Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод планування обробки даних в розподілених комп’ютерних системах

(тема)

Виконав:

студент II курсу, групи СПм-21-2

Мірошниченко Р.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: зав. каф. Коваленко А.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Мірошниченко Регіні Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Метод планування обробки даних в розподілених комп’ютерних системах

затверджена наказом по університету від “ 03 ” квітня 2023 р. № 318 СТ

2. Термін подання студентом роботи до екзаменаційної комісії 10 травня 2023 р.

3. Вхідні дані до роботи Файл формату *.txt, який містить дані, програмні та алгоритмічні засоби мови програмування С++

4. Перелік питань, що потрібно опрацювати у роботі

1) Аналіз предметної області

2) Моделювання процесу планування обробки даних в розподілених комп’ютерних системах

3) Розробка методу планування обробки даних в розподілених комп’ютерних системах

4) Дослідження планування обробки даних в розподілених комп’ютерних системах

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 13 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	04.04.23-07.04.23	
2	Збір і аналіз вимог для тестів на проникнення	08.04.23-13.04.23	
3	Розробка методу планування обробки		
	даних в розподілених комп'ютерних		
	системах	14.04.23-18.04.23	
4	Тестування розробки	19.04.23-25.04.23	
5	Оформлення матеріалів кваліфікаційної роботи	26.04.23-09.05.23	
6	Подання кваліфікаційної роботи керівникові та	10.05.23-11.05.23	
	її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-		
	вання	12.05.23-16.05.23	

Дата видачі завдання 03 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ зав.каф. Коваленко А.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 68 с., 16 рис., 7 табл., 1 дод., 21 джерел.

РОЗПОДІЛЕНІ СИСТЕМИ, ТЕХНОЛОГІЯ GRID, ОБРОБКИ ДАНИХ

Метою кваліфікаційної роботи є підвищення оперативності планування завдань в розподіленій обчислювальній системі.

Об'єкт дослідження – процес планування та виконання завдань розподіленій в обчислювальній системі.

Предмет дослідження – методи планування та виконання завдань в розподіленій обчислювальній системі.

ABSTRACT

Master’s thesis: 68 p., 16 pic., 3tab., 21 sources.

DISTRIBUTED SYSTEMS, GRID TECHNOLOGY, DATA PROCESSING

The major goal of this thesis is to increase the efficiency of task planning in a distributed computing system.

The object of research is the process of planning and execution of tasks in a distributed computing system.

The subject of the research is methods of planning and performing tasks in a distributed computing system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ВВ – віртуальний вузол;

ВО – віртуальна організація;

ВР – використання ресурсів;

ВЧ – виконання черги;

ІС – інформаційна система;

ІТ – інформаційна технологія;

ПЗП – постійно запам’ятовуючий пристрій

МКР – модель керованого ресурсу;

МО – математичне очікування;

ОЗП – оперативно запам’ятовуючий пристрій

ОС – операційна система;

ПОС – паралельна обчислювальна система;

РМ – ранговий метод;

РОС – розподілена обчислювальна система;

СУБД – система управління базами даних;

СУР – система управління ресурсами

ЦП – центральний процесор

ЧС – частотний метод;

FCFS – First Come First Served;

NLP – None Lineal Programming;

QoS – Quality of service.

ВСТУП

Останнім часом з’являється все більше наукових досліджень присвячених розподіленим системам. Істотно зросла роль факторів обумовлених часом, виділеним для передобробки та обробки вхідної інформації, котра надходить з різноманітних джерел, часом, необхідним для ухвалення рішення, базуючись на прийнятій інформації та часом, необхідним для доведення прийнятого рішення до його виконавців. Існують об'єкти, у яких час обробки даних складає одиниці секунд, а іноді й менше однієї секунди. У таких обчислювальних системах всі задіяні процеси повинні бути орієнтованими на зменшення часу виконання своїх функцій, тобто час є оптимізуємм параметром. Функціонування сучасних обчислювальних систем може здійснюватися у реальному часі. Для цього необхідно забезпечувати найшвидкішу обробку надходячої інформації. Отже на системи управління обробкою інформаційних потоків та потоків даних накладаються такі умови: високопродуктивність, простота, надійність у експлуатації. На сьогодні цим умовам відповідає декілька технологій розподіленої обробки даних, зокрема вже довгий час є популярною технологія Грід, розглядаєма разом з використанням системи планування обробки даних.

Ефективність обробки даних у сучасних обчислювальних системах суттєво залежить від оперативності виконання окремих завдань управління інформаціймих потоками та потоками даних. Отже, актуальним є розв’язання науково-практичного завдання, котре полягає в оптимізації процесів у розподілених обчислювальних системах шляхом розробки методу планування завдань обробки даних системах із застосуванням Grid технологій.

Об'єкт дослідження – процес планування та виконання завдань розподіленій в обчислювальній системі.

Предмет дослідження – методи планування та виконання завдань в розподіленій обчислювальній системі.

Методи дослідження. У дипломній роботі використано: математичний апарат теорії графів, теорії дослідження операцій; методи теорії ймовірностей і математичної статистики, імітаційне комп'ютерне моделювання.

Мета роботи полягає у підвищенні оперативності планування завдань в розподіленій обчислювальній системі.

Для досягнення поставленої мети необхідно вирішити такі часткові задачі:

1) провести аналіз сучасних методів планування та виконання завдань в розподіленій обчислювальній системі;

2) провести моделювання процесу планування обробки даних в розподілених обчислювальних системах;

3) удосконалити метод планування обробки даних в розподілених обчислювальних системах;

4) провести дослідження методу планування обробки даних в розподілених обчислювальних системах.

Наукова новизна одержаних результатів полягає в тому, що удосконалено метод планування завдань в розподіленій обчислювальній системі, за рахунок використання Grid-технології.

Практична цінність роботи полягає в тому, що запропоновані в роботі методи удосконалюють функціональні можливості розподіленої обчислювальної системи за рахунок за рахунок використання Grid-технології.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розподілені обчислювальні системи на основі використанням Grid технологій

Доволі часто розподілену систему визначають як набір незалежних обчислювальних пристроїв, котрі з'єднані каналами зв'язку, причому з погляду користувача розглядаємлї системи розглядаються як єдине ціле. Але при цьому зовсім не враховується розподіленість програмне забезпечення. Тому визначення розподіленої системи ґрунтується на аналізі програмного забезпечення, яке створює таку систему.

Проаналізуємо узагальнену модель, що описує взаємодію між клієнтом та сервером, у котрій клієнт запускає процес обміну даними, посславши запит серверу. Сервер обробляє запит і за необхідності посилає відповідь клієнту (рисунок 1.1).

Рисунок 1.1 – Модель взаємодії клієнт сервер

Взаємодія в межах такої моделі клієнт- сервер може бути синхронною у випадку, якщо клієнт буде очікувати завершення обробки власного запиту сервером.

Якщо клієнт буде посилати на сервер запит та продовжувати виконання не очікуючи на відповідь сервера, то взаємодія є асинхронною. Модель клієнта та сервера може використовуватись як основа для формування різних взаємодій, зокрема для реалізації взаємодії компонент програмного забезпечення, котре утворює розглядаєму розподілену систему.

Тому що в реальній роботі користувачів розподіленої системи зазвичай цікавить доступ до конкретниxx даних, найбільш простим варіантом рознесення функцій для розподіленої системи між декількома комп'ютерами буде розподіл логічних частин програми між головною серверною частиною програми, котрв відповідає за запити до даних програми, і клієнтськими частинами, що знаходяться на декількох комп'ютерах.

Архітектуру застосунків, побудованих за вищенаведеним принципом, називають клієнт- серверною або дволанковою. Насправді такі системи можна не відносити до класифікації розподілених систем, але їх можна вважати тривіальними розподіленими системами. Більш складною клієнт-серверною архітектурою є триланкова архітектура, в котрій доступ до даних, інтерфейс користувача та логіка застосунка складають окремі коипоненти системи, які можуть виконуватися на незалежних комп'ютерах (рисунок 1.2).

Рисунок 1.2 – Триланкова архітектура

Запит користувача в таких системах у послідовному порядку обробляється клієнтським компонентом системи, сервером логіки програми та сервером баз даних. Проте у більшості випадків під розподіленою системою розуміють систему із складнішою архітектурою, ніж триланкова.

Таким чином, під розподіленою обчислювальною системою (РОС) часто визначають систему, у якій зростання багатоланкової архітектури проходить "завширшки", тобто запити користувача не обов’язково послідовно проходять від користувача до сервера баз даних.

Іншим прикладом розподіленоїобчислювальної системи є мережі, що додатково мають можливість організовувати прямий обмін даними між клієнтами системи. Такі системи на сьогодні є найбільш поширеними серед існуючих розподілених систем.

Зауважимо, що до розподіленої обчислювальної системи можна додавати складові різної архітектури, а для побудови РОС на сьогодні існує багато різноманітних технологій, серед яких вже доволі довго найпоширенишими є системи, котрі будуються за Грід -технологією.

Грід =технологія - включає в себе обчислювальні ресурси різних класів. Її можна визначити як географічно розподілену інфраструктуру, котра поєднує в собі множини ресурсів різних типів (ЦП, ОЗП, ПЗП, БД), доступ до яких користувач може отримати з будь-якрї точки, незалежно від місця їх розташування. Грід, як концепція, припускає в собі колективний поділ режим доступу до ресурсів у рамках глобально розподілених віртуальних організацій. В кожній організації можуть бути свої власні правила, а також політики користувачів.

Як відомо, термін Грід був запропонований Яном Фостером та Карлом Кессельманом, авторами першою книги щодо такої ідеологіїї використання комп'ютерних мереж для вирішення завдань. Свою ідею вони пояснювали якоюсь аналогією з електричними мережами (Power GRID),. побудованими таким чином, що кожен користувач може підключитися до розетки та отримати електрику, не розмірковуючи про те, звідкіля ця сама електрика

з'явилася. Такий же принцип закладений і у грід-мережах - користувач підключається до мережі, отримує в своє розпорядження обчислювальні ресурси, не роздумуючи, звідки вони з'являються і де перебувають. Він запускає обчислювальне завдання на виконання, не маючи гадки про те, які комп'ютери і в який точці земної кулі будуть її виконувати.

Для побудови повністю функціональної Грід-системи необхідно програмне забезпечення проміжного рівня (middleware), побудоване на базі існуючих інструментальних засобів, що надає високорівневі послуги завданням і користувачам. Створення та реалізація грід-технологій є складною науковою і практичною проблемою, що знаходиться на стику великої кількості науково-технічних напрямів.

Грід характеризується наступними властивостями:

- масштаби обчислювального ресурсу, які зазвичай перевершують ресурси окремого обчислювального вузла, обчислювального комплексу або суперкомп'ютера;
- гетерогенність середовища;
- географічний розподіл інформаційно-обчислювальних систем;
- об'єднання ресурсів, котрі не можуть керуватися централізовано;
- використання загальнодоступних, стандартних та відкритих, протоколів;
- забезпечення інформаційної безпеки.

До завданням, в рішеннях яких може використовуватися Грід - технології, відносяться:

- складне моделювання;
- спільна візуалізація великих наборів даних;
- розподілена обробка в цілях аналізу даних;
- зв'язок складного інструментарію з віддаленими вузлами.

Найбільш ефективно застосування Грід для рішення таких завдань:

- розподілені складні обчислення, рішення великих завдань, що вимагають велику кількість обчислювального ресурсу;

- високоточні обчислення, котрі дозволяють організувати ефективне використання обчислювальних ресурсів для невеликих завдань, при цьому утилізуючи простоюючі тимчасово мережні ресурси;

- проведення разових складних розрахунків;

- обчислення з великими обсягами розподілених даних, наприклад, в метеорології, астрономії;

- одночасна робота декількох взаємодіючих між собою завдань різних користувачів.

Грід-технології активно застосовуються як державними організаціями управління, оборони, а також приватними компаніями, зокрема, фінансовими та енергетичними. Область застосування Грід зараз охоплює ядерну фізику, захист навколишнього середовища, передбачення погоди. Ресурсом в грід - мережі є будь-які типи обчислювальних потужностей, систем зберігання, мережевих ресурсів; ресурси можуть бути поділені на фізичні і логічні;

Для нормального функціонування грід-система повинна забезпечувати: ідентифікацію виконуваної програми; авторизацію користувача; пошук ресурсів; опис ресурсів; резервування ресурсів; доступ до віддалених даних; розподіл ресурсів; виявлення неполадок.

