НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації та управління

"На правах рукопису"
УДК *519.854.2*

До захисту допущено
В.о. завідувача кафедри

_____ Олександр ПАВЛОВ
“ ____ ” _____ 20 *21* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-науковою програмою

«*Інформаційні управляючі системи та технології*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Оптимізація розміщення прямокутників на напівнескінченній стрічці»

Виконала:
студентка VI курсу, групи **ІС-91мн**
Дубіна Анастасія Володимирівна

Керівник:
проф., д.т.н., с.н.с.,
Гуляницький Леонід Федорович _____

Консультант:
доцент, к.т.н., доцент,
Жданова Олена Григорівна

Рецензент:
с.н.с., к.ф-м.н., с.н.с,
Ходзінський Олександр
Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка_____

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації та управління

Рівень вищої освіти – *другий (магістерський)*

Спеціальність – 126 «*Інформаційні системи та технології*»

Освітньо-наукова програма *«Інформаційні управляючі системи та технології»*

В.о.завідувача кафедри

_____ Олександр ПАВЛОВ

«__» _____2021 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Дубіні Анастасії Володимирівні

1. Тема дисертації «Оптимізація розміщення прямокутників на напівнескінченній стрічці», науковий керівник дисертації Гуляницький Леонід Федорович, професор, д.т.н., затверджені наказом по університету від «12» березня 2021 р. № 809-с

2. Строк подання студентом дисертації “ *10* ” *05* 20 *21* р.

3. Об’єкт дослідження – процес розміщення прямокутників на напівнескінченній стрічці.

4. Перелік завдань, які потрібно розробити:

- *виконати огляд існуючих методів розв’язування задач розміщення прямокутників;*
- *розробити алгоритм розв’язування задачі розміщення (жадібний, алгоритми локального та табу пошуку);*
- *розробити програмну реалізацію вищезгаданих алгоритмів;*
- *виконати аналіз отриманих результатів дослідження.*

5. Орієнтовний перелік графічного (ілюстративного) матеріалу

Плакат 1 Приклад роботи жадібного алгоритму 1 (ЖА1)
Плакат 2 Приклад роботи жадібного алгоритму 2 (ЖА2)
Плакат 3 Приклад роботи жадібного алгоритму 3 (ЖА3)
Плакат 4 UML-діаграма варіантів використання

Плакат 5 Блок-схема алгоритму локального пошуку
Плакат 6 Блок-схема алгоритму табуйованого пошуку
Плакат 7 Блок-схема алгоритму обміну

6. Орієнтовний перелік публікацій: *троє тез на міжнародних конференціях, одні тези на всеукраїнській конференції, одна стаття у фаховому виданні.*

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання “ *01* ” *лютого* 20 *21* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Аналіз результатів огляду існуючих рішень*	*05.02*	
2	*Порівняльний аналіз існуючих методів розв’язування задачі*	*19.02*	
3	*Формулювання змістовної постановки та математичної моделі задачі розміщення*	*01.03*	
4	*Розробка алгоритмів розв’язування задач розміщення*	*25.03*	
5	*Розробка інформаційної системи*	*01.04*	
7	*Проведення експериментальних досліджень розроблених алгоритмів*	*16.04*	
8	*Оформлення документації*	*19.04*	
9	*Подання роботи на попередній захист*	*20.04*	
10	*Подання роботи на основний захист*	*10.05*	

Студентка _____ *Анастасія ДУБІНА*

Науковий керівник _____ *Леонід ГУЛЯНИЦЬКИЙ*

РЕФЕРАТ

Магістерська дисертація: 134 с., 35 рис., 44 табл., 65 джерел,1 додаток.

Актуальність. Задача розміщення прямокутників на напівнескінченній стрічці в сучасному світі має практичне застосування та різні варіації: задача упаковки прямокутників, задача календарного планування та інші. В рамках календарного планування доцільно розглядати методи та моделі теорії розкладів. Оскільки більша частина задач теорії розкладів класифікується як задачі, що мають NP складність тому знаходження точних розв'язків за прийнятний час є неможливим. В такому випадку, можливий лише пошук локальних оптимумів, за допомогою наближених алгоритмів.

Задачі геометричного розміщення широко використовуються в різних сферах. Задача розміщення полягає у визначенні оптимального положення скінченної кількості геометричних об'єктів в заданих областях з урахуванням різноманітних обмежень. Технологічні процеси зазвичай використовують етап розкрою чи розміщення деталей. Цей етап важливий, оскільки можна досягти економії ресурсів, але він є трудомістким за рахунок необхідності пошуку оптимального розв'язку. Цей етап можна описати оптимізаційними задачами геометричного розміщення. До класичних задач досліджуваного типу відносяться задачі розкрою та упаковки, геометричного розміщення.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Ефективні методи розв'язання задач теорії розкладів» (Державний реєстраційний номер 0117U000919).

Мета дослідження – розробка та модифікація алгоритмів розміщення прямокутників на напівнескінченній стрічці для зменшення її довжини.

Для досягнення мети необхідно виконати наступні **завдання:**

– визначити клас задач, до якого відноситься дана задача;
– виконати огляд існуючих методів розв'язування задач розміщення прямокутників;

– розробити алгоритм розв’язування задачі розміщення (жадібний, алгоритми локального та табу пошуку);

– розробити програмну реалізацію вищезгаданих алгоритмів;

– виконати аналіз отриманих результатів дослідження.

Об’єкт дослідження – процес розміщення прямокутників на напівнескінченній стрічці.

Предмет дослідження – моделі та методи оптимізації розміщення прямокутників.

Наукова новизна одержаних результатів полягатиме удосконаленні існуючих алгоритмів розміщення прямокутників на напівнескінченній стрічці.

Публікації. Матеріали роботи опубліковані у збірниках: Наукового забезпечення технологічного прогресу XXI сторіччя: матеріали міжнародної наукової конференції (Т.2) (2020р, м. Чернівці), Proceedings of the International Scientific and Practical Conference (2019 р. м. Валета), Science and Technology of the XXI Century: Proceedings of the XXI International Students R&D Online Conference (2020 р, м.Київ), VI Всеукраїнська науково-практична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2021, м. Київ); та у фаховому виданні – Науковий вісник Ужгородського університету, серія «Математика і інформатика».

ПЕРЕСТАНОВКИ, РОЗМІЩЕННЯ ПРЯМОКУТНИКІВ, ТЕОРІЯ РОЗКЛАДІВ, КОМБІНАТОРНА ОПТИМІЗАЦІЯ, ЖАДІБНИЙ АЛГОРИТМ, ЛОКАЛЬНИЙ ПОШУК, ТАБУЙОВАНИЙ ПОШУК

ABSTRACT

Master's dissertation: 134 pp., 35 figs., 44 tables, 65 sources, 1 appendix.

Topicality. The problem of placing rectangles on a semi-infinite tape in the modern world has a practical application and various variations: the problem of packing rectangles, the problem of scheduling and others. Within the framework of calendar planning it is expedient to consider methods and models of the theory of schedules. Since most problems of schedule theory are classified as problems of NP complexity, it is impossible to find exact solutions in a reasonable time. In this case, only the search for local optimums is possible, using approximate algorithms.

Geometric placement problems are widely used in various fields. The task of placement is to determine the optimal position of a finite number of geometric objects in a given area, taking into account various constraints. Technological processes usually use the stage of cutting or placement of parts. This step is important because it can save resources, but it is time consuming due to the need to find the optimal solution. This stage can be described by optimization problems of geometric placement. The classical problems of the studied type include the problems of cutting and packaging, geometric placement.

Connection of work with scientific programs, plans, themes. The work was performed at the Department of Automated Information Processing and Control Systems of the National Technical University of Ukraine "Kyiv Polytechnic Institute. Igor Sikorsky »» within the theme "Effective methods for solving problems of schedule theory" (№ DR 0117U000919).

The purpose of the study is to modify and improve the algorithms for placing rectangles on a semi-infinite tape to reduce its length.

To achieve this goal must perform the following tasks:

- determine the class of tasks to which this task belongs;
- review existing methods for solving rectangle placement problems;

- develop an algorithm for solving the placement problem (greedy, local algorithms and search taboos);
- develop a software implementation of the above algorithms;
- perform an analysis of the results of the study.

The object of study is the process of placing rectangles on a semi-infinite tape.

The subject of research - models and methods of optimizing the placement of rectangles.

The scientific novelty of the obtained results will be improve existing algorithms for placing rectangles on a semi-infinite tape.

Publications. Materials of the work are published in the collections: Scientific support of technological progress of the XXI century: materials of the international scientific conference (Vol. 2) (2020, Chernivtsi), Proceedings of the International Scientific and Practical Conference (2019, Valletta), Science and Technology of the XXI Century: Proceedings of the XXI International Students R&D Online Conference (2020, Kyiv), VI All-Ukrainian scientific-practical conference of young scientists and students "Information systems and management technologies" (ISTU-2021, Kyiv); and in the professional edition - Scientific Bulletin of Uzhhorod University, series "Mathematics and Informatics".

PERMUTATION, PLACEMENT OF RECTANGLES, SCHEDULE THEORY, COMBINATORY OPTIMIZATION, GREEDY ALGORITHM, LOCAL SEARCH, TABU SEARCH

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ ОДИНИЦЬ ВЕЛИЧИН І ТЕРМІНІВ

БД	база даних
ЕОМ	електронно-обчислювальна машина
ОС	операційна система
ПК	персональний комп’ютер
СКВ	середньо-квадратичне відхилення
ТР	теорія розкладів
ЦФ	цільова функція

ВСТУП

Проблема оптимального розміщення виникає при проєктуванні різноманітних пристроїв, розміщення обладнання, вантажу, деталей для розкрою, оптимізації логістичної системи підприємства та ін.

Більшість практичних оптимізаційних задач управління ресурсами, розміщення, упаковки та розкрою зводяться до задачі розміщення скінченного числа набору прямокутних геометричних об’єктів в прямокутній ділянці розміщення. Ця задача має теоретичну та практичну цінність.

Під час планування розкладу на підприємстві іноді виникає потреба врахувати можливі часові проміжки при яких співробітники, чи машини не зможуть виконувати роботу. Для задач розкрою геометричних фігур, можлива ситуація наявності дефектів, які потрібно враховувати. Для таких задач необхідно розглядати вужчий клас розміщення – розміщення на стрічці із забороненими областями.

У першому розділі розглянуто класифікацію задач геометричного розміщення об’єктів, визначено їх особливості та наведено методи їх розв’язання.

У другому розділі сформульовано дві змістовні постановки задачі та запропоновано їх математичну модель. Для задачі розміщення прямокутників описано жадібний алгоритм та його модифікації, алгоритми локального та табуйованого пошуку, алгоритм обміну та алгоритм обміну з заборонами. Для кожного з алгоритмів наведено псевдокод та визначено переваги.

Результати експериментальних досліджень алгоритмів подано у третьому розділі. Наведено опис початкових параметрів експериментів та отриманих результатів проведених експериментів, визначено найкращий алгоритм для розв'язування задачі розміщення прямокутників.

У четвертому розділі описано вимоги до технічного та програмного забезпечення. Описано функціональну модель, побудовано діаграму пакетів, класів та компонентів. Наведено специфікацію функцій та опис структури бази даних. Розглянуто засоби розробки та наведено опис взаємодії із розробленим

застосуванням.

У п’ятому розділі наведено опис розробки стартап-проєкту, проведено технічний аудит ідеї, аналіз ринкових можливостей запуску, розроблено ринкову та маркетингову стратегію реалізації проєкту.

1 ОГЛЯД СУЧАСНИХ ДОСЛІДЖЕНЬ В ОБЛАСТІ РОЗМІЩЕННЯ ПРЯМОКУТНИКІВ НА НАПІВНЕСКІНЧЕННІЙ СТРІЧЦІ

1.1 Постановка задачі розміщення прямокутників

Задача розміщення прямокутників на напівнескінченній стрічці (також двовимірна задача, two-dimensional strip packing problem, 2DSPP) полягає в такому: потрібно розмістити задані прямокутники на напівнескінченній стрічці так, щоб довжина зайнятої частини смуги досягла мінімуму.

Згідно типології Дікхофф (Dyckhoff's) розкрою і упаковки [1], двовимірна задача розкрою-упаковки має тип 2 / N / O.

Задача 2DSPP виникла в середині XX століття і з тих пір не втрачає актуальності, оскільки до неї зводяться прикладні задачі розподілу двовимірного ресурсу, такі як:

- задача планування зайнятості;
- задача складання розкладів;
- задача розподілу пам’яті ЕОМ;
- власне проблеми упаковки і розміщення.

Разом з тим більшість робіт у зазначеній галузі присвячені наближеним методам розв’язання даної задачі. Це обумовлено тим, що 2DSPP є NP- складною задачею.

На даний момент в світі існує лише кілька підходів, що дозволяють розв’язувати двовимірну задачу розміщення точно [2].

Більшість точних методів розв’язання 2DSPP зводиться до перебору всієї множини допустимих розв’язків і знаходження серед них оптимуму. Такий перебір може бути доведений до ефективних алгоритмів [2] різного виду поліпшеннями.

Значна група алгоритмів поліпшеного перебору відноситься до методу гілок і меж (МГМ). S. Martello і D. Vigo застосували МВВ для розв’язування 2DSPP [3].

Точний метод розв’язування 2DSPP, орієнтований на невелику кількість

прямокутників, був так само розроблений А. І. Липовецьким. Він ввів поняття «зони» таким чином, що розв'язування двовимірної задачі розкрою-упаковки стало зводитися до перебору допустимих зон, а також показав, що завдання упаковки прямокутників в напівнескінченної смугу має складність $O(m!^2)$, де m - число прямокутників [4]. Використання ряду правил відсікання безперспективних гілок дозволило скоротити перебір. Цей метод був досліджений та реалізований В. В. Бухваловою [5].

S. P. Fekete, J. Schepers [6] запропонували подавати упаковки у вигляді набору інтервальних графів. Розроблений ними алгоритм здійснює перебір зазначених графів. В роботі показано, що одному інтервального графу відповідає деяке сімейство упаковок. Вони використовували цей підхід для розв'язання задачі про рюкзак.

1.2 Інший підхід до класифікації задач розміщення

Задачі розміщення використовуються у різних сферах діяльності: розкрій металевих листів, тканини, планування виробництва, розміщення товарів на складі магазину. Задачі геометричного розміщення класифікують відповідно до області застосування, метричних характеристик та орієнтації розміщення.

Розглянемо класифікацію оптимізаційних задачі розміщення об'єктів [8]:

- задачі розкрою листових матеріалів;
- задачі упакування;
- задачі календарного планування;
- задачі компонування обладнання в цехах і відсіках;
- задачі компоновочного синтезу апаратури;
- задачі розкрою натуральної шкіри, тканини;
- задачі розміщення джерел забруднюючих викидів з урахуванням рельєфу області розміщення;
- задачі розміщення об'єктів на площині з забороненими областями.

В постановках оптимізаційних задач розміщення об'єктів доцільно враховувати метричні характеристики цих об'єктів та їх форму, оскільки в процесі

розміщення вони можуть змінюватись. Розглянемо класифікацію задач із змінними метричними характеристиками [8]:

а) за характером зміни метричних характеристик об’єктів:

- пов’язані функціональними залежностями;
- можуть змінюватись незалежно один від одного в заданих діапазонах;

б) за результатом зміни метричних характеристик:

- вихідна просторова форма об’єктів залишається незмінною;
- вихідна просторова форма об’єктів змінюється;
- змінюється топологічні властивості об’єктів:
- змінюється топологічна розмірність;
- змінюється зв’язність.

Важливу роль під час розміщення об’єктів відіграє такий параметр, як кут повороту власної системи координат. Наявність кутового параметру суттєво змінює властивості задачі. Розглянемо класифікацію задач відносно кута повороту власної системи координат [8]:

- задача розміщення орієнтованих об’єктів;
- задача розміщення неорієнтованих об’єктів;
- задача розміщення об’єктів в анізотропній області.

1.2.1 Задача розкрою листових матеріалів

В задачах розкрою листових матеріалів, зазвичай розглядається використання ресурсів таких, як металеві листи, фанера, сталь, ДВП, ДСП, скло і т.п. Цю область розміщення відносять до ізотропних, оскільки на всій ділянці маємо однакові фізичні властивості незалежно від напряму. Під час дослідження задач розкрою листових матеріалів, розглядається задача розміщення геометричних об’єктів, врахувавши їх розміри, але не враховуючи їх просторові форми. Такі задачі розкрою також називають задачами одновимірного розміщення і допускають формулювання у вигляді задач лінійного програмування.

В [9] розглянута задача формування плану розкрою. План розкрою

складається з переліку способів розкрою цілісного листа матеріалу та інформації, яка частина матеріалу буде розкроєна наведеними способами. Способи розкрою підбираються таким чином, щоб план надавав заготовки в потрібній кількості. Групи заготовок, які необхідно виготовити необхідно обирати таким чином, щоб їх виготовлення могло бути сумісним. Заготовки можуть бути використані як для одного, так і для багатьох виробів. Основною технологічною вимогою визначають досягнення найменшого відсотку відходів.

Додатковими вимогами в формуванні плану розкрою можуть бути певні технологічні вимоги:

– із одного фрагменту матеріалу не створювати заготовки для декількох виробів;

– одні і ті самі заготовки не повторювати в декількох розкроїв;

– майстру, що виконує розкрій, необхідно якомога рідше виконувати переналагодження обладнання ;

– партія розкрою не була надто великою.

Не враховуючи можливі технологічні обмеження, можна визначити три основні типи задач розкрою:

– задачі максимального коефіцієнту корисного використання матеріалу, що надходить та може бути представлений у вигляді шматків однакового розміру;

– задачі зменшення відсотку відходів матеріалу, що замовляється та може бути представлений у вигляді шматків різного габариту;

– задачі зменшення відсотку відходів матеріалу, що наявний на складі та може бути представлений у вигляді шматків різного габариту.

Найчастіше зустрічаються задачі ортогонального розкрою, де в якості заготовок виступають об’єкти прямокутної форми, а матеріал надається у вигляді рулонів, прямокутних листів і т.п. В [10] наведена класифікація розкрою та математична постановка задачі.

Класифікація задач розкрою:

– одновимірний розкрій;

– гільйотинний розкрій напівнескінченної стрічки;

– гільйотинний розкрій листів.

Підхід до розв’язування 2DSPP був запропонований Ю. Г. Стояном і М. В. Новожиловою [7], які пропонують використовувати методи пошуку глобального мінімуму цільової функції для задачі 2DSPP.

Умови розміщення прямокутників описуються системою з $2m(m-1)$ лінійних нерівностей щодо координат полюсів прямокутників і такою ж кількістю цілочислових змінних. Отримана система нерівностей може бути розв'язана з використанням методів лінійного програмування.

Недоліком даного підходу є те обставина, що число нерівностей і цілочислових змінних швидко зростає зі збільшенням кількості прямокутників, що значно збільшує навантаження на обчислювальну частину розв’язування даної задачі.

Математична постановка задачі полягає в розміщенні прямокутних об’єктів різних розмірів на листах заданих розмірів з найменшими непотрібними залишками матеріалу при конкретних обмеженнях. Основними є геометричні обмеження:

– перевірка належності об’єктів до області розміщення;

– відсутність перетину об’єктів;

– направлення текстури для кожного об’єкту.

В [10] описані технологічні обмеження, що зумовлені специфікою обладнання різних підприємств:

– ширина ріжучої частини;

– довжина наскрізного різу;

– максимальна кількість обертів.

Для визначення класу задач розкрою в [11] наведені основні ознаки:

а) розмірність:

– одновимірний розкрій;

– двовимірний розкрій;

– тривимірний розкрій;

– N-мірний розкрій;

б) форма розкрою:

– розкроюються всі об’єкти на вибрані елементи (B);

– розкроюються вибрані об’єкти на всі елементі (V);

в) асортимент об’єктів, що розкроюються:

– один об’єкт, що розкроюється (O);

– ідентичні об’єкти (I);

– різні об’єкти (D);

г) асортимент елементів:

– невелика кількість різних елементів (F);

– безліч різних елементів (M);

– групи ідентичних елементів (R);

– неконгруентні фігури (C).

Подамо загальну класифікацію задач розкрою (рисунок 1.1).

Рисунок 1.1 - Загальна класифікація задач розкрою

Математичну постановку задачі можна подати так [12]. Нехай комплект складається з n заготовок кожного виду в кількості b_j , a_{ij} кількість заготовок виду j, що отримуються при розкрої однієї частини матеріалу i-м способом, c_i - цінність частини, що використовується в i-му розкрою, x_i - використання i-го розкрою. Таким чином задача формування плану розкрою зводиться до знаходження x_i, що відповідають обмеженням:

$$\sum_{i=1}^{N} a_{ij} x_i \geq b_i,\ j = \overline{1, N} \qquad (1.1)$$

$$x_i \geq 0\ ,\ i = 1,\ N, \qquad (1.2)$$

та оптимізують цільову функцію

$$\min \sum_{i=1}^{N} c_i x_i \qquad (1.3)$$

В розглянутій математичній постановці вимір в явному вигляді не фігурує, оскільки замість цього вводиться поняття вартості i-ї частини матеріалу. За рахунок

такого підходу легко розширити запропоноване формулювання для *n*-вимірної задачі, ввівши додаткові фактори.

Для розв'язування подібних задач розкрою використовують гібридний метод еволюційної метаевристики [13], [14], метод зон [15], рекурсивний метод [16], генетичний алгоритм [17], модифікований метод гілок та меж [18], алгоритм сумісної роботи групового декодеру та алгоритму імітаційного відпалу [19].

1.2.2 Задача упакування

Задача упакування дуже часто пов'язується з задачею розкрою. Відповідно до [10, 20-24] розглянемо класифікацію задач упакування.

Задачі одновимірного упакування:

- упакування рюкзака;
- одновимірне упакування.

Упакування в двовимірні контейнери:

- упакування кругів:
- круги в крузі;
- круги в квадраті;
- круги в рівнобедреному прямокутному трикутнику;
- круги в правильному трикутнику.

Упакування квадратів:

- квадрати в квадраті;
- квадрати в крузі.

Упакування прямокутників:

- прямокутники однакових розмірів;
- прямокутники різних розмірів.

Упакування в тривимірні контейнери:

- упакування сфер.

В задачі ортогональної упаковки об'єкти в контейнерах можуть бути наведені в різних моделях [25]: матрична модель, вузлова модель, блочна модель, модель

віртуальних об’єктів, модель потенційних контейнерів.

Одновимірне упакування

Задача одновимірної упаковки широко використовується в якості моделі розподілу ресурсів різного типу. Однією з відомих задач одновимірної упаковки є задача упакування рюкзака. Задача про рюкзак – одна із класичних задач дискретної оптимізації. Нехай маємо множину n предметів, кожний з яких має визначену c_i вартість та w_i вагу, де i – номер предмету.

Необхідно сформувати такий набір з цих предметів, який би мав найбільшу сумарну вартість серед усіх наборів та сумарна вага не перевищує P – місткість рюкзака [24].

Математичну постановку такої задачі можна сформувати так [14,19]. Необхідно знайти булевий вектор $(x_1, x_2, ..., x_n)$, який максимізує суму

$$\sum_{i=1}^{n} c_i x_i \ , \tag{1.4}$$

і при цьому виконувалась б нерівність

$$\sum_{i=1}^{N} a_i x_i \le P \ . \tag{1.5}$$

Загальна постановка задачі одновимірної упаковки така [22].

Нехай маємо множину елементів $E = \{e_1, e_2, ..., e_n\}$, розміри елементів $S(E) = \{s(e_1), s(e_2), ..., s(e_n)\}$та множину блоків $B = \{b_1, b_2, ..., b_n\}$, що мають розміри $S\ (B) = \{s(b_1),\ s(b_2),...,\ s(b_n)\}$. Необхідно розмістити елементи в блоки, заповнюючи кожний з блоків до максимально можливого рівня і мінімізуючи загальну кількість заповнених блоків.

Задача одновимірного продовженого упакування, яка відома як задача одновимірного прямокутно-орієнтованого упакування, використовується в блочній технології для розв’язання задачі двовимірної упаковки в стрічку [27].

Двовимірне упакування

Задача двовимірного упакування має велику кількість різновидів: двовимірна упаковка в напівнескінченну стрічку, лінійна упаковка по вазі, вартості і т.п.. В [23] розглянута задача розміщення прямокутників на напівнескінченній стрічці. Нехай маємо набір прямокутників з відомими розмірами кожного, стрічка, ширина якої відома. Необхідно розмістити таким чином, щоб прямокутники не накладалися один на одного і при цьому зменшувалася б загальна довжина стрічки, на якій розміщені прямокутники.

Тривимірне упакування

Задача тривимірної упаковки знайшла широке практичне застосування в різних сферах. Упакування коробок в контейнер, кузов транспортного засобу – одна з найскладніших проблем пакування, оскільки в реальному світі виникає ряд обмежень, що впливає на процес розміщення [20]. Розглянемо постановку задачі, що наведена в [28]. Нехай маємо область тривимірного простору з заданою шириною, висотою та довжиною, та маємо множину блоків різних розмірів в контейнері, що обмежений в паралелепіпеді. Необхідно розмістити блоки в контейнері таким чином, щоб зменшити об’єм пустоти в області упаковки.

В залежності від типу блоків, задачі можна класифікувати як однорідні та неоднорідні. Також окремо можна розглянути класифікацію задач, комбінуючи різні типи контейнерів та блоків, що наведено в [20].

Для розв’язування зазначених задач упакування використовують алгоритм паралельного табу пошуку [29], метод керованого локального пошуку [30], евристичний алгоритм [31], гібридний алгоритм [32], біонічні методи [28].

1.2.3 Задача календарного планування

Задача календарного планування є ефективним інструментом керування проєктам, ресурсами, обладнанням підприємства. Календарний план чітко описує етапи виробництва, їх тривалість, кількість та використання ресурсів. Задачі упорядкування робіт, що виконуються машинами, при використанні будь-яких ресурсів визначають як задачі теорії розкладів [33]. Основними елементами задачі

теорії розкладів є операція, робота та машина. Для формування задачі теорії розкладів необхідно визначити набір робіт, які треба виконати, та набір машин, що виконують операції.

В [34] описана класифікація задач теорії розкладів відносно кількості, конфігурації машин, порядком виконання робіт та додатковими обмеженнями.

Класичною задачею теорії розкладів є призначення робіт для виконання ідентичними машинами. Ця задача, зокрема, досліджувалася в [35]. Нехай маємо m машин ($M_i, i = 1, m$), які мають однакову конфігурацію та можуть працювати паралельно і одночасно, та n робіт ($J_j, j = 1, n$), які необхідно надати для виконання машинам так, щоб зменшити загальний час роботи машин.

Для розв'язування задач календарного планування використовують генетичний алгоритм [36, 27, 34], псевдополіноміальні алгоритми [38], гібридний алгоритм [39], метод гілок та меж [40], алгоритм мурашиної колонії [34].

1.2.4 Задача розміщення об'єктів на площині з забороненими областями

Задача розміщення об'єктів на площині з забороненими областями знайшла практичне застосування в різних сферах. Вважають, що цей тип задач є окремим випадком задач розкрою, упаковки та календарного планування, за рахунок додаткових обмежень. Наприклад, маємо металевий лист з якого необхідно зробити заготовки, але цей лист області з дефектами. Ці області заборонено використовувати для виготовлення заготовок. Тому необхідно розкроїти цей лист таким чином, щоб використана довжина листа була найменша. В рамка календарного планування заборонені області можна інтерпретувати як заплановане технічне обслуговування обладнання і тому використання обладнання в цей період заборонено. В [41] подано практичне застосування: розподіл територій для відео-контролю можливих лісових пожеж, забороненими областями є водойми та болотні місцевості.

Об'єкти, що розміщаються на площині можуть мати будь-які геометричні розміри: прямокутники, квадрати, круги, багатокутники і т.п. Відповідно,

заборонені області також можуть мати різні розміри. У [42] в якості області покриття розглядають багатокутник, а заборонені зони наведені у вигляді зв'язаних багатокутників. Описана система називається багатозв'язним ортогональним полігоном. Задача мінімального покриття прямокутної області з забороненими областями розглянута у [43], де покриття відбувається кругами. Практичне застосування розглянутої задачі є розміщенням телефонних станцій для покриття стільникового зв'язку.

В [44] розглянута задача календарного планування, забороненими періодами є заданий час простою.

Для розв'язування розглянутих задач доцільно використовувати ті самі методи і алгоритми, як для задач розкрою, упаковки та календарного планування.

1.2.5 Задача компоновочного синтезу апаратури

Під час проектування чи створення систем матеріальних об'єктів важливо враховувати їх геометричні характеристики. Основними задачами компоновочного синтезу є задачі структурного та геометричного синтезу [45]. Серед задач структурного синтезу при проектуванні станків можна виділити два характерні класи: задачі покриття і розбиття. Геометричний синтез при компоновочном проектуванні конструкції включає задачі розміщення та трасування. Типовою задачею розміщення є визначення планування ділянки з обладнанням , автоматичної лінії чи цеха. Задачі трасування тісно пов'язані з задачами розміщення конструктивних моделей і зміст полягає в визначенні геометрії з'єднань. В [46] описана математична постановка задачі синтезу компоновочної схеми базових підтримуючих конструкцій.

1.2.6 Задача розкрою натуральної шкіри, тканини

Задачу розкрою натуральної шкіри та тканини можна віднести до розглянутої вище задачі розкрою. Нехай маємо множину комплектів деталей, задача розміщення зводиться до такого розміщення в задані області, що забезпечить найкраще їх розміщення. При розкрої натуральної шкіри на деталі взуття, тканини на деталі

одягу вважають задачею несистемного чи нерегулярного розміщення фігур. Формування розкрою деталей на натуральному матеріалі ускладнює задачу розміщення об’єктів, оскільки натуральні матеріали є анізотропними областями [47].

1.2.7 Задача розміщення пожежонебезпечних об’єктів з урахуванням рельєфу області розміщення

Сучасна діяльність людини дуже сильно впливає на навколишнє середовище і є причиною нових небезпек. За рахунок пожеж відбувається забруднення атмосфери, водойми забруднюються попелом, лісові пожежі знищують види рослин та тварин, збільшується ризик виникнення ураганів та тайфунів. Під час розміщення пожежонебезпечних об’єктів доцільно враховувати рельєф розміщення та наслідки можливого негативного впливу. У [48] розглянута постановка оптимізаційної задачі розміщення пожежонебезпечних об’єктів. Нехай маємо багатозв’язну область і відома множина випуклих заборонених областей. Тому розміщення пожежонебезпечних об’єктів зводиться до того, щоб максимальна загальна концентрація аерозольних викидів під час пожеж була найменшою. В [49] модель забруднення подана у вигляді восьмикутника, що гарантує зменшення концентрації забруднення за межами, у випадку, коли областю є рівнина і будь-який рельєф відсутній. Розглянута модель зводиться до оптимізаційної задачі розміщення орієнтованих восьмикутників.

1.3 Підходи до формалізації та розв’язування задач розміщення прямокутників

Задачі розміщення прямокутників на напівнескінченній стрічці в сучасному світі має практичне застосування та різні варіації: задача упаковки прямокутників, задача календарного планування та інші. В рамках календарного планування доцільно розглядати методи та моделі теорії розкладів. Оскільки більша частина задач теорії розкладів класифікуються як задачі, що мають NP складність тому побудова оптимального розв’язку неможлива. В такому випадку, можливий лише пошук локальних оптимумів, за допомогою евристичних алгоритмів.

В [50] розглянута задача ортогональної упаковки прямокутників в одну нескінченну стрічку (strip packing problem). Подібна задача актуальна та має практичне застосування в різних сферах: розкрій матеріалу для виготовлення текстильної продукції, розміщення мікросхем під час розробки нових електронних приладів, розміщення задач для обчислення на кластері. В цій роботі для упаковки використовують неперервну стрічку, не поділяючи її на рівні. Прямокутники, які необхідно розмістити мають різну ширину та довжину. Заборонено накладати ці прямокутники та їх перевертання, тому розміщення виконують таким чином, щоб зменшити їх загальну висоту. В основі методу упаковки були розглянуті два алгоритми, що виконувались послідовно: генетичний алгоритм, модифікований алгоритм Нелдера-Міда та алгоритм ущільнення упаковки. Результатами цих досліджень стали розроблена програма мовою C++ та результати проведених експериментів зі зміною розмірів прямокутників та їх кількістю. Початкові параметри для роботи генетичного алгоритму варто обирати спираючись на співвідношення розмірів початкових прямокутників (їх ширини) та ширини стрічки. Запропонований алгоритм швидко виконує упаковку, якщо ширина прямокутників значно менша за ширину стрічки. Запропонований алгоритм може використовуватись у багатовимірних задачах. В алгоритмі доцільно використати розпаралелювання обчислень. Недоліком цього алгоритму є те, що в ньому не враховується , що ділянка стрічки може містити певні недоліки, тому такі ділянки заборонено використовувати для упаковки. Наведений алгоритм ортогональної упаковки можна використати для розміщення прямокутників на напівнескінченній стрічці лише тоді, коли стрічка має лише один рівень, а ширина прямокутників рівна ширині стрічці та відсутні ділянки з забороною на розміщення прямокутників.

В [51] розглянута математична модель задачі теорії розкладів. Елементами цієї моделі є роботи, що мають тривалість виконання, момент надходження у систему, та машини, що можуть бути ідентичним чи пропорційними, працюють паралельно. З обмежень визначено лише те, що робота виконується лише 1 раз, 1

машиною і без переривань. Для моделювання реального часу доцільно використовувати періоди та врахування витрат на затримку виконання робіт. Для побудови розкладу був розглянутий алгоритм, який дозволяє зменшити експлуатаційні витрати, чи затримку витрат і тд. В запропонованому алгоритмі обчислюється зважені суми: сумарний час обробки, та сумарні експлуатаційні витрати, витрати на затримку та вартість затримки з часу прибуття. Найголовнішим є те, що роботи можуть прибувати на обробку в будь-який момент часу і тому, в найгіршому випадку сформується n(s+m+1)+s рівнянь та nms двійкових змінних. Результатом роботи алгоритму є призначення роботи на виконання машинами та встановлення точного часу, коли всі завдання повинні бути оброблені.

В [34] планування являється однією з форм прийняття рішень, які регулярно використовуються в багатьох галузях виробництва. Для організації планування в компаніях використовують математичні прийоми та евристичні методи. В цій роботі описані основні моделі та задачі теорії розкладів, які актуальні в сучасному світі та знайшли широке використання при організації роботи різних компаній. В даній публікації для опису організації виробництва використовується набір робіт, кожна робота має свій час виконання, та машини, що виконують роботи. Наведені методи побудови розкладів для різних класів задач ТР. Один з розглянутих методів - метод критичного шляху описаний для виконання робіт послідовними машинами і спрямований на те, щоб зменшити час виконання машинами всіх робіт. Алгоритм, реалізується так, що роботи на виконання машинами обираються таким чином, щоб зменшити найбільший час завершення виконання робіт машинами. В главі «Інтервал планування, бронювання та розклад» розглянута модель бронювання, де існує певна кількість заходів, де кожен захід має час початку, час завершення і його вагу. В цій моделі враховані певні простої чи затримки в часі, але мета максимізувати кількість заходів. Алгоритми, що наведені в роботі є наближеними, доцільно використати їх для дослідження проблеми розміщення прямокутників на напівнескінченній стрічці, модифікувавши таким чином, щоб врахувати наявність паралельних машин, що можуть працювати одночасно та затримки.

Отже, з розглянутих публікацій можна стверджувати, що задача розміщення прямокутників на напівнескінченній стрічці знайшла практичне застосування в різних сферах діяльності. За рахунок наявності «дірок» на стрічці можна стверджувати, що задача актуальна, але більшість наукових публікацій не охоплюють дану проблему.

