result.append((node, False))
        # перевірка з обертанням
        if self.allow_rotation and node.can_fit(part.height,
part.width, self.kerf):
            result.append((node, True))

        return result

    def _place_part_in_node(self, node: GuillotineNode, part: Part,
rotated: bool) -> None:
        node.used = True
        node.part = part

        if rotated:
            part.rotated = True
            pw, ph = part.height, part.width
        else:
            part.rotated = False
            pw, ph = part.width, part.height

        part.x = node.x
        part.y = node.y

        remaining_width = node.width - pw - self.kerf
        right_x = node.x + pw + self.kerf
        remaining_height = node.height - ph - self.kerf
        top_y = node.y + ph + self.kerf

        # ВАРІАНТ A (горизонтально-перший): right має висоту ph, top
має ширину node.width
        rightA = (remaining_width, ph) if remaining_width > 0 else
None
        topA = (node.width, remaining_height) if remaining_height > 0
else None
        largestA = max((rightA[0] * rightA[1]) if rightA else 0,
(topA[0] * topA[1]) if topA else 0)

        # ВАРІАНТ B (вертикально-перший): right має повну висоту, top
має ширину pw
        rightB = (remaining_width, node.height) if remaining_width > 0
else None
```

```
        topB = (pw, remaining_height) if remaining_height > 0 else
None
        largestB = max((rightB[0] * rightB[1]) if rightB else 0,
(topB[0] * topB[1]) if topB else 0)

        # Вибір варіанту згідно політики
        if self.split_policy == "vertical_first":
            use_B = True
        elif self.split_policy == "horizontal_first":
            use_B = False
        else:  # auto
            use_B = largestB > largestA + 1e-6

        # Присвоюємо дочірні вузли згідно вибору
        if use_B:
            if remaining_width > 0:
                node.right = GuillotineNode(x=right_x, y=node.y,
width=remaining_width, height=node.height)
            if remaining_height > 0:
                node.top = GuillotineNode(x=node.x, y=top_y, width=pw,
height=remaining_height)
        else:
            if remaining_width > 0:
                node.right = GuillotineNode(x=right_x, y=node.y,
width=remaining_width, height=ph)
            if remaining_height > 0:
                node.top = GuillotineNode(x=node.x, y=top_y,
width=node.width, height=remaining_height)

    # Метрики та експорт
    def used_area(self) -> float:
        return sum(p.area for p in self.placed_parts)

    def sheet_area(self) -> float:
        return (self.sheet_width - 2 * self.trim) * (self.sheet_height
- 2 * self.trim)

    def efficiency(self) -> float:
        if self.sheet_area() == 0:
            return 0.0
        return 100.0 * self.used_area() / self.sheet_area()

    def export_to_json(self, filename: str, extra_meta: Optional[Dict]
= None) -> None:
        """Експорт розкрою цього листа у JSON (один лист)."""
        data = {
            "sheet": {"width": self.sheet_width, "height":
self.sheet_height, "trim": self.trim, "kerf": self.kerf},
            "strategy": self.strategy_name,
            "efficiency_percent": self.efficiency(),
            "parts": [
```

```
                {
                    "id": p.id,
                    "label": p.label,
                    "x": p.x,
                    "y": p.y,
                    "width": (p.height if p.rotated else p.width),
                    "height": (p.width if p.rotated else p.height),
                    "rotated": p.rotated,
                }
                for p in self.placed_parts
            ],
        }
        if extra_meta:
            data["meta"] = extra_meta
        with open(filename, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=2)


# ======================
#   Адаптивний оптимізатор (гіпер-евристика)
# ======================
class AdaptiveOptimizer:
    """Керує запуском різних стратегій сортування та збирає результати
(може пакувати на кілька листів)."""

    def __init__(self, sheet_width: float, sheet_height: float, trim:
float = 10.0, kerf: float = 3.5, allow_rotation: bool = True) -> None:
        self.sheet_width = sheet_width
        self.sheet_height = sheet_height
        self.trim = trim
        self.kerf = kerf
        self.allow_rotation = allow_rotation