Для доступу до обчислювальних ресурсів потрібні:

- механізми для запуску програми і моніторингу її виконання;

- механізми для визначення апаратних і програмних характеристик, а також визначення поточного стану (наприклад, робочої завантаження).

Для доступу до мережгих ресурсів потрібні:

- механізм контролю над ресурсами, призначеними для мережвого трафіку (пріоритети, резервування);

- функції визначення характеристик і завантаження мережі.

Вузлами у грід-системах зазвичай є віддалені обчислювальні кластери (ОбКл), котрі можуть відрізнятися по власних характеристиках. Кожний ОбКл можна розглядати як окрему масштабуєму обчислювальну систему лекількох комп'ютерів під керуванням диспетчера даного ОбКл.

Також грід-система може і такі окремі вузли як станції візуалізації та сховища даних.

Для побудови грід-систем використовують програмний комплекс Globus Toolkit, котрий складається із служб управління завданнями, служб збору даних, служб безпеки GSI та служб керування даними. До біосфери Globus Toolkit входять інтерфейси API та SDK мов програмування C++, Java, Python), програми для роботи з сертифікатами.

1.2 Кластерна архітектура Грід-систем

Основним завданням Грід-систем є координація розрізнених ресурсів, причому безпосередньо ресурси не мають загального центру керуваннння, тобто грід-система може займатися допустимо балансуванням загального навантаження. Виходячм з цього можна відзначити, що система управління ресурсами статичного кластера не є системою Грід, так як здійснює централізоване управління усіма вузлами даного кластера, маючи до них повний доступ. Грід-системи мають обмежений доступ до ресурсів, який залежить від власників ресурсу, але при планування використання ресурсів може динамічно створювати тимчасові кластери для виконання окремого завдання або групи завдань. Отже планування виконання завдань в Грід-системі є дуже важливою задачею, яка орієнтована на властивості та архітектуру Грід. Особливостями Грід-систем є такі:

1) повсюдний, стандартний, надійний та дешевий доступ до ресурсів;

2) складність інфраструктури прихована від користувача (прозорість);

3) легкість інтеграції нових ресурсів;

4) динамічний перерозподіл навантаження та відмовостійкість

5) управління складним ансамблем ресурсів.

Грід можуть об’єднувати одну або декілька віртуальних груп, які складаються низкою користувачів або організацій, об'єднаних загальними правилами колективного доступу до певних обчислювальних ресурсів.

Наприклад, це можуть бути провайдери прикладних послуг, провайдери послуг зберігання; учасники промислового консорціуму, що фінансують створення нового літака; учасники багаторічних, великих міжнародних об'єднань у галузі фізики високих енергій. На рисунку1.3 наведений приклад Гріл-системи, що об’єднує дві віртуальні групи: ВГ1 – група, що об’єднує розробників моделей складних 3D об’єктів та ВГ2 – група, що об’єднує колективи, орієнтовані на аналіз супутникової інформації. Бачимо, що крім обчислювальних ресурсів груп ВГ1 (блакитний кольор) та ВГ2 (рожевий кольор) даний Грід задіяв ще кілька сторонніх ресурсів (бузковий кольор).

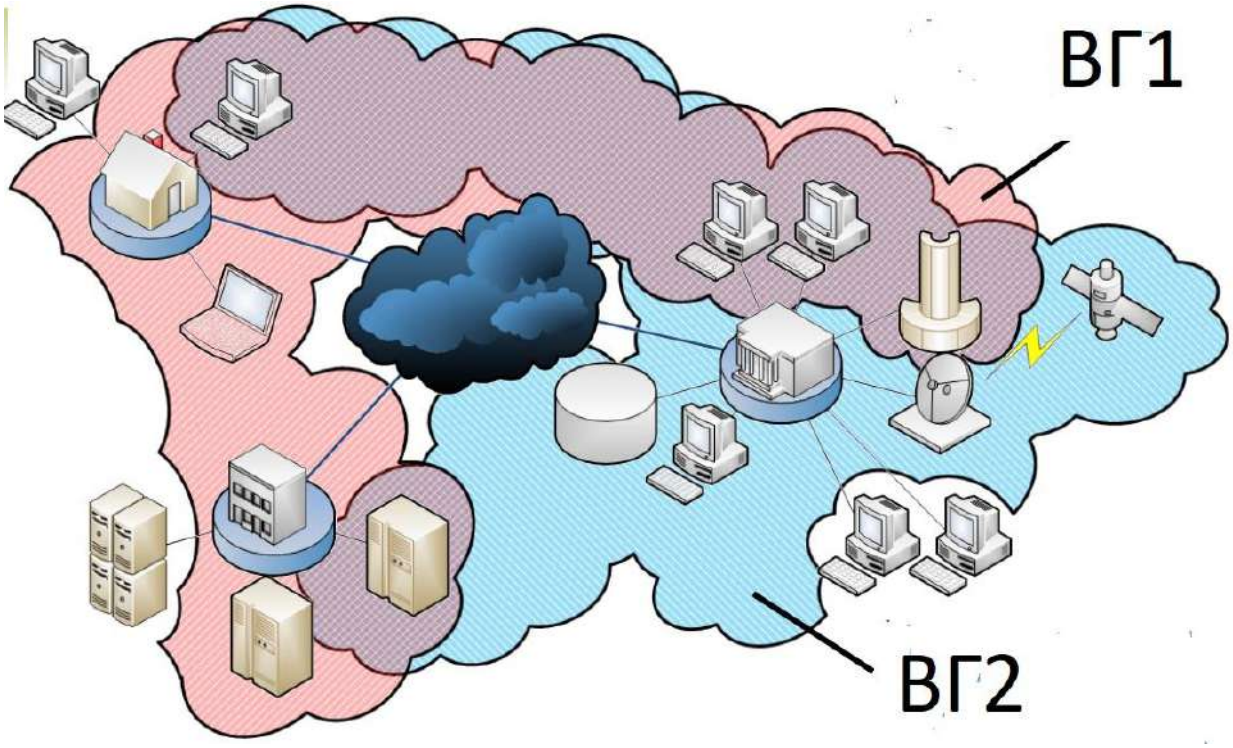

Рисунок 1.3 – Приклад Грід-системи

Архітектура Грід формується, виходячи з необхідності забезпечення інтероперабельності – можливості взаємодії між будь-якими потенційними учасниками даного процесу та наявності набору загальних протоколів, які б визначали механізми за допомогою яких учасники віртуальних груп

домовляються, встановлюють, керують використовують відношення розподілу ресурсів.

Архітектура Грід відображає, яким чином взаємодіють між собою складові системита та складається із низки взаємопов'язаних інтерфейсів, сервісів і протоколів.

Основне завдання – спільне використання віртуальними групасми або окремими користувачами наявних обчислювальних ресурсів. Рівні протоколів Грід наведені на рисунку 1.4, при цьому кожний рівень може використовувати складові будь-якого з рівнів, що розташовані нижче. При цьому в рамках прийнятою концепції протоколи можуть вільно розвиватися.

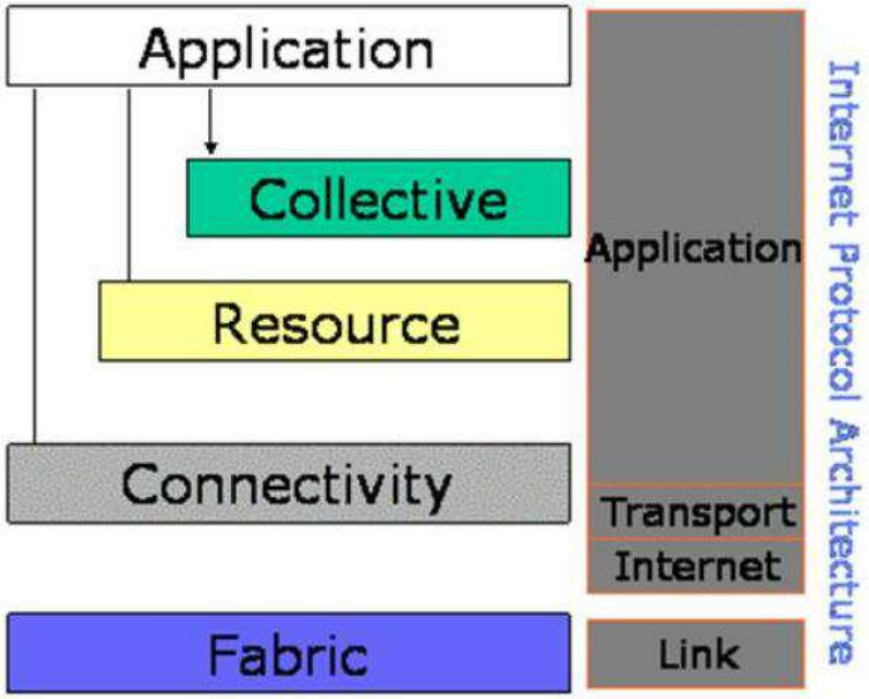

Рисунок 1.4 – Багаторівнева архітектура Грід

На базовому рівні (Fabric) визначаються служби, що забезпечують безпосередній доступ до ресурсів, використання яких розподілено за допомогою протоколів Грід.

До основних типів базових мережних ресурсів відносяться:

- обчислювальні ресурси;
- ресурси пам’яті;
- інформаційні ресурси.

Рівень зв'язку (connectivity) визначає комунікаційні протоколи та протоколи автентифікації. Даний рівень забезпечує передачу даних між ресурсами базового рівня.

Базові протоколи зв'язку, що використовуються на даному рівнв, засновані на стеку протоколів TCP/IP:

- Internet (IP, ICMP);
- транспортні протоколи (TCP, UDP);
- прикладні протоколи (DNS, OSRF…).

Основні вимоги до протоколам безпеки Грід є такими:

- єдиний вхід;
- делегування прав користувача;
- інтеграція з локальними системами безпеки;
- орієнтована на користувача політика взаємодії.

Рівень ресурсів (resource) або ресурсний рівень реалізує такі протоколи, котрі забезпечують виконання наступних функцій:

- узгодження політик щодо безпеки використання ресурсу;
- процедура ініціації задіяного ресурсу;
- моніторинг стану задіяного ресурсу;
- контроль над задіяним ресурсом;
- облік використання задіяного ресурсу.

До протоколів ресурсного рівня відносяться такі:

1) нформаційні протоколи – використовуються для отримання інформації про структуру та стан ресурсу.

2) протоколи управління – використовуються для узгодження доступу до ресурсів, що розділяються, визначаючи вимоги та допустимі дії щодо відношення до ресурсу, такі, як підтримка запиту на резервування, можливість створення процесів, організація доступу до даних).

Колективний рівень (collective) відповідає за глобальну інтеграцію різних наборів ресурсів:

- служби каталогів;
- служби спільного виділення, планування та розподілу ресурсів (Brokering);
- служби моніторингу та діагностики;
- служби реплікації даних.

До системи колективного рівня відносяться такі:

- Грід-орієнтовані системи програмування (наприклад, MPI).
- системи формування бізнес-потоків (Workflow).
- служби пошуку ПЗ (NetSolve, Ninf).

І на останок розглянемо рівень застосунків (applications). На цьому рівні розташовуються користувацькі програми, що виконуються в середовищі віртуальних груп. Вони можуть використовувати ресурси, що знаходяться на нижніх шарах архітектури Грід.

Слід звернути увагу на те, що як фізичні, так і тимчасові віртуальні кластери Грід можуть бути гетерогенними, мати різну архітектуру і комплектацію.

Враховуючи все вищенаведене, в наступному підрозділі детально опишемо процес обробки запитів безпосередньо в кластері Грід.

1.3 Обробка запитів в кластері Грід

Гетерогенність впливає на те, що обчислювальні ресурси не можна обирати довільно, а треба спланувати відповідно до вимог завдання. Також при плануванні обробки запитів на відповідний ресурс треба визначити час задіяння ресурсу, виходячи з часу виконання завдання. Хоч дане значення у будь-якому разі є приблизним, та виходить із певної продуктивності розглядаємого ресурсу, воно важливе, виходячи з таких причин:

- більш тозна оцінка сприяє більшій ефективності алгоритми

планування;

- дане значення є обмеженням, завдяки котрому при виникненні проблем завдання примусово можна завершити.

Виконуючи плануванні виконання завдання Грід відбирає необхідні ресурси, виходячи із власної інформаційної бази, котра містить інформацію щодо складу та характеристик ресурсів системи.