Оскільки, для поданої задачі можна використати модель ТР, необхідно розглянути методи комбінаторної оптимізації, які були розроблені для задач ТР. Серед відомих підходів та методів можна виділити наступні [52]:

– евристичні методи;
– метод динамічного програмування;
– метод гілок та меж;
– метаевристичні методи;
– графічний метод;
– методи випадкового пошуку (глобального випадкового пошуку, локальної варіації);
– методи цілочислового програмування (лінійного, нелінійного, булевого);
– метод програмування в обмеженнях;
– гібридні методи;
– генетичні алгоритми та еволюційні стратегії.

Евристичні (наближені) методи мають широке використання в наш час, оскільки вони дозволяють знаходити розв'язки складних задач, враховуючи одну або декілька властивостей оптимального розв’язку, на основі яких відбувається скорочення перебору можливих розв’язків. Оскільки у кожної задачі можуть бути особливі властивості оптимальних розв’язків, тому евристичні алгоритми в різних задачах можуть бути різними [53]. До переваги цих методів можна віднести те, що будується розв’язок задачі, зменшується кількість варіантів перебору розв’язків. Недоліком подібних методів можна вказати те, що побудований розв’язок задачі

може бути близьким до оптимального розв’язку, але не завжди оптимальним. Тому, для задання початкового варіанту розміщення прямокутників на напівнескінченній стрічці буде досліджено жадібний алгоритм, а для його покращення будуть використані додаткові модифікації. Зазвичай, жадібний алгоритм лежить в основі інших алгоритмів для отримання початкового розв’язку.

Метаевристичні методи з’явилися за рахунок покращення евристичних алгоритмів. Вважають, що такі методи були запозичені з інших наук [58].

Наприклад, ідея генетичного алгоритму запозичена з генетики, де кожен розв’язок покращується за рахунок мутації попередніх розв’язок, або їх схрещувань. Ідея методу мурашиних колоній була отримана з біології, і заснована на спостереженнях за життям мурах [57]. Мурахи намагаються знайти їжу, що знаходиться близько до мурашника, але ця інформація їм невідома, тому вони залишають один одному «феромонні сліди», запах яких можуть використовувати інші мурахи і йти в тому ж напрямі, куди ведуть більшість мурашиних слідів.

Метод бджолиних колоній визначають одним з нових методів оптимізації і характеризується високою ефективністю [64]. Ідея цього методу полягає в тому, що для вибору нових місць з нектаром бджоли-розвідники відправляються на пошуки, знайшовши так джерело, бджола повертається у вулик і повідомляє іншим бджолам про розташування цього нектару, та про запаси нектару. Частина бджіл вирушила за нектаром, а інша частина вирушила до місця, про яке повідомила інша бджола. Перевагою цього методу є те, що він не чуттєвий до локальних оптимумів [56] .

Перевагами метаевристичних методів є те, що вони володіють двома особливостями:

– в результаті їх роботи будуються декілька розв’язків;

– відображають загальний підхід;

– побудова кожного розв’язку заснована на накопичуванні знань о якості попередніх отриманих розв’язків.

Метод гілок та меж використовує послідовну паралельну схему побудови

дерева можливих варіантів Спочатку шукається допустимий план і для кожного допустимого варіанту визначається верхня межа цільової функції. Гілки дерева можливих варіантів, для яких верхня межа нижче наближеного розв’язку і з подальшого розгляду виключаються [59].

Перевагою гібридних алгоритмів є те, що вони будуються на основі наближених алгоритмів та їх комбінацій. Для розв’язання багатьох комбінаторних задач доцільно використовувати метаевристичні методи: генетичні алгоритми, локальний пошук з заборонами, мурашині колоній та інші. На основі цих алгоритмів можна побудувати нові модифіковані алгоритми. В деяких випадках використання гібридних покращує ефективність існуючих метаевристичних алгоритмів. Наприклад, для пошуку оптимального розкладу мурашиним алгоритмом необхідно в середньому 27 мурах, а гібридному – 5 [60]

Методи глобального випадкового пошуку та методи локальної варіації класифікують до наближених алгоритмів, оскільки на кожній ітерації вибір нового розв’язку відбувається випадковим чином, в результаті реалізації певного випадкового процесу [56]. Більшість алгоритмів глобальної оптимізації були засновані на комбінація локальних алгоритмів та визначенні початкових точок, що забезпечували глобальність. Основна ідея локального пошуку – перебір всіх допустимих розв’язків, близьких до оптимального розв’язку. Одним із недоліків методів локального пошуку – їх неповнота: пошук може зупинитися у локальному мінімуму, що може не бути глобальним розв’язком [54].

Висновки до розділу

У першому розділі здійснено літературний огляд за темою дослідження. Розглянуто та проаналізовано підходи до класифікації задач розміщення геометричних фігур. Наведено практичне застосування задачі геометричного розміщення на прикладах розкрою на шкіряних заготовках та розміщення пожежонебезпечних об’єктів.

Досліджено моделі для розглянутих задач та наведені методи їх

розв’язування. Для задачі розміщення прямокутників на напівнескінченній стрічці доцільно використовувати модель задачі теорії розкладів.

Для задачі розміщення прямокутників визначено методи, які доцільно використовувати: метаевристичні, випадкового пошуку та генетичні.

2 МОДЕЛІ ТА МЕТОДИ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ РОЗМІЩЕННЯ ПРЯМОКУТНИКІВ НА НЕСКІНЧЕННІЙ СТРІЧЦІ

2.1 Змістовна постановка задачі

Меблевий завод займається виробленням різної фурнітури. Для виготовлення різних видів меблів використовують дошки з дерева різної довжини, але однакової ширини та товщини.

Завод має необмежений ресурс пресованої деревини, з якої вирізаються дошки. Майстер по дереву має значення довжин дошок і його задача спланувати процес вирізки. Лист пресованої деревини умовно поділяють на декілька ярусів, що мають однакову ширину, яка відповідає ширині дошок, і довжину. Листи пресованої деревини можуть мати на деяких ділянках брак, що зумовлено видом дерева. Ці ділянки з браком заборонено використовувати для вирізки дошок і подальшого виробництва меблів, тому майстер повинен врахувати це.

Майстер формує план вирізки дошок, починаючи з певного краю листа. Його задача розмістити дошки, які необхідно вирізати, врахувавши їх довжини, місця з браком на листі та економне використання деревини, тому процес розміщення відбувається таким чином, щоб використовувався найменший обсяг ресурсів.

Отже, основною метою є формування розміщення, зменшуючи загальну довжину використаних дошок.

Задачу розміщення прямокутників на напівнескінченній стрічці можна трактувати, як задачу теорії розкладів ввівши наступні позначення у таблиці 2.1.

Таблиця 2.1 – Позначення для задачі розміщення прямокутників

Терміни розміщення прямокутників	Терміни теорії розкладів	Змістовна постановка задачі
Напівнескінченній стрічка	Множина машин, яківиконують роботу	Співробітники
Множина прямокутників	Множина робіт, які необхідно надати на виконання	Замовлення
Дірки	Час технічного обслуговування машин	Заплановані зустрічі

У компанії з ремонту комп'ютерів працює команда спеціалістів, яку очолює

керівник. На початку дня для команди фахівців видається перелік замовлень, які необхідно виконати.

Кожне замовлення має різну складність, це впливає на час їх виконання. Замовлення можуть виконуватись паралельно різними співробітниками та в різній послідовності, але одне замовлення виконується лише одним співробітником. Співробітнику заборонено починати виконувати нове замовлення, якщо він не завершив попереднє.

Робочий день у спеціалістів починається в один і той самий час, але закінчується по-різному для кожного і лише тоді, коли співробітник виконає всі свої замовлення.

Протягом дня у кожного фахівця на різний час заплановані зустрічі. На момент зустрічей співробітник не повинен мати незавершеного замовлення та не може починати виконувати нове. Задача тімліда на початку дня, врахувавши складність кожного замовлення, розподілити їх між фахівцями таким чином, щоб загальний час виконання усіх замовлень був найменшим.

2.2 Математична постановка задачі

Для опису задачі розміщення прямокутників на напівнескінченній стрічці можна використати дві моделі: модель упаковки та розкрою в стрічку, або модель задачі теорії розкладів.

Наведемо модель упаковки та розкрою прямокутників в стрічку.

Нехай маємо множину, що складається з N прямокутників $A=\{a_1, a_2, ..., a_N\}$, які необхідно розмістити на напівнескінченній стрічці , що складається з K однакових за шириною рівнів, на кожному з рівнів маємо максимальну кількість дірок – M. На k-му рівні задана відстань x_{kj} від початку стрічки до j-ї дірки, маємо, що $k=\overline{1,K}$, $j=\overline{1,M}$; b_{kj} – довжина j-ї дірки на k-му рівні, z_i – відстань від початку стрічки до початку i-го прямокутника, де $i=\overline{1,N}$ та y_i- номер рівня, на якому міститься i-й прямокутник, $i=\overline{1,N}$ [1].

Нехай $x_{y_i 0} = b_{y_i 0} = 0$, $x_{y_i M+1} = \pm\infty$, тоді математичну модель надамо в наступному вигляді:

$$x_{k_j} \geq 0,\ b_{k_j} \geq 0,\ k = \overline{1,K},\ j = \overline{1,M}, \tag{2.1}$$

$$\alpha_i > 0,\ z_i \geq 0,\ i = \overline{1,N}, \tag{2.2}$$

$$y_i \in \{1,...,K\},\ i = \overline{1,N}, \tag{2.3}$$

Для кожного i і усіх $s \in \{1,...,N\}$, таких, що $y_i = y_s$, $z_i < z_s$:

$$z_i + \alpha_i \leq z_s, \tag{2.4}$$

для кожного i, $i = \overline{1,N}$ існує лише одне і тільки одне $l \in \{0,1,...,M\}$, таке, що:

$$x_{y_i l} + b_{y_i l} \leq z_i \leq x_{y_i l+1} - \alpha, \tag{2.5}$$

$$\max_{i=1,N}(z_i + \alpha_i) \longrightarrow \min. \tag{2.6}$$

Враховуючи специфікацію обмежень використання традиційних методів нелінійного програмування для розв'язування задачі (2.1)-(2.6) зазвичай неможливо чи не раціонально. Для більш ефективного пошуку розв'язку розглянемо задачу розкрою в комбінаторному просторі. Розв'язок поставленої задачі (розміщення) однозначно описуються векторами $z = (z_i)$ та $y = (y_i)$, $i = \overline{1,N}$. Якщо l-й прямокутник в розміщенні R(Z,Y) розміщений останнім на k-му рівні, то величина $f_k = z_i + \alpha_l$ рівна довжини частини k-го рівня, що зайнята прямокутниками. Максимальна довжина частини рівня, зайнята прямокутниками, виступає критерієм якості розміщення:

$$f(R(Z,Y)) \equiv \max_{k=1,K} f_k = \max_{i=1,N}(z_i + \alpha_i). \tag{2.7}$$

Задача полягає в пошуку варіанту розміщення R*(Z,Y), при якому довжина зайнятої частини стрічки була б мінімальна:

$$f(R^*(Z,Y)) \equiv \max_{\{R(Z,Y)\}} f(R(Z,Y)). \tag{2.8}$$

Зведемо цю задача до оптимізаційної задачі з використанням перестановок. В подальшому з прямокутником a_i асоціюємо його номер i. Розглянемо простір перестановок:

$$P_k = \{p = (p_1, ..., p_N) : p_s \in \{1, ..., N\}, p_s \neq p_t, s \neq t, s, t = \overline{1, N}\}, \quad (2.9)$$

де елементи перестановки р визначаються по рівності $p_s = i$, i – номер прямокутника, s – місце i–го прямокутника в перестановці p, $i, s = \overline{1, N}$. Кожній $p \in P_N$ поставимо у відповідність розміщення R'(Z',Y'), побудоване за φ–алгоритмом:

– i – й крок алгоритму ($i = \overline{1, N}$);

– в розміщення включається прямокутник з номером p_i. Для цього визначається найменш завантажений рівень, тобто такий, при розміщенні на якому даного прямокутника p_i, відповідно z_{p_i} було б мінімальним. Якщо таких рівнім декілька, то обираємо рівень з найменшим номером.

– в розміщення включається прямокутник з номером p_i . Для цього визначається найменш завантажений рівень, тобто такий, при розміщенні на якому даного прямокутника , відповідно було б мінімальним. Якщо таких рівнім декілька, то обираємо рівень з найменшим номером.

Значення нижньої границі цільової функції дозволяє оцінити точність отриманих результатів розв'язання.

Вважаємо, що всі дірки розміщені таким чином, що всі величини $\overline{x}_{kj} = x_{kj} + b_{kj}$, $k = \overline{1, K}, j = \overline{1, M}$, менше мінімально можливого значення довжини зайнятої частини стрічки.

Обчислимо нижню границю C цільової функції $f(R(z, y))$ за наступним алгоритмом.

КРОК 1. Візьмемо $C_0 = 0$, $C_i = \frac{1}{K}\sum_{i=1}^{N} a_i$

КРОК 2. $j = 1$

КРОК 3. Побудуємо множину Q_j:

$$Q_j = \{ b_{km} : C_{j-1} \leq x_{km} \leq C_j - b_{km}, k = \overline{1, K}, m = \overline{1, M}\}.$$

КРОК 4. Якщо $Q_j \neq 0$, то обчислюємо величини

$$B_j = \frac{1}{K} \sum_{b_{km} \in Q_j} b_{km}\,,\ C_{j+1} = C_j + B_j.$$

Вважаємо $j = j + 1$ і переходимо до пункту 3; якщо $Q_j = 0$, то ітеративний процес закінчується, вважаємо $\bar{C} = C_j$ і переходимо до пункту 5.

КРОК 5. Побудуємо множину V:

$$V = \{d_j^k : d_j^k < \min_{i=\overline{1,N}} a_i\,, k = \overline{1,K}\,, j = \overline{1,M-1}\},$$

де $d_j^k = x_{kj+1} - x_{kj} + b_{kj}$, та обчислюємо величину

$$D = \frac{1}{K} \sum_{d_j^k \in V} d_j^k$$

КРОК 6. Обчислюємо нижню границю $C = \bar{C} + D$.

2.3 Жадібний алгоритм та його модифікації

Для розв'язання поставленої задачі доцільно розглянути жадібний алгоритм [61], де на кожному кроці, формуючи черговий компонент вектору розв'язку, робить лише локально найкращий вибір. Оскільки кінцевий розв'язок, отриманий жадібним алгоритмом, не завжди буде оптимальним, тому розглянемо класичний жадібний алгоритм та дві його модифікації.

В кожному алгоритмі вхідними даними є кількість прямокутників (N) та їх розміри (a_i), кількість рівнів (K) та інформація про дірки на кожному з рівнів. Розглянемо особливості кожного алгоритму.

2.3.1 Жадібний алгоритм без модифікацій (ЖА1)

Особливість жадібного алгоритму без модифікацій (ЖА1) в тому, що прямокутники впорядковуються за не зростанням їх довжин. Тобто, першим буде розміщений найдовший прямокутник, а останнім – найкоротший. На кожному етапі розглядаються всі можливі варіанти розміщення і кожен наступний прямокутник буде розміщатися на тій позиції, де загальна довжина зайнятої частини стрічки зміниться мінімально або ніяк.

Наведемо псевдокод жадібного алгоритму ЖА1.

1 **procedure** greedy_1

2 ***α*:=** впорядковані прямокутники (a_i) за не зростанням їх довжин;

3 **while** не всі прямокутники розміщені **do**

4 обрати прямокутник a_l;

5 s_i:= довжина рівня *i*, якщо там буде розміщено прямокутник;

6 *j*:= номер рівня *i*, де s_j найменше або при розміщенні не змінилось;

7 на рівень *j* помістити прямокутник a_l;

8 **end while**;

9 **return** найдовшу довжину рівня;

10 **end**

Примітки

Якщо у рядку 6 отримано декілька рівнів, обирається той, що має найменший номер.

В розглянутому алгоритмі, прямокутники розміщуються таким чином, щоб всі наступні прямокутники незначно впливали на максимальну довжину рівнів, або зовсім мне впливали. Це надає можливість розміщувати останні прямокутники між іншими прямокутникам та найближчими до них дірками.

2.3.2 Жадібний алгоритм з оберненим порядком вибору прямокутників (ЖА2)

Алгоритм ЖА2 є модифікацією попереднього за рахунок того, що прямокутники впорядковуються за не спаданням їх довжин. Отже, першим буде розміщуватися прямокутник з найменшою довжиною, а останній – з найбільшою.

Наведемо псевдокод алгоритму ЖА2.

1 **procedure** greedy_2

2 ***α*:=** впорядковані прямокутники (a_i) за не спаданням їх довжин;

3 **while** не всі прямокутники розміщені **do**

4 обрати прямокутник a_l;

5 s_i:= довжина рівня *i*, якщо там буде розміщено прямокутник;

6 *j*:= номер рівня *i*, де s_j найменше або при розміщенні не змінилось;

7 на рівень *j* помістити прямокутник a_l;

8 **end while**;

9 **return** найдовшу довжину рівня;

10 **end**

На відміну від попереднього, в розглянутому алгоритмі необхідно компактно розмістити в першу чергу прямокутники з найменшою довжиною. Цей підхід надає можливість одразу перекрити існуючі місця між дірками.

2.3.3 Жадібний алгоритм з випадковим порядком вибору прямокутників (ЖА3)

Алгоритми, в яких прямокутники обираються у визначеному порядку, рідше знаходять розв'язок із задовільною точністю. Більшість наближених алгоритмів отримують кращі результати за рахунок використання випадкових процесів. Тому, необхідно розглянути модифікацію класичного алгоритму (ЖА3) за рахунок упорядкування робіт випадковим чином.

Наведемо псевдокод алгоритму ЖА3.

1 **procedure** greedy_3

2 ***α*:=** впорядковані прямокутники (a_i) випадковим чином;

3 **while** не всі прямокутники розміщені **do**

4 обрати прямокутник a_l;

5 s_i:= довжина рівня *i*, якщо там буде розміщено прямокутник;

6 *j*:= номер рівня *i*, де s_j найменше або при розміщенні не змінилось0;

7 на рівень *j* помістити прямокутник a_l;

8 **end while**;

9 **return** найдовшу довжину рівня;

10 **end**

Перевагами розглянутого алгоритму є те, що прямокутники можуть надходити в систему не одночасно, а в довільний момент часу, на відміну від ЖА1, ЖА2, де необхідно знати довжини прямокутників для їх упорядкування. ЖА3 надає можливість розміщувати прямокутники в реальному часі.

2.3.4 Приклади отриманих розміщень прямокутників алгоритмами ЖА1, ЖА2, ЖА3.

Розглянемо задачі розміщення прямокутників на напівнескінченній стрічці з дірками.

Приклад 1

Нехай маємо *N*=10 прямокутників, з довжинами *A*={1, 3, 4, 6, 1, 2, 4, 6, 7, 1} та стрічку з *K*=4 рівнями. На кожному рівні маємо M=3 дірки (2.14).

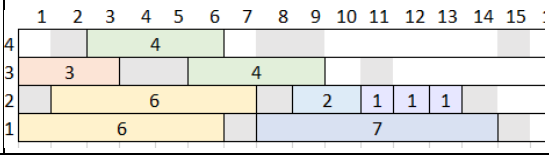
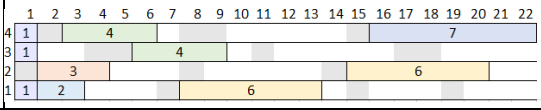
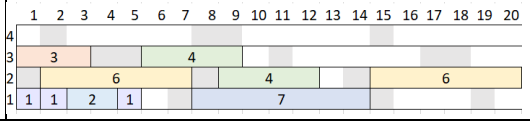
$$b=\begin{pmatrix}1 & 2 & 1\\ 2 & 1 & 2\\ 1 & 1 & 1\end{pmatrix},\ x=\begin{pmatrix}1 & 7 & 14\\ 3 & 10 & 16\\ 0 & 7 & 13\\ 6 & 14 & 18\end{pmatrix}, \qquad (2.14)$$

де b_{kj} - довжина *j*-й дірки на *k*-м рівні та x_{kj} - відстань від початку стрічки до *j*-ї дірки на *k*-му рівні, де $k=\overline{1,K}$, $j=\overline{1,M}$.

Результати роботи алгоритмів наведемо у таблиці 2.2.

Для розглянутої задачі, де на стрічці багато дірок доцільно використовувати ЖА1, проте ЖА3 показав кращі результати за ЖА2.

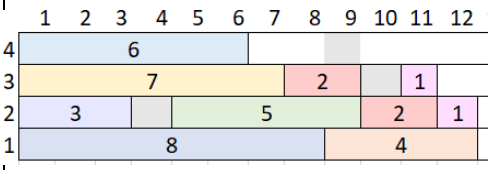
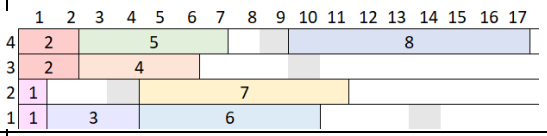
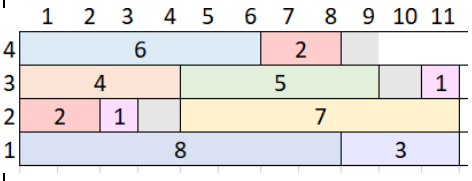
Таблиця 2.2 – Отримані результати

Алгоритм	Порядок прямокутників	ЦФ	Розміщення
ЖА1	7, 6, 6, 4, 4, 3, 2, 1, 1, 1	14	
ЖА2	1, 1, 1, 2, 3, 4, 4, 6, 6, 7	22	
ЖА3	1, 7, 3, 6, 6, 1, 2, 4, 4, 1	20	

Приклад 2

Нехай маємо N=10 прямокутників, з довжинами A={8, 4, 3, 2, 5, 7, 6, 1, 1, 2} та стрічку з K=4 рівнями. На кожному рівні маємо M=1 дірку. Результати роботи алгоритмів наведемо у таблиці 2.3.

Таблиця 2.3 – Отримані результати

Алгоритм	Порядок прямокутників	ЦФ	Розміщення
ЖА1	8, 7, 6, 5, 4, 3, 2, 2, 1, 1	12	
ЖА2	1, 1, 2, 2, 3, 4, 5, 6, 7, 8	17	
ЖА3	8, 4, 6, 2, 7, 5, 3, 1, 1, 2	11	

Отже, для розглянутої задачі найкраще розміщення було отримане алгоритмом ЖА3.

2.4 Алгоритм локального пошуку

Для NP-складних задач локальний пошук дозволяє знаходити наближені розв’язки і часто є єдиним можливим способом розв’язання задач, де точні методи безсилі через надто великий обсяг простору можливих розв'язків. Використання локального пошуку в комбінаторній оптимізації відбулося у 1956 році для розв'язування задачі комівояжера [62], а в наступні роки сфера використання локального пошуку була розширена та концепція алгоритму була застосована в різноманітних задачах. Для багатьох задач теорії розкладів, розміщення, покриття та інших задач алгоритми локального пошуку дозволяють отримувати кращі результати, в порівнянні з іншими алгоритмами, особливо за умови обмеження у часі. Розглянемо схему алгоритму [63].

КРОК 1. Генерація початкового припустимого розв'язку x, який обираємо як поточний варіант.

КРОК 2. Чергова ітерація.

КРОК 3. Формуємо окіл $L(x)$ поточного варіанта й точно чи наближено знаходимо елемент $y \in L(x)$, який є субоптимальним розв'язком у цьому околі. Якщо $y \neq x$, то знайдений елемент оголошується черговим поточним варіантом (здійснюється переприсвоєння $x \leftarrow y$) і починається чергова ітерація, інакше - повернення на п. 3.

КРОК 4. Завершення роботи алгоритму: x - локальний розв'язок, якщо на останній ітерації здійснюється вичерпний пошук в околі.

Принциповими моментами реалізації конкретних алгоритмів локального пошуку є:

1. визначення околів $L(x)$;
2. генерація чергової точки околу $y \in L\,(x)$;
3. критерій завершення перегляду точок у поточному околі та переходу до наступного;
4. спосіб обчислення величини зміни цільової функції при переході до

нового поточного варіанта;

5. критерій завершення;
6. формування початкового наближення.

Ефективність алгоритмів локального пошуку істотно залежить від вибору відповідного типу околу $L(x)$. Чим більше окіл, тим імовірніше отримати кращий результат, але розширення околів швидко стає непрактичним.

У п.1 використовуються метричні околи чи околи, які утворені алгоритмічно. Пошук в околі може бути здійснений шляхом повного перебору сусідніх точок, або здійснити перехід до першого знайденого варіанта, який поліпшує цільову функції.

Для організації перебору точок околу при переході до нової ітерації використовують лінійний генератор або кільцевий генератор.

Критерієм завершенням локального пошуку може бути одна із умов: вичерпання ліміту часу або заданої кількості ітерацій, відсутність поліпшуючої точки в поточному околі.

Наведемо псевдокод алгоритму задачі розміщення прямокутників:

```
1   procedure local_search
2   x^0 := деякий припустимий розв'язок;
3   h:=0;
4   while окіл L( x^h ) поточного варіанта x^h не переглянутий повністю або
    не вичерпаний часовий ліміт do
5   y:= генерація чергової точки околу L( x^h );
6   Δ = f (x^h ) - f ( y);
7   if Δ > 0 then
8   h:=h+1;
9   x^h := y ;
10  end if;
11  end while;
12  return x := x^h ;
```

13 **end**

Примітки:

У рядку 2 припустимий розв’язок – початкова перестановка.

У рядку 4 окіл поточного варіанта визначається з множини перестановок, отриманих шляхом транспозиції елементів поточної перестановки.

Перегляд точок поточного околу завершується і здійснюється перехід до наступної ітерації, коли знаходиться перший же розв'язок, який покращує значення цільової функції, що і відображено у рядках 7-10.

Значенням цільової функції для даного розв'язку (перестановки) є правий край найвіддаленішого від початку прямокутника у розміщенні, яке побудоване зазначеною процедурою (placement).

Критерієм завершення алгоритму вважаємо такі умови:

– відсутність поліпшуючої точки в околі поточного розв’язку;

– вичерпання заданого ліміту часу.

Генерування точок околу поточної розв'язку здійснюється шляхом транспозицій його компонентів. При переході до нової ітерації процес генерування продовжується, починаючи з тієї транспозиції, на якій завершився процес у попередньому околі, а при досягненні останньої пари компонент транспозиції починаються із компонент 1 і 2, якщо перегляд точок в околі не завершено.

Нехай на ітерації h маємо перестановку $x^h = (x_1, ..., x_n)$, а $y \in L(x^h)$ утворюється із x^h шляхом транспозиції компонент i та j, $i \neq j$. Тоді покладаємо $x^{h+1} = y$ і генерування точок із околу $L(x^{h+1})$ починаємо із транспозиції i та j+1, якщо j<n, або ж присвоюємо i=1, j=2 в іншому разі. При цьому слід підраховувати кількість згенерованих точок околу, щоб вона не перевищила C_n^2 – кількості всіх точок околу при такому його означенні, що і використовується в одній із умов завершення роботи алгоритму.

Псевдокод алгоритму побудови розміщення прямокутників (placement), що

реалізує принцип "розміщувати вліво-вниз".

1 **procedure** placement

2 x := перестановка, для якої необхідно побудувати розміщення;

3 i :=1;

4 **while** $i < n$ **do** {доки всі прямокутники у перестановці x не розглянуто}

5 обрати прямокутник x_i з перестановки x;

6 s_j :=відстань до правого краю прямокутника x_i на рівні j, якщо його розмістити найближче до останнього прямокутника чи дірки, дотримуючись умови неперетинання;

7 l := найменший номер рівня, де s_l мінімальне;

8 на рівень l помістити прямокутник x_i;

9 i :=i +1;

10 **end while**;

11 **return** розміщення прямокутників;

12 **end**

Примітки

У рядках 6 та 8 прямокутник розміщається впритул до лівого краю останнього прямокутника на обраному рівні (чи дірки, якщо лівіше неї розмістити прямокутник не можна).

2.5 Алгоритм табуйованого пошуку

Головна ідея табуйованого пошуку полягає в зміні поточного варіанта розв’язку задачі та запам’ятовуванні послідовності таких змін та їх використанням. Всі зроблені модифікації розв'язків фіксуються в списку заборон.

Пошук починається з початкового варіанта розв’язку і поступово поліпшується локальними модифікації. Розв’язок, який був відвіданий нещодавно, включається в список заборон і далі не розглядається як кандидат на відвідування.

Для поданої задачі розглянемо метод строго табуювання, який запам’ятовує всі розв’язки з метою заборони потрапляння в уже відвідані точки простору розв’язків. Наведемо псевдокод алгоритму.

1 **procedure** tabu_search
2 x^0 := деякий припустимий розв’язок;
3 T:= ∅;
4 h:=0;
5 **while** окіл $L(x^h)$ поточного варіанта x^h не переглянутий повністю або не вичерпаний часовий ліміт **do**
6 y:= генерація чергової точки околу $L(x^h)$, що не належить списку заборон T;
7 $\Delta := f(x^h) - f(y)$;
8 **if** $\Delta > 0$ **then**
9 $h:=h+1$;
10 $x^h := y$;
11 $T:=T \cup x^h$;
12 **end if**;
13 **end while**;
14 **return** $x = x^h$;
15 **end**

Примітки:

У рядку 2, як і в алгоритмі локального пошуку, припустимий розв’язок – початкова перестановка.

У рядку 3 вказується множина розв’язків, що потрапили у список заборон і не розглядатимуться повторно.

У рядку 6 відбувається генерація нової точки околу, що відсутня у списку заборон.

Якщо поточна точка околу покращує значення ЦФ, то вона зберігається у

списку заборон у рядку 11.

Процедури обчислення значення ЦФ, генерування нової точки околу та критерій завершення роботи алгоритму використовуються ті самі, що наведені для алгоритму локального пошуку.

2.6 Алгоритм обміну

Під час побудови розміщень прямокутників, на стрічці можуть з'явитися рівні найбільшої та найменшої довжин. Для покращення значення цільової функції доцільно зробити повний обмін прямокутниками на цих рівнях.

Алгоритм обміну надає можливість повністю змінити положення прямокутників на двох рівнях з метою покращення розв'язку.

На кожній ітерації обираються два будь-які рівні, виконуючи обмін прямокутників необхідно проаналізувати чи отримали кращий розв'язок, або варіант розміщення без погіршення ЦФ. Якщо це виконалось, то запам'ятовуємо обраний розв'язок як кращий і переходимо до наступної ітерації. Якщо ні – обираємо інші рівні та повторюємо алгоритм знову [65].

Наведемо псевдокод алгоритму обміну (АПО).

```
1  procedure full_ex
2  побудувати початкове розміщення прямокутників і визначити як
   початковий розв'язок;
3  while розв'язок не оптимальний локально, або не вичерпані часові
   ресурси do
4   обрати два рівні випадковим чином;
5   зробити на обраних рівнях обмін прямокутниками;
6   if поточний результат кращий за розв'язок then прийняти його за
   розв'язок
7  end if;
```

8 **end while;**

9 **return** отриманий розв’язок;

10 **end**

Примітки

У рядку 2 початкове розміщення прямокутників формується випадковим чином, враховуючи обмеження. Часовими ресурсами визначаємо час роботи алгоритму на моменті зациклення або кількість ітерацій, що не надають покращення розв’язку. Рівні обираються (рядок 4) випадковим чином. Обмін прямокутниками (рядок 5) виконується, використовуючи всі прямокутники з двох рівнів, формуючи нове розміщення на обраних рівнях за принципом «зліва-знизу» [65].

2.7 Алгоритм обміну зі списком заборон

Оскільки алгоритм обміну може привести до зациклень або повторного розгляду рівнів, доцільно розглянути його модифікацію на основі формування списку заборон (табу).

Основою алгоритму обміну зі списком заборон (АПОТС) є алгоритм обміну, але він покращений за рахунок формування списку заборон. Списком заборон – є розв’язки, що були отримані на попередніх кроках табу-пошуку. Якщо отриманий розв’язок наявний в списку заборон, то надалі він не розглядається. За рахунок цього списку зменшиться кількість зациклень.

Наведемо псевдокод алгоритму АПОТС.

1 **procedure** full_taboo_ex

2 побудувати початкове розміщення прямокутників і визначити як розв’язок;

3 **while** розв’язок не оптимальний локально, або не вичерпані часові ресурси **do**

4 обрати два будь-які рівні;

5 зробити на обраних рівнях обмін прямокутниками;

6 перевірити чи не має розв’язку в списку заборон;

7 **if** поточний результат кращий за розв’язок **then** прийняти його за розв’язок, та внести попередній до списку заборон;

8 **else** надалі не розглядати ці рівні, якщо не було виконано обмінів прямокутниками;

9 **end if;**

10 **end while;**

11 **return** отриманий розв’язок;

12 **end**

Примітки

У рядку 2 початкове розміщення прямокутників формується випадковим чином, враховуючи обмеження. Відповідно до алгоритму АПО , часовими ресурсами визначаємо час роботи алгоритму на моменті зациклення або кількість ітерацій, що не надають покращення розв’язку. Рівні обираються (рядок 4) випадковим чином. Обмін прямокутниками (рядок 5) виконується, використовуючи всі прямокутники з двох рівнів, формуючи нове розміщення на обраних рівнях за принципом «зліва-знизу». В рядку 6 список заборон порожній, поки не буде знайдений локально кращий розв’язок. В рядку 7 список заборон формується на основі попередніх, гірших розв’язків, які вдалося перевершити поточним розв’язком.

Висновки до розділу

В другому розділі наведено дві змістовні постановки задачі розміщення прямокутників та їх математичну модель.

Для розв’язування задачі запропоновано жадібні алгоритми з модифікаціями.

Для алгоритмів ЖА1, ЖА2 та ЖА3 наведено приклади побудови розміщення прямокутників та розглянуто переваги кожного з них.

Для дослідження розв’язків в околі розроблено алгоритми локального та табуйованого пошуку, а також визначено переваги кожного з них.

На основі розглянутих алгоритмів запропоновано алгоритм обміну та алгоритм обміну з заборонами, де відбувається повний обмін прямокутниками на двох рівнях.

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

В попередньому розділі були розглянуті алгоритми розв'язування задачі розміщення прямокутників. Для порівняння ефективності алгоритмів проведемо експериментальні дослідження. Результати експериментів наведемо у таблицях та графіках. Експерименти будуть проведені на тестових задачах з різними параметрами: кількістю рівнями на стрічці, кількістю прямокутників та дірок.

Метою досліджень є оцінка ефективності алгоритмів, їх порівняння, виявлення їх недоліків і переваг.

Експерименти виконувались на персональному комп’ютері з наступними характеристиками: процесор Intel Core i7-8565U, 1.8 GHz з оперативною пам`яттю 8 GB та операційною системою Microsoft Windows 10 Home (версія 10.0.18363).

Для цього сформуємо задачі розміщення прямокутників, кількість яких вибиралася із діапазону 10-100, на напівнескінченних стрічках з кількістю рівнів від 5 до 20. Для кожного типу задачі згенеруємо 100 індивідуальних задач. Згенеровані довжини прямокутників не перевищують 15, кількість дірок на кожному рівні не перевищує 3, а довжина кожної дірки є 1. Для генерації довжин прямокутників та положення дірок використовувався датчик випадкових чисел з рівномірним розподілом. Для роботи алгоритмів використовуються однакові початкові наближення.

Результатами експериментів є час роботи алгоритмів в секундах, відсоток середнього покращення значення цільової функції. На початку та після завершення роботи кожного алгоритму для кожної задачі визначається найдовша відстань від початку стрічки на кожному з рівнів до правого краю прямокутників (довжина найдовшого рівня). Середнє покращення вказує на те, на скільки відсотків зменшується довжина найдовшого рівня.

3.1 Порівняння ЖА1, ЖА2, ЖА3 для задач з різними параметрами

Для дослідження ефективності жадібного алгоритму (ЖА1) та його модифікацій (ЖА2 та ЖА3) проведемо експериментальні дослідження та

представимо результати у таблицях 3.1-3.3. Результатами експериментів є час роботи алгоритму, рекордне (найкраще знайдене) значення цільової функції, дисперсія, середньо-квадратичне відхилення та довірчі інтервали. Вхідними параметрами задачі є кількість рівнів та прямокутників.