        # Доступні стратегії сортування (ключ функції)
        self.strategies: Dict[str, Callable[[Part], float]] = {
            "Area Desc": lambda p: -p.area,
            "Longest Side": lambda p: -p.longest_side,
            "Perimeter": lambda p: -p.perimeter,
            "Quantity Desc": lambda p: -p.batch_count,  # НОВА:
пріоритет за кількістю
        }

    def _pack_all_to_sheets(self, parts_sorted: List[Part],
placement_strategy: str, split_policy: str = "auto") ->
List[GuillotinePacker]:
        """Пакує усі наявні копії деталей на стільки листів, скільки
потрібно."""
        remaining = [copy.deepcopy(p) for p in parts_sorted]
        packers: List[GuillotinePacker] = []

        while remaining:
```

```
            packer = GuillotinePacker(
                sheet_width=self.sheet_width,
                sheet_height=self.sheet_height,
                trim=self.trim,
                kerf=self.kerf,
                allow_rotation=self.allow_rotation,
                strategy=placement_strategy,
                strategy_name="",
                split_policy=split_policy,
            )

            not_placed: List[Part] = []

            # Перевірка: кожна деталь повинна вміститися у порожній
лист (інакше - помилка)
            for p in remaining:
                fits_empty = packer.root.can_fit(p.width, p.height,
packer.kerf) or (
                    packer.allow_rotation and
packer.root.can_fit(p.height, p.width, packer.kerf)
                )
                if not fits_empty:
                    raise ValueError(f"Part id={p.id}
({p.width}x{p.height}) не вміщується в робочу зону листа
{packer.root.width}x{packer.root.height}")

            # спробувати покласти в поточний лист
            for p in remaining:
                if not packer.fit(p):
                    not_placed.append(p)

            packers.append(packer)

            # без прогресу — зупиняємося (запобігання петлі)
            if len(not_placed) == len(remaining):
                break

            remaining = not_placed

        return packers

    def simulate_strategy(self, parts: List[Part], strategy_name: str,
placement_strategy: str = "best_fit", split_policy: str = "auto") ->
Tuple[List[GuillotinePacker], List[Part]]:
        """
        Запускає симуляцію для однієї стратегії сортування.
        Повертає список packer-ів (по одному на лист) та відсортований
список деталей (копії).
        """
        sort_key = self.strategies[strategy_name]
        parts_copy = [copy.deepcopy(p) for p in parts]
```

```
        parts_copy.sort(key=sort_key)

        packers = self._pack_all_to_sheets(parts_copy,
placement_strategy, split_policy=split_policy)
        for p in packers:
            p.strategy_name = f"{strategy_name} [{placement_strategy}]
({split_policy})"
        return packers, parts_copy

    def run_all_strategies(self, parts: List[Part]) -> Tuple[str,
Dict[str, Dict]]:
        """Запускає усі стратегії, повертає ім’я найкращої та словник
результатів."""
        results: Dict[str, Dict] = {}
        best_strategy_name: Optional[str] = None
        best_efficiency: float = -1.0

        for name in self.strategies.keys():
            for placement in ["best_fit", "first_fit"]:
                for split_policy in ["auto", "vertical_first",
"horizontal_first"]:
                    packers, placed_parts =
self.simulate_strategy(parts, name, placement_strategy=placement,
split_policy=split_policy)

                    total_used = sum(p.used_area() for p in packers)
                    total_area = sum(p.sheet_area() for p in packers)
                    eff = (100.0 * total_used / total_area) if
total_area > 0 else 0.0
                    total_placed = sum(len(p.placed_parts) for p in
packers)

                    key = f"{name} [{placement}] ({split_policy})"
                    results[key] = {
                        "packers": packers,
                        "placed_parts": placed_parts,
                        "efficiency": eff,
                        "sheets": len(packers),
                        "placed_count": total_placed,
                    }

                    if eff > best_efficiency:
                        best_efficiency = eff
                        best_strategy_name = key