При плануванні з урахуванням пріоритетів використовуються алгоритми, рацююч на приципах FCFS або FIFO). Але при цьому виникає завдяки малим завданням фрагментація ресурсів. Тому Грід поділяє ресурси на два таких класи: відчужувані та, причому невідчужувані ресурси, хоча і не входять до інформаційної бази, але ускладнюють планування, виходячи з того, що вони можуть одночасно отримати завдання з двох потоків:

1) глобальний потік завдань, котрий керується планувальником Грід;

2) локальний потік завдань, котрі запускаються не користувачами Грід.

Хоча локальний потік завдань не контролюється планувальником Грід, але він повинен враховувати даний потік в процесі планування глобальних завдань.

В процесіобробки запитів завдань Грід орієнтується на такі основні етапи, котрі виконуються автоматично:

1) формування пріоритетів завдань з одночасним постійним моніторингом ресурсного завантаження системи та процесу виконання поточних завдань;

2) оперативний перегляд черги завдань на виконання;

3) формування тимчасового пулу ресурсів для поточного завдання;

4) доставляння даних на виконавчі ресурси поточного завдання;

5) процес виконання поточного завдання;

6) доставляння вихідних результатів поточного завдання або до користувача, або до сховища даних.

Користувач участі у цих етапах не приймає, отже можна розглядати кластери Грід-системи (фізичні та тимчасові віртуальні) у якості єдиного

операційного середовища.

Для зменшення ймовірності втрат або затримок в обслуговуванні завдань у випадку обмежених розмірів системної черги (що практично є наявним у більшості Грід-систем) необхідно при плануванні враховувати інтенсивність надходження завдань та швидкість їх обслуговування.

При звичайному функціонуванні системі управління Грід необхідно мати можливість отримувати одночасний доступ до певної кількості вільних ресурсів у поточний момент часу. Крім того, множина ресурсів повинна бути декомпозованою таким чином, щоб один ресурс системи не міг виконувати одночасно більше одного завдання.

Для оцінки ефективності планування Грід-системою виконання завдань розглянемо коефіцієнт використання ресурсів K_{BP}. Даний коефіцієнт визначає частину загальної кількості ресурсів, які необхідні завданням, котрі стоять у черзі, що буде використовуватися. При $K_{BP} = 0$ жоден ресурс не буде використаний, якщо $K_{BP} = 1$, то будуть використані всі ресурси, для котрих у черзі є запити. Але, звісно, такий варіант можливий тільки теоретично, отже наше зводиться до знаходження $K_{BP} \to 1$. У загальному вигляді можна записати:

$$K_{BP} = \frac{N_B}{N_O}, \tag{1.1}$$

де N_B - число ресурсів, які задіяні при певному варіанті; N_O - кількість ресурсів, до яких існують завдання з черги.

При врахуванні пріоритетів завдань коефіцієнт K_{BP} буде залежати від способу вибірки (тільки чисельник в (1.1)).

Нехай Y_i є максимальним значення пріоритету завдань із поточної черги завдань на виконання, котрим потрібний ресурс R_i, M - число ресурсів. Тоді знаменник в (1.1) набуде вигляду:

$$\sum_{i=1}^{M} Y_i \,. \tag{1.2}$$

Для реалізації обслуговування завдань застосуємо спосіб групової вибірки (одночасне обслуговування декількох завдань на різних ресурсах з максимальним сумарним пріоритетом).

Розглянемо множину $\left\{\vec{X}\right\}$, котру складають всі варіанти вибірок завдань з поточної черги. Нехай її елемент $\vec{X}$ є таким:

$$\vec{X} = \{x_1, x_2, ..., x_p, ..., x_N\},\ p = \overline{1..N}, \tag{1.3}$$

де N - число завдань в черзі; x_p - булева змінна. Якщо завдання $З_р$ вибрано в цьому варіанті, то $X_р = 1$, якщо немає, то $X_р = 0$.

Нехай β_p - пріоритет завдання $З_р$. Тоді чисельник виразу в (1.1) набуде такого вигляду:

$$\sum_{p=1}^{N} \beta_p x_p \,. \tag{1.4}$$

З урахуванням (1.2) і (1.4) отримуємо:

$$K_{BP} = \frac{\sum_{p=1}^{N} \beta_p x_p}{\sum_{i=1}^{M} Y_i}. \tag{1.5}$$

Для того щоб коефіцієнт K_{BP} прийняв одиничне значення, необхідно щоб в (1.5) чисельник був рівним знаменнику. Але такий варіант є тільки теоретичним. Вважаючи на те, що знаменник у виразі (1.5) у фіксований момент часу має постійне значення, завдання зводиться до максимізації исельника, що можна обрати як головний критерій формування поточної групової вибірки.

2 МОДЕЛЮВАННЯ ПРОЦЕСУ ПЛАНУВАННЯ ОБРОБКИ ДАНИХ В РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

2.1 Формальний опис процедури планування

Введемо функцію, яка відображає множину завдань *T* на множину ресурсів *R* у розподіленій обчислювальній системі:

$$\Phi(Match):T\times R\times CLTh\to R^{+},$$

де *Match* – матриця, котра задає відповідність завдань із множини *T*, які плануються до виконання, ресурсам із множини *R* з урахуванням пропускної спроможності множини каналів передачі даних *CLTh*, які пов'язують сплановані завдання і обчислювальні вузли, що володіють потрібними ресурсами. За допомогою цієї матриці може бути сформований функціонал оптимізації і обмеження.

В попередньому розділі була обгрунтована основна вимога для побудови цільової функції – сума пріоритетів β_k всіх завдань вибірки x_k повинна бути максимальною, тобто:

$$F=\sum_{k=1}^{p}\beta_k x_k\to\max, \qquad (2.1)$$

де змінна *p* задає розмір поточної вибірки, а змінна *k* ідентифікує конкретне завдання.

Введемо A_{kg} - двійкову змінну, котра дорівнює одиниці при умові використання завданням C_k ресурсу R_g, а в іншому випадку дорівнює нулю; B_g - кількість ресурсів типу R_g. При введених позначеннях можна ввести низку обмежень, пов'язаних з кількісними характеристиками всіх задіяних ресурсів (кількістю *M*):

$$\sum_{k=1}^{p} A_{kg} x_k \le B_g,\ g = \overline{1..M}\ . \qquad (2.2)$$

Тепер можна сформулювати завдання на планування вибірки із черги завдань, тобто для цього треба знайти таку вибірку $\vec{X}$ із множини $\{\vec{X}\}$, для котрої сформований функціонал (2.1) буде приймати максимальне значення при умові виконанні всієї низки обмежень (2.2).

Розглянемо принцип реалізації отриманного завдання на конкретному прикладі для невеличкої черги у 7 завдань. Кожне із завдань у чрзі має якийсь пріоритет та сформовані вимоги до ресурсів системи вимагає використання ресурсів певного типу, тобто чергу можна описати трьома векторами, що складаються із 7 елементів:

$C_{\text{черги}}$ = (C_1, C_2, C_3, C_4, C_5, C_6, C_7);

$P_{\text{черги}}$ = (1, 3, 2, 2, 4, 1, 1);

$R_{\text{черги}}$ = ((1, 3),) (2), (1, 4), (4), (1), (5), (3, 4)).

Зауважимо, що вектор $R_{\text{черги}}$ складається із списків номерів потрібних ресурсів із доступної множини ресурсів {R_1, R_2, R_{13}, R_4, R_5)/

Виходячи з умои задачі складемо перелік завдань, що потребують кожен із доступних ресурсів:

R_1 потребуть завдання (C_1, C_1, C_1);

R_2 потребує завдання (C_2);

R_3 потребуть завдання (C_1, C_7);

R_4 потребуть завдання (C_3, C_4, C_7);

R_5 потребує завдання (C_6).

Етапи вирішення задачі.

Розглянемо процес планування запуску задач покроково.

Крок 1. Сформуємо функціонал (2.1):

$$F = C_1 + 3C_2 + 2C_3 + 2C_4 + 4C_5 + C_6 + C_7 \to \max\ . \qquad (2.3)$$

Ресурсні обмеження (2.2) при цьому можна записвти таким чином,

виходячи з правила: один ресурс – одне завдання:

$$C_1 + C_3 + C_5 \le 1, \tag{2.4}$$

$$C_1 + C_7 \le 1, \tag{2.5}$$

$$C_3 + C_4 + C_7 \le 1. \tag{2.6}$$

Обмеження (2.4) сформоване для ресурсу R_1, обмеження (2.5) сформоване для ресурсу R_3, а обмеження (2.6) сформоване для ресурсу для R_4.

Найкраще значення функйіоналу (2.3) на першому кроці отримаємо при вирішенні завдань C_2, C_4, C_5 та C_6.

Крок 2. Внесемо зміни до функціоналу (2.3) враховуючи результати першого кроку:

$$F = C_1 + C_3 + C_7 \to max. \tag{2.7}$$

При цьому зміняться і обмеження (2.4) – (2.6):

$$C_1 + C_3 \le 1; \tag{2.8}$$

$$C_1 + C_7 \le 1; \tag{2.9}$$

$$C_3 + C_7 \le 1. \tag{2.10}$$

Рішення даного завдання показує, що на цьому кроці найкращим варіантои є обслуження завдання C_3.

Крок 3. Функціонал (2.7) спростився до виразу

$$F = C_1 + C_7 \to max. \tag{2.11}$$

Крім того, залишилося тільки одне обмеження (2.8). тому очевидно, що на цьому кроці потрібно обслужити завдання C_1.

Крок 4. На цьому кроці з черги запускаємо останнє завдання c_7.

Для розглянутого приклада виконання черги було здійснено за чотири кроки. буде виконано в чотири етапи. Як бачимо, виникає необхідність

скорочення улькості кроків при виконанні черги за рахунок використання нетрівіальних підходів, як наприклад, підхід до удосконалення меоду групової вибірки через введення індивідуальної сегментацієї для завдань, котрі знаходяться у черзі, зметою розбиття їх на підзадачі.

2.2 Удосконалення методу групової вибірки за рахунок введення індивідуальної сегментації

Введеня індивідуальної сегментацієї окремих завданьбудемо проводити покроково. На поточному кроці будемо розбивати на підзадачі ті завдання, котрі використовують декілька ресурсів та мают серед подібних найвищі приоритети.

Введемо такі позначення: C_{kg} - підзадача завдання C_k, котра звертається до ресурсу R_g, при цьому має пріоритет C_{kg}; $\{\vec{X}\}$ - множина всіх можливих варіантів вибору підзадач з поточної черги, окремий елемент цієї мнохин $\vec{X}$ - це один з варіантів вибору підзадач $\vec{X} = \{x_{11}, x_{12}, ..., x_{kg}, ..., x_s\}$, де $k = \overline{1..p}$, $g = \overline{1..M}$; p - число завдань в черзі; x_{kg} - змінна, котра дорівнює одиниці у випадку, якщо була обрана відповідна підзадача C_{kg}, і дорівнює нулю у протилежному випадкуі. При цьому функціонал (2.1) розшириться за рахунок нових доданків:

$$F(x) = \sum_{j=1}^{p_1} L_{1j} S_1(L_n^1) + \sum_{j=1}^{p_2} L_{2j} S_2(L_n^2) + ... + \\ + \sum_{j=1}^{p_k} L_{kj} S_k(L_n^k) + ... + \sum_{j=1}^{p_n} L_{nj} S_n(L_n^n) \to max, \tag{2.12}$$

де у змінній $Sr(L_n^r) = S_1 + ... + S_{p_r}$ сумуються всі можливі добутки змінних рангу r, тобто $S_r = X_p X_к ... X_m$; $p_r = \dfrac{n!}{r!(n-r)!}$ - кількість таких комбінацій;

L_{rj} - коефіцієнти в добутках S_r, щокотрі містять містять r змінних.

Обмеження (2.2) при цьому приймуть такого вигляду:

$$\sum_{k=1}^{p} A_{kg} x_{kg} \le B_g,\ g = \overline{1..M}, \tag{2.13}$$

тобто отримане складне завдання математичного програмування з двійковими змінними. Для прикладу, розглянутого вище, функціонал (2.3) зміниться за рахунок нових доданків, що заамістять базові завдання:

$$\begin{gathered} F = C_{11} + C_{13} + 3C_2 + 2C_{31} + 2C_{34} + 2C_4 + 4C_5 + C_6 + C_{73} + \\ + C_{74} + C_{11}C_{13} + 2C_{31}C_{34} + C_{73}C_{74} \to \max, \end{gathered} \tag{2.14}$$

а обмеження (2.4) – (2.6) набудуть такого вигляду:

$$C_{11} + C_{31} + C_5 \le 1; \tag{2.15}$$

$$C_{13} + C_{73} \le 1; \tag{2.16}$$

$$C_{34} + C_4 + C_{74} \le 1. \tag{2.17}$$

Незважаючи на візуально збільшену складність задачі, кількість кроків скоротиться до трьох.