Для задач, де кількість рівнів та прямокутників однакова експерименти не були проведені.

Таблиця 3.1 – Результати роботи ЖА1

Рівні	Пр.	Час(мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	
5	10	0,022	16	8,714	2,952	17,048	22,952
5	15	0,046	16	16,237	4,03	16,97	25,03
5	20	0,074	24	21,526	4,64	23,36	32,64
5	25	0,116	30	16,647	4,08	31,92	40,08
5	30	0,182	37	20,646	4,544	43,456	52,544
5	35	0,238	44	19,055	4,365	48,635	57,365
5	40	0,301	49	25,584	5,058	56,942	67,058
5	45	0,385	59	27,259	5,221	62,779	73,221
5	50	0,423	62	30,852	5,554	69,446	80,554
5	55	0,547	71	27,412	5,236	75,764	86,236
5	60	0,682	78	32,734	5,721	85,279	96,721
5	65	0,704	78	48,599	6,971	90,029	103,971
5	70	0,829	88	55,412	7,444	97,556	112,444
5	75	0,93	93	45,511	6,746	103,25	116,746
5	80	1,059	105	46,347	6,808	112,19	125,808
5	85	1,191	104	71,849	8,476	117,52	134,476
5	90	1,526	115	53,707	7,329	125,67	140,329
5	95	1,526	124	53,029	7,282	133,72	148,282
5	100	1,631	128	70,428	8,392	140,61	157,392
10	10	0	0	0	0	0	0
10	15	0,076	12	0,1	0,316	12,684	13,316
10	20	0,076	13	3,601	1,898	14,102	17,898
10	25	0,159	13	9,63	3,103	15,897	22,103
10	30	0,207	19	4,515	2,125	20,875	25,125
10	35	0,249	23	3,695	1,922	24,078	27,922
10	40	0,333	25	6,839	2,615	27,385	32,615
10	45	0,387	29	6,679	2,584	30,416	35,584
10	50	0,45	30	10,204	3,194	33,806	40,194
10	55	0,626	34	9,356	3,059	37,941	44,059
10	60	0,684	37	10,165	3,188	41,812	48,188
10	65	0,756	41	12,142	3,485	45,515	52,485
10	70	0,866	45	9,56	3,092	47,908	54,092
10	75	1,042	49	14,555	3,815	53,185	60,815
10	80	1,181	50	18,277	4,275	54,725	63,275

Продовження таблиці 3.1

Рівні	Пр.	Час(мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	
10	85	1,288	55	12,111	3,48	59,52	66,48
10	90	1,408	56	18,092	4,253	62,747	71,253
10	95	1,564	63	14,947	3,866	67,134	74,866
10	100	1,749	66	16,066	4,008	68,992	77,008
15	10	0	0	0	0	0	0
15	15	0	0	0	0	0	0
15	20	0,108	12	0,095	0,308	12,692	13,308
15	25	0,116	12	1,655	1,286	11,714	14,286
15	30	0,201	13	3,308	1,819	13,181	16,819
15	35	0,239	14	3,036	1,743	15,257	18,743
15	40	0,356	16	3,571	1,89	18,11	21,89
15	45	0,397	18	4,563	2,136	20,864	25,136
15	50	0,508	22	3,326	1,824	23,176	26,824
15	55	0,623	23	5,834	2,415	25,585	30,415
15	60	0,651	25	6,696	2,588	27,412	32,588
15	65	0,745	28	4,788	2,188	29,812	34,188
15	70	0,876	30	4,704	2,169	32,831	37,169
15	75	1,067	32	5,586	2,363	35,637	40,363
15	80	1,119	34	6,27	2,504	36,496	41,504
15	85	1,285	37	5,08	2,254	39,746	44,254
15	90	1,427	37	7,735	2,781	41,219	46,781
15	95	1,598	38	7,537	2,745	44,255	49,745
15	100	1,677	43	6,875	2,622	46,378	51,622
20	10	0	0	0	0	0	0
20	15	0	0	0	0	0	0
20	20	0	0	0	0	0	0
20	25	0,136	12	0,22	0,469	12,531	13,469
20	30	0,207	12	0,329	0,574	12,426	13,574
20	35	0,236	12	1,038	1,019	11,981	14,019
20	40	0,378	13	2,362	1,537	13,463	16,537
20	45	0,46	14	2,019	1,421	15,579	18,421
20	50	0,503	17	2,679	1,637	17,363	20,637
20	55	0,63	17	3,681	1,919	20,081	23,919
20	60	0,696	17	5,277	2,297	20,703	25,297
20	65	0,789	20	3,441	1,855	23,145	26,855
20	70	0,935	23	3,038	1,743	25,257	28,743
20	75	1,064	24	4,125	2,031	25,969	30,031
20	80	1,183	26	2,55	1,597	28,403	31,597
20	85	1,348	27	4,539	2,13	29,87	34,13
20	90	1,685	29	3,906	1,976	31,024	34,976
20	95	1,792	29	3,944	1,986	33,014	36,986
20	100	1,862	32	3,892	1,973	35,027	38,973

Тут і далі СКВ – це середньоквадратичне відхилення.

На основі отриманих результатів для алгоритму ЖА1 побудуємо графік залежності часу роботи алгоритму від параметрів задачі та представимо на рисунку 3.1.

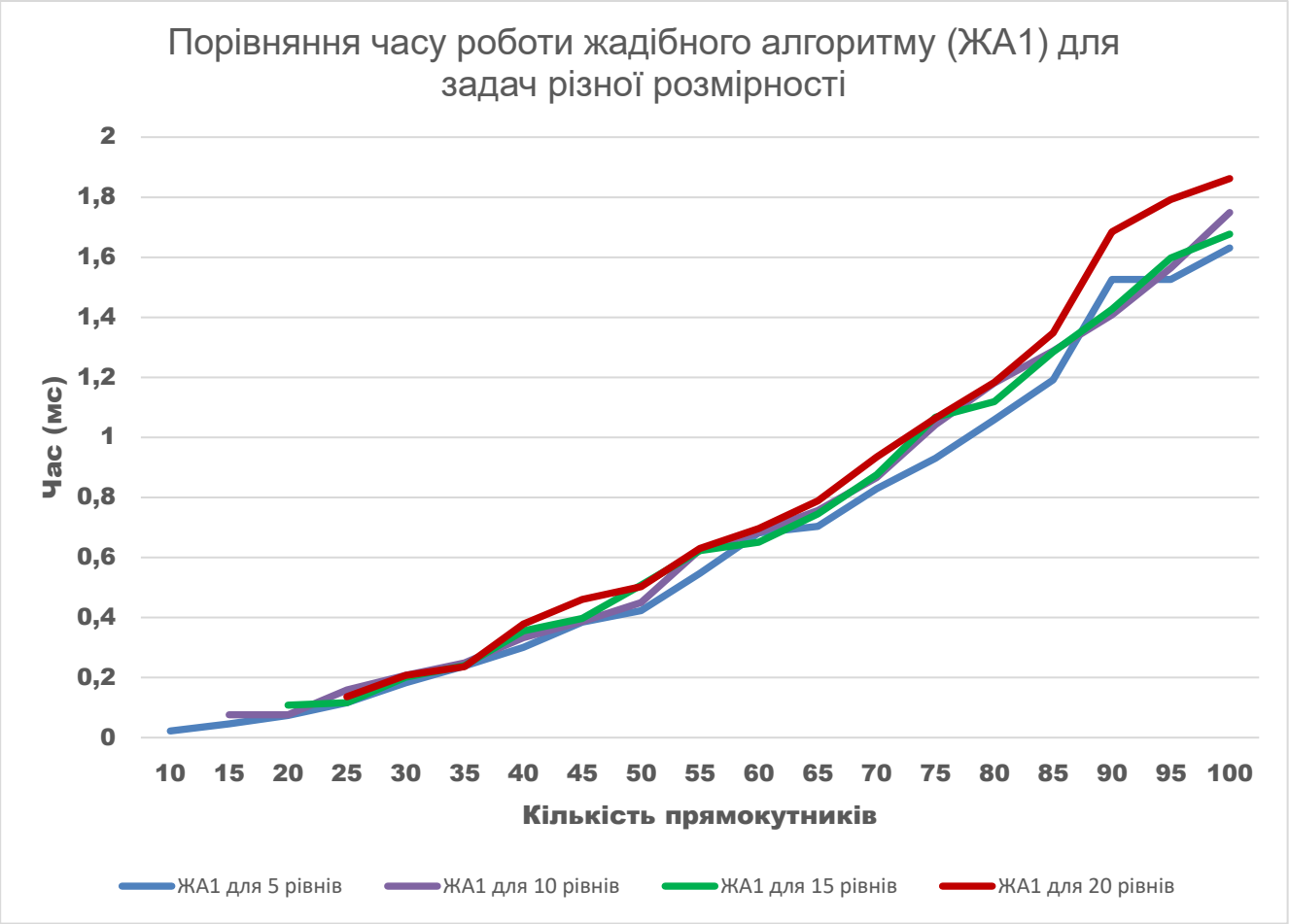


Рисунок 3.1 – Графік порівняння часу роботи алгоритму ЖА1 для задач різної розмірності

Час роботи жадібного алгоритму ЖА1 залежить від початкових параметрів задачі: від кількості рівнів стрічки. Тому для задач з 20 рівнями та 100 прямокутниками час роботи алгоритму може бути більшим, ніж для задачі з 15 та менше рівнями.

Результати дослідження ефективності жадібного алгоритму з модифікацією ЖА2 наведені у таблиці 3.2. Результатами експерименту є час роботи алгоритму, рекордне значення цільової функції, дисперсія, середньо-квадратичне відхилення та довірчі інтервали.

Таблиця 3.2 – Результати роботи ЖА2

Рівні	Пр.	Час (мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	
5	10	0,014	19	6,477	2,545	20,455	25,545
5	15	0,042	24	7,758	2,785	22,215	27,785
5	20	0,038	30	15,404	3,925	28,075	35,925
5	25	0,1	39	10,269	3,204	39,796	46,204
5	30	0,144	41	22,064	4,697	47,303	56,697
5	35	0,109	50	18,219	4,268	53,732	62,268
5	40	0,16	54	26,29	5,127	61,873	72,127
5	45	0,203	63	22,498	4,743	68,257	77,743
5	50	0,248	66	32,055	5,662	76,338	87,662
5	55	0,289	76	32,954	5,741	80,259	91,741
5	60	0,428	84	33,991	5,83	90,17	101,83
5	65	0,415	84	45,684	6,759	95,241	108,759
5	70	0,455	93	52,319	7,233	102,77	117,233
5	75	0,471	99	48,762	6,983	110,02	123,983
5	80	0,621	111	47,155	6,867	118,13	131,867
5	85	0,677	110	72,305	8,503	122,5	139,503
5	90	0,857	124	54,853	7,406	131,59	146,406
5	95	0,834	129	54,343	7,372	139,63	154,372
5	100	0,921	132	69,854	8,358	145,64	162,358
10	10	0	0	0	0	0	0
10	15	0,058	15	2,312	1,52	17,48	20,52
10	20	0,052	17	4,328	2,081	18,919	23,081
10	25	0,104	21	6,423	2,534	21,466	26,534
10	30	0,133	25	3,702	1,924	26,076	29,924
10	35	0,169	28	4,059	2,015	29,985	34,015
10	40	0,212	31	4,805	2,192	33,808	38,192
10	45	0,25	36	5,141	2,267	36,733	41,267
10	50	0,294	37	9,774	3,126	39,874	46,126
10	55	0,345	40	9,437	3,072	43,928	50,072
10	60	0,432	43	9,8	3,131	47,869	54,131
10	65	0,45	46	11,922	3,453	51,547	58,453
10	70	0,463	51	9,585	3,096	53,904	60,096
10	75	0,57	55	13,141	3,625	58,375	65,625
10	80	0,751	56	18,322	4,28	60,72	69,28
10	85	0,78	61	12,956	3,599	65,401	72,599
10	90	0,851	63	17,309	4,16	67,84	76,16
10	95	0,923	66	15,184	3,897	73,103	80,897
10	100	1,021	71	15,209	3,9	76,1	83,9
15	10	0	0	0	0	0	0
15	15	0	0	0	0	0	0
15	20	0,094	15	1,879	1,371	14,629	17,371
15	25	0,118	16	2,282	1,511	16,489	19,511

Продовження таблиці 3.2.

Рівні	Пр.	Час (мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	Рівні
15	40	0,246	22	2,866	1,693	24,307	27,693
15	45	0,23	23	3,312	1,82	26,18	29,82
15	50	0,361	27	3,045	1,745	29,255	32,745
15	55	0,385	29	4,873	2,207	30,793	35,207
15	60	0,477	31	6,49	2,548	32,452	37,548
15	65	0,496	34	4,193	2,048	35,952	40,048
15	70	0,589	36	5,02	2,24	38,76	43,24
15	75	0,633	37	5,041	2,245	41,755	46,245
15	80	0,706	40	6,802	2,608	43,392	48,608
15	85	0,773	43	5,685	2,384	45,616	50,384
15	90	0,836	44	7,21	2,685	47,315	52,685
15	95	0,967	45	6,954	2,637	50,363	55,637
15	100	1,032	49	6,917	2,63	52,37	57,63
20	10	0	0	0	0	0	0
20	15	0	0	0	0	0	0
20	20	0	0	0	0	0	0
20	25	0,143	14	1,743	1,32	14,68	17,32
20	30	0,146	16	1,736	1,317	16,683	19,317
20	35	0,243	17	1,023	1,011	17,989	20,011
20	40	0,239	18	1,181	1,087	18,913	21,087
20	45	0,328	21	1,871	1,368	22,632	25,368
20	50	0,369	22	2,278	1,509	23,491	26,509
20	55	0,458	24	2,356	1,535	25,465	28,535
20	60	0,493	23	2,796	1,672	26,328	29,672
20	65	0,531	27	3,021	1,738	29,262	32,738
20	70	0,599	29	2,707	1,645	31,355	34,645
20	75	0,699	30	3,13	1,769	32,231	35,769
20	80	0,772	31	2,752	1,659	34,341	37,659
20	85	0,873	34	4,149	2,037	35,963	40,037
20	90	1,003	35	3,599	1,897	38,103	41,897
20	95	1,112	35	3,456	1,859	39,141	42,859
20	100	1,179	38	4,535	2,13	40,87	45,13
15	30	0,122	18	1,437	1,199	19,801	22,199
15	35	0,175	21	2,806	1,675	22,325	25,675

Для отриманих результатів побудуємо графік залежності часу роботи алгоритму від параметрів задачі та представимо на рисунку 3.2.

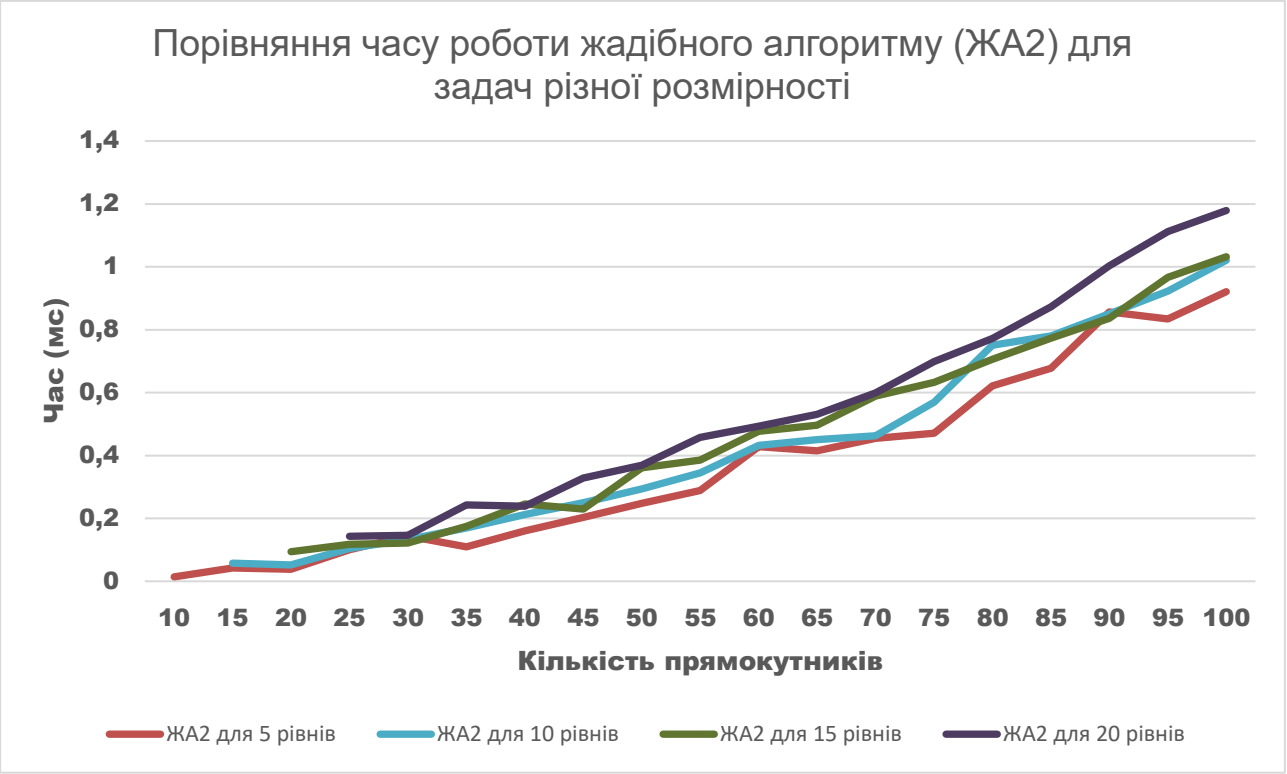


Рисунок 3.2 – Графік порівняння часу роботи алгоритму ЖА2 для задач різної розмірності

Відповідно до отриманих результатів можна зробити висновок, що параметри задачі впливають на час роботи алгоритмів, де кількість прямокутників перевищує 65.

У таблиці 3.3 наведені результати експерименту дослідження ефективності жадібного алгоритму з модифікацією ЖА3.

Таблиця 3.3 – Результати роботи ЖА2

Рівні	Пр.	Час(мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	
5	10	0,046	18	6,477	2,545	20,455	25,545
5	15	0,036	24	7,758	2,785	22,215	27,785
5	20	0,062	29	15,404	3,925	28,075	35,925
5	25	0,06	35	10,269	3,204	39,796	46,204
5	30	0,155	39	22,064	4,697	47,303	56,697
5	35	0,153	49	18,219	4,268	53,732	62,268
5	40	0,177	50	26,29	5,127	61,873	72,127
5	45	0,214	62	22,498	4,743	68,257	77,743
5	50	0,263	65	32,055	5,662	76,338	87,662
5	55	0,341	72	32,954	5,741	80,259	91,741

Продовження таблиці 3.3.

Рівні	Пр.	Час(мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	Рівні
5	60	0,38	81	33,991	5,83	90,17	101,83
5	65	0,389	82	45,684	6,759	95,241	108,759
5	70	0,439	90	52,319	7,233	102,77	117,233
5	75	0,516	97	48,762	6,983	110,02	123,983
5	80	0,584	108	47,155	6,867	118,13	131,867
5	85	0,637	109	72,305	8,503	122,5	139,503
5	90	0,843	117	54,853	7,406	131,59	146,406
5	95	0,832	127	54,343	7,372	139,63	154,372
5	100	0,909	131	69,854	8,358	145,64	162,358
10	10	0	0	0	0	0	0
10	15	0,036	15	2,312	1,52	17,48	20,52
10	20	0,09	13	4,328	2,081	18,919	23,081
10	25	0,076	17	6,423	2,534	21,466	26,534
10	30	0,154	22	3,702	1,924	26,076	29,924
10	35	0,158	27	4,059	2,015	29,985	34,015
10	40	0,205	29	4,805	2,192	33,808	38,192
10	45	0,24	33	5,141	2,267	36,733	41,267
10	50	0,317	35	9,774	3,126	39,874	46,126
10	55	0,331	38	9,437	3,072	43,928	50,072
10	60	0,379	42	9,8	3,131	47,869	54,131
10	65	0,481	43	11,922	3,453	51,547	58,453
10	70	0,59	50	9,585	3,096	53,904	60,096
10	75	0,632	51	13,141	3,625	58,375	65,625
10	80	0,689	52	18,322	4,28	60,72	69,28
10	85	0,737	55	12,956	3,599	65,401	72,599
10	90	0,82	60	17,309	4,16	67,84	76,16
10	95	0,874	66	15,184	3,897	73,103	80,897
10	100	0,981	70	15,209	3,9	76,1	83,9
15	10	0	0	0	0	0	0
15	15	0	0	0	0	0	0
15	20	0,09	14	1,879	1,371	14,629	17,371
15	25	0,115	13	2,282	1,511	16,489	19,511
15	30	0,164	15	1,437	1,199	19,801	22,199
15	35	0,177	16	2,806	1,675	22,325	25,675
15	40	0,263	21	2,866	1,693	24,307	27,693
15	45	0,293	22	3,312	1,82	26,18	29,82
15	50	0,281	25	3,045	1,745	29,255	32,745
15	55	0,404	29	4,873	2,207	30,793	35,207
15	60	0,446	27	6,49	2,548	32,452	37,548
15	65	0,485	31	4,193	2,048	35,952	40,048
15	70	0,54	35	5,02	2,24	38,76	43,24
15	75	0,65	34	5,041	2,245	41,755	46,245
15	80	0,715	40	6,802	2,608	43,392	48,608
15	85	0,751	41	5,685	2,384	45,616	50,384

Продовження таблиці 3.3.

Рівні	Пр.	Час(мс)	Рекорд	Дисперсія	СКВ	Довірчі інтервали	Рівні
15	90	0,817	42	7,21	2,685	47,315	52,685
15	95	0,95	45	6,954	2,637	50,363	55,637
15	100	0,989	48	6,917	2,63	52,37	57,63
20	10	0	0	0	0	0	0
20	15	0	0	0	0	0	0
20	20	0	0	0	0	0	0
20	25	0,124	14	1,743	1,32	14,68	17,32
20	30	0,194	16	1,736	1,317	16,683	19,317
20	35	0,203	16	1,023	1,011	17,989	20,011
20	40	0,233	15	1,181	1,087	18,913	21,087
20	45	0,263	20	1,871	1,368	22,632	25,368
20	50	0,331	19	2,278	1,509	23,491	26,509
20	55	0,414	21	2,356	1,535	25,465	28,535
20	60	0,49	22	2,796	1,672	26,328	29,672
20	65	0,541	21	3,021	1,738	29,262	32,738
20	70	0,618	26	2,707	1,645	31,355	34,645
20	75	0,689	27	3,13	1,769	32,231	35,769
20	80	0,752	30	2,752	1,659	34,341	37,659
20	85	0,843	29	4,149	2,037	35,963	40,037
20	90	1,016	32	3,599	1,897	38,103	41,897
20	95	1,112	33	3,456	1,859	39,141	42,859
20	100	1,158	37	4,535	2,13	40,87	45,13

На рисунку 3.3 подано залежність часу від параметрів задачі.

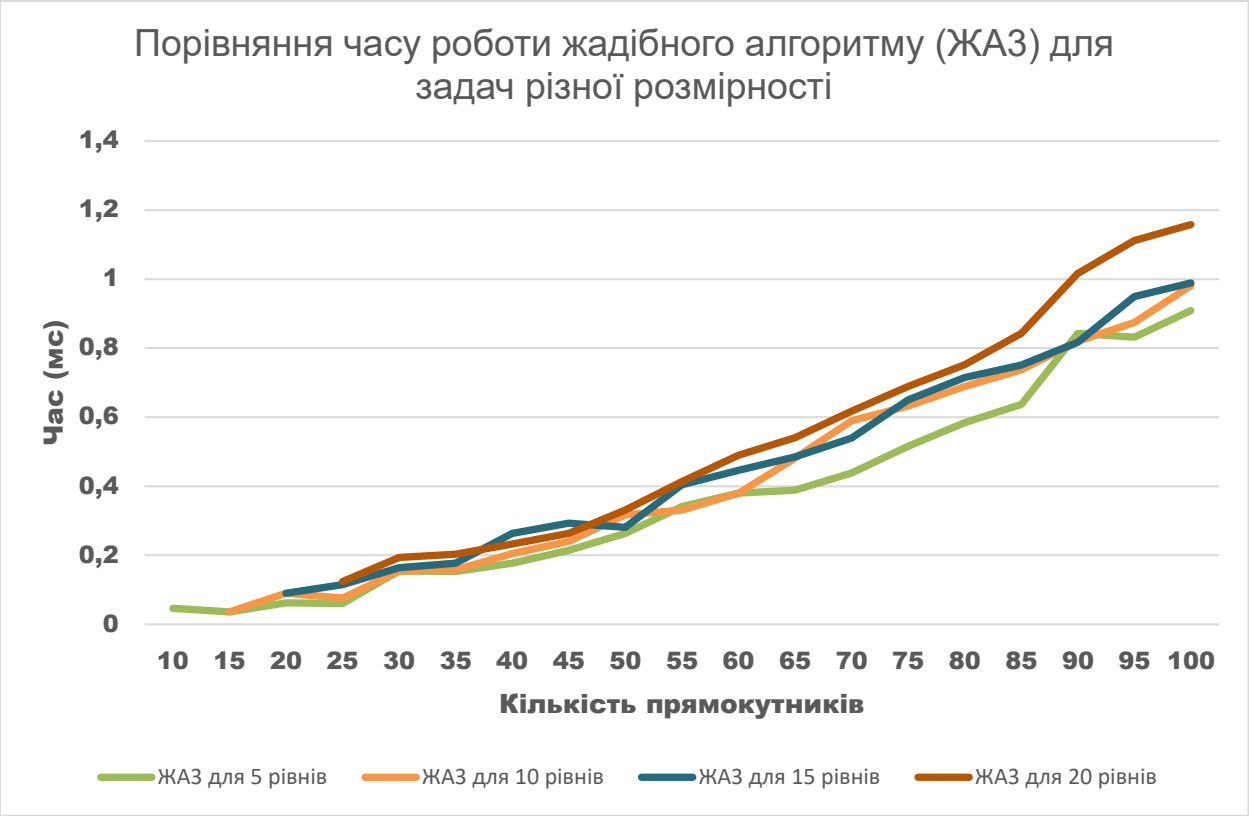


Рисунок 3.3 – Графік порівняння часу роботи алгоритму ЖА3 для задач різної розмірності

Для задач, дек кількість прямокутників перевищує 60 – час роботи алгоритму для різної кількості рівнів збільшується.

Серед розглянутих жадібних алгоритмів час роботи ЖА3 найменший, а найбільший у ЖА1. Це зумовлено тим, що для алгоритмів ЖА1 та ЖА2 необхідно більше часу, оскільки відбувається впорядковування прямокутників.

Порівняємо отримані рекордні значення цільової функції для задач з різними параметрами. Графік залежності рекордного значення ЦФ від параметрів задачі та алгоритмів наведено на рисунку 3.4.

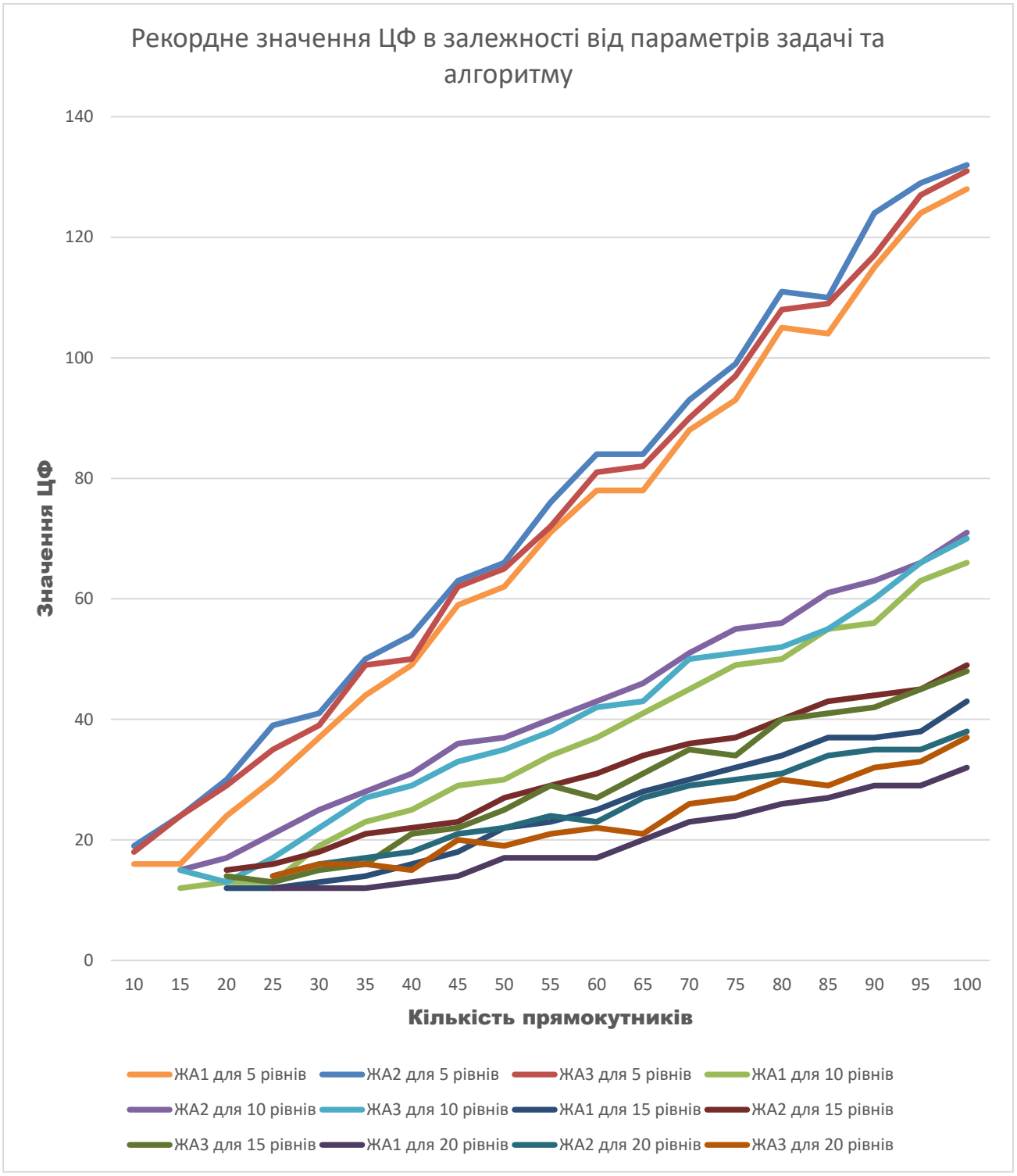


Рисунок 3.4 – Порівняння рекордних значень цільової функції для різних параметрів задач

Для всіх задач з різними вхідними параметрами, найкраще значення цільової функції було отримано жадібними алгоритмом без модифікацій (ЖА1). Рекордні значення, що були отримані алгоритмом ЖА3 в деяких випадках співпадали зі

значеннями, отриманими алгоритмом ЖА3 (наприклад, для задач, де кількість прямокутників 40, а рівнів 5; 5 рівнів та 55 прямокутників та інші). Алгоритм ЖА2 показав не найкращі результати, проте в подальшому його можна покращити за рахунок алгоритму обміну.

Подані результати демонструють, що ЖА1 працює повільніше, ніж ЖА2 і ЖА3. Це обумовлено тим, що виконується більше перевірок перед розміщенням чергового прямокутника. Час роботи алгоритмів ЖА2 і ЖА3 мають не значну різницю в часі, в порівнянні з алгоритмом ЖА1.

При розміщенні 300 і більше прямокутників алгоритмом ЖА2 кількість кращих розміщень більше, ніж при розміщенні ЖА3. Для отримання кращого розв’язку необхідно використовувати ЖА1, але при цьому збільшується час розв’язання.

Отже серед розглянутих алгоритмів алгоритм ЖА1 має найкраще значення цільової функції, проте час роботи алгоритму перевищує час роботи алгоритму ЖА3. Отже, для задач, де необхідно отримати найкраще значення цільової функції доцільно використовувати алгоритм ЖА1, якщо відсутні обмеження на використання часових ресурсів, в іншому випадку доцільно використовувати алгоритм ЖА3, оскільки алгоритм працює набагато швидше і результати алгоритму можуть бути отримані ті самі, що й алгоритмом ЖА1.

3.2 Дослідження алгоритмів локального та табуйованого пошуку

Проведемо експериментальне дослідження та порівняємо алгоритми локального та табу пошуку. Для проведення експериментів було встановлено 1000 ітерацій для завершення роботи алгоритмів.

На рисунках 1-2 подані результати експериментів для задач з 5, 10, 15 та 20 рівнями та кількістю прямокутників від 5 до 100. Першою частиною результатів є час роботи алгоритмів та середнє покращення (таблиця 3.4). Середнє покращення вказує на те, на скільки відсотків зменшується довжина найдовшого рівня у порівнянні із початковим варіантом розміщення.

Таблиця 3.4 – Результати експерименту, частина 1

Параметри задачі		Час (мс)		Середнє покращення (%)	
Рівні	Прямокутники	ЛП	ТП	ЛП	ТП
5	10	7,8	7	6,5	21,3
5	15	12,7	16,1	0,4	17,8
5	20	6,2	10	1,1	16,3
5	25	15,2	4,2	1	4,1
5	30	21	16,8	4,4	8,7
5	35	34,4	9,8	3	11
5	40	55,5	5	2,7	12,3
5	45	62,3	13,2	3,4	5,5
5	50	99,9	5,7	1,3	3,8
5	55	144,5	3,5	2,5	5
5	60	150	4,8	1,3	1,9
5	65	190,4	4,9	3,6	5,1
5	70	236,5	4,4	0,9	1,2
5	75	313,9	2,7	3,1	5
5	80	356,2	4,7	0,4	2,7
5	85	479,4	3,8	0,3	1
5	90	572,7	2,1	0,9	2,9
5	95	786	1,4	0,7	2,6
5	100	865,1	0,7	0,1	0,4
10	10	0	0	0	0
10	15	8,4	0,1	0,7	16,9
10	20	12,9	15,8	1	0,1
10	25	24,4	8,3	2,4	12,1
10	30	34,8	16,6	8,5	13
10	35	68	14,9	4,1	5,7
10	40	100,1	8,8	3	9,1
10	45	155,5	1,4	0,4	4
10	50	189,5	16,7	1,7	11,5
10	55	243,9	7,8	0,3	0,9
10	60	335,5	12,3	1,5	3,4
10	65	456,5	5,5	1,4	0,2
10	70	540,8	3	3,5	7,4
10	75	601,1	7	3,7	7,6
10	80	715,1	9,4	4,6	6,1
10	85	786,6	5,8	3	6,7
10	90	975,3	7,4	5,8	7,8
10	95	1091,5	3,9	3,5	9,8

Продовження таблиці 3.4.

Параметри задачі		Час (мс)		Середнє покращення (%)	
Рівні	Прямокутники	ЛП	ТП	ЛП	ТП
15	15	0	0	0	0
15	20	16,9	0,8	1	0,7
15	25	36,6	18,3	2,6	8,1
15	30	72	10,7	4	10,8
15	35	109,6	26,3	6,4	11,4
15	40	155,9	6	8,7	12,6
15	45	237,7	16	4,5	10,7
15	50	299,7	21	10,8	14,7
15	55	357,6	5,2	2,7	4
15	60	524,5	13,8	7,5	8,5
15	65	651,2	8,4	7,1	15,5
15	70	848,6	15,1	3,7	4,9
15	75	1007,1	7,6	4,9	6
15	80	1252,6	7,2	4,7	6,1
15	85	1521,1	6,7	5,6	6,3
15	90	1641,4	9,3	7,2	8,6
15	95	1968	2,8	7,9	8,5
15	100	2259,2	10,8	0,6	2
20	10	0	0	0	0
20	15	0	0	0	0
20	20	0	0	0	0
20	25	50,1	11,6	1,1	2,6
20	30	85,6	10,4	2	3,3
20	35	99,9	8,1	2,2	3,4
20	40	185,7	15,8	4,8	5,4
20	45	251,9	8,5	5,9	8
20	50	352,1	12,4	6,3	8,4
20	55	456,1	7	11,3	11,2
20	60	600,2	11,6	16,9	19,4
20	65	837,5	23,5	10,1	23,7
20	70	1058	10,4	10,7	17,1
20	75	1322,3	9,4	13,3	13,8
20	80	1641,8	12,6	3,4	4,8
20	85	1764	13,2	3,4	8,1
20	90	2162,8	8,5	13	13,5
20	95	2550,5	11,4	5,7	11,6
20	100	2816,9	14,3	13	13,1

Для ілюстрації результатів експерименту на рисунках 3.5-3.6 наведемо графіки залежності часу роботи алгоритму від параметрів задачі та алгоритмів.