        assert best_strategy_name is not None
        return best_strategy_name, results


# ======================
#   Візуалізація результату
```

```
# ======================
def visualize_result(packer: GuillotinePacker, title: str = "Cutting
Layout", highlight: bool = False) -> None:
    fig, ax = plt.subplots(figsize=(10, 7))
    edge_color = 'red' if highlight else 'black'
    edge_width = 3 if highlight else 2

    ax.add_patch(plt.Rectangle((0, 0), packer.sheet_width,
packer.sheet_height, fill=False, edgecolor=edge_color,
linewidth=edge_width))
    ax.add_patch(plt.Rectangle((packer.trim, packer.trim),
packer.sheet_width - 2 * packer.trim, packer.sheet_height - 2 *
packer.trim, fill=False, edgecolor="gray", linestyle="--"))

    color_map: Dict[str, Tuple[float, float, float]] = {}
    def get_color(label: str):
        if label not in color_map:
            h = hashlib.md5(label.encode('utf-8')).digest()
            r = h[0] / 255.0
            g = h[1] / 255.0
            b = h[2] / 255.0
            color_map[label] = (r, g, b)
        return color_map[label]

    free_leaves: List[GuillotineNode] = []
    def collect_free_leaves(node: Optional[GuillotineNode]) -> None:
        if node is None:
            return
        if (not node.used) and (node.right is None) and (node.top is
None):
            free_leaves.append(node)
            return
        collect_free_leaves(node.right)
        collect_free_leaves(node.top)
    collect_free_leaves(packer.root)

    def merge_intervals(intervals: List[Tuple[float, float]]) ->
List[Tuple[float, float]]:
        if not intervals:
            return []
        intervals = sorted(intervals)
        merged = [intervals[0]]
        for a, b in intervals[1:]:
            last_a, last_b = merged[-1]
            if a <= last_b + 1e-6:
                merged[-1] = (last_a, max(last_b, b))
            else:
                merged.append((a, b))
        return merged

    eps = max(1.0, packer.kerf)
```

```
    def draw_vertical_segment_clipped(x: float, y0: float, y1: float)
-> None:
        blocked: List[Tuple[float, float]] = []
        for rect in free_leaves:
            if rect.x - eps <= x <= rect.x + rect.width + eps:
                a = max(y0, rect.y - eps)
                b = min(y1, rect.y + rect.height + eps)
                if a < b - 1e-6:
                    blocked.append((a, b))
        blocked = merge_intervals(blocked)
        cursor = y0
        for a, b in blocked:
            if a > cursor + 1e-6:
                ax.plot([x, x], [cursor, a], color='black',
linewidth=1.2, zorder=3)
            cursor = max(cursor, b)
        if cursor < y1 - 1e-6:
            ax.plot([x, x], [cursor, y1], color='black',
linewidth=1.2, zorder=3)

    def draw_horizontal_segment_clipped(y: float, x0: float, x1:
float) -> None:
        blocked: List[Tuple[float, float]] = []
        for rect in free_leaves:
            if rect.y - eps <= y <= rect.y + rect.height + eps:
                a = max(x0, rect.x - eps)
                b = min(x1, rect.x + rect.width + eps)
                if a < b - 1e-6:
                    blocked.append((a, b))
        blocked = merge_intervals(blocked)
        cursor = x0
        for a, b in blocked:
            if a > cursor + 1e-6:
                ax.plot([cursor, a], [y, y], color='black',
linewidth=1.2, zorder=3)
            cursor = max(cursor, b)
        if cursor < x1 - 1e-6:
            ax.plot([cursor, x1], [y, y], color='black',
linewidth=1.2, zorder=3)

    def draw_cuts_from_nodes(node: Optional[GuillotineNode]) -> None:
        if node is None:
            return
        if node.right is not None:
            x = node.right.x - packer.kerf / 2
            y0 = node.right.y
            y1 = node.right.y + node.right.height
            draw_vertical_segment_clipped(x, y0, y1)
        if node.top is not None:
            y = node.top.y - packer.kerf / 2
```