Отже, удосконалення методу групової вибірки за рахунок введення індивідуальної сегментації скорочує кількість кроків відповідного алгоритму та дозволяє збільшити темпи розвантаження черг. На це і будемо спиратися при розробці методу планування обробки даних в розподілених обчислювальних системах

3 РОЗРОБКА МЕТОДУ ПЛАНУВАННЯ ОБРОБКИ ДАНИХ В РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

3.1 Графовий ранговий підхід планування обробки даних

Завдання знаходження оптимального рішення задачі математичного програмування (2.12) – (2.13) при великих розмірах черг є доволі скдадним і у більшості випадків має неприпустиму обчислювальну складність. Тому у даному випадку загальні методи розв’язання подібних завдань використовувати не можно, потрібно знайти підходи, котрі враховують специфіку завдання.

На сьогодні розподілені обчислювальні системи, такі, як, наприклад, Грід-системи, хмарні та туманні технології, є одним з найважливіших напрямків розвитку глобальних і регіональних мереж, де вони сприяють більш раціональному використанню обчислювальних ресурсів. При цьому більшість таких систем для розв’язання задачі планування виконання завдань, що знаходяться у системних чергах, використовують різноманітні власні методи розв’язання. Бажано було б знайти єдиний підхід до розв’язання задач такого класу.

Розглянемо метод, котрий базується на графовому ранговому підході. У цьому підході використовується стягнуте дерево шляхів графа. В свою чергу це дозволяє ефективно розпаралелювати процеси виділення ресурсів завданням, які знаходяться у черзі на виконання.

Суттєво, що при такому підході алгоритм не залежить ні від архітектури, на від типів завдань.

Визначимо множину елементів $\Omega = \{\omega_I\}$ та її підмножини $L_i \in \Omega$.

Також визначимо відношення *R*, котре будує ізоморфізм між елементами (або підмножинами L_i), із множині Ω та об’єктами, що розглядаються.

Крім тог розглянемо розбиття множини Ω на деякі сімейства підмножин $\{L_i\}$. таке, що їх об’єднання є множиною Ω:

$$\bigcup_i L_i = \Omega. \tag{3.1}$$

Об’єкти $\{l_i\}$ ізоморфні підмножинам $L_i \in \Omega$ і також є розбиттям множини Ω:

$$\bigcup_i l_i = \Omega. \tag{3.2}$$

Відношення P визначає вагові характеристики об'єднань сімейств підмножин $L_k \cup L_p \in \Omega$. Дані характеристики пов’язані із властивостями $\{v\}$ об'єктів $\{l_i\}$. Оптимальне значення забезпечує властивість $v^* \in \{v\}$.

Нехай граф з паралельною ярусною структурою D описує всі об'єднання сімейств L_i (рисунок 3.1).

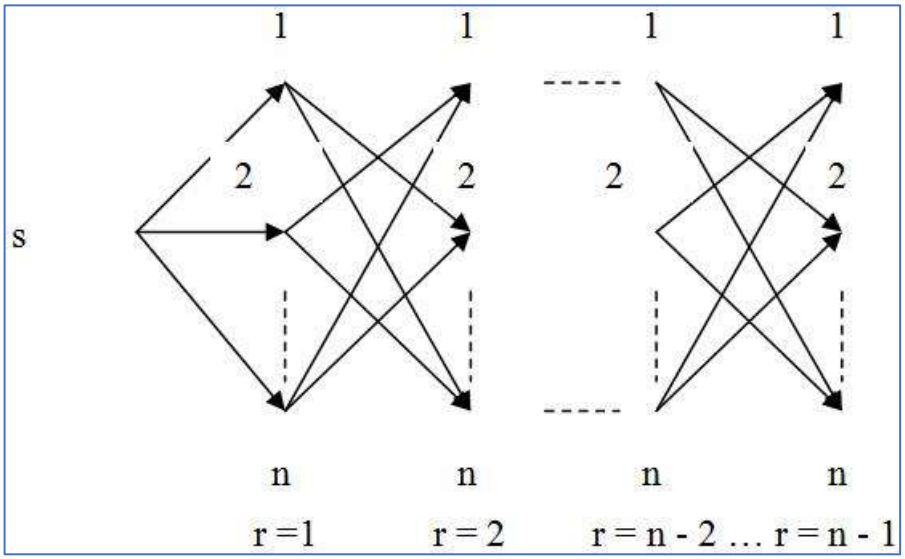

Рисунок 3.1 – Граф паралельної ярусної структури D

Кожна з верши графа D відповідає базоврму елементу l_i. Суттєвою характеристикою маршруту, що не виходить за межі графіа D являється ранг

маршруту *r*. Це кількістьчисло ребер, котрі утворюють маршрут. До поточної вершини *i* у графі є маршрути рангів *від 1 до n* - 1, а кожний маршрут μ_{st}, який задовольняє вимогам до побудови *R* та містить вершини (*j, p,..., k, t*), формується за допомогою об'єднань елементів $(1_j \cup 1_p \cup ... \cup 1_k \cup 1_t)$ відповідно до правила *R*, що визначають деякий об'єкт $L_i \in \Omega$. Довжина цього маршруту $d(\mu_{st})$ визначається за правилами *P*. Отже, множина всіх маршрутів $m_{si}(r)$ графа D, котрі задовольняють вимогам побудови *R* формує всі допустимі рішення початкового завдання щодо виділення об'єкту, котрий має властивості $v^* \in \{v\}$, Як. вихідну вершину графа D використаємо фіктивну вершину *S*, яка буде ототожнюватися із початковим станом нашої системи. Отже максимальний ранг маршруту, що не виходить за межі графа D дорівнюватиме *n*, причому система не змінить своїх властивостей при додаванні до базових елементів вершини *S*.

Множина складних систем, що володіють різними властивостями $\{F_i\}$, може бути відображено за допомогою деякого підмножини графів $\{G_i\}$. Розглянемо довільний *n* вершинний граф $G(V,E) \in \{G_i\}$, який описує стан системи $F \in \{F_i\}$. Вершини $\{i\} \in V$ – це можливі стани системи. Маршрути в графі $G(V,E)$ – послідовність вершин $\{v_i\}$ і ребер $\{(i,j)\}$, визначають можливість досягнення стану $i = p$ із вихідного стану *s*. У графі $G(V,E)$ максимальне значення рангу $r = n-1$. Множини маршрутів m_{sj}^r, $j = (\overline{1,n})$, містять можливі способи досягнення стану *j*. Нехай кожному елементу із множини $\{l_i\}$ відповідає вершина графа $G(V,E)$. У цьому випадку об'єктам $\{L_j\}$ відповідають всі множини об'єктів, котрі будуються на множині ребер *V* по правилах *R* і *P*. Для кожного об'єкту формуємо $m+1$ вагову характеристику, де *m* характеристик є другорядними, з обмеженнями $\{b_i\} i = (1,2,...,m)$, а один є визначальним показником якості об'єкта із

властивістю *v*.

Таким чином, шляху μ_{sj}^{r} в графі D відповідає об'єкт L_j, який може бути побудований з r базових елементів відповідних вершин $\{v_j\}$, включаючи елемент j. А множини маршрутів m_{sj}^{r}; $j=\overline{(1,n)}$ визначають множину об'єктів L_j, які можна побудувати з r базових елементів відповідних вершин $\{v_j\}$, включаючи елемент j.

3.2 Постановка завдання на планування обробки даних

Розглянемо породжуючу функцію $F(x)$, котра дорівнює виразу:

$$F(x)=\sum_{k=1}^{n}\sum_{j=1}^{p_k}C_{k,j}S_{k,j}, \tag{3.3}$$

де $S_{k,j}$ - добуток k різних змінних; $C_{k,j}$ - цілочисельні коефіцієнти, що стоять при $S_{k,j}$, $p_k=C_n^k=\frac{n!}{k!(n-k!)}(k=1\div n)$.

З її допомогою можна задавати всі складові задачі математичного програмування. Нехай H – множина функцій з різними доданками в (3.2), які можна отримати вважаючи рівними нулю різні $C_{k,j}$. Тоді потужність цієї множини є кінцевою та дорівнює 2^{p_Σ}, де

$$p_\Sigma=1+\frac{1}{2}\left(\sum_{k=1}^{n}\frac{n!}{k!(n-k)!}\left(\frac{n!}{k!(n-k)!}+1\right)\right). \tag{3.4}$$

Слід зазначити, що за допомогою співвідношення (3.3) можна представити цільову функцію завдання планування обробки даних:

$$f(X_1,...,X_n)=\sum_{k=1}^{n}\sum_{j=1}^{p_k}C_{k,j}S_{k,j}=f^*\to\max, \tag{3.5}$$

Також за її допомогою формуються нетривіальні обмеження для завдання планування обробки даних:

$$g_{\ell}(X_1,..,X_n)=\sum_{k=1}^{n}\sum_{j=1}^{p_k}T_{k,j}S_{k,j}\le b_l;\quad \ell=(\overline{1,m}), \tag{3.6}$$

де $C_{k,j};T_{k.j};b_l$ – позитивні цілі числа.

Розглянемо повно зв'язний граф $G(X,E)$, в якому вершинами графа є змінні X_i. Виділимо в графі G довільну кліку $Q=X_pX_r...X_m$, котра складається з вершин кількістю r, де $r<n$, та будемо розглядати включення $Q\subseteq S_{k,j}\in f(X_1,X_2,...,X_n)$, а також $Q\subseteq S_{k,j}\in g(X_1,X_2,...,X_n)$. Кожне включення можна охарактеризувати сумами коефіцієнтів $C_{k,j}$, що стоять при $S_{k,j}$ в функціоналі $f(X_1,X_2,...,X_n)$ і сумами коефіцієнтів $T_{k,j}$, що стоять при $S_{k,j}$ в обмеженнях $g_j(X_1,X_2,...,X_n)$. Зауважимо, що поточна кліка Q характеризується відповідною вагою за функціоналом $f(X_1,X_2,...,X_n)$ і не більше ніж m вагами по обмеженням.

3.3 Розв’язання задачі планування обробки даних на основі рангового підходу

Розглянемо вихідний граф $G(V, E)$ як симетричне дерево маршрутів D (стягнене дерево маршрутів). Дерево всіх маршрутів D містить $(n-1)$ горизонталь та $(n-1)$ ярус. Виходячи із цього, для будб-якої вершини j множина маршрутів, до цієї вершини є такою:

$$m_s(j)=m_{sj}^{r=1}\cup....\cup m_{sj}^{r=n-1};\quad j=1..n-1, \tag{3.7}$$

де $m_{sj}^{r}=\{\mu_{sj}^{r}\}$ - підмножини шляхів з будь-якої вершини в деяку вершину графа $G(V,E)$,шо має ранг r.

Але дерево всіх маршрутів можна побудовати. Орієнтуючись на

конкретну вершину *i* графа. Тоді вершина *s* = *i*, та у цьому випадку *i*-та горизонтальна низка буде виключена зі *D*. В якості прикладу побачимо, що при *i* = 2 розглядаємо стягнуте маршрутів дерево *D* змінить вигляд таким чином, як це показано на рисунку 3.2.