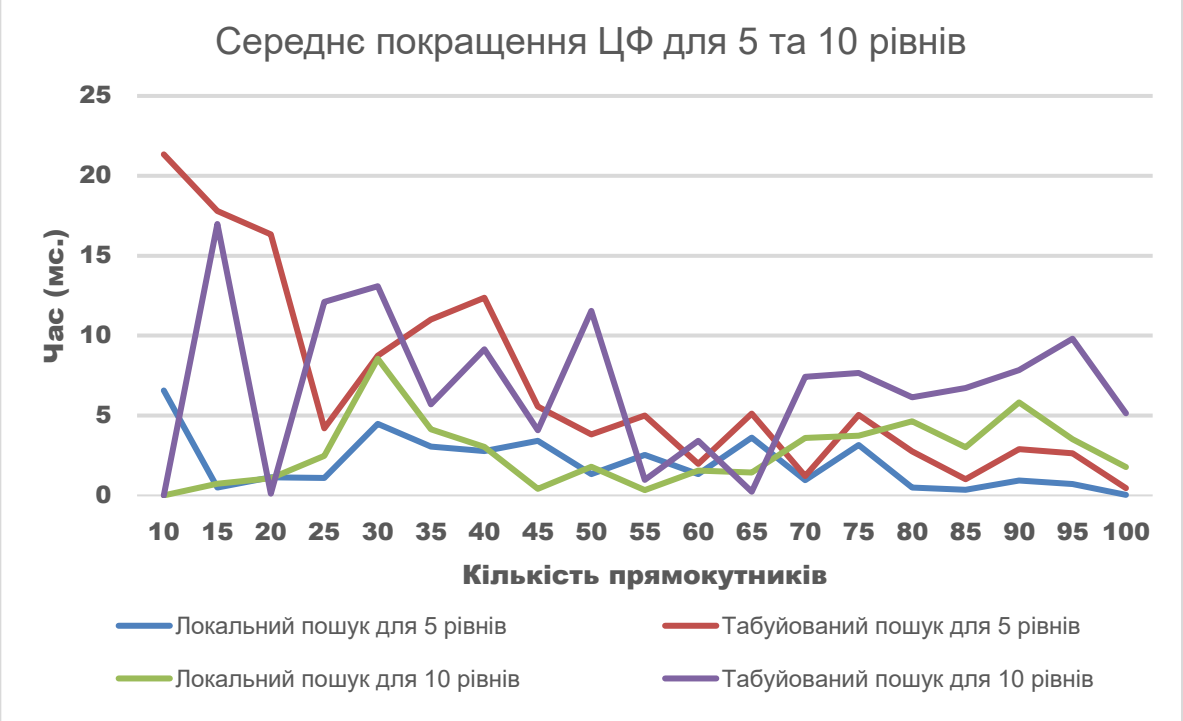


Рисунок 3.5 – Середнє покращення значення цільової функції для 5 та 10 рівнів

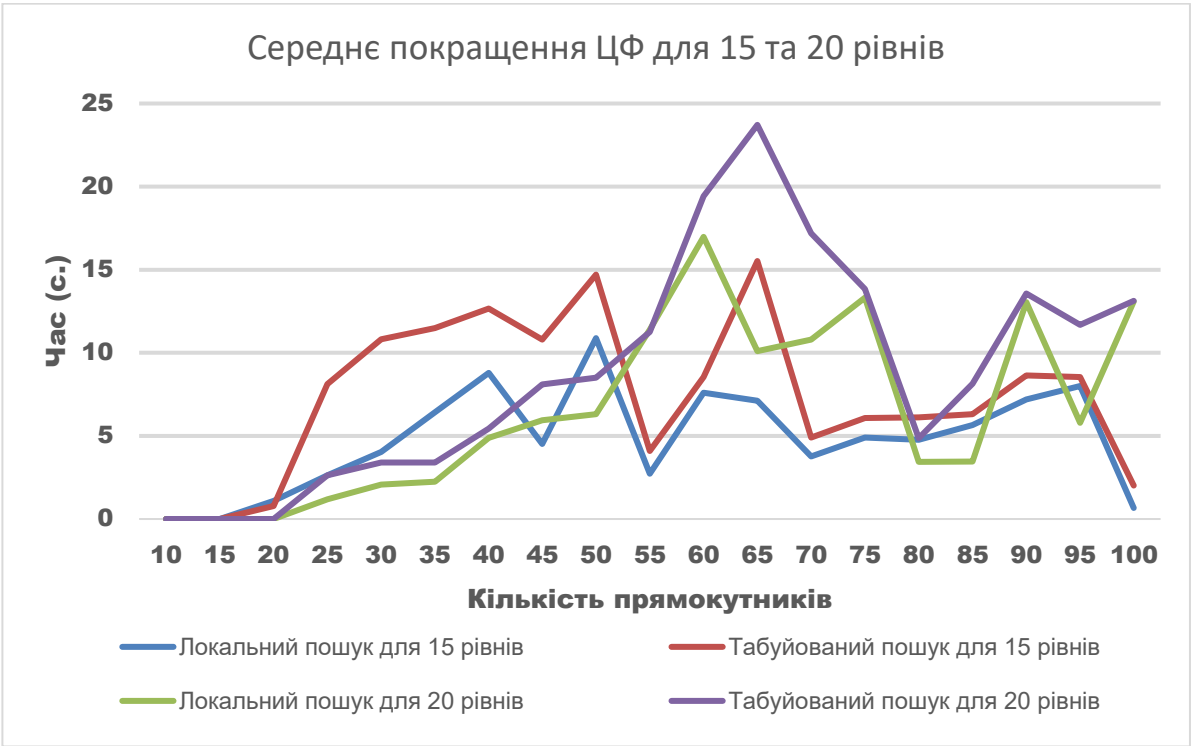


Рисунок 3.6 – Середнє покращення значення цільової функції для 15 та 20 рівнів

Отже, алгоритм локального пошуку програє алгоритму табуйованого пошуку у відсотку середнього покращення значення цільової функції для задач з кількістю прямокутників меншою за 60. Це зумовлено тим, що для задач невеликої

розмірності, алгоритм табуйованого пошуку не розглядає околи із табу списку, тим самим збільшуючи кількість різних околів, які будуть переглянуті.

Для задач з кількістю прямокутників більше 60 та кількістю рівнів 5 – відсоток середнього покращення значення цільової функції не перевищує 5% кожним алгоритмом, а для задач із кількістю рівні 10 – не перевищує 10%. Це зумовлено тим, що генерується велика кількість прямокутників з невеликою різницею довжин сторін, тому кількість околів, які можна переглянути – перевищує задані часові ліміти.

Найкращі результати у середньому покращені показав алгоритм табуйованого пошуку для задачі, яка складається з 20 рівнів та кількістю прямокутників 55-70, проте в інших випадках переваги у середньому покращенні значення цільової функції алгоритм табуйованого пошуку не очевидні.

Для задачі із 15 рівнями алгоритми табуйованого пошуку має незначне середнє покращення у порівнянні з алгоритмом локального пошуку для кількості прямокутників, що перевищує 50, окрім задачі для 65 прямокутників.

На рисунках 3.7 та 3.8 наведено графіки залежності часу роботи алгоритмів від параметрів задачі (кількості рівнів стрічки та прямокутників).

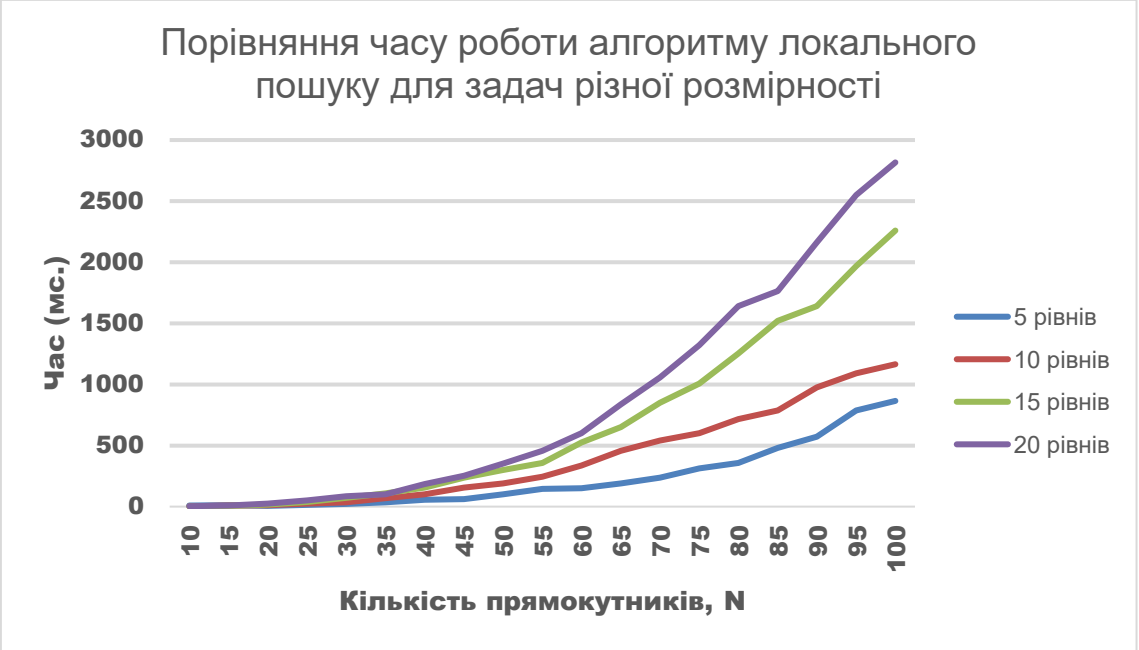


Рисунок 3.7 – Час роботи алгоритму локального пошуку

Рисунок 3.8 – Час роботи алгоритму табуйованого пошуку

Час роботи алгоритму локального пошуку значно вищий за час роботи алгоритму табуйованого пошуку для задач з кількістю прямокутників більше за 30. Це зумовлено тим, що алгоритм табуйованого пошуку повторно не розглядає околи і тому, отримуючи нові околи без покращення – зупиняє свою роботу, в той час, коли алгоритм локального пошуку продовжує розглядати інші околи, доки не закінчиться заданий ліміт ітерацій для околу.

З рисунку 3 видно, що для задач, де збільшується кількість прямокутників, час роботи алгоритму локального пошуку зростає. Це зумовлено тим, що збільшується кількість околів, які можна розглянути.

Із рисунку 4 випливає, що для час роботи алгоритму табуйованого пошуку буде найменшим для будь-якої кількості прямокутників у випадку, коли стрічка має 5 рівнів. Максимальний час роботи алгоритму табуйованого пошуку більший для задач, де стрічка має 15 та 20 рівнів, проте не перевищує час роботи алгоритму локального пошуку.

У таблиці 3.5 подані інші результати експерименту: рекордне значення цільової функції, середньоквадратичне відхилення (СКВ), дисперсія та довірчі інтервали.

Таблиця 3.5 – Результати експерименту, частина 2

Задача		Локальний пошук					Табуйований пошук				
Рівні	Прямокутники	Рекорд	СКВ	Дисперсія	Довірчі інтервали		Рекорд	СКВ	Дисперсія	Довірчі інтервали	
5	10	10	2.9	8.5	13.0	18.9	9	3.3	11.0	11.6	18.3
5	15	18	2.5	6.7	19.4	24.5	19	2.5	6.5	19.4	24.5
5	20	24	3.0	9.1	24.9	31.0	22	4.8	23.3	25.1	34.8
5	25	30	3.8	14.7	34.1	41.8	29	3.8	15.1	33.1	40.8
5	30	36	5.7	32.7	38.2	49.7	34	5.6	32.4	37.3	48.6
5	35	47	5.3	28.1	47.6	58.3	48	3.8	15.0	47.1	54.8
5	40	54	5.6	31.9	56.3	67.6	51	3.8	14.7	55.1	62.8
5	45	60	4.8	23.4	65.1	74.8	57	5.5	31.0	60.4	71.5
5	50	65	5.5	30.8	70.4	81.5	67	4.0	16.6	69.9	78.0
5	55	73	4.8	23.8	74.1	83.8	73	5.0	25.5	76.9	87.
5	60	78	7.9	63.0	84.0	99.9	80	6.2	39.3	86.7	99.2
5	65	86	8.5	72.3	86.4	103.5	77	7.6	58.9	88.3	10.6
5	70	94	6.3	40.8	101.6	114.3	82	9.1	83.4	94.8	113.1
5	75	102	6.6	44.2	104.3	117.6	95	8.1	66.5	105.8	122.1
5	80	112	5.7	32.7	113.2	124.7	102	7.1	51.4	108.8	123.1
5	85	120	5.0	25.3	121.9	132.0	113	8.3	70.0	113.6	130.3
5	90	116	7.5	56.4	125.4	140.5	117	6.8	47.5	127.1	140.8
5	95	125	9.1	83.2	126.8	145.1	130	7.1	50.9	133.8	148.1
5	100	136	10.9	120.8	139.0	160.9	132	9.4	88.5	136.5	155.4
10	10	0	0	0	0	0	0	0	0	0	0
10	15	12	1.1	1.4	11.8	14.1	10	1.8	3.3	11.1	14.8
10	20	13	2.3	5.7	14.6	19.3	12	2.9	8.8	13.0	18.9
10	25	18	2.4	5.9	18.5	23.4	17	1.8	3.4	18.1	21.8
10	30	20	2.1	4.6	21.8	26.1	20	2.5	6.4	19.4	24.5
10	35	24	2.9	8.7	25.0	30.9	23	2.4	6	24.5	29.4
10	40	25	2.9	8.6	27.0	32.9	28	2.6	7.2	28.3	33.6
10	45	27	2.8	8.1	31.1	36.8	31	2.4	5.8	32.5	37.4
10	50	32	2.9	8.5	36.0	41.9	32	3.3	11.5	35.6	42.3
10	55	39	2.5	6.2	39.4	44.5	33	3.7	14.0	39.2	46.7
10	60	44	1.4	2	45.5	48.4	39	3.0	9.2	43.9	50.0
10	65	41	3.5	12.2	46.4	53.5	41	2.7	7.3	45.2	50.7
10	70	50	3.0	9.5	51.9	58.0	44	3.4	12.0	49.5	56.4
10	75	54	3.2	10.8	53.7	60.2	51	3.6	13.4	53.3	60.6
10	80	53	4.7	22.2	53.2	62.7	54	4.6	21.4	54.3	63.6
10	85	61	2.6	7.0	62.3	67.6	57	3.3	10.9	57.6	64.3
10	90	62	3.3	10.9	63.6	70.3	63	3.8	14.9	65.1	72.8
10	95	63	4.4	20	67.5	76.4	65	3.9	15.6	68.0	75.9
10	100	72	2.7	7.5	71.2	76.7	69	3.2	10.5	72.7	79.2
15	10	0	0	0	0	0	0	0	0	0	0

Продовження таблиці 3.5.

Задача		Локальний пошук					Табуйований пошук				
Рівні	Прямокутники	Рекорд	СКВ	Дисперсія	Довірчі інтервали		Рекорд	СКВ	Дисперсія	Довірчі інтервали	
15	20	12	0.2	0.1	12.7	13.2	11	0.7	0.5	12.2	13.7
15	25	13	1.9	3.9	13.0	16.9	13	1.9	3.7	13.0	19.9
15	30	15	1.7	3.0	16.2	19.7	15	1.1	1.4	15.8	18.1
15	35	18	1.4	2.1	18.5	21.4	16	2.3	5.4	18.6	23.3
15	40	20	1.6	2.5	21.3	24.6	20	2.0	4.3	21.9	26.0
15	45	22	1.9	3.8	24.0	27.9	20	1.9	3.8	23.0	26.9
15	50	25	1.7	2.9	26.2	29.7	24	2.1	4.5	25.8	30.1
15	55	27	2.0	4.2	28.9	33.0	26	1.7	2.8	27.2	30.7
15	60	29	1.8	3.2	30.1	33.8	29	1.6	2.8	30.3	33.6
15	65	31	2.2	4.9	32.7	37.2	31	2.1	4.7	32.8	37.1
15	70	33	1.7	3.1	35.2	38.7	32	3.1	9.9	33.8	40.1
15	75	35	2.3	5.6	35.6	40.3	35	2.6	7.2	37.3	42.6
15	80	36	3.0	9.4	39.9	46.0	39	2.8	8.2	38.1	43.8
15	85	41	2.7	7.7	42.2	47.7	37	2.6	7.2	40.3	45.6
15	90	42	3.2	10.4	43.7	50.2	45	2.0	4.0	44.9	49.0
15	95	45	3.1	9.7	45.8	52.1	46	2.6	6.8	48.3	53.6
15	100	50	2.1	4.4	50.9	55.1	48	3.5	12.9	48.4	55.5
20	10	0	0	0	0	0	0	0	0	0	0
20	15	0	0	0	0	0	0	0	0	0	0
20	20	0	0	0	0	0	0	0	0	0	0
20	25	11	0.5	0.3	12.4	13.5	12	0.8	0.6	12.1	13.8
20	30	11	1.4	2.1	11.5	14.4	12	1.4	2.0	11.5	14.4
20	35	13	2.3	5.3	13.6	18.3	13	1.8	3.5	14.1	17.8
20	40	17	1.5	2.2	17.4	20.5	16	0.9	0.9	17.0	18.9
20	45	18	1.4	2.2	19.5	22.4	16	1.8	3.3	17.1	20.8
20	50	18	2.5	6.3	19.4	24.5	18	2.0	4.3	18.9	23.0
20	55	20	1.3	1.8	21.6	24.3	21	1.7	2.9	22.2	25.7
20	60	23	1.0	1.0	23.9	26.0	22	2.0	4.2	22.9	27.0
20	65	24	2.2	5.2	24.7	29.2	23	1.7	3.1	25.2	28.7
20	70	25	2.4	5.8	26.5	31.4	26	1.1	1.4	26.8	29.1
20	75	28	1.4	2.2	29.5	32.4	28	2.6	6.8	28.3	33.6
20	80	30	1.3	1.7	31.6	34.3	29	2.1	4.8	30.8	35.1
20	85	31	1.4	1.9	32.5	35.4	32	1.7	3.0	32.2	35.7
20	90	34	3.0	9.5	33.9	40.0	33	1.4	2	33.5	36.4
20	95	33	2.0	4.2	36.9	41.0	36	1.7	2.8	36.2	39.7
20	100	36	2.1	4.5	38.8	43.1	38	1.8	3.5	38.1	41.8

На рисунку 3.9 подано графік залежності рекордного значення цільової функції від параметрів задачі для кожного алгоритму.

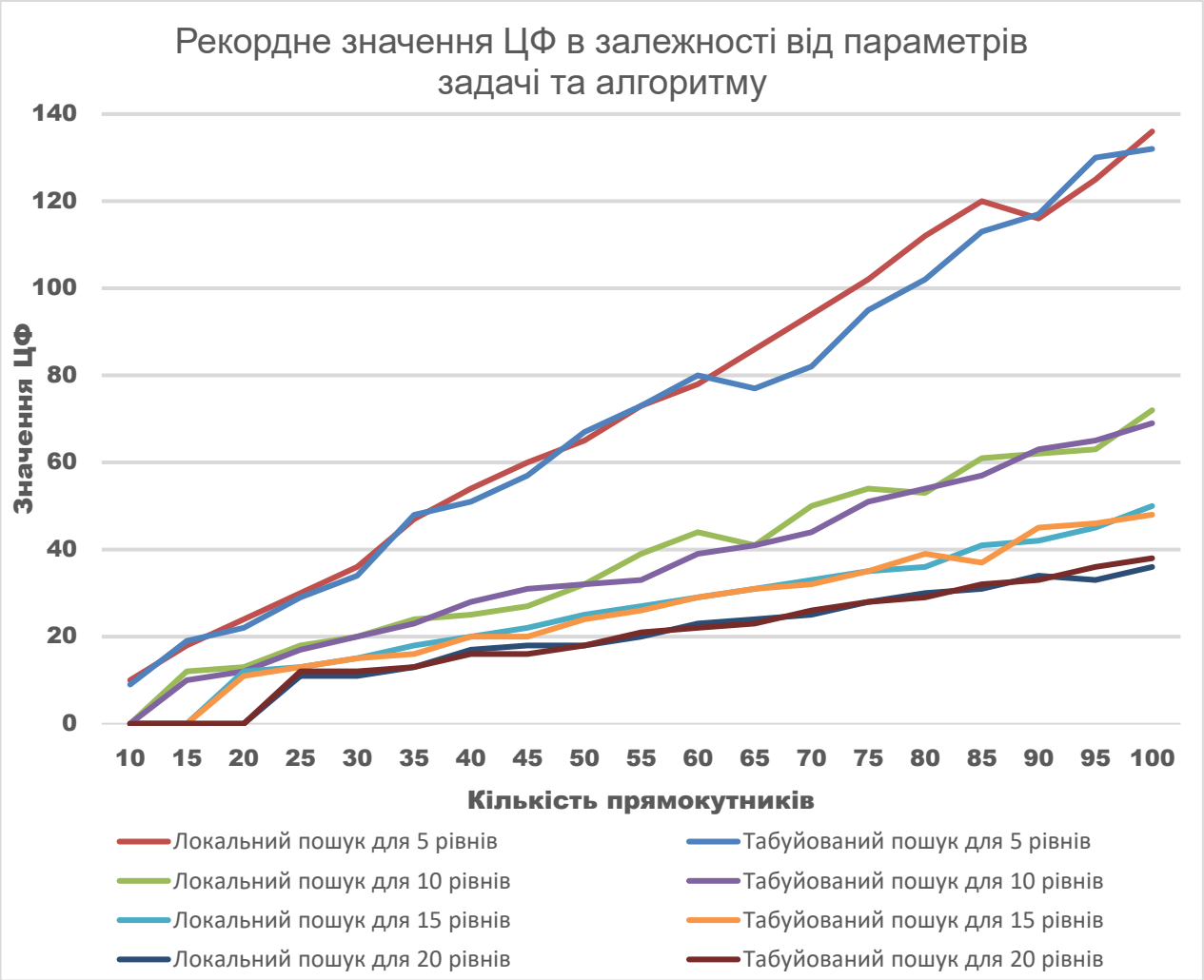


Рисунок 3.9 – Рекордне значення цільової функції

Рекордні значення цільової функції для задач з кількістю рівнів 15 та 20 майже однакові, проте, якщо кількість прямокутників більша за 90 – кращі рекордні значення отримані алгоритмом локального пошуку. Для задач з кількістю рівнів 5 та 10 алгоритмом локального пошуку можна отримати краще рекордне значення.

Отже, для розглянутої задачі розміщення прямокутників на напівнескінченній стрічці алгоритм табуйованого пошуку показав кращі результати, ніж алгоритм локального пошуку.

До переваг алгоритму локального пошуку можна віднести такі: можливість перегляду всіх точок в околі, для деяких задач – найкраще значення цільової функції. Недоліками алгоритму є час роботи алгоритму та гірше значення середнього покращення значення цільової функції.

До переваг алгоритму табуйованого пошуку можна віднести час роботи алгоритму, середнє покращення значення цільової функції. Недоліком алгоритму

табуйованого пошуку є менше знайдене рекордне значення цільової функції.

Час роботи алгоритму локального пошуку є суттєвим недоліком, оскільки у реальних задачах є обмеження у часових ресурсах, тому за вказаний час можна не отримати найкращого розв’язку. Для таких випадків доцільно використовувати алгоритм табуйованого пошуку, оскільки середнє покращення вище, ніж у алгоритмі локального пошуку.

3.3 Дослідження алгоритмів обміну та обміну із заборонами

Проведемо експериментальне дослідження та порівняємо алгоритми обміну та обміну із заборонами. Для проведення експериментів було встановлено обмеження в 1000 ітерацій для завершення роботи алгоритмів.

Результатами експериментів є час роботи алгоритмів, рекордне значення цільової функції, середнє покращення, дисперсія, середньо-квадратичне відхилення та довірчі інтервали, що наведені у таблицях 3.6-3.7.

Таблиця 3.6 – Результати роботи алгоритму обміну

Рівні	Пр.	Час (мс)	Рекорд	Сер. покр	Дисперсія	СКВ	Довірчі інтервали	
5	10	1,218	16	8,68	5,969	2,443	14,557	19,443
5	15	1,795	18	5,36	35,402	5,95	18,05	29,95
5	20	2,393	29	5,84	18,173	4,263	28,737	37,263
5	25	3,75	32	4,6	21,538	4,641	35,359	44,641
5	30	4,229	39	2,42	26,823	5,179	42,821	53,179
5	35	5,196	47	2,92	17,713	4,209	51,791	60,209
5	40	7,066	55	2,92	24,051	4,904	55,096	64,904
5	45	9,256	60	2,02	20,72	4,552	65,448	74,552
5	50	10,283	67	2,28	34,041	5,834	73,166	84,834
5	55	12,509	73	2,44	42,122	6,49	79,51	92,49
5	60	13,672	81	3	24,965	4,996	87,004	96,996
5	65	14,678	81	1,98	56,287	7,502	89,498	104,502
5	70	18,296	92	1,5	50,612	7,114	98,886	113,114
5	75	18,113	99	1,36	58,602	7,655	103,345	118,655
5	80	20,335	95	1,22	77,16	8,784	112,216	129,784
5	85	27,489	107	1,2	68,817	8,296	117,704	134,296
5	90	25,858	121	0,74	81,207	9,011	126,989	145,011
5	95	30,014	123	1,06	78,523	8,861	135,139	152,861

Продовження таблиці 3.6.

Рівні	Пр.	Час (мс)	Рекорд	Сер. покр	Дисперсія	СКВ	Довірчі інтервали	
15	40	15,183	19	6,98	4,059	2,015	21,985	26,015
15	45	15,088	21	5,92	4,126	2,031	23,969	28,031
15	50	17,123	25	5,72	5,563	2,359	26,641	31,359
15	55	20,407	27	5	5,666	2,38	28,62	33,38
15	60	20,585	27	4,22	6,622	2,573	31,427	36,573
15	65	22,684	29	5,4	6,711	2,59	33,41	38,59
15	70	24,322	34	4,58	4,967	2,229	35,771	40,229
15	75	29,033	37	4,14	4,867	2,206	38,794	43,206
15	80	28,956	38	4,18	7,381	2,717	41,283	46,717
15	85	32,098	39	4,2	8,281	2,878	42,122	47,878
15	90	35,018	43	3,1	10,531	3,245	44,755	51,245
15	95	35,56	47	3,14	7,46	2,731	48,269	53,731
15	100	39,574	49	2,78	9,876	3,143	50,857	57,143
20	10	0	0	0	0	0	0	0
20	15	0	0	0	0	0	0	0
20	20	0	0	0	0	0	0	0
20	25	12,126	12	5,18	1,587	1,26	11,74	14,26
20	30	12,683	12	8,74	5,071	2,252	12,748	17,252
20	35	13,912	12	7,44	4,983	2,232	15,768	20,232
20	40	18,085	16	7,04	3,402	1,845	18,155	21,845
20	45	21,982	16	7	5,122	2,263	18,737	23,263
20	50	21,992	19	6,36	3,02	1,738	21,262	24,738
20	55	24,619	19	4,84	4,041	2,01	22,99	27,01
20	60	28,332	23	5,94	4,532	2,129	23,871	28,129
20	65	31,85	24	5,1	5,783	2,405	26,595	31,405
20	70	35,043	27	3,04	4,132	2,033	28,967	33,033
20	75	39,11	29	3,56	4,694	2,167	29,833	34,167
20	80	43,16	31	3,08	4,908	2,215	31,785	36,215
20	85	40,256	29	3,86	7,984	2,826	33,174	38,826
20	90	44,842	30	3,88	8,836	2,973	35,027	40,973
20	95	58,679	35	4,2	5,533	2,352	37,648	42,352
20	100	60,299	35	3,3	8,041	2,836	39,164	44,836
5	100	32,92	121	1,18	86,553	9,303	140,697	159,303
10	10	0	0	0	0	0	0	0
10	15	3,59	11	13,1	4,245	2,06	11,94	16,06
10	20	4,35	15	9,84	5,161	2,272	14,728	19,272
10	25	5,779	19	8,18	3,277	1,81	20,19	23,81
10	30	7,07	21	5,56	6,423	2,534	22,466	27,534
10	35	9,018	22	6,08	14,556	3,815	25,185	32,815
10	40	9,616	26	5,26	10,825	3,29	29,71	36,29
10	45	12,442	31	5,9	7,851	2,802	34,198	39,802
10	50	13,846	33	4,9	11,998	3,464	35,536	42,464
10	55	15,845	38	4,14	8,67	2,944	41,056	46,944
10	60	26,131	38	4,6	17,484	4,181	43,819	52,181

Продовження таблиці 3.6.

Рівні	Пр.	Час (мс)	Рекорд	Сер. покр	Дисперсія	СКВ	Довірчі інтервали	
10	65	21,412	43	4,1	13,037	3,611	47,389	54,611
10	70	23,8	47	3,02	16,221	4,028	50,972	59,028
10	75	24,858	49	3,2	10,271	3,205	55,795	62,205
10	80	24,683	49	3,28	27,985	5,29	56,71	67,29
10	85	27,523	57	3,12	14,133	3,759	62,241	69,759
10	90	30,004	63	2,8	14,518	3,81	66,19	73,81
10	95	31,583	67	2,58	17,367	4,167	68,833	77,167
10	100	34,033	66	1,88	26,745	5,172	73,828	84,172
15	10	0	0	0	0	0	0	0
15	15	0	0	0	0	0	0	0
15	20	8,996	12	11,78	1,704	1,306	11,694	14,306
15	25	9,319	12	7,9	9,468	3,077	12,923	19,077
15	30	10,351	16	7,6	3,747	1,936	17,064	20,936
15	35	12,581	15	7,6	4,73	2,175	19,825	24,175

На рисунку 3.10 побудовано графік залежності часу алгоритму обміну від параметрів задач.

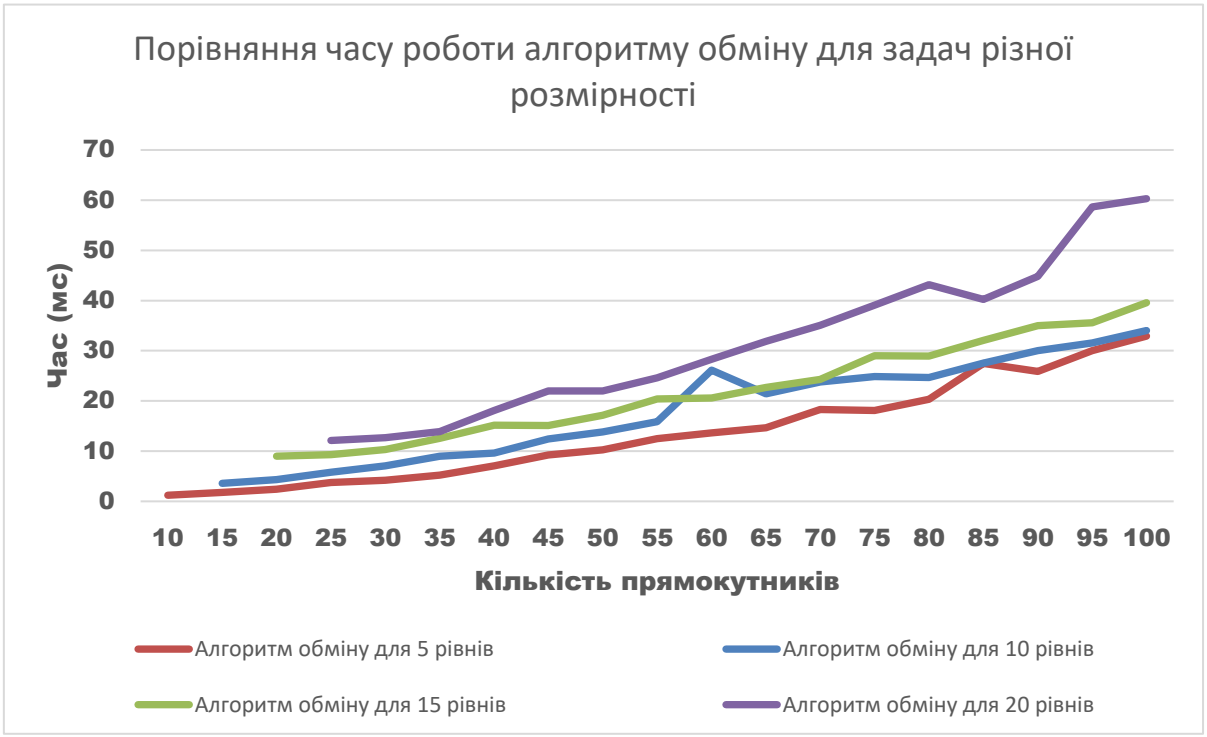


Рисунок 3.10 – Час роботи алгоритму обміну

Час роботи алгоритму обміну залежить від параметрів задачі: кількості прямокутників та кількості рівнів на стрічці. Експериментально було визначено, що час роботи алгоритму для 60 прямокутників для 10 рівнів може перевищувати

час роботи для 15 рівнів. Час роботи алгоритму для задачі з кількістю прямокутників 85 для 5 та 10 рівнів однаковий. Це зумовлено тим, що були зациклення.

У таблиці 3.7 наведено результати алгоритму обміну з заборонами.

Таблиця 3.7 – Результати роботи алгоритму обміну з заборонами

Рівні	Пр.	Час(мс)	Рекорд	Сер. покр	Дисперсія	СКВ	Довірчі інтервали	
5	10	1,536	15	13,1	22,649	4,759	16,241	25,759
5	15	2,394	22	8,12	5,536	2,353	24,647	29,353
5	20	3,85	27	8,02	17,388	4,17	27,83	36,17
5	25	6,084	28	4,02	13,486	3,672	35,328	42,672
5	30	7,707	34	4,38	28,189	5,309	41,691	52,309
5	35	8,739	45	2,38	32,165	5,671	49,329	60,671
5	40	9,398	52	2,44	31,234	5,589	58,411	69,589
5	45	9,435	62	2,66	29,618	5,442	64,558	75,442
5	50	10,513	58	1,98	44,327	6,658	70,342	83,658
5	55	12,239	73	2,4	42,796	6,542	74,458	87,542
5	60	14,227	74	2,06	47,551	6,896	86,104	99,896
5	65	15,718	86	2,1	28,882	5,374	93,626	104,37
5	70	18,97	95	0,96	52,76	7,264	99,736	114,26
5	75	22,631	100	1,4	40,164	6,337	106,663	119,33
5	80	23,068	107	1,04	47,506	6,892	114,108	127,89
5	85	23,614	106	0,98	80,647	8,98	121,02	138,98
5	90	26,732	119	1,44	61,959	7,871	127,129	142,87
5	95	31,483	129	1,18	38,612	6,214	138,786	151,21
5	100	29,722	136	0,98	86,904	9,322	141,678	160,32
10	10	0	0	0	0	0	0	0
10	15	3,211	11	12,18	3,158	1,777	11,223	14,777
10	20	4,311	14	10,28	4,222	2,055	14,945	19,055
10	25	5,642	18	5,88	7,276	2,697	19,303	24,697
10	30	10,599	20	7,4	7,649	2,766	22,234	27,766
10	35	12,008	26	6,1	5,102	2,259	26,741	31,259
10	40	15,412	26	5,7	9,535	3,088	29,912	36,088
10	45	15,838	29	4,26	10,973	3,313	33,687	40,313
10	50	20,486	33	4,96	12,263	3,502	37,498	44,502
10	55	19,223	35	4,8	12,643	3,556	41,444	48,556
10	60	21,49	39	5,26	12,908	3,593	45,407	52,593
10	65	22,779	44	4,18	17,362	4,167	47,833	56,167
10	70	25,056	48	3,52	14,257	3,776	51,224	58,776
10	75	26,801	51	3,42	13,367	3,656	55,344	62,656
10	80	35,344	56	2,68	12,898	3,591	57,409	64,591
10	85	33,076	58	2,86	15,204	3,899	62,101	69,899

Продовження таблиці 3.7.