```
            x0 = node.top.x
            x1 = node.top.x + node.top.width
            draw_horizontal_segment_clipped(y, x0, x1)
        draw_cuts_from_nodes(node.right)
        draw_cuts_from_nodes(node.top)

    draw_cuts_from_nodes(packer.root)

    # Підсвічуємо ВСІ вільні області (щоб приховати в них небажані
штрихи)
    for rect in free_leaves:
        ax.add_patch(plt.Rectangle((rect.x, rect.y), rect.width,
rect.height,
                                   facecolor=(0.2, 0.45, 0.95, 0.22),
edgecolor='none', zorder=3.5))

    # Малюємо деталі над всім
    for p in packer.placed_parts:
        if p.x is None or p.y is None:
            continue
        w = p.height if p.rotated else p.width
        h = p.width if p.rotated else p.height
        color = get_color(p.label)
        rect = plt.Rectangle((p.x, p.y), w, h, facecolor=(color[0],
color[1], color[2], 0.45),
                             edgecolor=color, linewidth=1.2, zorder=4)
        ax.add_patch(rect)
        ax.text(p.x + w / 2, p.y + h / 2, f"{p.label}\nID:{p.id}",
ha='center', va='center',
                fontsize=8, color='black', zorder=6,
bbox=dict(facecolor='none', edgecolor='none'))

        dim_w = f"{int(round(w))} мм"
        dim_h = f"{int(round(h))} мм"
        tick_len = 6
        ax.plot([p.x + 5, p.x + w - 5], [p.y + h, p.y + h],
color='black', linewidth=0.9, zorder=5)
        ax.plot([p.x + 5, p.x + 5], [p.y + h - tick_len / 2, p.y + h +
tick_len / 2], color='black', linewidth=0.9, zorder=5)
        ax.plot([p.x + w - 5, p.x + w - 5], [p.y + h - tick_len / 2,
p.y + h + tick_len / 2], color='black', linewidth=0.9, zorder=5)
        ax.text(p.x + w / 2, p.y + h + 3, dim_w, ha='center',
va='bottom', fontsize=8,
                bbox=dict(facecolor='none', edgecolor='none'),
zorder=6)
        ax.plot([p.x, p.x], [p.y + 5, p.y + h - 5], color='black',
linewidth=0.9, zorder=5)
        ax.plot([p.x - tick_len / 2, p.x + tick_len / 2], [p.y + 5,
p.y + 5], color='black', linewidth=0.9, zorder=5)
        ax.plot([p.x - tick_len / 2, p.x + tick_len / 2], [p.y + h -
5, p.y + h - 5], color='black', linewidth=0.9, zorder=5)
```

```
        ax.text(p.x - 6, p.y + h / 2, dim_h, ha='right', va='center',
fontsize=8, rotation=90,
                bbox=dict(facecolor='none', edgecolor='none'),
zorder=6)

    # TRIM & layout
    ax.plot([packer.trim, packer.trim], [0, packer.sheet_height],
color='black', linewidth=0.8, zorder=2)
    ax.plot([packer.sheet_width - packer.trim, packer.sheet_width -
packer.trim], [0, packer.sheet_height], color='black', linewidth=0.8,
zorder=2)
    ax.plot([0, packer.sheet_width], [packer.trim, packer.trim],
color='black', linewidth=0.8, zorder=2)
    ax.plot([0, packer.sheet_width], [packer.sheet_height -
packer.trim, packer.sheet_height - packer.trim], color='black',
linewidth=0.8, zorder=2)
    ax.tick_params(axis='both', which='both', bottom=False,
left=False, labelbottom=False, labelleft=False)
    ax.set_aspect("equal", adjustable="box")
    ax.set_xlim(-40, packer.sheet_width + 20)
    ax.set_ylim(-20, packer.sheet_height + 40)
    ax.set_xlabel("X, мм")
    ax.set_ylabel("Y, мм")
    ax.set_title(f"{title} — ККД листа: {packer.efficiency():.2f}%")
    plt.tight_layout()
    plt.show()