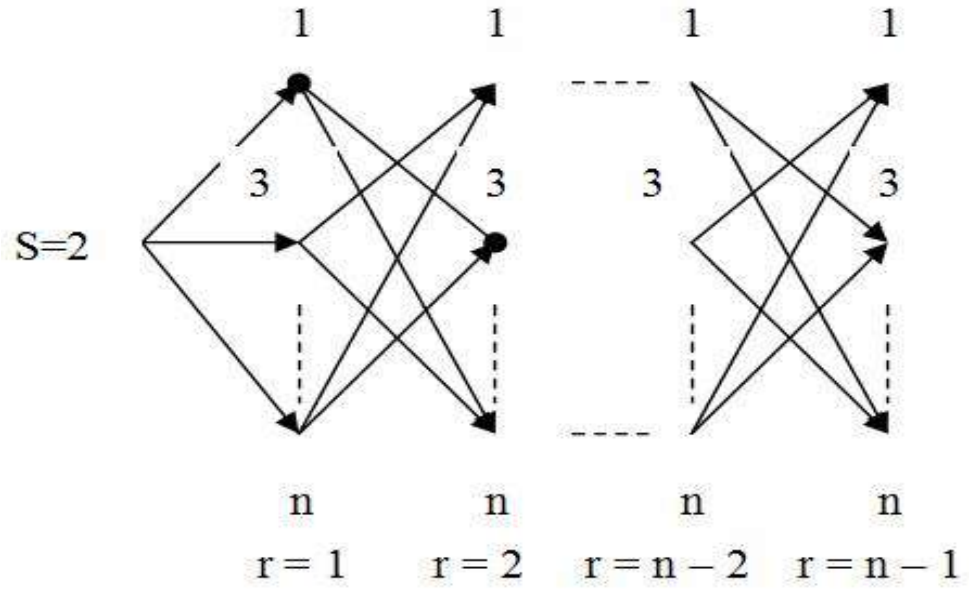

Рисунок 3.2 – Стягнуте дерево маршрутів графа *D* з відною вершини *S* = 2

Побудоване таким чином стягнуте дерево маршрутів використовується для формування однопрохідних алгоритмів рішення завдання з цільовою функцією (3.5) та низкою нетривіальних обмежень (3.6), а безпосередньо дерево (рисунок 3.2) використаємо для того, щоб побудувати *n*-прохідні алгоритми.

Отже, можна з будь-якої вершини s поетапно будувати маршрути $\{\mu_{sj}^{r}\}$ довільного рангу аж до рангу $r = n-1$. У завданні з цільовою функцією (3.5) та низкою нетривіальних обмежень (3.6) під станом системи розуміємо якісь різні способи, котрі об'єднують вершини графа *D* у кліки. Тоді кожному маршруту $\{\mu_{sj}^{r}\}$, котрий має ранг *r* у графі *D*, який проходить через вершини $\left(v_h, v_k ... v_p\right)$, у вихідному графі завдання, що розв'язується, відповідає кліка з

$r=1$ вершини $\left(X_h X_k ... X_p\right)$, що характеризується відповідним вагою по функціоналу $f(X_1, X_2, ..., X_n)$ та не більше чим m вагами з обмеженнями $g_l(X_1, X_2, ..., X_n); l=(\overline{1,m})$. Вагові характеристики $\left\{d_{sj}^{r-1}\right\}$ довільної кліки $Q^{r-1}(j)$, що складається з $r-1$ - вершини і яка визначається одним із маршрутів $\mu_{sj}^r \in m_{sj}^r$, що мають ранг *r*, обчислюються за вагами функціоналу з використанням підсумовування коефіцієнтів у підмножині $L_f=\{C_{k,j}\}$, що стоять при доданках $S_{k,j}$ в функціоналі, тобто б відповідала умовам $Q \subseteq S_{k,j} \in f(X_1, X_2, ..., X_n)$. Такимо ж чином можна визначити вагові характеристики по вагах обмежень, підсумовуючи коефіцієнти підмножини $L_B=\{T_{k,j}\}$, котрі стоять при доданках $S_{k,j}$ в обмеженнях, тобто б відповідала умовам $Q \subseteq S_{k,j} \in g(X_1, X_2, ..., X_n)$. Таким чином, вагові характеристики клік $Q^{r-1}(j)$ характеризуються великою кількістю маршрутів m_{sj}^r за вагами функціоналу та обмежень визначаються відповідно виразами:

$$d_{sj}^{f_{r-1}} = \sum_{C_{k,j} \in L_f} C_{k,j}\,;\; d_{sj}^{B_{r-1}} = \sum_{T_{k,j} \in L_B} T_{k,j}\,. \qquad (3.8)$$

Якщо на основі підмножин маршрутів $m_{sj}^{r=1}$ в графі *D* будувати підмножини $m_{sj}^{r=2}$ і так далі до маршрутів $m_{sj}^{r=n-1}$ рангу *r* = *n* - 1, то ми змушені будемо побудувати $(n-1)!$ маршрутів, тому для формування маршрутів вводиться процедура *A*, котра дозволяє відсіч неперспективні маршрути. Для цього у процедурі *A* можна використати принцип оптимізації, прямуючи до будб-якої вершини *p*, при формуванні маршрутів рангу m_{sp}^{r+1} на основі маршрутів попереднього рангу m_{sj}^r, , який у даному завданн можна визначити таки рекурентним співвідношенням:

$$\mu_{sp}^{r+1} = \max_j \{\{\mu_{sj}^r\} \cup (j, p)\};\; j=(\overline{1,n});\; p=(\overline{1,n});\; j \neq p\,, \qquad (3.9)$$

де позначеню, що (j, p) є ребром графа D, а n - це кількість різних вершин у графі D.

Опишемо процедуру побудови n - прохідних і однопрохідних процедур вирішення завдання з цільовою функцією (3.5) та низкою нетривіальних обмежень, заданих у виразі (3.6) відповідно на стягнутих деревах, котрі наведені на рисунках 3.1 і 3.2.

Процедура A_1 (n-прохідна).

Етап 1. Задаємо змінну $s := i$; з вершини s побудуємо всі можливі маршрути рангу $r = 1$ до всіх вершин графа D (рисунок 3.2), що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l = (\overline{1, m}),;$ але зауважимо, що довжини за вагами функціоналу та низкою обмежень будемо обчислювати відповідно до співвідношеня (3.9).

Етап 2. Поточний ранг r. Будуємо всілякі можливі маршрути рангу $r := r + 1$ задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l = (\overline{1, m}),$ з використанням рекурентного співвідношення (3.9). Перевірку обмежень та вибір маршруту, що є максимальноим за вагами функціоналу, здійснюємо, використовуючи обчислення довжин маршрутів за вагами функціоналу та обмежень, що відповідають співвідношеннями (3.8). Якщо при застосуванні рекурентного співвідношення (3.9) виникли декілька маршрутів, що мають однакову довжину, тоді надо такі маршрути продовжти розглядати на наступному ранзі.

Етап 3. Проводимо перевірку $m_{sj}^{r+1} = \varnothing$. При її виконання маршрут μ_{sj}^{*r} максимальної довжини, котрий був отриманим на рангі r, буде локальним максимумом розв'язуваної даним алгоритмом завдання, у іншому випадку перейдемо до наступного етапу.

Етап 4. Проводимо перевірку $i := n - 1$. При її невиконанні $i := i + 1$ і переходимо до виконання етапу 1, інакше процедура A_1 буде закінчувати виконання, тоді з множини локальних екстремумів $\left\{\mu_{sj}^{*r}\right\}$ вибирається

глобальний екстремум μ_{sj}^{**r}, що відповідає оптимальному вирішенню завдання (3.5) – (3.6).

Для того, щоб знижзити час виконання цього алгоритму, можлна використати однопрохідний варіант для цієї процедури, яка формується на основі стягнених дерева маршрутів, наведеного на рисунку 3.1. При цьому однопрохідне процедура A_2 має вигляд, описаний нижче.

Процедура A_2 з одним проходом.

Етап 1. Базуючись на вершині s будуємо всілякі можливі маршрути, що мають ранг $r=1$, до всіх вершин сформованого графа D, зображеного на рисунку 3.2, що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l=(\overline{1,m})$, при цьому всі характеристики функціоналу і обмежень будемо розраховувати за співвідношеннями (3.9).

Етап 2. Використовуючи маршрут поточного рангу r, будуємо всілякі можливі маршрути рангу $r:=r+1$, що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l=(\overline{1,m})$, використовуючи рекурентне співвідношення (3.9). Слід зазначити, що якщо в процесі застосування рекурентного співвідношення (3.9) виникнеь декілька маршрутів однакової довжини, то необхідно всі ці маршрути продовжити на наступному ранзі.

Етап 3. Проводимо перевірку $m_{sj}^{r+1}=\varnothing$. При її виконанні маршрут μ_{sj}^{*r} максимальної довжини, що бувотриманий на ранзі r, буде локальним максимумом розв'язуваної задачі, інакше перейдемо до наступного етапу.

Етап 4. Проводимо перевірку рангу $r=n$. При її невиконанні то переходимо до виконання етапу 2, інакше процедура A_2 закінчується, а маршрут μ_{sj}^{*r} – це маршрут максимальної довжини, який був отриманий на ранзі $r=n$, відповідає оптимальному вирішенню завдання (3.4).

Ще одним варіантом зменшення часової складності алгоритмів на основі процедур A_1 і A_2 можна розглянути процедури A' і A'', які

відрізняються від A_1 і A_2 тим, що локальні екстремуми виділяються не у кожній множині.

Процедура A'.

Етап 0. Починаємо з присвоєння значення змінній $i := 1$.

Етап 1. Присвоюємо $s := i$ та з вершини s побудуємо всілякі можливі маршрути рангу $r = 1$ до всіх вершин графа (рисунок 3.2), що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l = (\overline{1,m})$, при цьому довжини за вагами функціоналу і обмеженням обчислюються відповідно до співвідношення (3.9). Далі виділяємо найдовший маршрут на цьому ярусі.

Етап 2. Спираючись на найдовший маршрут рангу r побудуємо всілякі можливі маршрути рангу $r := r + 1$, що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l = (\overline{1,m})$, з використанням рекурентного співвідношення (3.8). Перевірку обмежень і маршруту проводимо за допомогою співвідношення (3.9).

Етап 3. Перевіряємо умову $m_{sj}^{r+1} = \varnothing$. При її виконанні маршрут μ_{sj}^{*r} максимальної довжини, що був отриманий на ранзі r, буде локальним екстремумом завдання. Якщо дана умова не виконується, то переходимо до виконання наступного етапу.

Етап 4. Перевіряємо умову $i := n - 1$. При її невиконанні $i := i + 1$ і переходимо до виконання етапу 1, інакше процедура A' закінчить роботу, а із множини локальних екстремумів $\{\mu_{sj}^{*r}\}$ вибирається глобальний, що відповідає оптимальному вирішенню завдання (3.5). – (3.6).

Процедура A''.

Етап 1. З вхідної вершини s будуємо всілякі можливі маршрути рангу $r = 1$ до всіх вершин графа D, відповідно до того, як це показано на рис. 3.1, що задовольняють обмеженням (3.6) $g_l(X_1, X_2, ..., X_n); l = (\overline{1,m})$, довжини за вагами для функціоналу і обмежень обчислюємо відповідно до співвідношень (3.9) та знаходимо на ярусі найдовший маршрут.

Етап 2. Базуючись на найдовшому маршруті поточного рангу r, котрий був побудований на попередньому етапі, будуємо всілякі можливі маршрути рангу $r := r+1$, що задовольняють обмеженням $g_l(X_1, X_2, ..., X_n); l = (\overline{1,m})$, використовучт рекурентні співвідношення (3.9). При цьому перевірка обмежень (3.6) і вибір максимального маршруту за вагами функціоналу (3.5) буде здійснюватися на базі обчислень довжин маршрутів за вагами функціоналу (3.5) та низки нетривіальних обмежень (3.6) відповідно до рекурентного співвідношення (3.9).

Етап 3. Проводимо перевірку $m_{sj}^{r+1} = \varnothing$. При її виконанні маршрут μ_{sj}^{*r} максимальної довжини, отриманий на ранзі r, є локальним екстремумів розв'язуваного завдання, якщо ні, то будемо виконувати наступний етап.

Етап 4. Проводимо перевірку рангу $r = n$. При її невиконанні переходимо до виконання етапу 2, у іншому разі процедура A'' закінчує роботу, а маршрут μ_{sj}^{*r} максимальної довжини, котрий був отриманий на ранзі $r = n$, буде відповідати оптимальному вирішенню завдання з цільовою функцією (3.5) та обмеженнями (3.6).

Надалі позначимо через A_5 алгоритм, реалізований на основі багатопрохідної процедури, через A_4 алгоритм, реалізований на основі однопрохідної процедури. Через A_2 і через A_3 алгоритм реалізований на основі однопрохідної процедури A'', який охоплює всі способи виділення локальних екстремумів.