Рівні	Пр.	Час(мс)	Рекорд	Сер. покр	Дисперсія	СКВ	Довірчі інтервали	Рівні
15	35	13,363	19	8	3,07	1,752	19,248	22,752
15	40	14,936	19	7,78	4,526	2,127	20,873	25,127
15	45	17,63	21	4,84	5,553	2,356	23,644	28,356
15	50	17,463	24	5,06	5,457	2,336	26,664	31,336
15	55	21,182	27	4,42	5,249	2,291	29,709	34,291
15	60	37,528	29	4,04	7,796	2,792	31,208	36,792
15	65	28,669	29	4,66	8,504	2,916	33,084	38,916
15	70	28,808	35	4,32	6,041	2,458	36,542	41,458
15	75	37,571	37	3,78	6,867	2,621	38,379	43,621
15	80	31,618	35	3,6	11,506	3,392	39,608	46,392
15	85	39,062	41	3,46	8,704	2,95	43,05	48,95
15	90	35,47	43	3,04	9,031	3,005	45,995	52,005
15	95	39,033	45	2,9	9,847	3,138	47,862	54,138
15	100	41,142	47	2,58	12,189	3,491	50,509	57,491
20	10	0	0	0	0	0	0	0
20	15	0	0	0	0	0	0	0
20	20	0	0	0	0	0	0	0
20	25	11,131	12	8,56	0,425	0,652	12,348	13,652
20	30	13,681	12	11,22	3,404	1,845	13,155	16,845
20	35	16,796	13	7,24	5,478	2,341	14,659	19,341
20	40	23,761	16	6,98	4,145	2,036	17,964	22,036
20	45	27,973	18	6,5	3,298	1,816	19,184	22,816
20	50	32,309	19	5,5	3,51	1,874	21,126	24,874
20	55	35,319	22	4,68	3,383	1,839	24,161	27,839
20	60	35,96	23	5,72	3,536	1,881	25,119	28,881
20	65	35,508	25	4,28	4,202	2,05	26,95	31,05
20	70	43,381	25	4,62	6,567	2,563	27,437	32,563
20	75	44,704	25	3,64	6,653	2,579	29,421	34,579
20	80	41,535	29	4,04	4,547	2,132	31,868	36,132
20	85	38,235	33	4,1	4,358	2,087	33,913	38,087
20	90	46,105	33	3,16	5,888	2,426	36,574	41,426
20	95	51,958	34	2,62	5,672	2,382	37,618	42,382
20	100	49,562	38	3,78	5,359	2,315	39,685	44,315
10	95	37,628	67	2,76	16,858	4,106	69,894	78,106
10	100	38,975	70	2,06	19,904	4,461	73,539	82,461
15	10	0	0	0	0	0	0	0
15	15	0	0	0	0	0	0	0
15	20	9,73	12	11,46	2,025	1,423	11,577	14,423
15	25	10,383	12	7,86	4,404	2,098	13,902	18,098
15	30	11,544	15	7,24	5,391	2,322	17,678	22,322

Відповідно до отриманих результатів побудуємо графік залежності часу роботи алгоритму від параметрів задачі (рисунок 3.11).

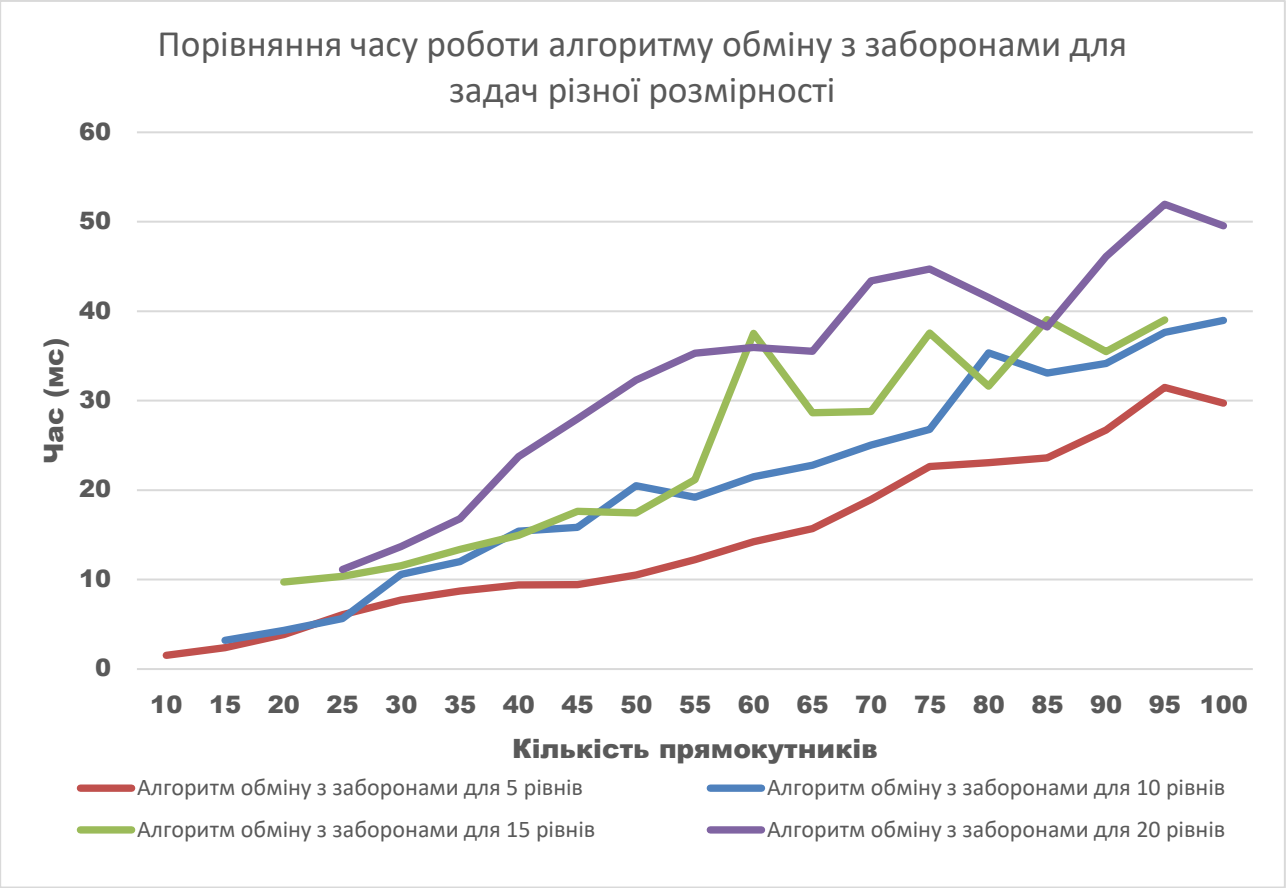


Рисунок 3.11 – Час роботи алгоритму обміну з заборонами

Час роботи алгоритму обміну з заборонами залежить від параметрів задачі (кількості прямокутників та рівнів на стрічці).

Для задач, де кількість рівнів 5-10 та прямокутників 5-25, кількість рівнів 10-15 та прямокутників 30-50 – час роботи алгоритмів однаковий. Це зумовлено тим, що алгоритм обміну із заборонами не буде розглядати повторно рядки, які не покращили значення цільової функції на попередніх ітераціях.

На рисунку 3.12 наведено графік залежності рекордного значення цільової функції від параметрів задачі та обраного алгоритму (обміну чи обміну з заборонами).

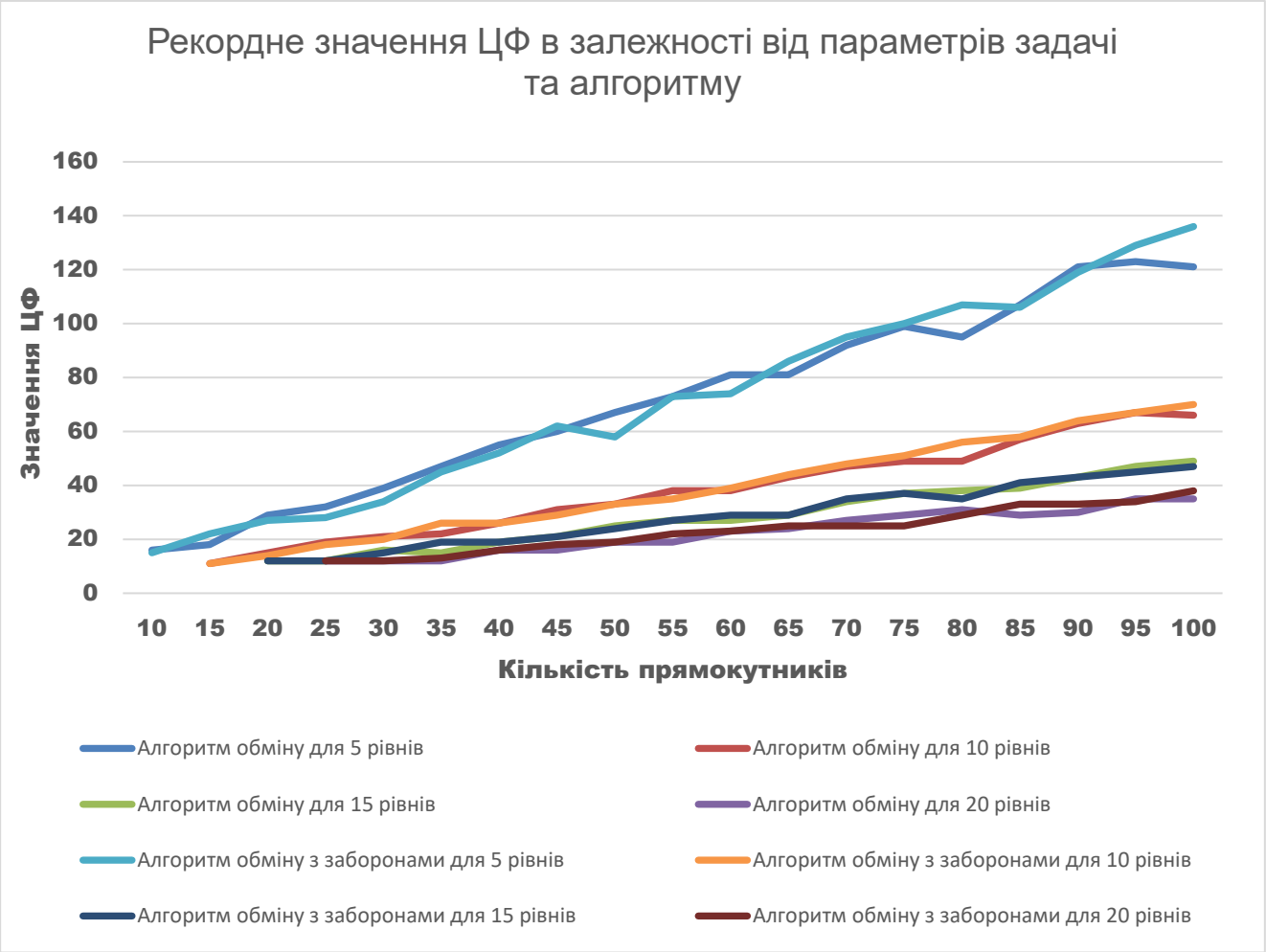


Рисунок 3.12 – Рекордне значення цільової функції

Для задач з різними параметрами алгоритми обміну та обміну з заборонами мають незначні відмінності у рекордних значеннях цільових функцій.

Для задач з кількістю рівнів 5 та кількістю прямокутників 75-100 кращі результати показав алгоритм обміну. Це зумовлено тим, що алгоритм обміну може мати зациклення, в той час, коли алгоритм обміну з заборонами не розглядає повторно рівні з розміщенням прямокутників, які були розглянуті раніше.

На рисунках 3.13-3.14 подано графіки залежності середнього покращення від алгоритму та параметрів задачі.

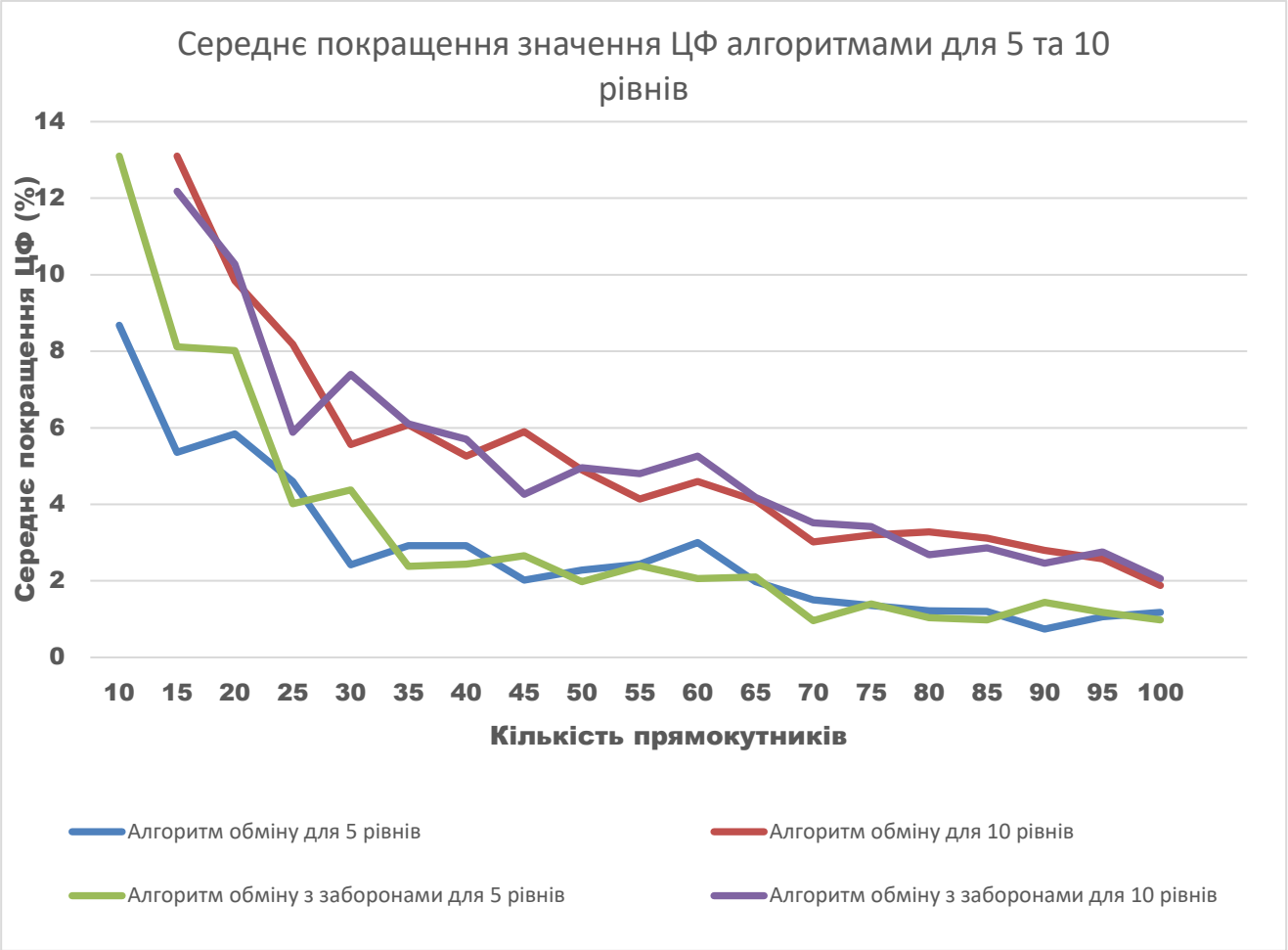


Рисунок 3.13 – Середнє покращення значення цільової функції алгоритмами для 5 та 10 рівнів

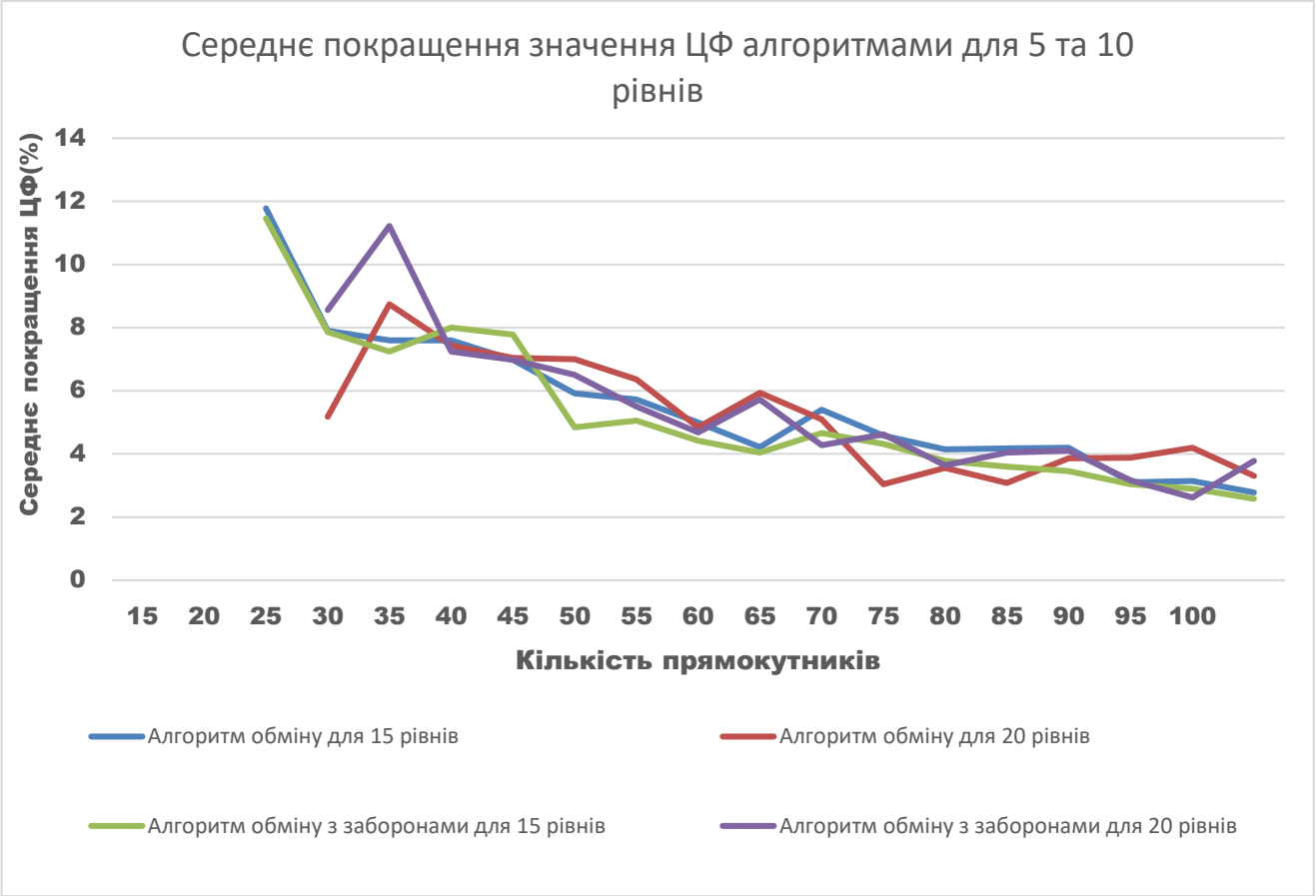


Рисунок 3.14 – Середнє покращення значення цільової функції алгоритмами для 15 та 20 рівнів

Для задач з кількістю рівнів 5 та кількістю прямокутників менша за 35 можна спостерігати краще середнє покращення алгоритмом обміну з заборонами. Це зумовлено тим, що для задач невеликої розмірності алгоритм обміну має змогу розглянути більшу кількість можливих пар рівнів і отримати кращий результат. Для кількості прямокутників, що більша за 35 – середнє покращення було отримане майже однакове.

Для задач з кількістю рівнів – 10-20, середнє покращення алгоритмів майже однакове. Це зумовлено тим, що зі збільшенням рівнів збільшується кількість варіантів вибору рівнів та обміну прямокутниками.

Для задач з 20 рівнями та кількістю прямокутників, що не перевищують 35 алгоритмом обміну з заборонами було отримано краще середнє покращення.

Відповідно до проведеного дослідження, можна виділити переваги та

недоліки кожного з алгоритмів. Алгоритм обміну розглядає повторно рівні та розміщує прямокутники, на відміну від алгоритму обміну з заборонами. Перевагами алгоритму обміну з заборонами є час роботи та середнє покращення.

Алгоритм обміну доцільно використовувати, збільшивши кількість ітерацій, якщо часовий ресурс не має обмежень.

3.4 Порівняння ефективності алгоритмів

Серед розглянутих алгоритмів визначимо найкращі алгоритми для подальшого порівняння. Відповідно до отриманих експериментально результатів, надалі будуть досліджуватися наступні алгоритми: жадібний алгоритм без модифікацій (ЖА1), алгоритм табуйованого пошуку та алгоритм обміну із заборонами.

Відповідно до отриманих даних побудуємо графіки залежності часу роботи алгоритму від параметрів задачі та представимо на рисунках 3.15-3.16.

Рисунок 3.15 – Порівняння часу роботи алгоритмів для 5 та 10 рівнів

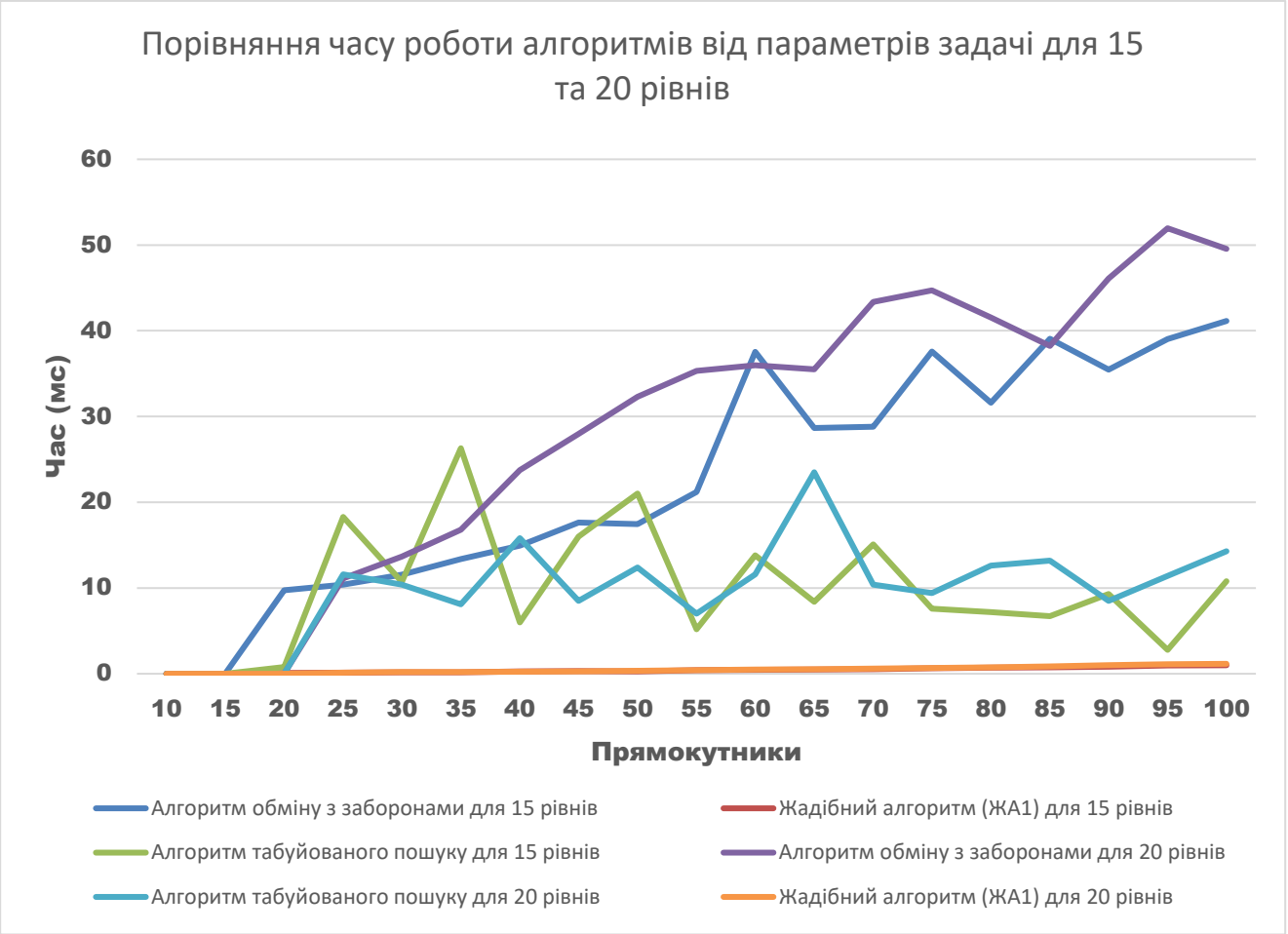


Рисунок 3.16 – Порівняння часу роботи алгоритмів для 15 та 20 рівнів

Серед розглянутих алгоритмів найшвидше працює жадібний алгоритм без модифікацій, оскільки основними кроками алгоритму є розміщення прямокутників.

Найбільший час роботи алгоритмів для задач, де кількість прямокутників перевищує 40, спостерігається у алгоритму обміну з заборонами. Це обумовлено тим, що для задач із великою кількістю прямокутників витрачається час на впорядкування прямокутників та розміщення їх на рівнях стрічки.

Для задач, де стрічка має 5-10 рівнів та кількість прямокутників не перевищує 40, найдовше працює алгоритм табуйованого пошуку. Це зумовлено тим, що для заданої кількості прямокутників та рівнів розглядається набагато більше точок околу, ніж кількість пар рівнів для алгоритму обміну з заборонами.

Порівняємо рекордні значення отримані алгоритмами та представимо їх на рисунку 3.17.

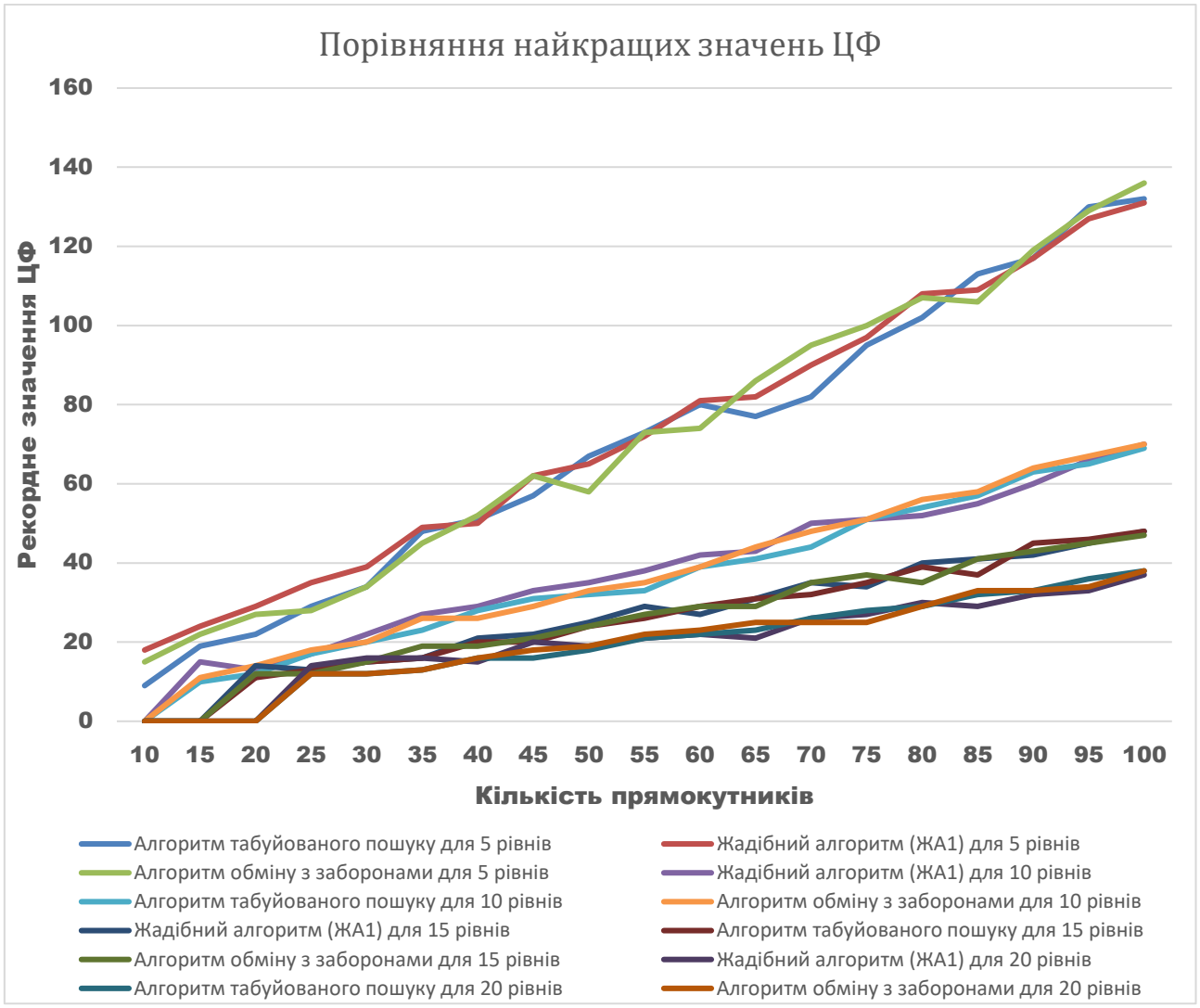


Рисунок 3.17 – Порівняння рекордних значень цільової функції

Для задач, де кількість прямокутників не перевищує 20 та 60-80, а кількість рівнів стрічки 5 – найкраще значення цільової функції було отримано алгоритмом табуйованого пошуку. Найкращий результат, що був отриманий алгоритмом обміну, можна спостерігати для задач, де кількість рівнів 5 та кількість прямокутників 35, 50-60. Для задачі, у якої стрічка містить 5 рівнів, а кількість прямокутників перевищує 90, найкращий результат був отриманий жадібним алгоритмом. Це зумовлено тим, що алгоритм табуйованого пошуку не розглянув достатню кількість околів.

Для задач, де стрічка має 10 рівнів алгоритми табуйованого пошуку та обміну з заборонами показали однакові рекордні значення для більшості задач. Для задач, де кількість прямокутників перевищує 75 було отримано найкраще значення

цільової функції жадібним алгоритмом без модифікації.

Для задач, де кількість рівнів 15-20 алгоритми показали майже однакові рекордні значення цільової функції. Це зумовлено тим, що для алгоритмів табуйованого пошуку та обміну необхідно збільшити кількість ітерацій. Найкращі результати були отримані алгоритмом обміну з заборонами для задач, де кількість прямокутників становить 25-35 та 70-80. Це зумовлено тим, що на кожному кроці розглядалися рівні перестановка прямокутників на яких значно покращувала значення цільової функції.

Порівняємо середнє покращення, що було отримано алгоритмами обміну з заборонами та табуйованого пошуку. Оскільки жадібний алгоритм реалізований за принципом «зліва-вниз», за яким відбувається розміщення прямокутників для вказаних алгоритмів, то будемо порівнювати на скільки відсотків в середньому покращується значення цільової функції. Результати подано на рисунках 3.18 та 3.19.

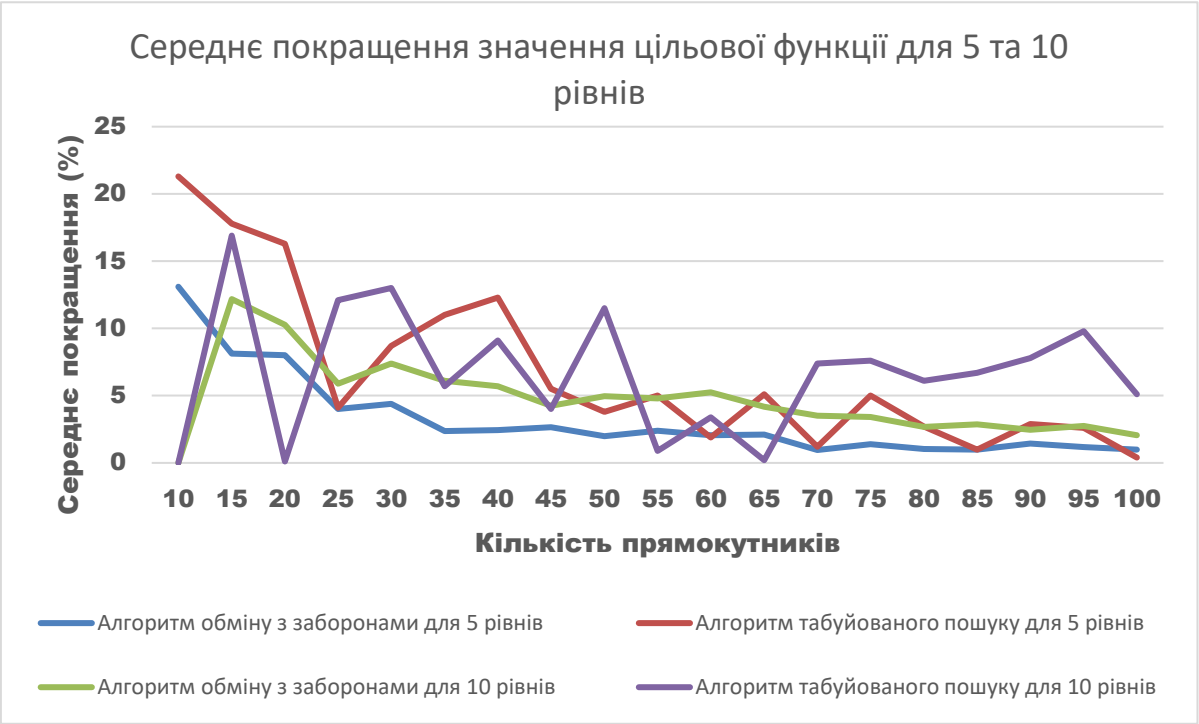


Рисунок 3.18 – Порівняння середнього покращення алгоритмами табуйованого пошуку та обміну

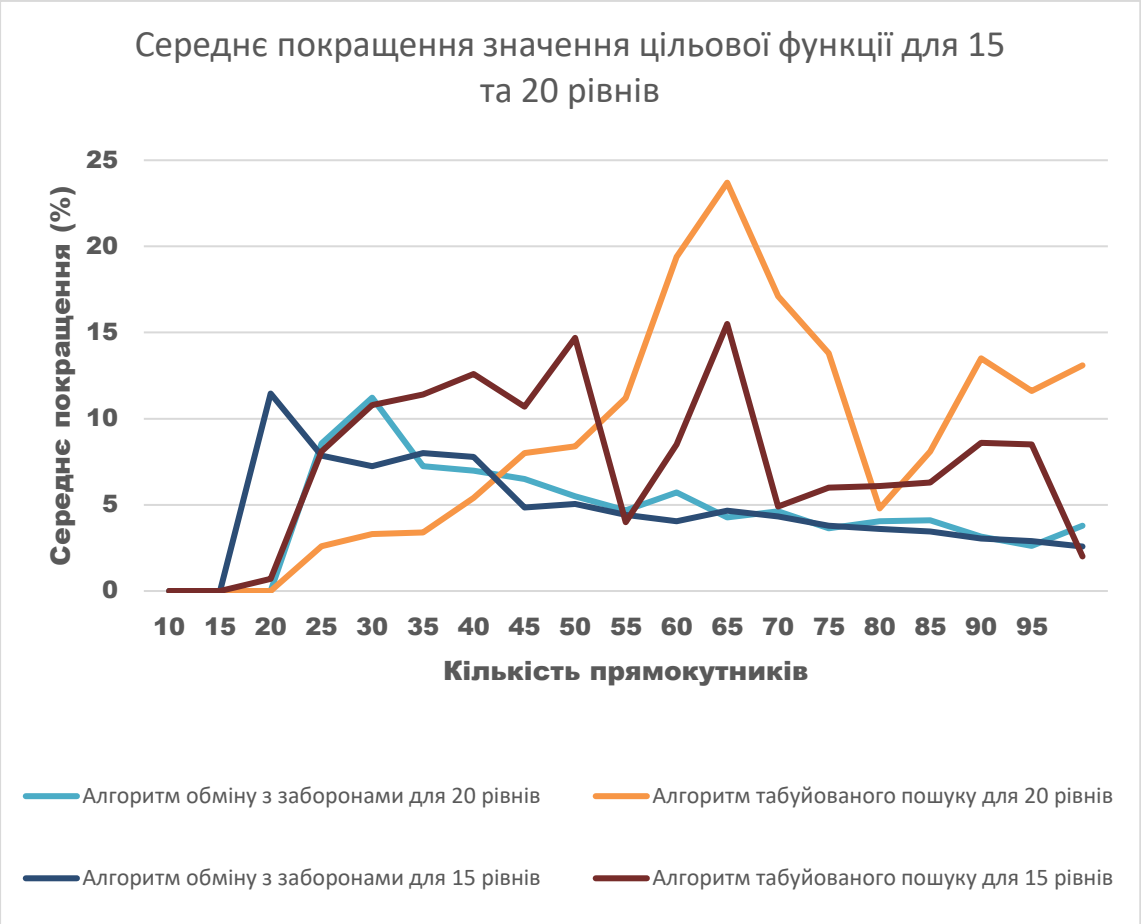


Рисунок 3.19 – Порівняння середнього покращення алгоритмами табуйованого пошуку та обміну

Для більшості задач найбільше середнє покращення цільової функції було отримано алгоритмом табуйованого пошуку. Для задач, де кількість рівнів становить 15-20, а кількість прямокутників 15-25.