def print_strategy_comparison(results: Dict[str, Dict]) -> None:
    print("\nПорівняння стратегій сортування деталей:")
    print("=" * 80)
    print(f"{'Стратегія':30s} | {'ККД, %':>8s} | {'Листів':>6s} |
{'Розміщених':>10s}")
    print("-" * 80)
    for name, info in results.items():
        eff = info["efficiency"]
        sheets = info["sheets"]
        placed = info["placed_count"]
        print(f"{name:30s} | {eff:8.2f} | {sheets:6d} | {placed:10d}")
    print("=" * 80)


# ======================
#   MAIN: запуск всього процесу
# ======================
def main() -> None:
    # Параметри листа
    sheet_width = 2800.0
    sheet_height = 2070.0
    trim = 10.0
    kerf = 3.5
```

```
    parts = build_test_order()
    optimizer = AdaptiveOptimizer(sheet_width=sheet_width,
sheet_height=sheet_height, trim=trim, kerf=kerf, allow_rotation=True)

    best_strategy_name, results = optimizer.run_all_strategies(parts)
    print_strategy_comparison(results)

    # Детальна інформація про найкращу стратегію (агрегат по листах)
    best_info = results[best_strategy_name]
    best_packers: List[GuillotinePacker] = best_info["packers"]
    total_used = sum(p.used_area() for p in best_packers)
    total_area = sum(p.sheet_area() for p in best_packers)
    best_eff = (100.0 * total_used / total_area) if total_area > 0
else 0.0
    total_placed = sum(len(p.placed_parts) for p in best_packers)

    print(f"\nНайкраща стратегія: {best_strategy_name}")
    print(f"Агрегований ККД: {best_eff:.2f}% на {len(best_packers)}
листах")
    print(f"Розміщено деталей: {total_placed} з {len(parts)}")

    # Експорт усіх листів найкращої стратегії у cutting_job.json
    export_data = {"strategy": best_strategy_name, "total_sheets":
len(best_packers), "sheets": []}
    for i, p in enumerate(best_packers):
        export_data["sheets"].append({
            "sheet_index": i + 1,
            "efficiency_percent": p.efficiency(),
            "parts": [
                {
                    "id": part.id,
                    "label": part.label,
                    "x": part.x,
                    "y": part.y,
                    "width": (part.height if part.rotated else
part.width),
                    "height": (part.width if part.rotated else
part.height),
                    "rotated": part.rotated,
                }
                for part in p.placed_parts
            ]
        })
    with open("cutting_job.json", "w", encoding="utf-8") as f:
        json.dump(export_data, f, ensure_ascii=False, indent=2)
    print("Експортовано найкращу стратегію у файл: cutting_job.json")

    # Вивести список стратегій від найкращої до гіршої (за ККД) з ККД
по листах
    print("\nСписок стратегій (від найкращої до гіршої):")
```

```
    print("=" * 80)
    sorted_results = sorted(results.items(), key=lambda kv:
kv[1]["efficiency"], reverse=True)
    for idx, (name, info) in enumerate(sorted_results, 1):
        eff = info["efficiency"]
        packers = info.get("packers", [])
        sheets = len(packers)
        placed = info.get("placed_count", 0)
        per_sheet = ", ".join(f"{p.efficiency():.1f}%" for p in
packers)
        per_sheet_display = f"[{per_sheet}]" if per_sheet else ""
        print(f"{idx:2d}. {name:40s} | ККД: {eff:6.2f}% | Листів:
{sheets:3d} | Розміщено: {placed:4d} {per_sheet_display}")
    print("=" * 80)

    # Показати лише карти найкращої стратегії
    print(f"\nВідображаю найкращу стратегію: {best_strategy_name}")
    for i, p in enumerate(best_packers):
        visualize_result(p, title=f"BEST {best_strategy_name} — лист
{i+1}/{len(best_packers)} (ККД: {p.efficiency():.2f}%)",
highlight=True)


if __name__ == "__main__":
    main()
```