3.4 Оцінка часової складності розроблених процедур $\{A\}$

При роботі процедур $\{A\}$ кількість маршрутів, котрі будуються на якомусь ранзі r, не може перевищити значення $(n-1)(n-1)$. Максимальний ранг r довільного маршруту не перевищує $(n-1)$, і число циклів, що виконується процедурою A_1, є n. Тому після n циклів, кількість маршрутів,

котрі побудує процедура A_1 не зможе перевершити $(n-1)(n-1)(n-1)n \approx n^4$, а число оброблених векторів n^5. Врахувючи кількість доданків (k) в функціоналі (3.5) та число обмежень (m), часова складність алгоритму не перевищить в гіршому випадку $O\left(n^5 k(m+1)\right)$. Якщо розв'язок завдання здійснемо за один прохід або процедури A_2, або процедури A'', але виділяючи найбільш довгий маршрут на поточному ярусі, то складність процедур A_2 і A'' не перевищать відповідно $O\left(n^4 k(m+1)\right)$ і $O\left(n^3 k(m+1)\right)$. Отже, алгоритми A_5 A_4, A_3 мають відповідно часову складність, котра не перевищує в гіршому випадку таких значень, як $O\left(n^5 k(m+1)\right)$, $O\left(n^4 k(m+1)\right)$ і $O\left(n^3 k(m+1)\right)$.

Отже, запропонований метод розв'язання планування обробки даних на основі рангового підходу дозволяє з використанням низки запропонованих алгоритмів отримати шуканий план за поліноміальний час з необхідною точністю.

4 ДОСЛІДЖЕННЯ МЕТОДУ ПЛАНУВАННЯ ОБРОБКИ ДАНИХ В РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

4.1 Оцінка часової складності складових методу

Як головні параметри алгоритмів, застосовуємо декілька простих виразів і декілька оброблених векторів, також часових витрат і відносну $П$ похибку отриманого наближеного рішення, яка визначається зі співвідношення:

$$П = \frac{|f(x) - f(x^*)|}{f(x^*)}, \tag{4.1}$$

де f - це цільова функція, котра визначена на множині M; x - це наближене рішення задачі; x^* - це плануєме оптимальне рішення.

В експерементальній частині використовувався алгоритм A_5 на основі багатопрохідної процедури A_1, алгоритм A_4 на основі однопрохідної процедури A_2 і алгоритм A_3 на основі однопрохідної процедури A''. При дослідженні коефіцієнти в функціоналі і обмеженнях синергувалися за рівномірном розподілом у діапазоні від 0 до 10, та обмеження мали діапозон від 0 до 20. Для оцінки складності алгоритмів за середнім часом на кожну означену ділянку (крапку) та погрішність алгоритмів обчислювалось приблизно 50 тестових завдань. Були отримані результати з ймовірністю майже 95%. Для прикладу точного алгоритму було взято алгоритм для розвязання задач лінійного та нелінійного програмування. Він зібран на базі використання для визначення перспективних рішень та гіпотез рангового підходу. Головна його мета відккинути неперспективні маршрути методу гілок та меж. Він дозволив зняти погрішності для нескладних за обчисленням задач (де не більше $n = 70$. Графіки залежності похибки від розмірності (n) завдань, що розв'язувались та від кількості обмежень (m) в прикладі що наведен у другому розділі представлені на рисунках 4.1 – 4.3.

Проаналізувавши ми бачимо, що погрішність алгоритмів з великим числом обмежень m значно зменшується, при тому що за збільшенням n зростає також і $m \geq 50$. Погрішність алгоритмів нормалізується і для задач лінійного програмування. Вона не перевищує 2%, та для задач квадратичного доходить до 10%. Розв'язання тестових завдань підтвердило, що чим більше діапазон змін різних коефіцієнтів у функціональній частині та обмеженнях, то це призводить зменьшення погрішностей алгоритмів. Також перехід від одного порядку нелінійностей до більш високого, майже не буде призводити збільшення погрішностей при малому числі обмежень. Та зі збільшенням кількості обмежень зростання погрішностей швидко компенсується. Дослідження (експерементальне) тимчасової складності визначило (рисунок 4.3), що число векторів що оброблюються не залежить від кількості обмежень алгоритмів A_5, A_4, A_3 тимчасова складність є величиною порядку відповідно заданих значень.

Розгляенмо приклади роз'язання задачі.

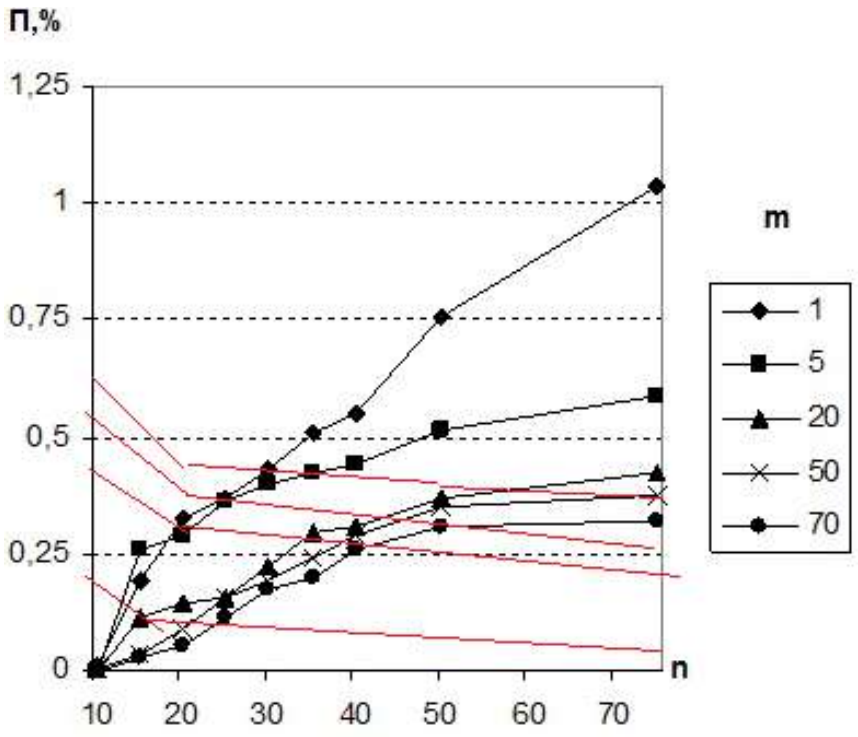

Рисунок 4.1 – Погрішності алгоритму A_5 в залежності від кількості обмежень

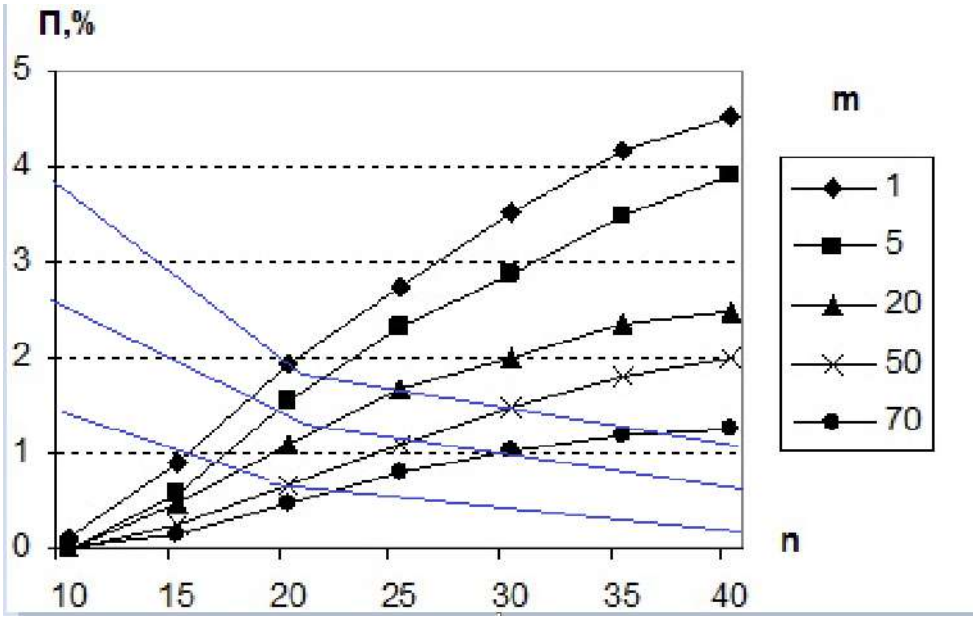

Рисунок 4.2 – Погрішність алгоритму в залежності від кількості обмежень

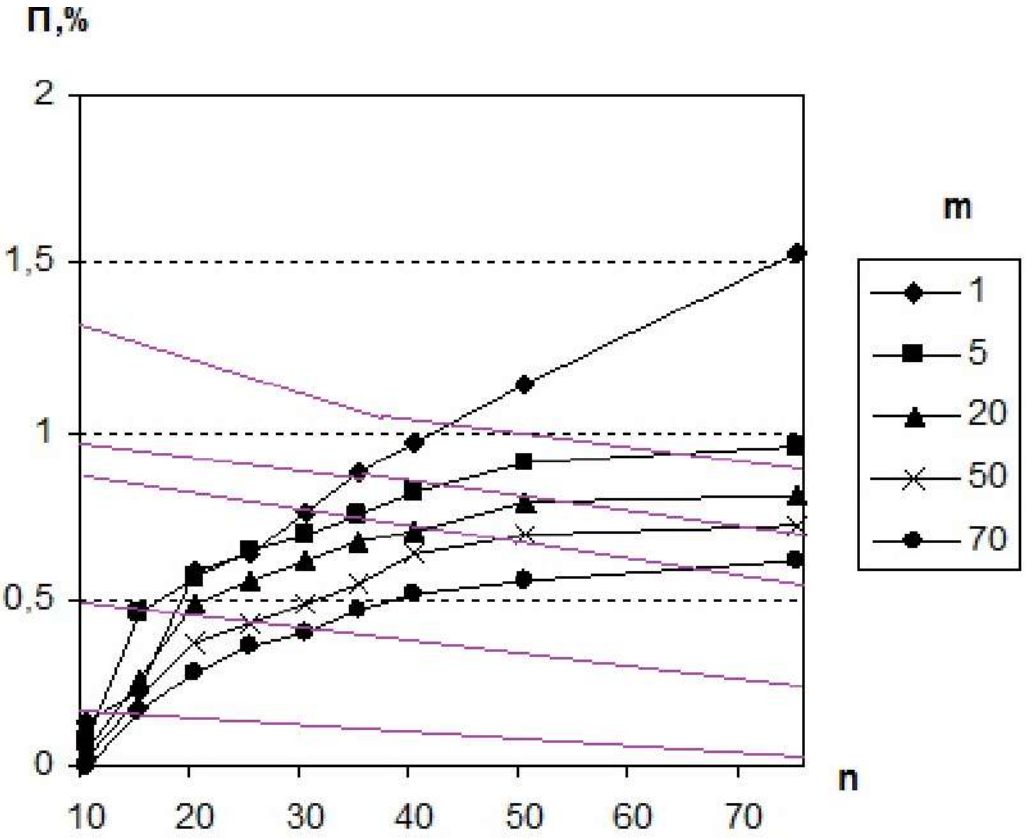

Рисунок 4.3 – Погрішність алгоритму при використанні булевого програмування

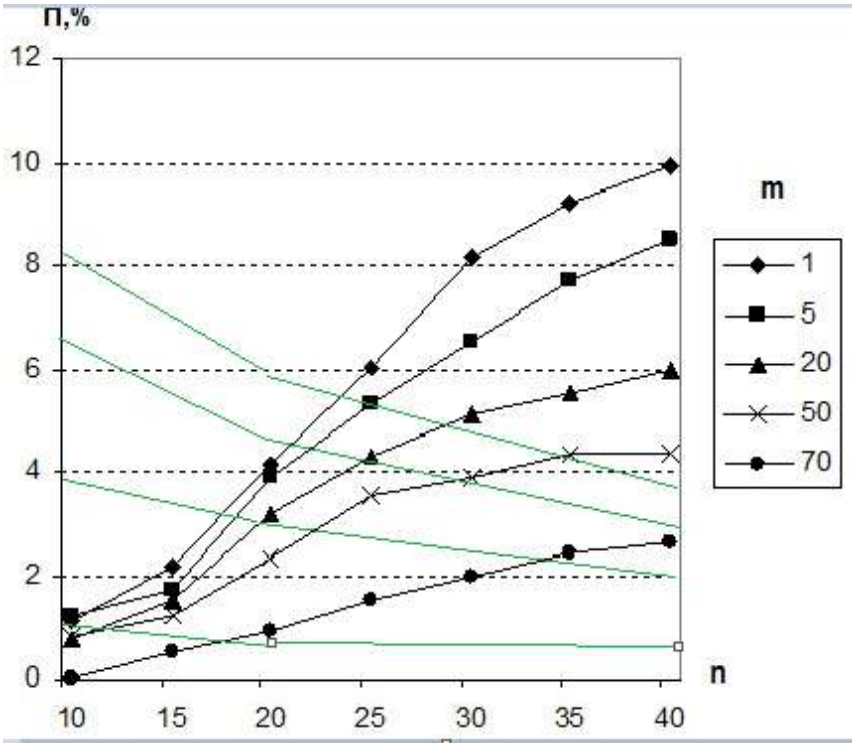

Рисунок 4.4– Залежність погрішності алгоритму (число обмежень)