Для задач, де кількість прямокутників рівна кількісті рівнів на стрічці – алгоритми табуйованого пошуку та обміну з заборонами не застосовуються, оскільки всі прямокутники будуть розміщені по одному на кожному з рівнів. Для інших випадків алгоритми табуйованого пошуку та обміну з заборонами значно покращують початкове значення цільової функції.

Отже, серед розглянутих алгоритмів доцільно виділити алгоритми табуйованого пошуку та обміну з заборонами. Оскільки для більшості задач спостерігалося значне покращення значення цільової функції. Проте, доцільно виділити час роботи жадібного алгоритму без модифікацій, оскільки цей алгоритм

працює набагато швидше за інші, проте значно програє у покращенні значення цільової функції.

Відповідно до отриманих результатів, алгоритм табуйованого пошуку має значні переваги за алгоритм обміну, оскільки час роботи алгоритму значно менший для задач більшої розмірності, а отримане значення цільової функції значно краще. Проте, для задач невеликої розмірності, де кількість рівнів не перевищує 10, а кількість прямокутників 25, доцільно використовувати алгоритм обміну з заборонами, оскільки можна отримати краще рекордне значення цільової функції.

Висновки до розділу

В розділі проведено аналіз розроблених жадібного алгоритму та його модифікацій, локального та табуйованого пошуку, обміну та обміну з заборонами.

Для перевірки ефективності алгоритмів розв'язування задачі розміщення прямокутників на напівнескінченній стрічці проведено серію обчислювальних експериментів. Задачі, які використовувались для проведення експериментів, були згенеровані випадковим чином.

На основі проведених експериментів можна зробити висновки:

– серед розглянутих алгоритмів, які використовувались в експериментах, найкращий результат показав алгоритм табуйованого пошуку для задач великої розмірності;

– для задач невеликої розмірності переваги у значеннях цільової функції мав алгоритм обміну з заборонами.

4 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до технічного забезпечення

Програмний продукт складається з комплексу базових функцій дослідження задач розміщення прямокутників та їх оптимізації, побудови розміщення та підсистеми проведення експериментів. Систему можуть використовувати одночасно 15 людей незалежно один від одного.

Вимоги до технічних засобів:

а) комп’ютер:

1) частота процесора – 2.0 ГГЦ ;
2) об’єм накопичувача – 128 Гб;
3) об’єм оперативної пам’яті – 2 Гб;
4) кількість ядер – 4.

б) встановлене програмне забезпечення:

1) операційна система Windows 7/8/10;
2) MS SQL сервер та MS SQL Management Studio/Heidi SQL;
3) .NET Framework 4.8;
4) Visual Studio / Visual Code;
5) MS Excel.

в) комплектація комп’ютерної периферії:

1) мишка дротова/бездротова;
2) клавіатура стандартна;
3) монітор.

4.2 Архітектура програмного забезпечення

4.2.1 Діаграма пакетів

Відповідного до архітектури програмного забезпечення, на рисунку 4.1 наведено діаграму пакетів.

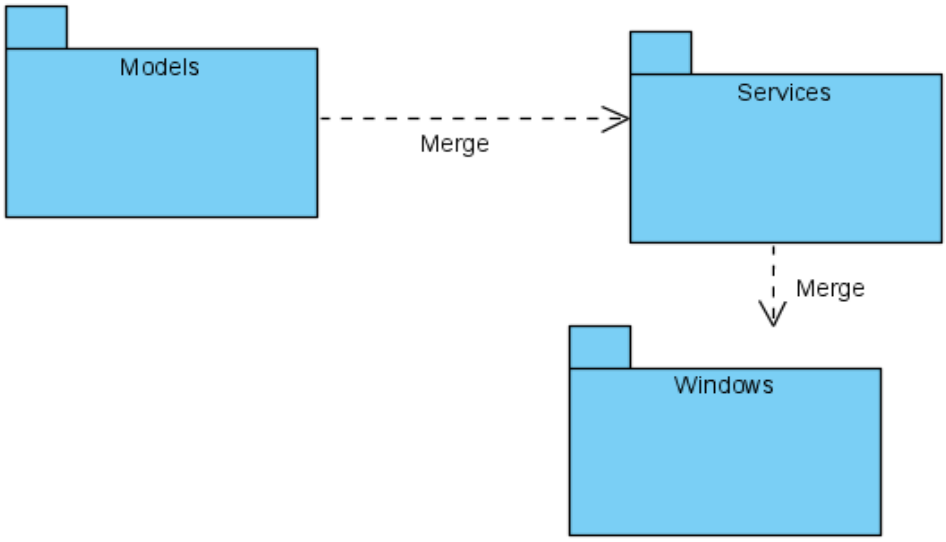


Рисунок 4.1 – Діаграма пакетів

Розглянемо опис вмісту папок з необхідними файлами.

Пакет Models складається з моделей, які зберігаються в базі даних та необхідні для роботи алгоритмів. Розширення файлів пакету – .cs. Набір класів пакету Models: Experement.cs, Rectangle.cs, Result.cs, Tap.cs.

Пакет Services складається з сервісів, які реалізують алгоритми розв'язання задачі, генерації початкових даних та проведення експериментів. Розширення файлів пакету – .cs. Набір класів пакету Services: GreedyService.cs, LocalFullSearch.cs, LocalFullSearchTaboo.cs, LocalSearchService.cs, OutputDataService.cs, RectangleService.cs, TaboSearchService.cs, TapeService.cs.

Пакет Windows складається з файлів представлення (вікна користувацького інтерфейсу). Розширення файлів пакету – .xaml та .xaml.cs. Набір елементів пакету Windows:

1) Experements.xaml;
2) Experements.xaml.cs;
3) GreedyAlg.xaml;
4) GreedyAlg.xaml.cs;
5) MainPage.xaml;
6) MainPage.xaml.cs;

7) ResolveTask.xaml;
8) ResolveTask.xaml.cs;
9) TapGeneration.xaml;
10) TapGeneration.xaml.cs;
11) TapNTapGeneration .xaml;
12) TapNTapGeneration .xaml.cs;
13) TaskGeneration.xaml;
14) TaskGeneration.xaml.cs.

4.2.2 Опис функціональної моделі

Розглянемо функціональну модель системи, що представлена діаграмою варіантів використання у графічному матеріалі.

Актором системи є користувач, який може використовувати наступні функції системи:

Розв'язання задачі – користувач має можливість отримати розв'язок задачі розміщення прямокутників, вказавши вхідні дані.

Введення вхідних даних – користувач обирає спосіб створення вхідних даних (генерація або введення власних даних).

Генерація даних – користувач вказує необхідні дані для генерації задачі: кількість прямокутників, або перелік прямокутників, кількість рівнів та кількість дірок на стрічці.

Редагування даних – користувач вводить вхідні дані та має можливість редагувати їх.

Збереження вихідних даних – користувач має можливість зберегти дані в базу даних чи в файл.

Планування експериментів – користувач має можливість експериментально дослідити ефективність алгоритмів, задавши параметри проведення експериментів.

Генерація набору задач – користувач генерує набір задач, необхідних для проведення експериментів, вказавши параметри задач (кількість рівнів, прямокутників, дірок і т.п.).

Вибір параметрів експерименту – користувач обирає параметри алгоритмів для яких проводитимуться експерименти.

Вибір типу експерименту – користувач обирає алгоритми, для яких буде проводити дослідження.

Формування результатів – користувач має можливість отримати результати експерименту та зберегти їх у базі даних чи у файлі.

Для наданих варіантів використання визначимо функціональні вимоги до системи і наведемо у таблиці 4.1.

Таблиця 4.1 – Функціональні вимоги

Варіант використання	**Функціональна вимога**	**Пріоритет**
Розв’язання задачі	1. Система надає можливість користувачу розв’язати задачу.	Високий
Введення вхідних даних	1.1 Система надає можливість користувачу ввести дані вручну або згенерувати.	Високий
	1.1.1 Система надає можливість користувачу ввести перелік довжин прямокутників.	Високий
	1.1.2 Система надає можливість користувачу вказати кількість рівнів на стрічці.	Високий
	1.1.3 Система надає можливість користувачу вказати кількість дірок на стрічці.	Високий
Генерація даних	1.2 Система надає можливість користувачу згенерувати дані.	Високий
	1.2.1 Система надає можливість користувачу вказати кількість рівнів стрічки для генерування.	Високий
	1.2.2 Система надає можливість користувачу вказати кількість прямокутників.	Високий
	1.2.3 Система надає можливість користувачу вказати межі довжин сторін прямокутника.	Високий
Редагування даних	1.3 Система надає можливість користувачу редагувати введені дані.	Високий
Збереження вихідних даних	1.4 Система надає можливість користувачу зберігати згенеровані дані у таблиці бази даних.	Високий
Планування експериментів	2. Система надає можливість користувачу провести експерименти.	Високий

Продовження таблиці 4.1

Варіант використання	Функціональна вимога	Пріоритет
Генерація набору задач	2.1 Система надає можливість користувачу згенерувати набір задач.	Високий
	2.1.1 Система надає можливість користувачу вказати кількість прямокутників.	Високий
	2.1.2 Система надає можливість користувачу вказати кількість рівнів стрічки.	Високий
	2.2 2.1.3 Система надає можливість користувачу вказати кількість дірок на рівнях.	Високий
Вибір параметрів експерименту	2.3 Система надає можливість користувачу обрати алгоритми для проведення експериментів.	Високий
Вибір типу експерименту	2.4 Система надає можливість користувачу вказати кількість ітерацій для роботи алгоритмів.	Високий
Формування результатів	2.5 Система надає можливість користувачу оброблювати результати експериментів.	Високий
	2.5.1 Система надає можливість користувачу переглядати результати експериментів.	
	2.5.2 Система надає можливість користувачу зберігати результати експерименту у файл.	

4.2.3 Діаграма компонентів

Програмний продукт складається з певного набору компонентів, які пов’язані з іншими компонентами різними зв’язками або залежностями. На рисунку 4.2 наведена схема компонентів.

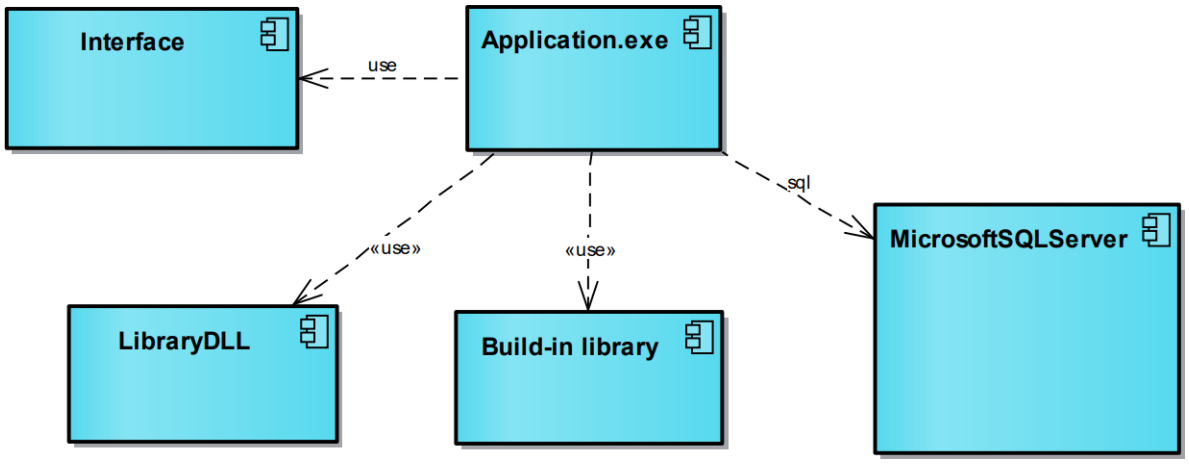


Рисунок 4.2 – Діаграма компонентів

Опис компонентів

Application.exe – виконуваний файл, який працює, використовуючи інші компоненти.

MicrosoftSQL Server – сервер бази даних, де зберігаються дані для роботи програми.

Компонент Interface складається з набору інтерфейсів програми, що допомагають з представленнями даних користувача.

LibraryDLL – бібліотека мови C#, яка описує більшість методів, необхідних для реалізації програми.

Build-in library – вбудовані бібліотеки.

Застосування складається з N-layer-ної архітектури, яка складається з трьох рівнів: рівень представлення даних, рівень обробки даних та рівень даних.

Рівень представлення даних відповідає за відображення даних користувачу, за рахунок їх отримання з рівня обробки. Користувач має прямий доступ лише до рівня представлення даних.

Рівень обробки даних відповідає за отримання даних, їх обробку та передачу на рівень представлення чи рівень даних. Цей рівень також називають рівнем логіки програми, оскільки складається з основної логіки програми.

Рівень даних відповідає за збереження даних у моделі та в базу даних. На цьому рівні не відбувається обробка цих даних.

Перевагою обраної архітектури є розділення функцій збереження, обробки та представлення даних.

4.2.4 Діаграма класів

Програмний продукт містить набір класів, які представляють моделі, що зберігаються в базі даних, набір сервісів для реалізації логіки алгоритмів та набір класів представлення.

Для збереження даних в базі даних використовуються класи та платформа EntityFramework та принцип Code First.

Для проведення експериментів та розв’язання задач були створені класи:

– Tap – описує модель стрічки, яка зберігається в базі даних та використовується сервісами;

– Rectangle – описує модель прямокутників, яка зберігається в базі даних та використовується сервісами;

– Experement – описує модель параметрів експериментів, які проводитимуться користувачем;

– Result – описує модель результатів експериментів;

– TaboSearchService – описує сервіс реалізації алгоритму локального пошуку;

– OutPutDataService – описує сервіс виводу даних;

– LocalFullSearchTabo – описує сервіс реалізації алгоритму повного перебору із заборонами;

– LocalFullSearch – описує сервіс реалізації алгоритму повного перебору;

– GreedyService – описує сервіс реалізації жадібних алгоритмів;

– RectangleService – описує сервіс генерації прямокутників;

– TapeService – описує сервіс створення стрічки;

– LocalSearchService – описує сервіс локального пошуку;

– Experements – описує представлення вікна проведення експериментів;

– ResolveTask – описує представлення вікна розв’язування задачі;

– TapGeneration – описує представлення вікна генерації;

– MainPage – описує представлення вікна головної сторінки;

– TapNTaskGeneration – описує представлення вікна генерації задачі;

– TaskGeneration – описує представлення вікна створення прямокутників;

На рисунках 4.3 – 4.5 представлені діаграми класів та зв’язки між ними.

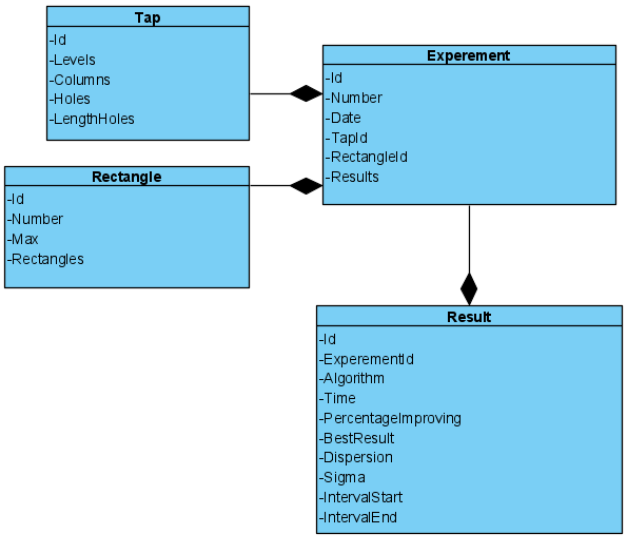


Рисунок 4.3 – Діаграма класів для моделей системи

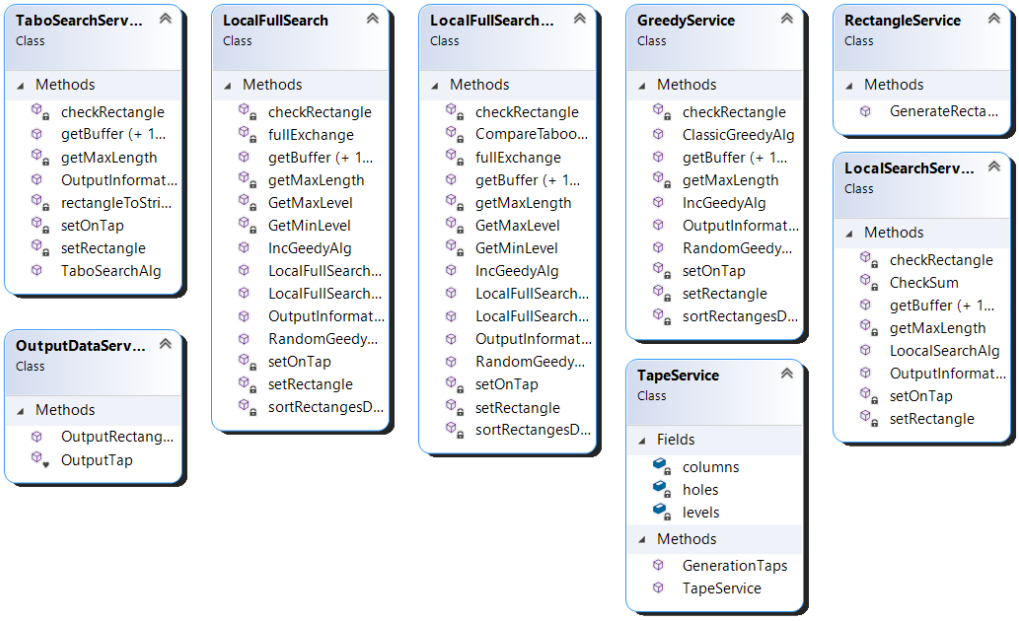


Рисунок 4.4 – Діаграма класів для сервісів системи

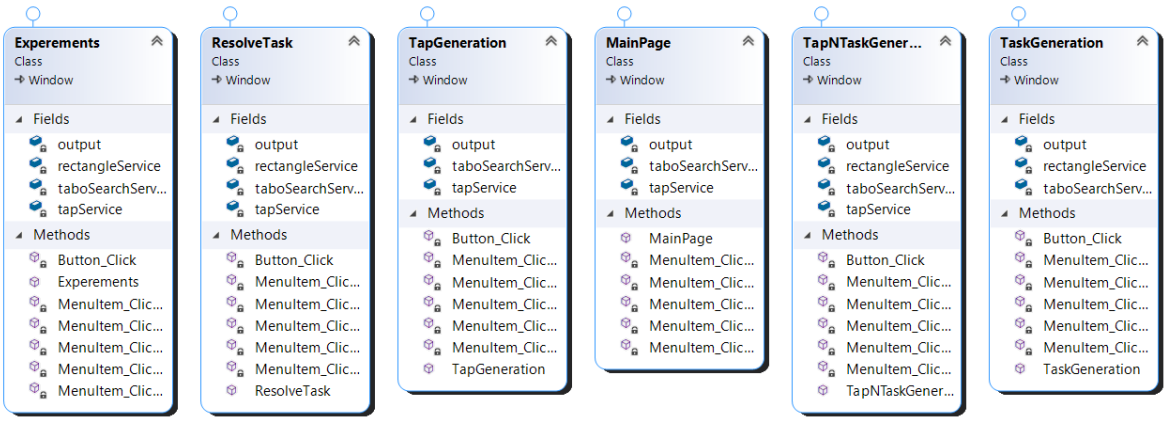


Рисунок 4.5 – Діаграма класів для представлення системи

4.2.5 Специфікація функцій

Згідно з існуючими пакетами у програмній реалізації наведемо основні компоненти та опис елементів пакетів.

Розглянемо пакет Services, який містить набір необхідних сервісів для роботи програми.

Розглянемо клас GreedyService, який описує роботу жадібного алгоритму та його модифікацій. У таблиці 4.2 наведемо опис методів класу.

Таблиця 4.2 – Методи класу GreedyService

Назва	Опис
public (double, double) ClassicGreedyAlg(int limit, int[][] tap, int[] rectangles)	Реалізація жадібного алгоритму без модифікацій. Вхідними даними є стрічка та набір прямокутників. Вихідними даними є час роботи алгоритму та значення цільової функції.
public (double, double) IncGeedyAlg(int limit, int[][] tap, int[] rectangles)	Реалізація жадібного алгоритму з модифікацією (впорядкування прямокутників по зростанню їх довжин). Вхідними даними є стрічка та набір прямокутників. Вихідними даними є час роботи алгоритму та значення цільової функції.
public (double, double) RandomGeedyAlg(int limit, int[][] tap, int[] rectangles)	Реалізація жадібного алгоритму з модифікацією (прямокутники впорядковані випадковим чином). Вхідними даними є стрічка та набір прямокутників. Вихідними даними є час роботи алгоритму та значення цільової функції.
private int[] sortRectangesDesc(int[] rectangleBuffer)	Повертає впорядкований набір прямокутників за спаданням їх довжин. Вхідними даними є набір прямокутників.
private int[] sortRectangesInc(int[] rectangleBuffer)	Повертає впорядкований набір прямокутників за зростанням їх довжин. Вхідними даними є набір прямокутників.
private int getMaxLength(int[][] bufferTapCurrent)	Повертає довжину найдовшого рівня стрічки. Вхідними даними є стрічка із розміщеними прямокутниками.
private int[][] setOnTap(int[][] bufferTap, int[] rectangleBuffer)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.
private void setRectangle(int[][] tap, int level, int rectangle, int colStart)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка, набір прямокутників та позиція розміщення (номер рівня та номер стовпчика).

Для реалізації алгоритму обміну використовуються методи із класу LocalFullSearch.

Наведемо опис алгоритмів у таблиці 4.3.

Таблиця 4.3 – Методи класу LocalFullSearch

Назва	Опис методу
public (double, double) LocalFullSearchRandomAlg(int limit, int[][] tap, int[] rectangles)	Реалізація алгоритму обміну. Вхідними даними є стрічка та набір прямокутників. Вихідними даними є час роботи алгоритму та значення цільової функції.
private (int[][], int) fullExchange(int[][] bufferTapCurrent, int pos1, int pos2, int Fmax)	Повертає стрічку та значення цільової функції під час обміну прямокутниками на 2х рівнях. Вхідними даними є стрічка з прямокутниками, номери рівнів та значення цільової функції для стрічки.
private int getMaxLength(int[][] bufferTapCurrent)	Повертає найдовшу довжину стрічки. Вхідними даними є стрічки та прямокутники
private int[][] setOnTap(int[][] bufferTap, int[] rectangleBuffer)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.
private void setRectangle(int[][] tap, int level, int rectangle, int colStart)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка, набір прямокутників та позиція розміщення (номер рівня та номер стовпчика).
private bool checkRectangle(int[] level, int rectangle, int colStart)	Вказує, чи можна розмістити прямокутник на обраній позиції. Вхідними даними є рівень, прямокутник та номер стовпчика.
public int[] getBuffer(int[] placement)	Повертає копію перестановки. Вхідними даними є перестановка.
public int[][] getBuffer(int[][] tap)	Повертає копію стрічки. Вхідними даними є стрічка.
public void OutputInformation(int[][]tap, int[] rectangle)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.

Клас LocalFullSearchTaboo описує реалізацію алгоритму обміну з заборонами. Детальний опис класу наведений у таблиці 4.4.

Таблиця 4.4 – Методи класу LocalFullSearchTaboo

Назва	Опис
private void addTabo**(**List<int**[][]**> taboList**,** int**[][]** bufferTapCurrent**)**	Додає перестановку в список заборон. Вхідними даними є список заборон та стрічка з розміщеними прямокутниками.
private bool checkTabo**(**int pos1**,** int pos2**,** int**[][]** bufferTapCurrent**,** List<int**[][]**> taboList**)**	Перевіряє список заборон. Вхідними даними є стрічка, номери рівнів та список заборон.
public **(**double**,** double**)** LocalFullSearchRandomAlg**(**int limit**,** int**[][]** tap**,** int**[]** rectangles**)**	Реалізація алгоритму обміну з заборонами. Вхідними даними є стрічка та набір прямокутників. Вихідними даними є час роботи алгоритму та значення цільової функції.
private **(**int**[][],** int**)** fullExchange**(**int**[][]** bufferTapCurrent**,** int pos1**,** int pos2**,** int Fmax**)**	Повертає стрічку та значення цільової функції під час обміну прямокутниками на 2х рівнях. Вхідними даними є стрічка з прямокутниками, номери рівнів та значення цільової функції для стрічки.
private int getMaxLength**(**int**[][]** bufferTapCurrent**)**	Повертає найдовшу довжину стрічки. Вхідними даними є стрічки та прямокутники
private int**[][]** setOnTap**(**int**[][]** bufferTap**,** int**[]** rectangleBuffer**)**	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.
private void setRectangle**(**int**[][]** tap**,** int level**,** int rectangle**,** int colStart**)**	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка, набір прямокутників та позиція розміщення (номер рівня та номер стовпчика).
private bool checkRectangle**(**int**[]** level**,** int rectangle**,** int colStart**)**	Вказує, чи можна розмістити прямокутник на обраній позиції. Вхідними даними є рівень, прямокутник та номер стовпчика.
public int**[]** getBuffer**(**int**[]** placement**)**	Повертає копію перестановки. Вхідними даними є перестановка.

В класі LocalSearchService наведена реалізація алгоритму локального пошуку. У таблиці 4.5 наведено опис методів.

Таблиця 4.5 – Методи класу LocalSearchService

Назва	Опис
public (double, double, double) LoocalSearchAlg(int limit, int[][] tap, int[] rectangles)	Реалізація алгоритму локального пошуку. Вхідними даними є стрічка, кількість ітерацій та набір прямокутників. Вихідними даними є час роботи алгоритму, середнє покращення та значення цільової функції.
private int getMaxLength(int[][] bufferTapCurrent)	Повертає найдовшу довжину стрічки. Вхідними даними є стрічки та прямокутники
private int[][] setOnTap(int[][] bufferTap, int[] rectangleBuffer)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.
private void setRectangle(int[][] tap, int level, int rectangle, int colStart)	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка, набір прямокутників та позиція розміщення (номер рівня та номер стовпчика).
private bool checkRectangle(int[] level, int rectangle, int colStart)	Вказує, чи можна розмістити прямокутник на обраній позиції. Вхідними даними є рівень, прямокутник та номер стовпчика.
public int[] getBuffer(int[] placement)	Повертає копію перестановки. Вхідними даними є перестановка.
public int[][] getBuffer(int[][] tap)	Повертає копію стрічки. Вхідними даними є стрічка.

Для однозначного виводу даних у різних вікнах, використовується клас OutputDataService. Опис методів класу наведений у таблиці 4.6.

Таблиця 4.6 – Методи класу OutputDataService

Назва	Опис
public void OutputRectanglesInformation(Models.Rectangle info, TextBox textPlace)	Вивід інформації про набір прямокутників. Вхідними даними є набір прямокутників та елемент виводу.
internal async Task OutputTap(Tap tap, int[][] taps, TextBox textPlace)	Вивід інформації про стрічку. Вхідними даними є стрічка та елемент виводу.

Для генерації прямокутників було створено окремий клас – RectangleService.

Розглянемо опис класу у таблиці 4.7.

Таблиця 4.7 – Методи класу RectangleService

Назва	Опис
public int**[]** GenerateRectangle**(**int minLength**,** int maxLength**,** int number**)**	Повертає згенерований набір прямокутників. Вхідними даними є найменша та найбільша довжина прямокутників та кількість прямокутників.

Клас TaboSearchService реалізує алгоритм табуйованого пошуку. У таблиці 4.8 наведено опис методів класу.

Таблиця 4.8 – Методи класу TaboSearchService

Назва	Опис
public **(**double**,** double**,** double**)** TaboSearchAlg**(**int limit**,** int**[][]** tap**,** int**[]** rectangles**)**	Реалізація алгоритму табуйованого пошуку. Вхідними даними є стрічка, кількість ітерацій та набір прямокутників. Вихідними даними є час роботи алгоритму, середнє покращення та значення цільової функції.
public string rectangleToString**(**int**[]** rectangleBuffer**)**	Конвертує набір прямокутників у рядок.
private int getMaxLength**(**int**[][]** bufferTapCurrent**)**	Повертає найдовшу довжину стрічки. Вхідними даними є стрічки та прямокутники
private int**[][]** setOnTap**(**int**[][]** bufferTap**,** int**[]** rectangleBuffer**)**	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка та набір прямокутників.
private void setRectangle**(**int**[][]** tap**,** int level**,** int rectangle**,** int colStart**)**	Повертає стрічку, на якій розміщені прямокутники. Вхідними даними є порожня стрічка, набір прямокутників та позиція розміщення (номер рівня та номер стовпчика).
private bool checkRectangle**(**int**[]** level**,** int rectangle**,** int colStart**)**	Вказує, чи можна розмістити прямокутник на обраній позиції. Вхідними даними є рівень, прямокутник та номер стовпчика.
public int**[]** getBuffer**(**int**[]** placement**)**	Повертає копію перестановки. Вхідними даними є перестановка.
public int**[][]** getBuffer**(**int**[][]** tap**)**	Повертає копію стрічки. Вхідними даними є стрічка.

Для генерації стрічки створено клас TapeService, в якому реалізовані методи для генерації та обробки стрічки. Методи наведені у таблиці 4.9.

Таблиця 4.9 – Методи класу TapeService

Назва	Опис
public int**[][]** GenerationTaps**(**int levels**,** int columns**,** int numberHole**)**	Повертає згенеровану стрічку з дірками. Вхідними даними є кількість рівнів та найбільша довжина стрічки, кількість дірок.

4.3 Засоби розробки

Для розробки інформаційної системи було обрано платформи .NET та WPF. Платформа WPF надає можливість використовувати традиційні мови .NET-платформи – C#. На відміну від WinForm, WPF – надає можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML. WPF взаємодіє з WinForm, отже маємо можливість використовувати традиційні елементи управління із WinForm. Основною перевагою WPF – можливість створювати нові елементи управління, анімації, стилі шаблони та теми.

C# – об’єктно-орієнтована мова програмування, яка має близький синтаксис до C++ та Java. C# підтримує поліморфізм, наслідування, перегрузку операторів, статичну типізацію. За рахунок об’єктно-орієнтованого підходу, маємо можливість побудувати великі та масштабовані інформаційні системи.

Платформа .NET є загальномовною середою виконання та підтримує декілька мов програмування: C#, F#, C++, VB.NET, та діалекти інших мов. Платформа є єдиною бібліотекою класів для мов програмування, які підтримує. Застосування на .NET мають перевагу у продуктивності за інші, які були реалізовані іншими технологіями.

XAML – декларативна мова розмітки, яка спрощує створення користувацького інтерфейсу. Вона забезпечує робочий процес, та дозволяє декільком розробникам працювати над користувацьким інтерфейсом та логікою програми, використовуючи різні засоби розробки.

Для роботи з даними використовується база даних MS SQL. Основними її перевагами є те, що вона є надійною БД, може розширюватися без зменшення швидкодії операцій з даними.

4.4 Опис структури бази даних

Структурна схема бази даних представлена на рисунку 4.6. На схемі встановлені зв’язки між таблицями, їх поля, первинні та зовнішні ключі. При проектуванні бази даних було визначено сутності: «Стрічки», «Прямокутники», «Експерименти», «Результати».

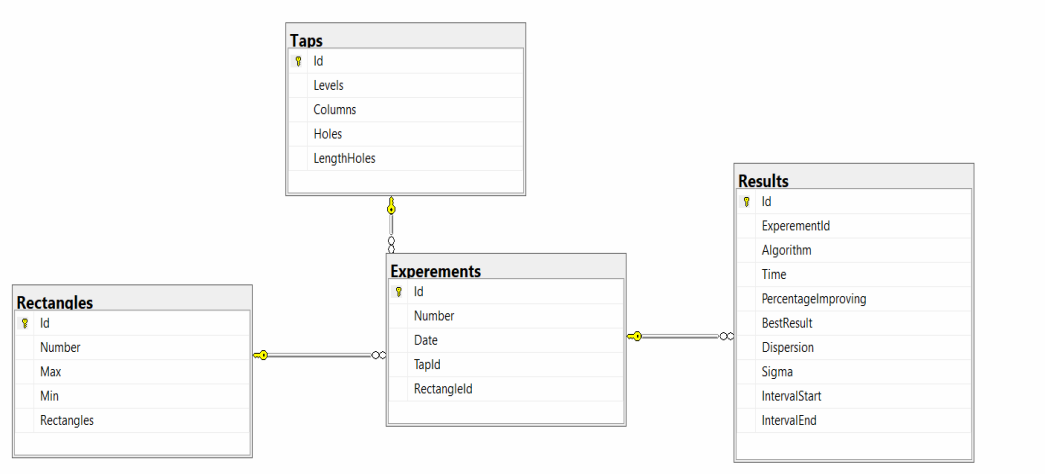


Рисунок 4.6 – Структурна схема бази даних

Розглянемо сутність «Стрічка» - описує параметри стрічки, необхідні для розв’язання задач та побудови розміщення. Наведемо наповнення опис таблиці бази даних «Taps» у таблиці 4.10.

Таблиця 4.10 – Сутність «Стрічка»

Код	Опис	Тип даних	Довжина	Не може бути NULL	Первинний ключ
Id	Унікальний ідентифікатор стрічки	uniqueidentifier		X	X
Levels	Кількість рівнів	Int			
Columns	Довжина стрічки	Int			
Holes	Кількість дірок на кожному рівні	Int			
LengthHoles	Максимальна довжина дірки	Int			

Розглянемо сутність «Прямокутники» у таблиці 4.11, яка описує параметри

прямокутників, які генеруються та приклад згенерованих прямокутників. Основні властивості:

- кількість прямокутників;
- найбільша довжина прямокутника;
- найменша довжина прямокутника.