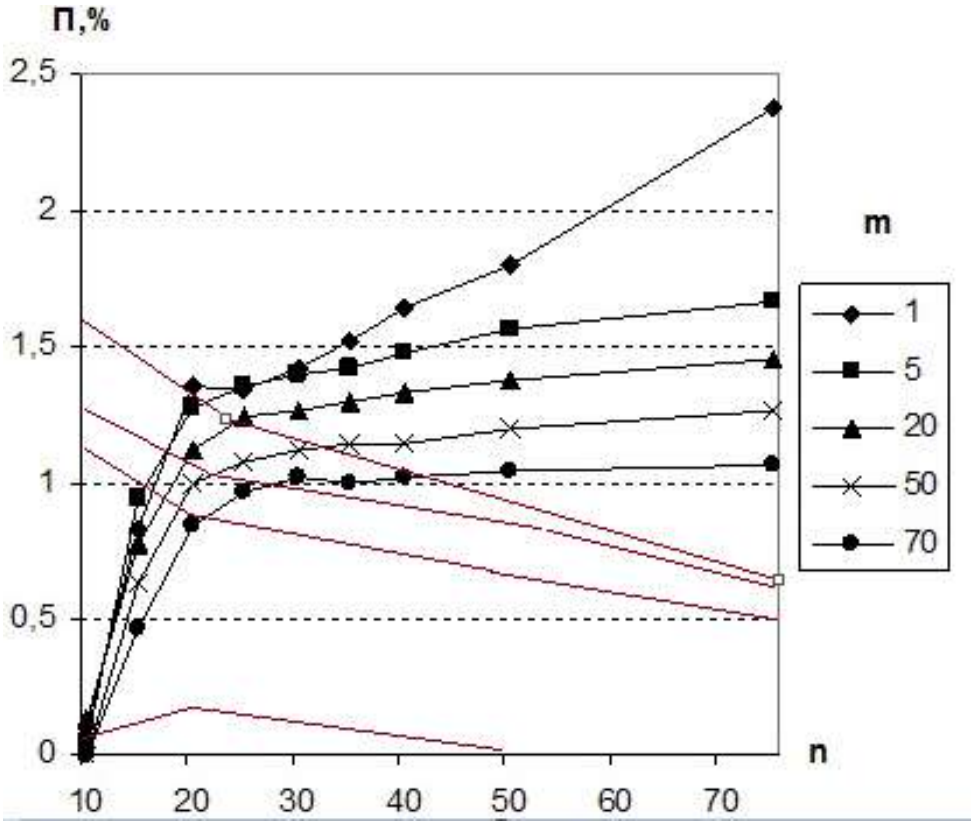

Рисунок 4.5 – Погрішність алгоритму A_3

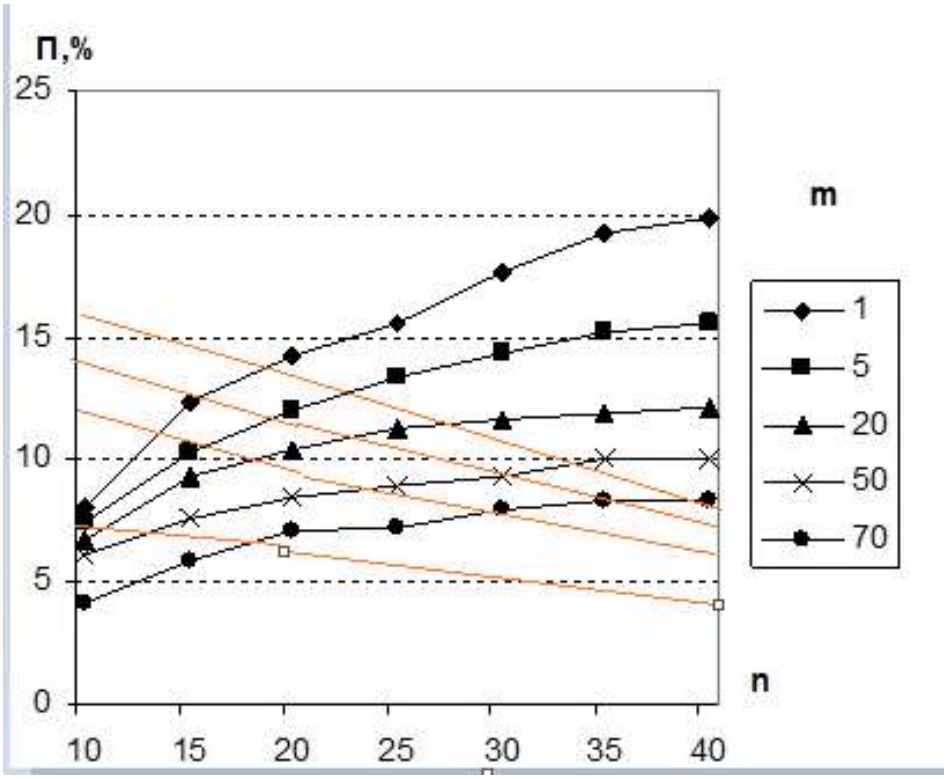

Рисунок 4.6 – Погрішності алгоритму A_3 в залезжності від різній числа обмежень (m)

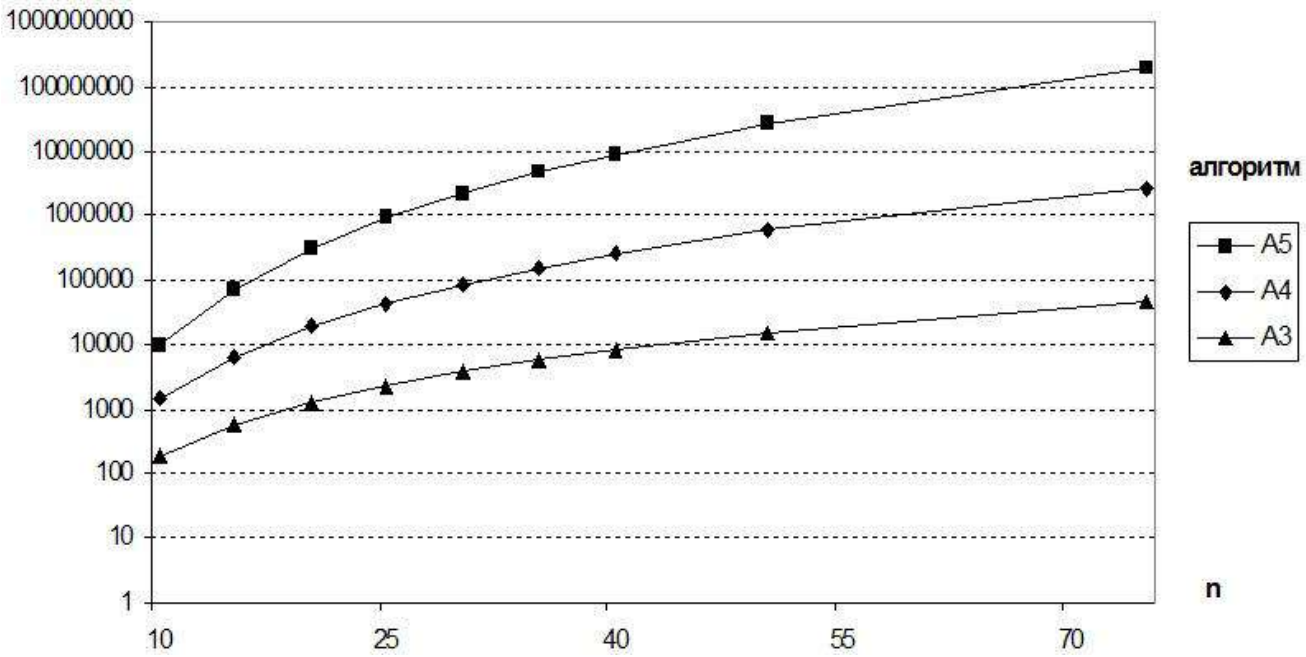

Рисунок 4.7 – Кількість обчислених векторів для всіх алгоритмів

4.2 Залежності похибок відповідних алгоритмів та ймовірнісні характеристики

Всі можливі рішення завдання надамо у вигляді графа *D*. Формування маршрутів представлено у таблиці 4.1, в котрій вершинам графа *D* відповідають відповідні чарунки.

Розглянемо процес формування. Спочатку означемо всі можливі рішення з вершини S до чотирьох вершин. Означемо їх $S1$, $S2$, $S3$, $S4$. Кожен маршрут має дві довжини. Перша довжина оцінюється за вагами коефіцієнтів у функціональній частині (перший в дужках). Та друга довжина вимірюється по вагах коефіцієнтів при обмеженнях.

Розрахунок довжини для маршруту $S1$. Перевіряємо, в функціоналі змінна x_1 є присутнбою чи її немає. Ця змінна не є присутньою не в функціоналі не в обмеженнях, тому довжини в маршруту $S1(0,0)$. Аналогічна ситуація з маршрутом $S2(0,0)$. Для маршруту $S3$ дивимося, в функціоналі змінна x_3 окремо присутня? Так присутня, при ній стоїть коефіцієнт 20, значить, довжина маршруту по функціоналу дорівнює 20.

В обмеженні окремої змінної x_3 немає, тому довжина маршруту по вагам обмежень буде дорівнювати 0, отже маршрут $S3$ має дві довжини (20,0). Для маршруту $S4$, перевіряємо, в функціоналі присутність змінної x_4. Вона є, у неї єкоефіцієнт 10, тому відповідна довжина маршруту дорівнює 10. Аналогічно проводимо перевірку присутності в обмеженні змінної x_4. Вона теж є з коефіцієнтом 4, тому, довжина відповідного маршруту дорівнює 4, отже маршрут $S4$ має дві довжини (10,4).

Далі на основі сформованих маршрутів $S1$, $S2$, $S3$, $S4$ будемо формувати всілякі можливі маршрути рангу, котрий більше поточного на одиницю, до вершин 1- 4. Довжини цих вершин за вагами не перевищують 10, тобто із вершини 1 можна потрапити до вершин 2, 3 та 4.

Таблиця 4.1 – Формування маршрутів графу

Номери вершин	1	2	3
1	Маршрут S1(0,0) 1	Маршрут S21(20,2) Маршрут S31(20,0) 1	Маршрут S321(40,2) 1
2	Маршрут S2(0,0) 2	Маршрут S12(20,2) Маршрут S32(20,0) Маршрут S42(10,7) 2	Маршрут S312(40,2) Маршрут S132(40,2) 2
3	S3(20,0) 3	Маршрут S13(20,0) Маршрут S23(20,0) 3	Маршрут S213(40,2) Маршрут S123(40,2) 3
4	Маршрут S4(10,4) 4	Маршрут S24(10,7) 4	4

Формуємо маршрут $S12$ та розраховуємо його довжину. Визначаємо x_1 x_2 треба зясувати вони єі чи ні. Якщо вони є, при множинні $x_1 x_2$ дорівнює 20, тому, відстань $S12 = 20$. Далі визначаємо x_1 і x_2 чи вони присутні. Якщо, змінні x_1 і x_2 присутні в обмеженнях у вигляді їх добутку, з коефіцієнтом 2, то, відстань $S12 = 4$, а значить відстань $S12 =$ (20 та 2).

Будуємо $S13$ та визначаємо відстані. Визначаємо присутність x_1 і x_3 Якщо при x_3 коефіцієнт 20, а спільно не має x_1 і x_3 то $S13 = 20$. Визначаємо x_1 і x_3 так же само. Тому тривалість маршруту 0, тобто маршрут $S13$ (20 та 0).

Будуємо $S14$ та визначаємо тривалість маршруту. Вивіряємо присутність x_1 і x_4. При x_4, коефіцієнт 10, а при іх сумісному добутку $x_1 x_4$ коефіцієнт 13, то, $S14 = 23$. Вивіряємо чи присутні змінні x_1 та x_4. Якщо присутні, при x_4 коефіцієнт 4, а при їх добутку коефіцієнт 7. Тому, $S14 = 11$. Тому, тривалість маршруту $S14 = 11$, отже $S14 =$ (23 та 11).

Далі будуємо всі можливі відстані від вершини 2, це вершини 1,3 і 4.