Таблиця 4.11 – Сутність «Прямокутники

Код	Опис	Тип даних	Довжина	Не може бути NULL	Первинний ключ
Id	Унікальний ідентифікатор набору прямокутників	uniqueidentifier		X	X
Number	Кількість прямокутників	Int			
Max	Найбільша довжина прямокутників	Int			
Min	Найменша довжина прямокутника	Int			
Rectangles	Набір прямокутників	Varchar			

Розглянемо сутність «Експеримент» у таблиці 4.12, яка описує параметри експерименту.

Таблиця 4.12 – Сутність «Експеримент»

Код	Опис	Тип даних	Довжина	Not NULL	Первинний ключ
Id	Унікальний ідентифікатор експерименту	uniqueidentifier		X	X
Number	Кількість тестових задач	Int			
Date	Дата експерименту	DateTime			
TapId	Ідентифікатор стрічки	uniqueidentifier			
RectanglesId	Ідентифікатор набору прямокутників	uniqueidentifier			

Для аналізу результатів експериментів використовують сутність «Результат», яка описує отримані результати. Основні властивості:

- параметри експерименту;
- обраний алгоритм;
- час роботи алгоритму;
- дисперсія;
- середньоквадратичне відхилення;
- довірчий інтервал;
- середнє покращення функції;
- найкращий результат.

У таблиці 4.13 наведемо опис сутності «Результат».

Таблиця 4.13 – Сутність «Результат»

Код	Опис	Тип даних	Довжина	Не може бути NULL	Первинний ключ
Id	Унікальний ідентифікатор експерименту	uniqueidentifier		X	X
Number	Кількість тестових задач	Int			
Date	Дата експерименту	DateTime			
TapId	Ідентифікатор стрічки	uniqueidentifier			
RectanglesId	Ідентифікатор набору прямокутників	uniqueidentifier			

4.5 Опис взаємодії з програмним забезпеченням

В цьому розділі наведено керівництво користувача, описано основні дії із системою та наведено можливі користувацькі помилки та варіанти їх усунення.

На головній сторінці застосування на рисунку 4.7 представлена інформація про автора і роботу, та наведено основні пункти меню.

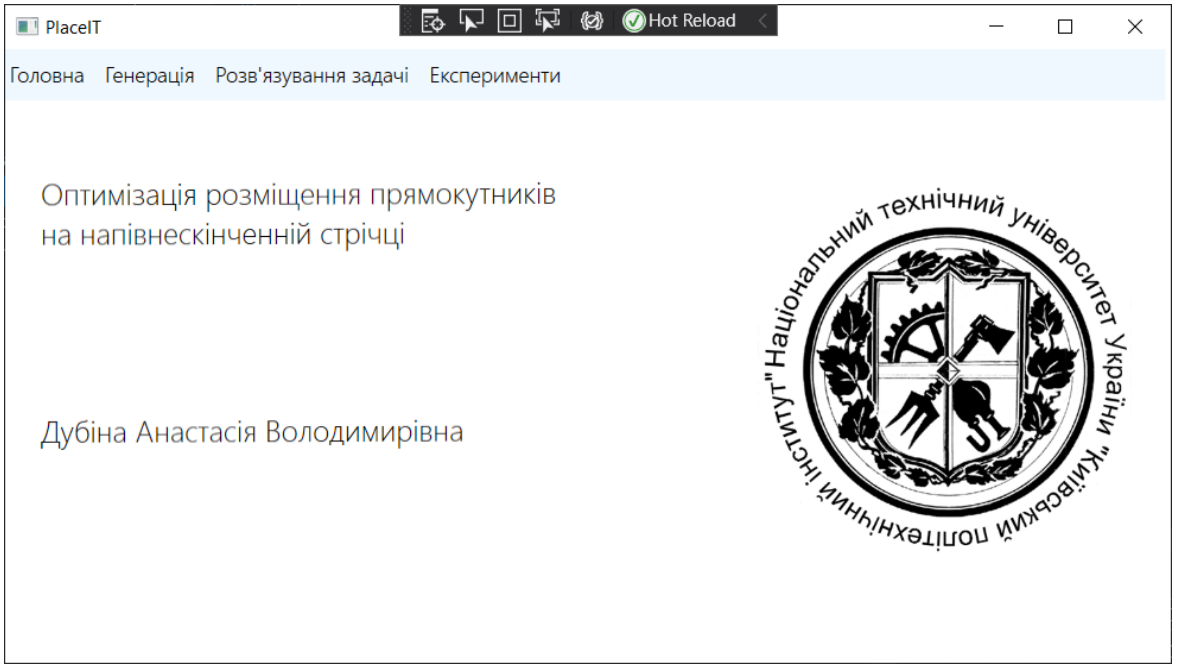


Рисунок. 4.7 – Головна сторінка програми

Для того, щоб отримати згенерувати приклади наборів прямокутників, стрічок та задач, необхідно натиснути на пункт меню «Генерація», як подано на рисунку 4.8, а потім обрати необхідний підпункт.

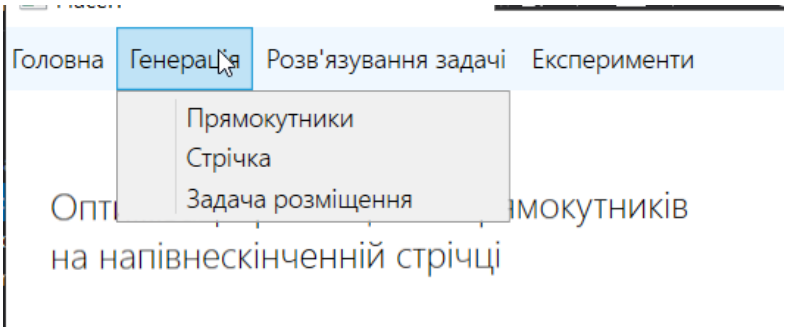


Рисунок 4.8 – Підпункти меню «Генерація»

Для генерації набору прямокутників необхідно обрати підпункт «Прямокутники». Після того, як користувач натисне на цю кнопку – відкриється нове вікно, як представлене на рисунку 4.9.

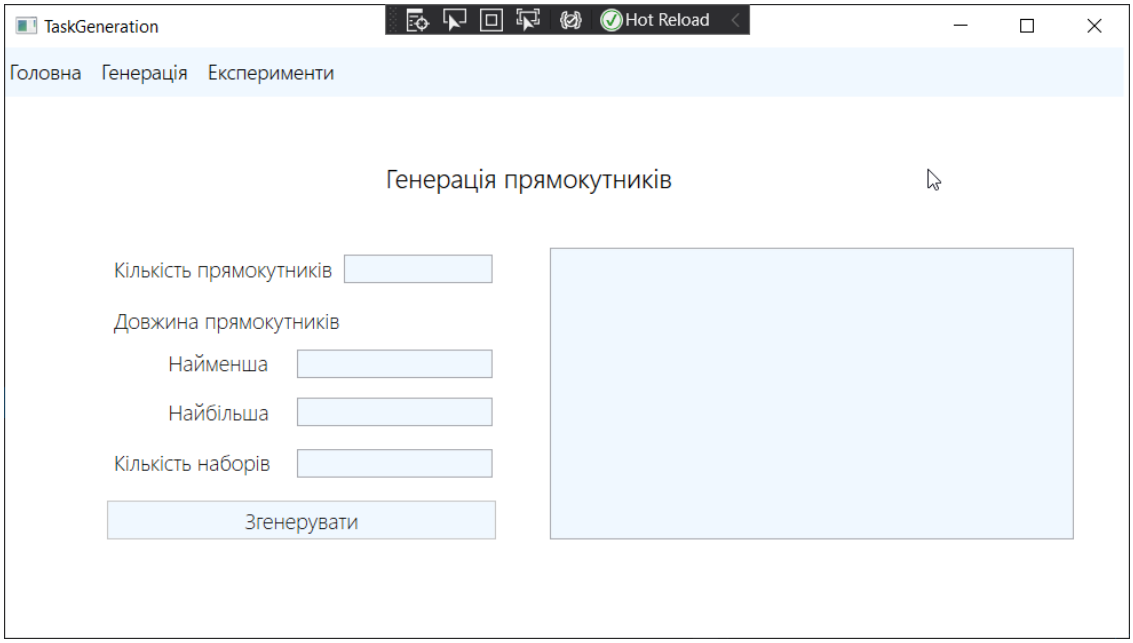


Рисунок 4.9 – Сторінка генерації прямокутників

У поля «Кількість прямокутників», «Найменша», «Найбільша» та «Кількість наборів» необхідно ввести натуральні числа. Введене значення у поле «Найменша» повинно бути строго меншим за значення, яке введено у поле «Найбільше». Для генерації прямокутників необхідно натиснути кнопку «Згенерувати». У випадку, якщо користувач введе не коректні дані або залишить порожніми поля, то у полі, що подано на рисунку 4.10, виведеться повідомлення про помилку. У випадку, якщо дані будуть введені правильно – користувач отримає згенеровані набори прямокутників (номер набору, кількість прямокутників та перелік довжин прямокутників).

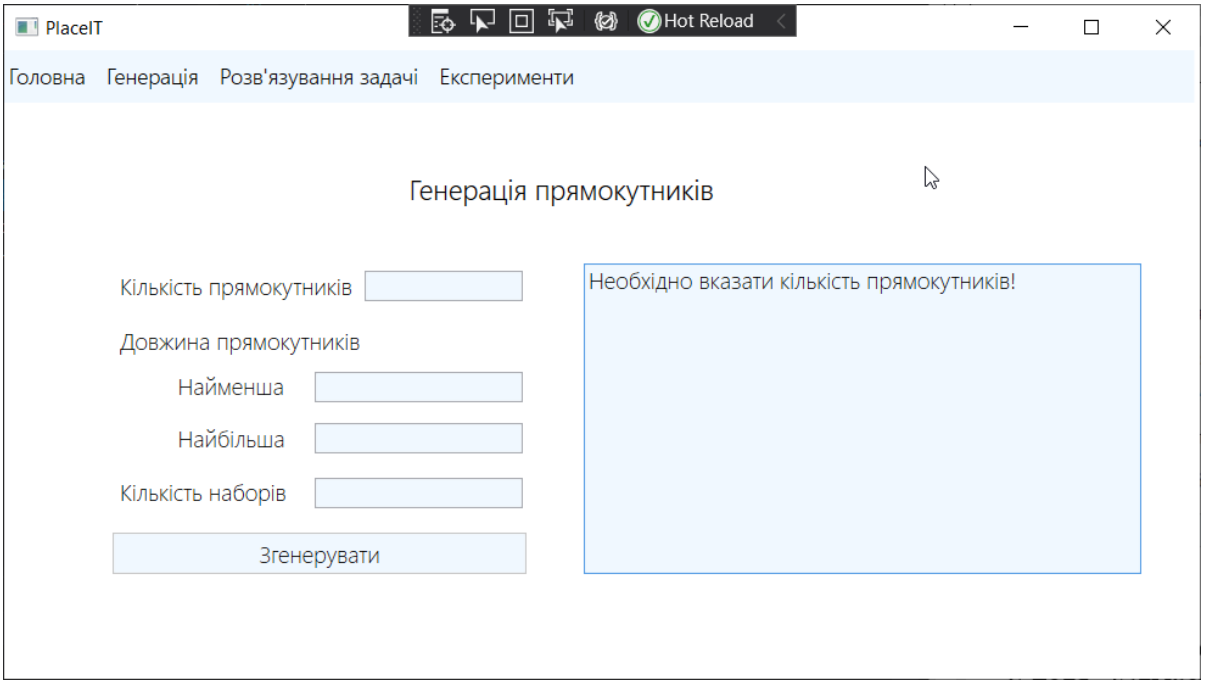


Рисунок 4.10 – Повідомлення про помилку під час генерації прямокутників

Для генерації стрічки, необхідно обрати у меню «Генерація» підпункт «Стрічка». У відкритому вікні, яке представлене на рисунку 4.11, необхідно у поля ввести натуральні числа. Для генерації стрічки необхідно натиснути кнопку «Згенерувати». Якщо дані були введено коректні – користувач отримає інформацію про стрічку (кількість рівні, дірок, та малюнок стрічки), інакше – отримає повідомлення про помилку.

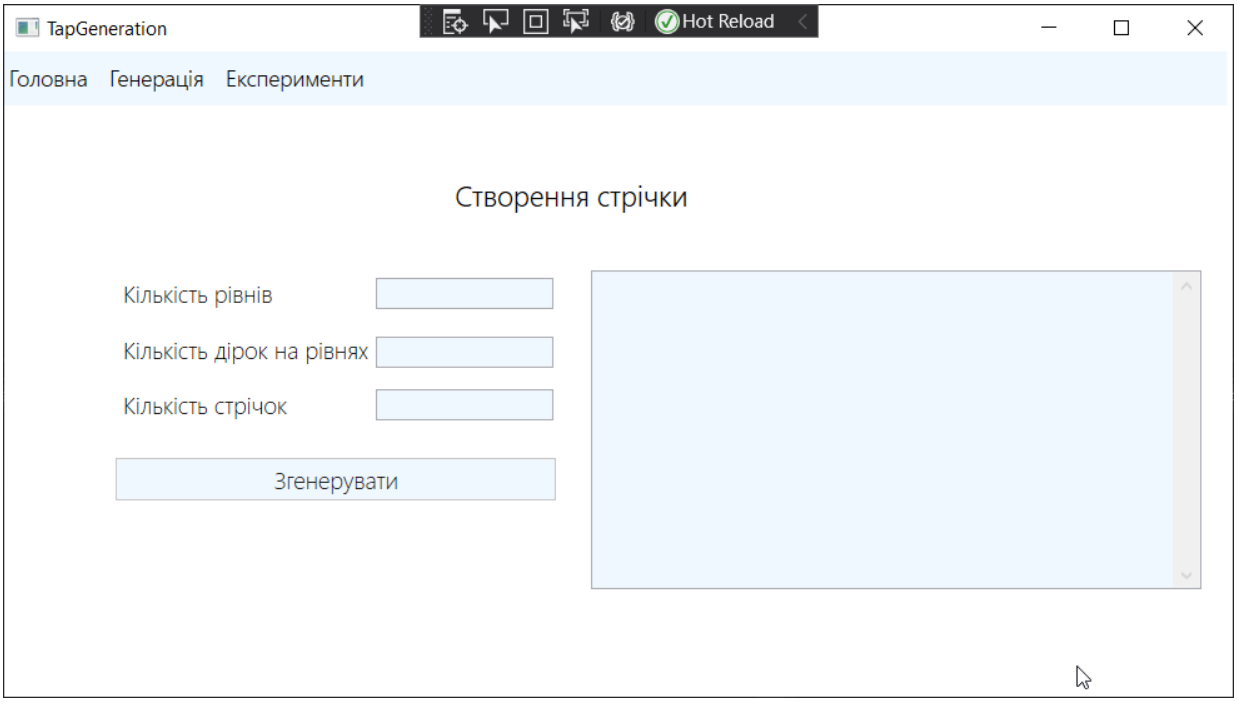


Рисунок 4.11 – Вікно створення стрічки

Для того, щоб згенерувати приклад тестової задачі необхідно в меню обрати «Генерація» та «Задача розміщення». У новому вікні, яке представлене на рисунку 4.12, необхідно вказати параметри прямокутників та стрічки. Всі вимоги, які були раніше наведені для вхідних даних, актуальні і для цього вікна. Для того, щоб згенерувати задачу, необхідно натиснути на кнопку «Згенерувати». Якщо дані будуть введено правильно – користувач отримає набір прямокутників та стрічку.

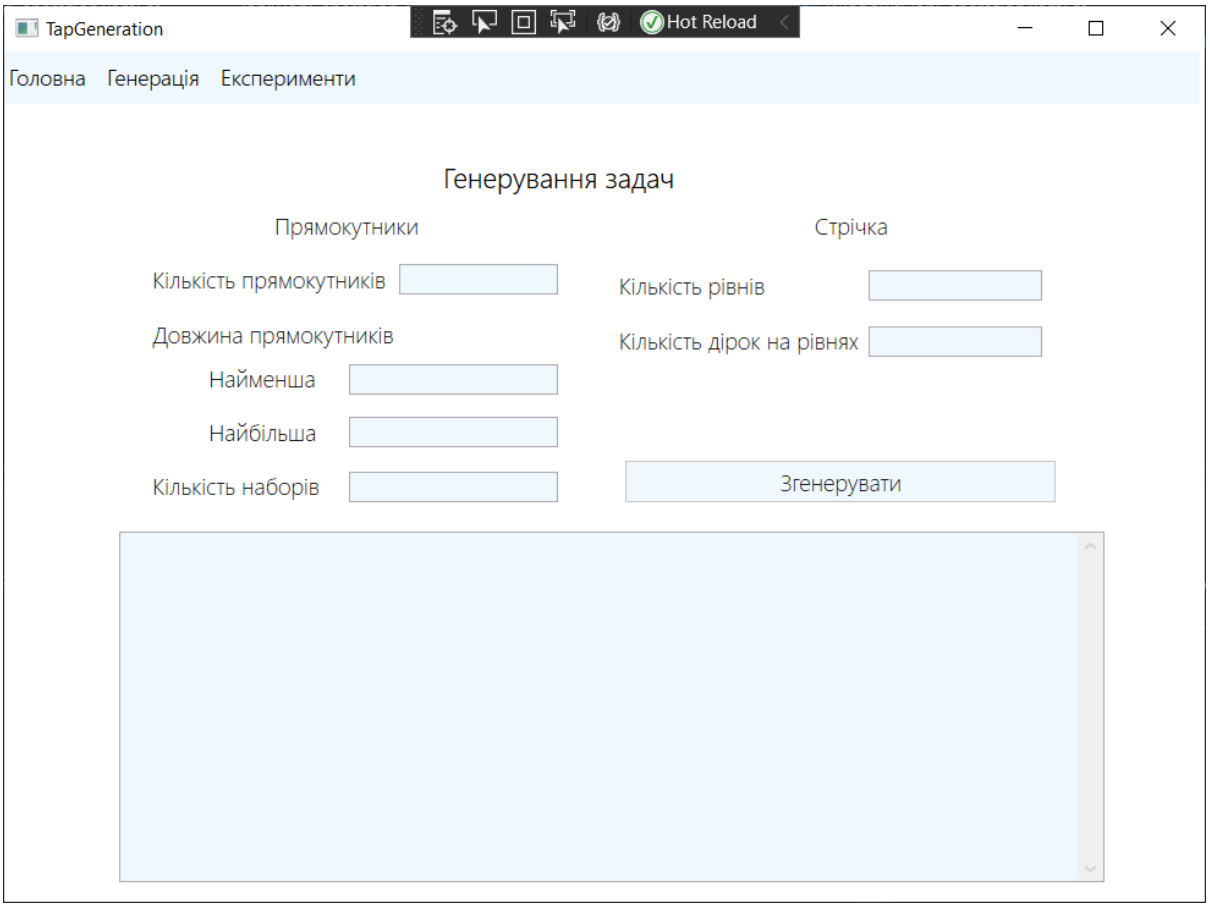


Рисунок 4.12 – Вікно створення тестової задачі

Для того, щоб розв’язати задачу необхідно обрати пункт меню «Розв’язування задачі». У вікні, яке з’явиться, на рисунку 4.13 необхідно ввести дані для створення стрічки. Якщо користувач бажає ввести власні довжини прямокутники – він може ввести їх через пробіл, кому у полі. Якщо користувач бажає, щоб система згенерувала довжини, необхідно обрати «згенерувати прямокутники» та вказати їх кількість у полі. Для розв’язання задачі необхідно обрати алгоритми серед наведених та натиснути на кнопку «Розв’язати». Якщо дані були введені правильно – користувач отримає результати розв’язання задачі,

інакше повідомлення про помилку.

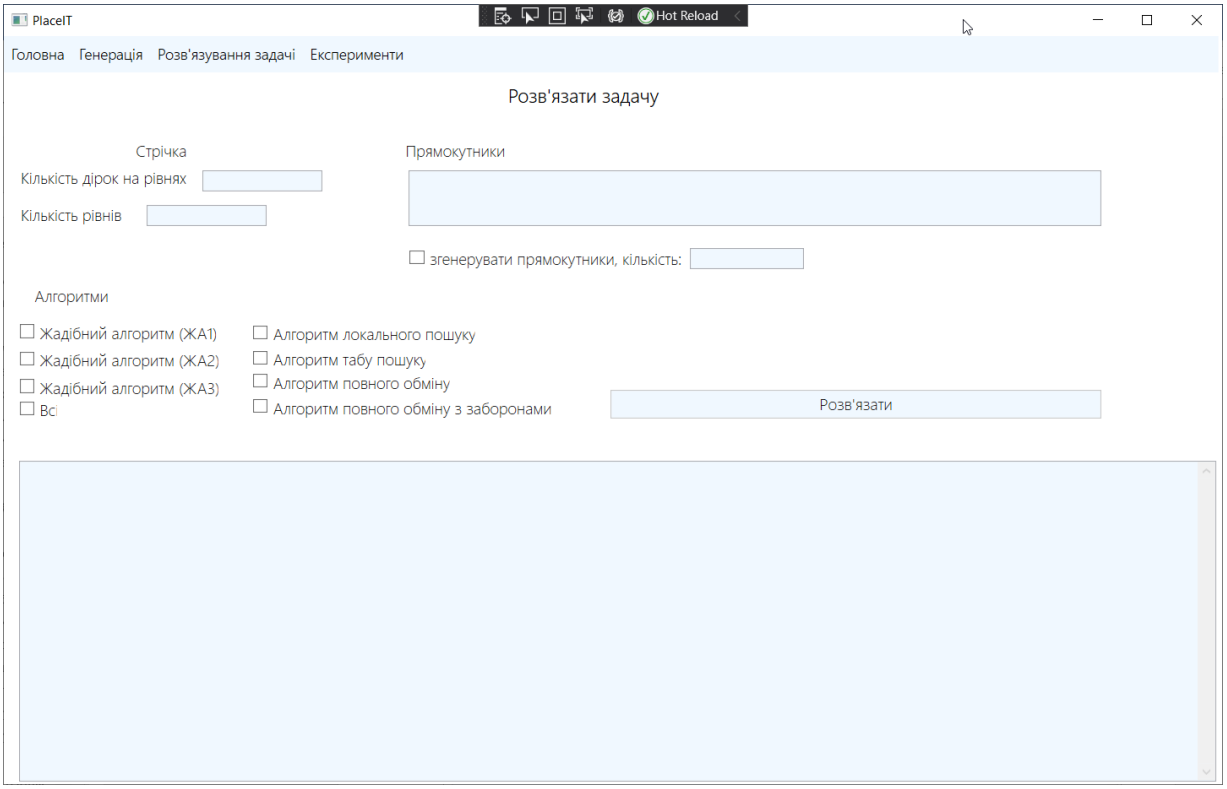


Рисунок 4.13 – Вікно для розв'язування задачі

Для проведення експериментів необхідно обрати в меню пункт «Експерименти». У новому вікні на рисунку 4.14 введення даних поділено на три блоки: Прямокутники, Стрічка та Алгоритми.
В блоці «Прямокутники» необхідно вказати найменшу та найбільшу довжини прямокутників та їх кількість. У полі «Від» необхідно вказати початкову кількість прямокутників для генерування. У полі «До» необхідно вказати кінцеву кількість прямокутників. У полі «Крок» необхідно вказати на скільки прямокутників буде більший кожен наступний набір прямокутників.

В блоці «Стрічка» необхідно вказати кількість дірок на рівні та кількість рівнів. У полі «Від» необхідно вказати початкову кількість рівнів, у полі «До» необхідно вказати кінцеву кількість рівнів, а у полі крок «Крок» необхідно вказати на скільки рівнів більше кожна наступна стрічка матиме.
В блоці «Алгоритми» необхідно обрати алгоритми для експериментів, вказати кількість тестових задач, які будуть згенеровані та у випадку, якщо будуть обрані алгоритми локального пошуку, табу пошуку чи обміну – необхідно вказати

кількість ітерацій.

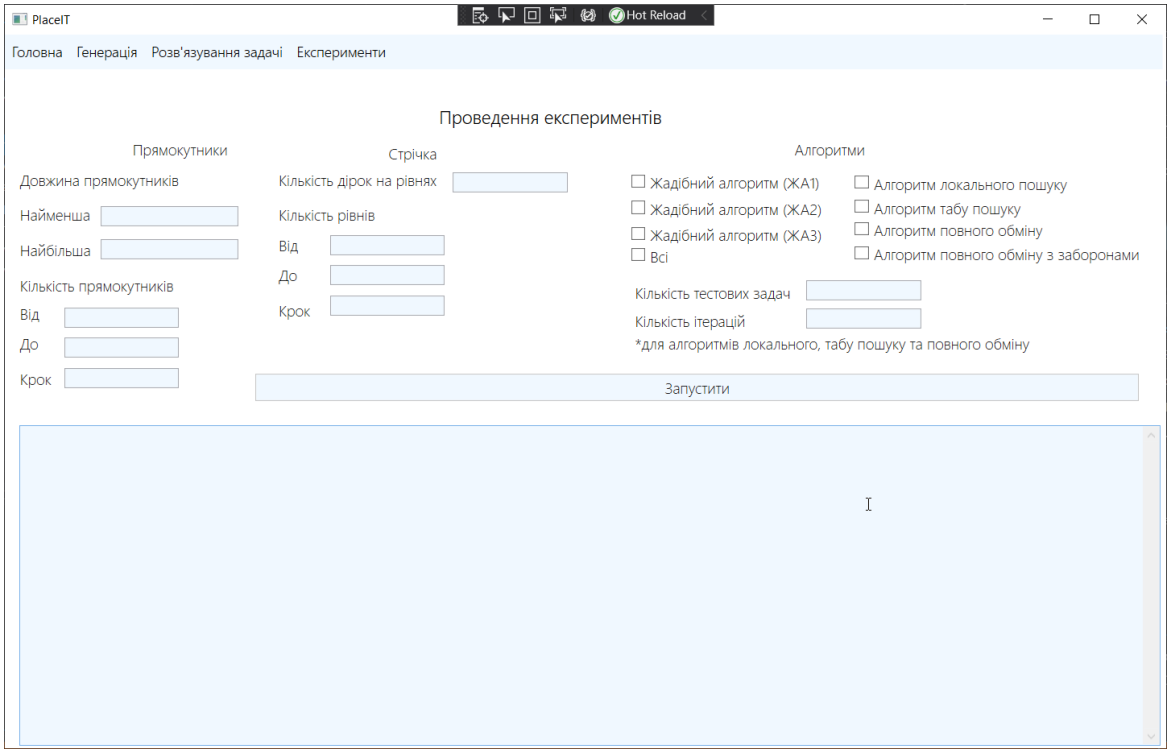


Рисунок 4.14 – Вікно проведення експериментів

Висновки до розділу

В розділі розглянуто варіанти використання системи на основі яких були сформульовані функціональні вимоги до програмного продукту. Визначено класи, склад та компоненти системи. Детально описано основні функції системи та побудовано діаграми використання, пакетів, класів та компонентів.

Визначено технічні вимоги до технічного забезпечення та сформовано перелік необхідного програмного забезпечення

Визначено структуру бази даних, наведено детальний опис таблиць та побудовано схему бази даних.

Розглянуто засоби розробки, наведено знімки розробленої системи та описано взаємодію із системою.

5 РОЗРОБКА СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Основною ідеєю проєкту є розробка і застосування інформаційної системи для розв’язування задачі розміщення прямокутників на напівнескінченній стрічці. Узагальнену опис ідей стартап-проєкту наведено у таблиці 5.1.

Таблиця 5.1– Опис ідеї проєкту

Зміст ідеї	Напрямки застосування
Надання програмного забезпечення для розв’язування задачі розміщення прямокутників на напівнескінченній стрічці.	1. Пошук оптимального розміщення прямокутників з метою зменшення використаної довжини стрічки.
	2. Оптимізація розміщення заготівок на листовому матеріалі з дефектами.
	3. Розподілення завдань між працівниками з врахуванням їх графіку.

Для того, щоб визначити потенційні техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів, необхідно сформувати перелік техніко-економічних властивостей та характеристик ідеї.

До цього переліку входять:

- вартість;
- швидкодія;
- зручність інтерфейсу;
- складність в експлуатації;
- надійність;
- швидкість розгортання;
- документованість;
- можливість роботи без інтернету;
- унікальність алгоритму.

Розглянемо аналоги, з якими далі будемо проводити порівняння. Існує велика кількість програмного забезпечення для формування оптимального розкрою.

Picaro IT. Програма розкрою, дозволяє створювати карти оптимізованого розкрою промислових матеріалів, наприклад, ДСП (при виготовленні меблів), скла або будь-яких інших листових або рулонних матеріалів.

Особливості та переваги:

- при тривалій роботі можливе накопичення залишків матеріалу, які можуть бути використані в подальшій роботі;
- можна вказати допустимий відсоток залишків;
- автоматично враховує витрати матеріалу під час розпилу, шліфування та іншу;
- формує автоматичне розміщення деталей на стрічці.

FieryCut. Програма для розкрою листового металу, формує оптимальну карту розкрою. Програму можна застосовувати для різних матеріалів.

Особливості та переваги:

- автоматичне створення контурів заготовок;
- підтримка вкладених контурів;
- розрахунок площі поверхні та маси заготовок;
- автоматичне розміщення деталей;
- враховує пріоритет розміщення;
- автоматичне створення траєкторії розкрої;
- розрахунок часу розкрою.

Cipiyc. Програма для розкрою листового матеріалу є високоефективною CAD / CAM, що об'єднує формування карти розкрою листа на довільні заготовки з генерацією оптимальних керуючих програм для верстатів з ЧПУ. Підтримується фігурний розкрій листового металу або прямокутний. Режим роботи – автоматичний і інтерактивний.

Особливості переваги:

- графічний редактор з інтерактивним проектуванням;
- формування фігурного та прямокутного розкрою матеріалу;
- оптимізація розкрою.

У таблиці 5.2 наведено результати аналізу характеристик ідей проєкту.

Таблиця 5.2 – Характеристики ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	PlaceIT	Picaro IT	FieryCut	Сірiус	W	N	S
1	Вартість	Безкоштовно	6000 грн	10000 грн	120000 грн			+
2	Швидкодія	Середня	Висока	Висока	Середня		+	
3	Зручність інтерфейсу	Висока	Висока	Висока	Середня		+	
4	Складність експлуатації	Інтуїтивно зрозуміла	Інтуїтивно зрозуміла	Високий поріг входження	Високий поріг входження			+
5	Можливість роботи без інтернету	Наявна	Відсутня	Наявна	Відсутня			+
6	Надійність	Висока	Невідомо	Середня	Висока		+	
7	Унікальність алгоритму	Є	Невідомо	Невідомо	Невідомо		+	
8	Документованість	Часткова	Часткова	Повна	Повна	+		

Програмні продукти, які існують на ринку, містять широкий функціонал для розв’язування задачі геометричного розкрою, перевантажений інтерфейс, середню надійність. Перевагами мого продукту Place IT є його простота в експлуатації та ціна.

5.2 Технологічний аудит ідеї проєкту

Проведемо аудит технології, за допомогою якої можна реалізувати ідею проєкту. Аналіз складових для визначення технологічної здійсненності проєкту подано у таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проєкту	Технології реалізації	Наявність технології	Доступність технологій
1	Зберігання розміщення	Структура даних	Необхідно розробити	Доступна
2	Використання алгоритмів розміщення прямокутників	Алгоритм розміщення	Необхідно розробити	Доступна
3	Використання алгоритмів покращення результатів	Жадібний, локального та табуйованого пошуку, обміну алгоритми	Необхідно розробити	Доступна

Оскільки всі технології для реалізації ідей доступні, ці технології необхідно розробити, проте є розуміння які сімейства алгоритмів необхідно взяти за основу для створення власних алгоритмів.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Наведемо попередню характеристику потенційного ринку стартап-проєкту (таблиця 5.4).

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку	Характеристика
1	Кількість головних гравців	1
2	Загальний обсяг продаж, грн/ум.од	Невідомий
3	Динаміка ринку (якісна оцінка)	Зростає
4	Специфічні вимоги до стандартизації та сертифікації	Відсутні
5	Середня норма рентабельності в галузі, %	52%

Ринок програмного забезпечення для розміщення прямокутників є достатньо привабливим, оскільки використовується у різних сферах виробництва. Тому існує потреба у створенні програмного продукту для оптимізації розміщення. Сформуємо орієнтований перелік вимог до товару у таблиці 5.5.

Таблиця 5.5– Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимога споживачів до товару
1	Пошук оптимального розміщення об'єктів на матеріалі	Керівники виробництва	Пошук автоматизації формування плану розкрою	Безпека даних, інтуїтивна зрозумілість, простота використання

Після визначення потенційних груп клієнтів проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проєкту, та факторів, що йому перешкоджають (таблиці 5.6-5.7).

Таблиця 5.6 – Фактори загроз

№	Фактор	Зміст загроз	Можлива реакція компанії
1	Нестабільність ринку	Нестабільна політична ситуація, війна, низький рівень економік	Вихід на європейський ринок
2	Швидкість розвитку технологій	Поява нових технологій	Періодичне впровадження нових технологій
3	Зростання конкуренції	Поява нових конкурентів	Створення оригінального функціоналу

Таблиця 5.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Інвестиції	Зацікавленість інших осіб у стартапі	Виконати вимоги інвесторів
2	Стати першопрохідцем з ідеєю	Стати універсальним застосуванням для різних підприємства	Реалізувати функціонал і стати першими на ринку
3	Розвивати ідею	Запроваджувати нові технології	Розвивати далі ідею

Визначимо загальні риси конкуренції на ринку та наведемо у таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється характеристика	Вплив на діяльність підприємства
1. Вказати тип конкуренції – чиста	Існують схожі програмні засоби, які широко направлені та досить складні у налаштуванні	Запропонувати нове рішення, якого немає на ринку
2. За рівнем конкурентної боротьби – національний	Багато фірм розробляють програми покращення формування карт розкрою	Охопити ринок України
3. За галузевою ознакою – внутрішньогалузева	Галузь ІТ	Впроваджування нових технологій
4. Конкуренція за видами товарів – між бажаннями	Існують сході програми, але для вузьких сфер	Аналізувати та впроваджувати бажання клієнтів
5. За характером конкурентних переваг – неціновa	Програмне забезпечення буде суттєво відрізнятися від звичайних програм створення карт розкрою	Підтримувати сучасний та зручний інтерфейс, впроваджувати новий функціонал
6. За інтенсивністю – не марочна	Фірма нова і невідома в Україні	Стати брендом

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Picaro IT, Сіріус	Розмір капіталовкладень	-	Наявність документації про продукт, контроль швидкодії	Є часткова заміна
Висновки	Висока інтенсивність	Є потенційні конкруренти, можоивість входу на ринок	-	Встановлюються вимоги до швидкодії	Частково перекривають функціонал програмного продукту за рахунок складних налаштувань, що не є основним функціоналом

Отже, проєкт є конкурентоспроможним, незважаючи на існування аналогів. Сильними сторонами проєкту є новий функціонал, простота використання, високий рівень автоматизації, привабливий та сучасний інтерфейс, якого немає у конкурентів.

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проєкту, вимог споживачів до товару та факторів маркетингового середовища визначимо перелік факторів конкурентоспроможності та наведемо у таблиці 5.10.

Таблиця 5.10 – Обгрунтування факторів конкурентоспроможності

№ п/п	Фактор	Обгрунтування
1	Динаміка галузі	Кожного для з’являються нові технології та стрімкий розвиток конкурентів
2	Постійні витрати	Необхідні постійні витрати на сервера, підтримка програмного продукту
3	Наявність торгівельних знаків	Мати унікальну і відому
4	Рівень автоматизації	Чим більший рівень автоматизації, тим менше затрат на підтримку та роботу

За визначеними факторами проведемо аналіз сильних та слабких сторін

стартап-проєкту у таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін «PlaceIT»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг конкурентів						
			-3	-2	-1	0	1	2	3
1	Динаміка галузі	18		+					
2	Постійні витрати	16					+		
3	Наявність торгівельних знаків	15						+	
4	Рівень автоматизації	19				+			

Далі проведемо SWOT-аналіз стартапу. Результати аналізу подано у таблиці 5.12.

Таблиця 5.12 – SWOT-аналіз стартап проєкту

Сильні сторони: низька ціна, доступний і інтуїтивний інтерфейс, задоволення потреб компаній.	Слабкі сторони: мало ресурсів та часу для створення дійсно якісного продукту.
Можливості: зацікавлення представників різних сфер діяльності.	Загрози: задоволення потреб клієнтів, агресивні дії великих конкурентів в галузі.

Розробимо альтернативи ринкової поведінки для виведення стартап-проєкту на ринок та орієнтовний оптимальних час їх ринкової реалізації з огляну на потенційні проєкти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів наведено у таблиці 5.13.

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива	Ймовірність отримання ресурсів	Строки реалізації
1	Стратегія спеціалізації	90%	1.4 роки
2	Стратегія диференціації	60%	1.5 роки
3	Позиціонування за співвідношенням «ціна-якість»	93%	1 рік
4	Стратегія зайняття конкурентної ніші	60%	1 рік

Найкращим варіантом буде стратегія позиціонування за співвідношенням «ціна-якість», оскільки ймовірність отримання ресурсів та строки реалізації є найкращими.

5.4 Розроблення ринкової стратегії проєкту

Для початку необхідно визначити опис цільових груп потенційних споживачів та надамо у таблиці 5.14.

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю	Готовність сприйняти продукт	Орієнтований попит	Інтенсивність конкуренції (від 1 до 10)	Простота входу в сегмент (від 1 до 10)
1	Працівники підприємств	+	100%	9	3

Використовуватиметься стратегія масового маркетингу, для обраних груп, оскільки вони зацікавлені у створюваному продукті.

Сформуємо базову стратегію розвитку для роботи в обраних сегментах ринку та наведемо у таблиці 5.15

Таблиця 5.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
1	Працюємо із всім ринком	Стратегія масового маркетингу	Високий рівень автоматизації, задоволення потреб клієнтів і реалізацію різного функціоналу, абсолютно новий функціонал, якого ще немає на ринку, зручність інтерфейсу	Стратегія диференціації

Наступним кроком визначаємо стратегії конкурентної поведінки та наведемо у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «Першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів або забирати існуючих конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так (в Україні)	Так	Ні	Стратегія лідера

На основі вимог споживачів з обраних сегментів до стартап-компанії та продукту, та в залежності від обраної базової стратегії розвитку і конкурентної поведінки розробимо стратегію позиціонування та представимо у таблиці 5.17.

Таблиця 5.17 – Стратегія позиціонування

№ п/п	Вимоги до товару цільової авдиторії	Базова стратегія розвитку	Ключові конкурентоспроможності позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію проєкту
1	Якість, зручний інтерфейс, простота у використанні, новий функціонал: формування розміщення різними алгоритмам	Стратегія диференціації	Задоволення потреб клієнтів, реалізація нового функціоналу, високий рівень автоматизації, зручність інтерфейсу	Зручність інтерфейсу, високий рівень автоматизації, задоволення потреб клієнтів

Для проєкту необхідна стратегія масового маркетингу. Як базову стратегію розвитку обирається стратегія диференціації, оскільки програмний продукт містить багато нового функціоналу, що відповідає вимогам споживачів. За стратегію конкурентної поведінки обрана стратегія лідерства, оскільки розглянуті аналоги не покривають всі сфери, тому буде можливість стати лідером.

5.5 Розроблення маркетингової програми стартап-проєкту

Сформуємо маркетингову концепцію продукту, який отримає споживач. Для цього необхідно підсумувати результати попередніх аналізів конкурентоспроможності товару. У таблиці 5.18 визначимо ключові переваги концепції потенційного товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Формування оптимального розміщення деталей на матеріалах	Економія ресурсів та зменшення залишків	Співвідношення якість-ціна, врахування потреб підприємства

Наведемо трьохрівневу маркетингову модель товару у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система оптимального розміщення прямокутників «PlaceIT»		
II. Товар у реальному виконанні	Властивості та характеристики	М/Нм	Вр/Тх/Тл/Е/Щр
	1. Якість 2. Простота у використання 3. Видача карт розкрою 4. Низька ціна	-	-
	Якість: тестування за документацією Пакування: відсутнє Марка: DreamTeamInc “PlaceIT”		
III. Товар із підкріпленням	До продажу: гарантія якості Після продажу: підтримка		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, а також використання технології обфускації програмного коду			

Далі необхідно визначити межі встановлення ціни (таблиця 5.20).

Таблиця 5.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари замінники	Рівень цін на товари аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення цін на послугу/товар
1	1000$	6000$	1.2 млрд $	600-6000

Далі необхідно визначити оптимальний канал збуту (таблиця 5.21)

Таблиця 5.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	5 одиниць на підприємство	Домовлятися з керівниками підприємств про вимоги та ціну	Перший рівень	Власні сили та через посередників

Останнім етапом є формування маркетингової комунікації (таблиця 5.22).

Таблиця 5.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій	Ключові позиції	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Орієнтовані на зручність та якість	Контент-маркетинг, реклама	Простота використання, ціна, новизна та якість	Показати переваги продукту та заохотити користуватися ним	Показати зручність продукції

Результатом підрозділу є маркетингова програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення. Щоб заохотити споживачів користуватися продуктом буде задіяна реклама, основна мета якої буде звернення уваги на переваги продукту.

Висновки до розділу

Проведено аналіз програмного продукту в якості стартап-проєкту. Існує потреба у створенні даного програмного продукту, оскільки наявний попит, а проєкт є рентабельним. На ринку наявна конкуренція, оскільки існують фірми конкуренти, тому вихід не буде легким. За рахунок того, що проєкт є програмним, його розробка не потребує додаткових витрат на різноманітне обладнання та матеріали. Проєкт має багато сильних сторін, які переважають над існуючими аналогами. Реалізуємо проєкт, шукаємо інвесторів, ведемо стратегію масового маркетингу, диференціації. Проєкт рентабельний і конкурентоспроможний, тож є цілком доцільним його подальше впровадження.

ВИСНОВКИ

У магістерській дисертації проведено аналіз постановок та підходів до розв’язування задач розміщення прямокутників на напівнескінченній стрічці із забороненими областями.

У роботі виконані наступні завдання:

– досліджено класифікацію задач розміщення;

– досліджено існуючі методи розв’язування задач розміщення прямокутників, виявлено їх переваги та недоліки;

– запропоновано 7 алгоритмів розв’язування задач розміщення прямокутників, які дозволяють оптимізувати розміщення прямокутників на напівнескінченній стрічці із врахуванням заборонених областей: жадібний алгоритм та його модифікації, алгоритм локального пошуку, алгоритм табуйованого пошуку, алгоритм обміну та алгоритм обміну із заборонами.

Розроблена і програмно реалізована система для розв’язування сформульованої задачі.

Експериментально досліджено ефективність запропонованих алгоритмів та визначено переваги і недоліки кожного з них. Встановлено, що результати, отримані алгоритмом табуйованого пошуку кращі за результати, отримані іншими алгоритмами.

За матеріалами дисертації було опубліковано 5 наукових робіт: 1 стаття та 4 тез доповідей на конференції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Dyckhoff, H. A Typology of cutting and packing problems / H. Dyckoff // European J. of Operational Research. 1990. Vol. 44. P. 145-159.

2. Wascher, G. An improved typology of cutting and packing problems / G.Wascher, H. Haussner, H. Schumann // Working Paper at Faculty of Economics and Management. Magdeburg : Otto von Guericke University. – 2005. – № 24. – P. 1109-1130

3. Beasley, J. E. An exact two-dimensional nonguillotine cutting tree search procedure / J. E. Beasley // Operational Research. 1985. Vol. 33. P. 49-64.

4. Martello, S. Exact solution of two-dimensional finite bin packing problem / S.Martello , D. Vigo // Managment Science. 1997. Vol. 35. P. 64-68.

5. Липовецкий, А. И. К оптимизации свободного размещения прямоугольников / А. И. Липовецкий // Автоматическое проектирование в машиностроении. Минск : ИТК АН БССР. – 1985. – С. 80-87.

6. Бухвалова, В.В. Задачи прямоугольного раскроя: метод зон и другие алгоритмы / В.В. Бухвалова. – СПб. : СПбГУ, 2001. – 96 с

7. Fekete, S.P. A combinatorial characterization of higher dimensional orthogonal packing / S.P. Fekete, J. Schepers // Mathematics of Operations Research. – 2004. – Vol. 29. – P. 353–368.

8. Чуб І.А, Новожилова М.В., Андронов В.А. Моделювання прикладних оптимізаційних задач розміщення об’єктів з метричними характеристиками, що змінюються: монографія / Чуб І.А., Новожилова М.В., Андронов В.А. – Харків: НУЦЗ України, 2017. – 167 с.

9. Канторович Л. В. Рациональный раскрой промышленных материалов / Л. В. Канторович, В. А. Залгаллер. – Новосибирск: Наука, 1971. – 300 с.

10. Марков В. Н. Критерии эффективности методов решения задачи раскрояупаковки плоских материалов / В. Н. Марков, Е. А. Руденко. // Научные труды КубГТУ. – 2014. – №6. – С. 316–322.

11. Dyckhoff, H. Cutting and Packing in Production and Distributing / H.

Dyckhoff, U. Finke. – Heidelberg : Physica Verlag, 1992. – 248 p

12. Косолап А. І. Ефективний метод оптимізації в задачах лінійного розкрою матеріалів / А. І. Косолап, Г. М. Кодола // Математичне моделювання. - 2018. – № 1. – С. 12–21.

13. Тарасов А. Е. Решение задачи одномерного раскроя материала различных длин на базе гибридизации эволюционных алгоритмов / А. Е. Тарасов. // Вестник УГАТУ. – 2007. – №4. – С. 111–115.

14. Балабанов В. Н. Эволюционный алгоритм решения задачи рационального раскроя рулонного материала / В. Н. Балабанов, Ю. А. Скобцов. // Известия ЮФУ. технические науки. – 2014. – №1. – С. 44–55.

15. Бухвалова, В.В. Задачи прямоугольного раскроя: метод зон и другие алгоритмы / В.В. Бухвалова. – СПб. : СПбГУ, 2001. – 96 с

16. Ермаченко, А. И. Рекурсивный метод для решения задачи гильотинного раскроя / А. И. Ермаченко, Т. М. Сиразетдинов // Принятие решений в условиях неопределенности : сб. науч. ст. – Уфа : Изд-во УГАТУ, 2000. – С. 35–39

17. Мухачева, А.С. Генетический алгоритм поиска минимума в задачах двумерного гильотинного раскроя / А.С. Мухачева, А.В. Чиглинцев // Информационные технологии. – 2001. – № 3. – С. 27–31

18. Мухачева Э.А. Модифицированный метод ветвей и границ: алгоритм и численный эксперимент для задачи одномерного раскроя/ Э.А. Мухачева,. В.М. Картак // Информационные технологии. – 2000. – № 9. – С. 15–22.

19. Исследование бизнес-процесса учета делового остатка при раскрое листовых материалов / Р. А.Файзрахманов, Р. Т. Мурзакаев, В. С. Шилов, А. В. Буркова. // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2013. – №7. – С. 143–148.

20. Юдаков П. В. Задача о трехмерной упаковке и методы ее решения.

Обзор / П. В. Юдаков. // Инженерный вестник. – 2015. – №6. – С. 552–581.

21. Потарусов Р. В. Проблема одномерной упаковки элементов / Р. В. Потарусов, В. М. Курейчик. // Известия ТРТУ. – 2006. – №6. – С. 88–93.

22. Мухлаева И.В. Решение задачи одномерной упаковки с помощью параллельного генетического алгоритма / И.В. Мухлаева // Перспективные информационные технологии и интеллектуальные системы.- 2000. - № 1. -С. 77 - 84.

23. Ванидовский В. А. Двумерная упаковка в полуограниченную полосу на основе моделирования адаптивного поведения муравьиной колонии / В. А. Ванидовский, О. В. Лебедев. // Известия южного федерального университета. технические науки. – 2014. – №7. – С. 34–42.

24. Introduction to Algorithms / T. H.Cormen, C. E. Leiserson, R. L. Rivest, C. Stein., 2009. – 1328 p.

25. Чеканин В.А., А.В. Чеканин. Модели конструирования ортогональной упаковки объектов / В.А.Чеканин, А.В. Чеканин // Информационные технологии и вычислительные системы. – № 2.– 2014. – С. 37-45

26. Пермякова Т.Л., Морозенко В.В. Комбинированный метод решения задачи о рюкзаке // Proceeding of the XIII-th Int. Conference «Knowledge-DialogueSolution» KDS 2007, Varna, V.1, pp.195–202

27. Картак В. М. Параллельный подход к решению задачи одномерной продолженной упаковки (1cbpp) с использованием технологии CUDA / В. М. Картак, А. В. Рипатти. // Вестник Башкирского университета.. – 2013. – №18. – С. 11–14.

28. Гладков Л. А. Решение задачи трехмерной упаковки разногабаритных объектов с использованием бионических методов / Л. А. Гладков, Н. В. Гладкова, Е. С. Скубриева. // Известия ЮФУ. Технические науки. – 2013. – №7. – С. 35–41.

29. Bortfeldt A. A parallel tabu search algorithm for solving the container loading problem / A. Bortfeldt, H. Gehring, D. Mack. // Parallel Comput. – 2003. – №29. – pp. 641–662.

30. Faroe O. Guided local search for the three-dimensional bin-packing problem. / O. Faroe, D. Pisinger, M. Zachariasen. // Informs journal on computing. – 2003. – №15. – С. 267–283.

31. Maarouf W. F. A New Heuristic Algorithm for the 3D Bin Packing Problem. / W. F. Maarouf, A. M. Barbar, M. J. Owayjan. // Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering. – 2008. – pp. 342–345.

32. Курейчик В.М. Алгоритмы одномерной упаковки элементов // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 8-11.

33. Юсупова Н. И. Об одной классификации задач составления расписаний / Н. И. Юсупова, О. Н. Сметанина, В. В. Никитин. // Вестник уфимского государственного авиационного технического университета. – 200. – №4. – С. 85–89.

34. Pinedo M. L. Theory, Algorithms, and Systems / Pinedo., 2012. – 676 p.

35. Liao C.J. Makespan Minimization for Multiple Uniform Machines / C.J. Liao, C.H. Lin // Computers. & Industrial Engineering. – 2008 – v. 54. pp. 983-992

36. Diana S. An Improved Genetic Algorithm for Resource Constrained Project Scheduling Problem / S. Diana, A. K. Pundir. // International Journal of Computer Applications. – 2013. – №78. – pp. 34–39.

37. Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling / Hartmann. // Naval Research Logistics. – 1998. – №45. – pp. 733–750.

38. Lazarev A. Solution algorithms for the total tardiness scheduling problem on a single machine / A. Lazarev, A. Kvaratskhelia, A. Tchernykh. // Workshop Proceedings of the ENC'04 International Conference. – 2004. – pp. 474–480.

39. Cheng T. Hybrid Algorithm for the Single Machine Total Tardiness. Problem / T. Cheng, A. Lazarev, E. Gafarov. // Computers & Operations Research. – 2009. – №36. – pp. 308–315.

40. Власов В. С. Метод ветвей и границ с эвристическими оценками для конвейерной задачи теории расписаний / В.С. Власов, М. Х. Прилуцкий // Вестник

Нижегородского университета им. Н. И. Лобачевского. - 2008. -№3. - С. 147-153.

41. Комяк В. М., Мунтян В. К. Постановка задачи оптимизации размещения пунктов наблюдения наземных систем видео-мониторинга лесных пожаров // Проблемы пожарной безопасности. Харьков : НУГЗУ. 2012. Вып 31. С.80–84.

42. Филиппова А. С. Нализ эффективности алгоритмов оптимального использования ресурсов на примере задач геометрического размещения и раскроя. / А. С. Филиппова, Э. И. Дяминова. // T & Transport. – 2015. – С. 26–35.

43. Филиппова, А. С. Задачи о минимальном покрытии ортогональных многоугольников с запретными участками / А. С. Филиппова, В. Ю. Кузнецов // М. : Информационные технологии. – 2008. – № 9. – С. 60–65

44. Гуляницкий Л.Ф., Туринский В.В. Математическая модель одного класса задач планирования работы независимых машин // Компьютерная математика. – 2014. – № 1. – С. 113 –118.

45. Васильев Г. Н. Автоматизация проектирования металлорежущих станков / Г. Н. Васильев. – Москва: Машиностроение, 1987. – 280 с.

46. Математическая постановка задачи синтеза компоновочной схемы базовых несущих конструкций / А.С. Кондрашов, В.И. Шелест // Технология и конструирование в электронной аппаратуре. — 2003. — № 2. — С. 11-14

47. Пластинин В. В. Задача автоматизированного размещения деталей одежды на кожевенном полуфабрикате и пути ее решения / В. В. Пластинин, И. И. Шапмина, О. А. Рашева. // Омский научный вестник. – 2003. – №1. – С. 137–139.

48. Чуб И. А. Математическая модель оптимизационной задачи размещения пожароопасных объектов с учетом рельефа области размещения / И. А. Чуб // Радіоелектроніка, інформатика, управління. - 2013. - № 1. - С. 88-93

49. Чуб І. А. Зниження рівня забруднення повітря викидами пожежі шляхом оптимального розміщення пожежонебезпечних об'єктів / І. А. Чуб // Науковий вісник НЛТУ України. - 2017. - Вип. 27(1). - С. 203-205.

50. Мартишин, С. А. Упаковка прямоугольников в полосу

модифицированным методом Нелдера-Мида с использованием генетического алгоритма / С. А.Мартишин, М. В. Храпченко // Труды Института системного программирования РАН. - 2010. - Т. 19. - С. 135-156.

51. Parra E. Practical Solution to Parallel Machine Scheduling Problems. / Parra // ptimization and Decision Science: Methodologies and Applications. / Parra. – Cham: Springer, 2017. – (Springer Proceedings in Mathematics & Statistics). – pp. 621–628.

52. Лазарев А. А. Теория расписаний. Задачи и алгоритмы. / А. А. Лазарев, Е. Р. Гафаров. – Москва: МГУ им. М. В. Ломоносова, 2011. – 222 с.

53. Конвей Р. В. Теория расписаний / Р. В. Конвей, В. Л. Максвелл, Л. В. Миллер. – Москва: Наука, 1975. – 360 с.

54. Галковська Л. О. Алгоритми розв'язання розподіленої задачі задоволення обмежень / Галковська Л.О. // Наукові записки НаУКМА. - 2013. - Т. 151 : Комп'ютерні науки. - С. 139-148.

55. Жиглявский А. А. Методы поиска глобального экстремума / А. А. Жиглявский, А. Г. Жилинскас. – Москва: Наука, 1991. – 247 с.

56. Кутянська В. І. Розв'язання однієї задачі розкрою алгоритмом оптимізації бджолиною колонією / В. І. Кутянська, Б. Г. Шаров. // Науковий вісник НЛТУ України. – 2010. – №8. – С. 290–294.

57. Штовба С.Д., Рудий О.М. Мурашині алгоритми оптимізації // Вісник ВПІ. – 2004. – № 4. – С. 62–69

58. Гуляницький Л.Ф., Мулеса О.Ю. До класифікації метаевристик // Пр. XXI Всеукр. наук. конф. "Сучасні проблеми прикладної математики та інформатики" (Україна, м. Львів, 24– 25 вересня 2015 р.). – Львів: Львівський національний університет ім.І.Франка. – 2015. – С. 139-142

59. Алексеев О. Г. Комплексное применение методов дискретной оптимизации / О. Г. Алексеев. – Москва: Наука, 1987. – 248 с.

60. Гафаров Е. Р. . Гибридный алгоритм решения задачи минимизации суммарного запаздывания для одного прибора / Е. Р. Гафаров. // Информационные технологии. – 2007. – №1. – С. 30–37.

61. Алгоритмы: построение и анализ, 2-е издание / Т.Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – Киев: Издательский дом «Вильямс», 2011. – 1296 с.

62. Fuentes A. Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic / A. Fuentes, G. Erick. // PloS one. – 2018. – №13

63. Song T. An iterated local search algorithm for the University Course Timetabling Problem / Song T. // Applied Soft Computing. – 2018. – №68.

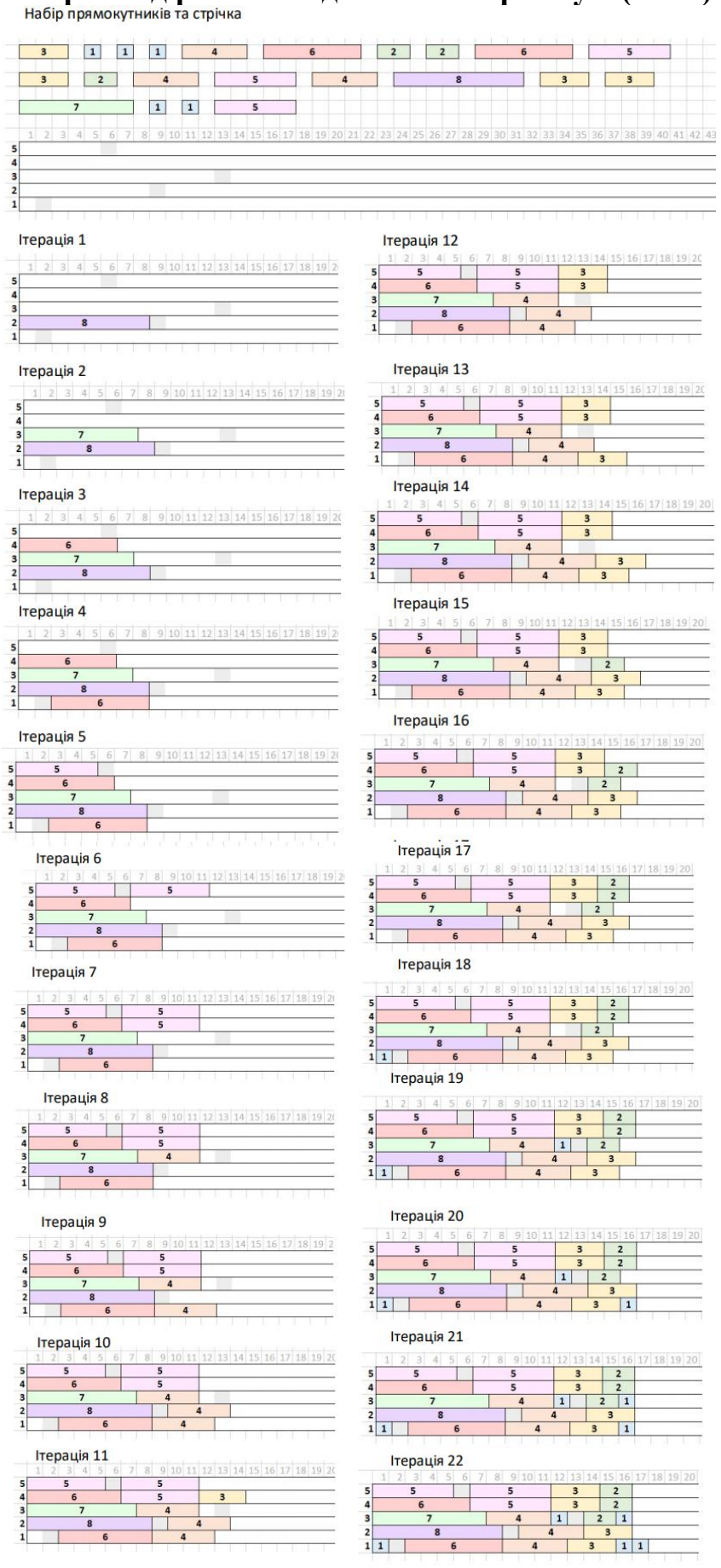
64. Karaboga, D. A comprehensive survey: artificial bee colony (ABC) algorithm and applications / Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. // Artificial Intelligence Review, - 2014, 42(1), С. 21-57.

65. Дубіна, А. В. Алгоритми локального пошуку та табу-пошуку для задач розміщення прямокутників/ Дубіна А.В., Гуляницький Л.Ф. // Наукове забезпечення технологічного прогресу XXI сторіччя: матеріали міжнародної наукової конференції (Т.2), 1 травня, 2020 рік. Чернівці, Україна: МЦНД, 2020. С. 48-53.

ДОДАТОК А

Графічний матеріал

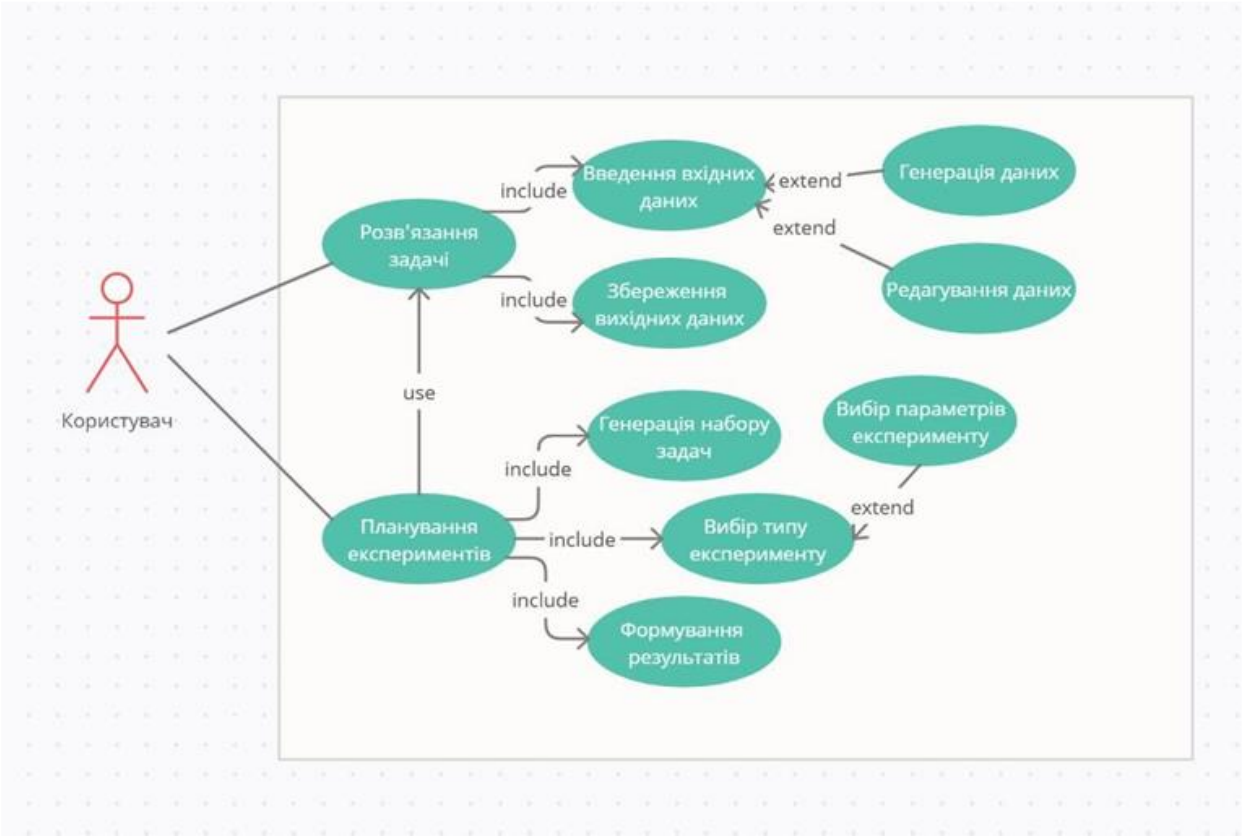
Плакат 1 Приклад роботи жадібного алгоритму 1 (ЖА1)



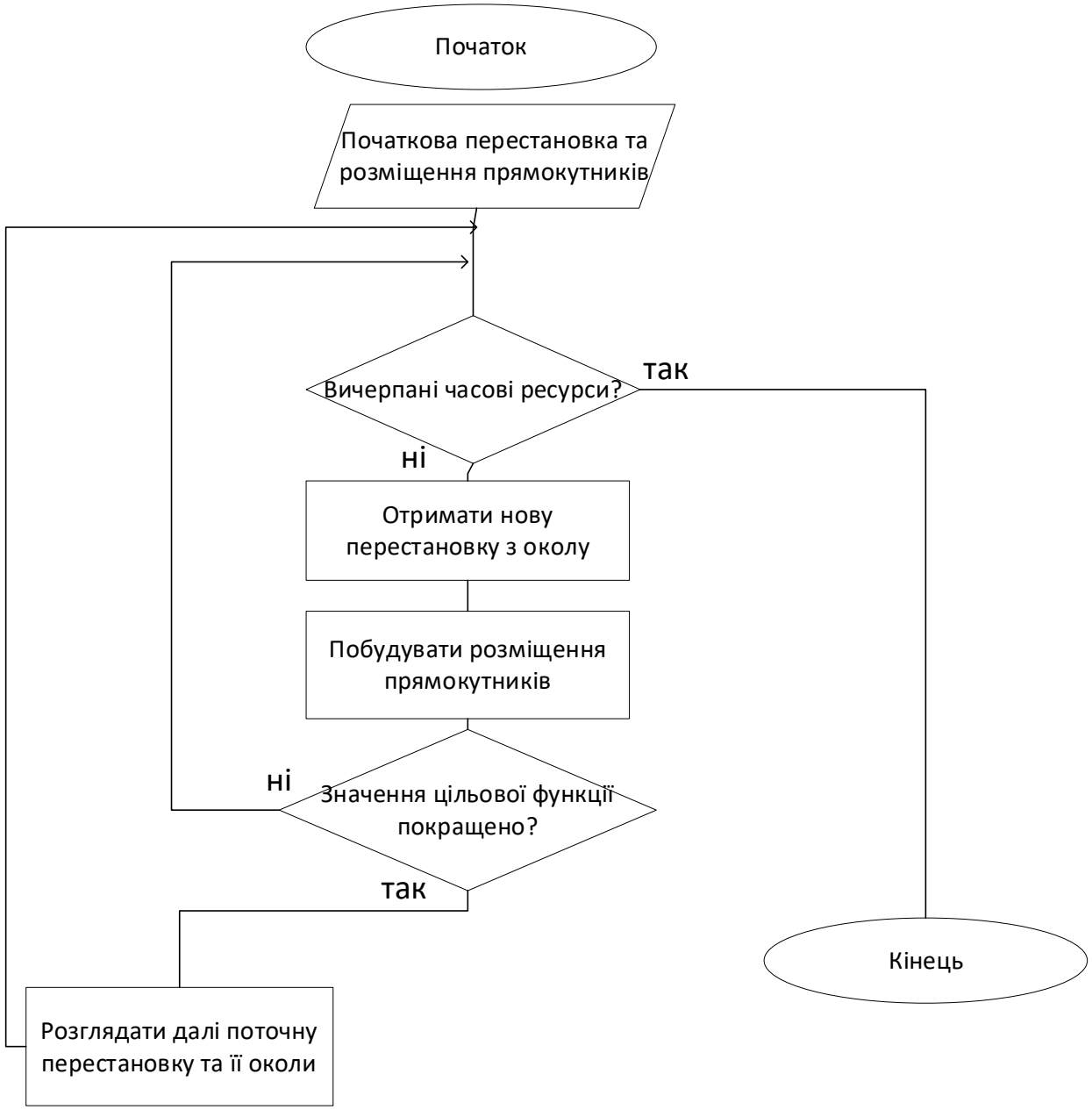
Плакат 2 Приклад роботи жадібного алгоритму 2 (ЖА2)

Плакат 3 Приклад роботи жадібного алгоритму 3 (ЖА3)

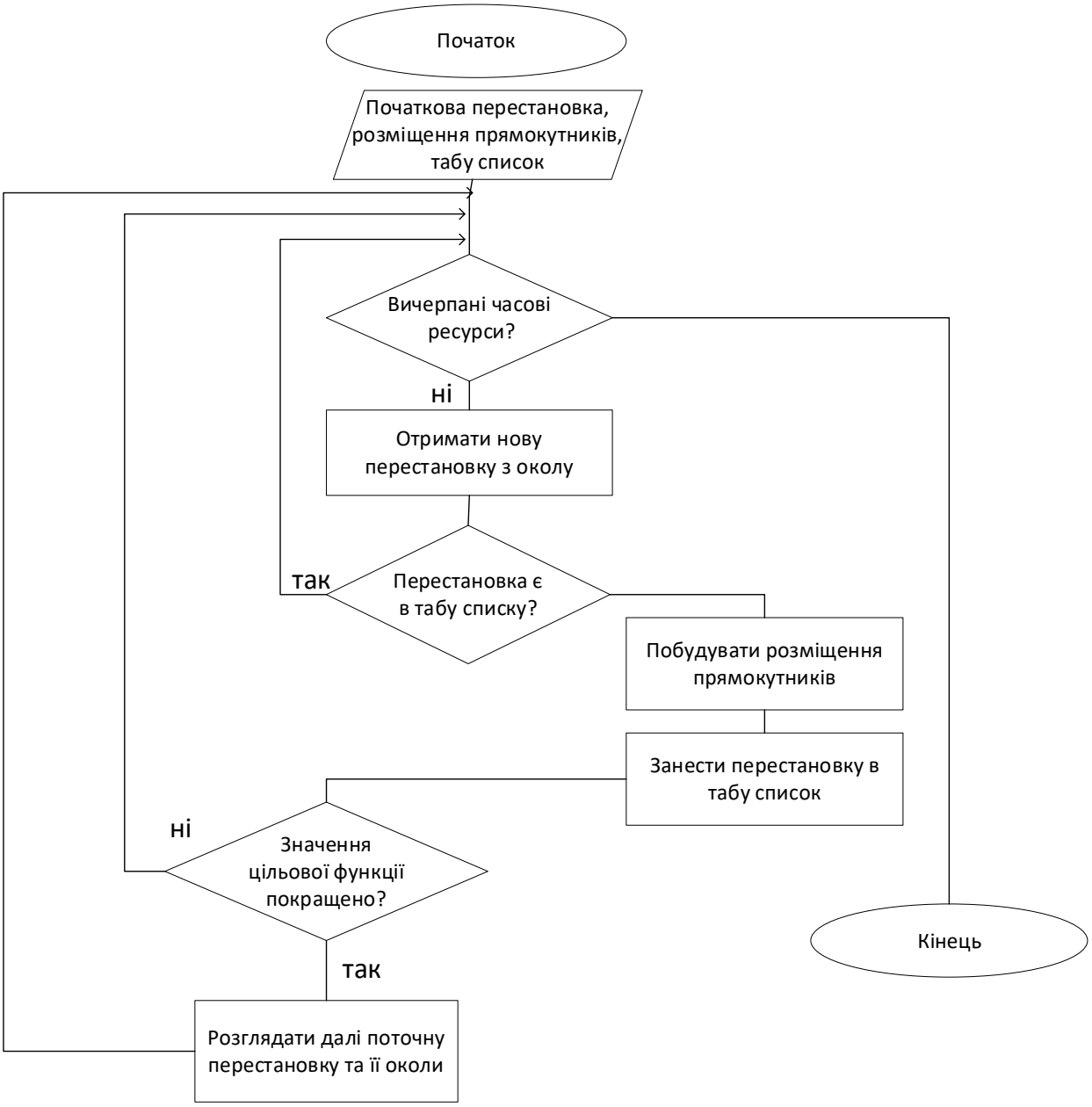
Плакат 4 UML-діаграма варіантів використання



Плакат 5 Блок-схема алгоритму локального пошуку



Плакат 6 Блок-схема алгоритму табуйованого пошуку



Плакат 7 Блок-схема алгоритму обміну