Будуємо відстань $S21$ і розрахуємо його довжини. Вивіряємо

присутність змінних x_1 і x_2, при їх добутку $x_1 x_2$ коефіцієнт 20, тому відстань $S21 = 20$. Вивіряємо, присутність в обмеженнях x_1 і x_2. Якщо x_1 і x_2 присутні то у їх добутку, (коефіцієнт 2), тому, відстань $S21 = 2$, тобто відстань $S21$ = (20 та 2).

Будуємо відстань $S23$ та розраховуємо його довжини. Визначаємо в функціоналі присутність змінних x_2 і x_3. Якщо пр x_3 коефіцієнт 20, а x_2 та x_3 відсутні, то, відстань $S23 = 20$. Вивіряємо присутності змінних x_2 і x_3 в обмеженнях. Якщо вони відсутні, то $S23 = 0$, тобто $S23$ = (20 та 0).

Будуємо відстань $S24$. Вивіряємо присутності x_2 і x_4, якщо при x_4, стоїть 10, а рузультату добутка $x_2 x_4$ немає, то $S24$ =10. Вивіряємо x_2 і x_4 є чи ні. Якщо, є x_4, з коефіцієнтом 4, а в x_2 коефіцієнт 3, то, $S24 = 7$. Тому, тривалість $S24 = 7$, тобто $S24$ = (10 та 7).

Далі будуємо маршрути від вершини 3, це маршрути 1,2 і 4.

Будуємо відстань $S31$ та вивіряємо x_1 і x_3. Якщо при x_3 коефіцієнт 20, а x_1 і x_3 відсутні, тому, $S31 = 20$. Вивіряємо присутність x_1 та x_3. Тому $S31 = 0$, тобто маршрут $S31$ = (20 та 0).

Формуємо маршрут $S21$ і визначаємо його довжини. Перевіряємо, в функціоналі змінні x_1 і x_2 окремо або спільно присутні чи ні? Так, при добутку $x_1 x_2$ стоїть коефіцієнт 20, отже, маршрут $S21$ по вагах функціоналу має вагу рівну 20. Перевіряємо, в обмеженнях змінні x_1 і x_2 окремо або спільно присутні чи ні? Так, в обмеженнях змінні x_1 і x_2 присутні у вигляді добутку $x_1 x_2$, при якому стоїть коефіцієнт 2, отже, маршрут $S21$ по вагах обмежень має вагу рівну 2, тобто маршрут $S21$ має дві довжини (20,2).

Формуємо маршрут $S23$ і визначаємо його довжини. Перевіряємо в функціоналі змінні x_2 і x_3 окремо або спільно присутні чи ні? Так, при змінній x_3 стоїть коефіцієнт 20, а спільно змінні x_2 і x_3 в функціоналі не присутні, отже, маршрут $S23$ по вагах функціоналу має вагу рівну 20. Перевіряємо в обмеженнях змінні x_2 і x_3 окремо або спільно присутні чи ні?

Ні, отже, довжина маршруту $S23$ по вагах обмежень дорівнює 0, тобто маршрут $S23$ має дві довжини (20,0)

Формуємо маршрут $S24$ і визначаємо його довжини. Перевіряємо в функціоналі змінні x_2 і x_4 окремо або спільно присутні чи ні? Так, при змінній x_4, що стоїть окремо стоїть коефіцієнт 10, а добутку $x_2 x_4$ в функціоналі немає, значить, маршрут $S24$ по вагах функціоналу має вагу рівну 10. Перевіряємо в обмеженнях змінні x_2 і x_4 окремо або спільно присутні чи ні? Так присутні, при змінної x_4, що стоїть окремо стоїть коефіцієнт 4, а при добутку x_2 в обмеженнях стоїть коефіцієнт 3, значить, маршрут $S24$ по вагах обмежень має вагу рівну 4 + 3 = 7. Отже, довжина маршруту $S24$ по вагах обмежень дорівнює 7, тобто маршрут $S24$ має дві довжини (10,7).

Далі формуємо всі можливі маршрути від вершини 3. Так з вершини 3 ми можемо потрапити в вершини 1,2 і 4.

Формуємо маршрут $S31$ і визначаємо його довжини. Перевіряємо в функціоналі присутність змінних x_1 і x_3 Уи змінній x_3 стоїть коефіцієнт 20, а спільно змінні x_1 і x_3 в функціоналі не присутні, отже, маршрут $S31$ по вагах функціоналу має вагу рівну 20. Перевіряємо в обмеженнях змінні x_1 і x_3 окремо або спільно присутні чи ні? Ні, отже, довжина маршруту $S31$ по вагах обмежень дорівнює 0, тобто маршрут $S31$ має дві довжини (20,0).

Формуємо маршрут $S32$ і визначаємо його довжину. Перевіряємо в функціоналі присутність змінних x_2 і x_3. У змінній x_3 стоїть коефіцієнт 20, а спільно змінні x_2 і x_3 в функціоналі не присутні, отже, маршрут $S32$ по вагах функціоналу має вагу рівний 20. Перевіряємо в обмеженнях присутність змінних x_2 і x_3. Ні, отже, довжина маршруту $S32$ по вагах обмежень дорівнює 0, тобто маршрут $S32$ має дві довжини (20,0).

Формуємо маршрут $S34$ і визначаємо його довжини: перевіряємо в функціоналі присутність змінних x_3 і x_4. У змінній x_4, що стоїть окремо

стоїть коефіцієнт 10, при змінній x_3, що стоїть окремо стоїть коефіцієнт 20, а при добутку $x_3\,x_4$ в функціоналі стоїть коефіцієнт 17. Отже, маршрут $S34$ за вагами функціоналу має вагу рівну 10 + 20 + 17 = 47. Перевіряємо в обмеженнях присутність змінних x_3 і x_4. Вони присутні, у змінній x_4, що стоїть окремо стоїть коефіцієнт 4, а при добутку $x_3\,x_4$ в обмеженнях стоїть коефіцієнт 10, отже, маршрут $S34$ по вагах обмежень має вагу рівну 4 + 10 = 13, отже, довжина маршруту $S34$ по вагах обмежень дорівнює 13, тобто маршрут $S34$ має дві довжини (47,13). Але він не відповідає обмеженням, отже виключається з розгляду.

Аналогічно формуються формуємо всі маршрути від вершини 4, з якої можна прийти до вершин 1,2 та 3.

Формуємо маршрут $S41$ і визначаємо його довжини аналогічним попередньому розгляду чином.. У змінній x_4, що стоїть окремо – коефіцієнт 10, а при добутку $x_1\,x_4$ в функціоналі – коефіцієнт 13. Отже, маршрут $S41$ має вагу 10 + 13 = 23. При перевірці змінних x_1 та x_4 виявилм, що змінній x_4 , відповідає коефіцієнт 4, а добутку $x_1\,x_4$ – коефіцієнт 7. Отже, маршрут $S41$ має вагу 7 + 4 = 11, тобто маршрут $S41$ має дві довжини (23,11), але оскільки довжина по вагам обмежень перевищує обмеження, то цей маршрут виключаємо.

Формуємо маршрут $S42$ і визначаємо його довжини. Перевіряємо в функціоналі присутність змінних x_2 і x_4. У окремій змінній x_4,є коефіцієнт 10, а добутку $x_2\,x_4$ в функціоналі немає, тому вага дорівнює тільки 10. Перевіряємо в обмеженнях присутність змінних x_2 і x_4. У окремій змінній є коефіцієнт 4, а у добутку $x_2\,x_4$ є коефіцієнт 3. Отже, маршрут $S42$ має вагу обмежень 4 + 3 = 7, тому маршрут $S42$ має дві довжини (10,7).

Формуємо маршрут $S43$ і визначаємо дві його довжини. Перевіряємо в функціоналі присутність змінних x_3 і x_4. У змінній x_4, що стоїть окремо стоїть коефіцієнт 10, при x_3, що стоїть окремо стоїть коефіцієнт 20, а добуток

$x_3\,x_4$ в функціоналі стоїть коефіцієнт 17. Отже, маршрут $S43$ по функціоналу має вагу 10 + 20 + 17 = 47. Далі проводимо перевірку в обмеженнях присутності змінних x_3 і x_4. У змінній x_4, що стоїть окремо стоїть коефіцієнт 4, а у добутку $x_3\,x_4$ в обмеженнях є коефіцієнт 10, отже, маршрут має вагу 4 + 10 = 13. Отже, довжина маршруту $S43$ по вагах обмежень дорівнює 13, тобто маршрут має дві довжини (47,13), але він не задовольняє обмеженням.

Отже, сформувані маршрути другого ярусу, переходимо до побудови всіляких можливіих маршрутів третього рангу. Починаємо з множини 1, яке містить два маршрути: $S21(20,2)$ і $S31(20,0)$. Далі переходимо до множин 2,3 і 4, формуючи при цьому всілякі можливі маршрути з множини 1 в 2: Маршрут $S21(20,2)$ викидаємо, тому що вершина 2 є у маршрут $S21(20,2)$, отже нам залишилося вибрати тільки один маршрут, що залишився $S31(20,0)$. В результаті отримаємо маршрут $S312$ довжина якого визначається наступним чином: перевіряємо в функціоналі присутність змінних x_1, x_2 и x_3, окремо і в комбінаціях x_1x_2; x_1x_3; x_2x_3 або у вигляді співмножника $x_1x_2x_3$. Є змінна x_3 з коефіцієнтом 20, при цьому є співмножник x_1x_2 з коефіцієнтом 20, отже, довжина маршруту $S312$ є такою: 20 + 20 = 40. Аналогічно довжина маршруту $S312$ по обмеженням дорівнює 2, тобто даний маршрут $S312$ володіє двома довжинами (40,2).

Аналогічно формуються всілякі можливі маршрути виходу з множини 1 до множини 3 та виходу з множини 1 в до множини 4.

На ранзі, що буде розглядатися за поточним рангом, не вдається побудувати ні одного маршруту, тому що вони не задовольняють обмеженням завдання. Отже, найдовгішим маршрутом є маршрут $S213$. Цей маршрут отримав дві довжини (40,2), тому вектор, який відповідає максимуму функціоналу (3.5), рівному 40, є таким: X=$\{x_1=1,\ x_2=1,\ x_3=1,\ x_4=0\}$, зазначимо, що для цього рішення значення обмеження буде не

більшим, ніж двійка. У випадку дублюючих один одного маршрутів на ярусі процес формування маршрутів буде відповідати варіанту, що наведений в таблиці 4.2.

Таблиця 4.2 – Побудови маршрутів графу з однаковими підмножинами вершин

Номери вершин	1	2	3
1	Маршрут S1(0,0) 1	Маршрут S31(20,0) 1	 1
2	Маршрут S2(0,0) 2	Маршрут S12(20,2) Маршрут S32(20,0) Маршрут S42(10,7) 2	Маршрут S312(40,2) 2
3	Маршрут S3(20,0) 3	 3	 3
4	Маршрут S4(10,4) 4	 4	 4

Якщо є m обмежень, то кожен маршрут буде описуватися більшою, ніж 2, кількістю довжин, тоді треба будерозраховувати за аналогічним принципом $m+1$ довжину. Тепер на першому ярусі сформуємо маршрути не рангу 1, а множини маршрутів у відповідності до функціоналу. Принцип отримання максимального маршруту буде аналогічним, а рішення завдання побачимо у таблиці 4.3.

Зазначимо, що аршрути $S41(33,11)$ та $S43(47,14)$ на ярусі 1 не формувалися, тому що вони не відповідали обмеженням.

Порівнючі таблицю 4.1 та 4.2 з таблиці 4.3 можемо побачити, що кількість маршрутів при оптимізації 2-го типу є меншою, тому зменшується і часова складність виконання процедури.

Проведений порівняльний аналіз обох варіантів оптимізації по

критеріям оцінки часової складності і похибки, ймовірності рішення задачі $P(T) = 1 - e^{-\frac{T_д}{T}}$, за деякий допустимий час $T_д$. Результати експериментального дослідження наведені нижче на рисунках 4.8 - 4.11.

Таблиця 4.3 – Рішення задачі

Номери вершин	1	2	3
1	Маршрут S21(20,2) 1	Маршрут S31(20,0) 1	 1
2	 2	Маршрут S32(20,0) Маршрут S42(10,7) 2	Маршрут S312(40,2) 2
3	Маршрут S3(20,0) 3	Маршрут S213(20,2) 3	 3
4	Маршрут S4(10,4) 4	 4	 4

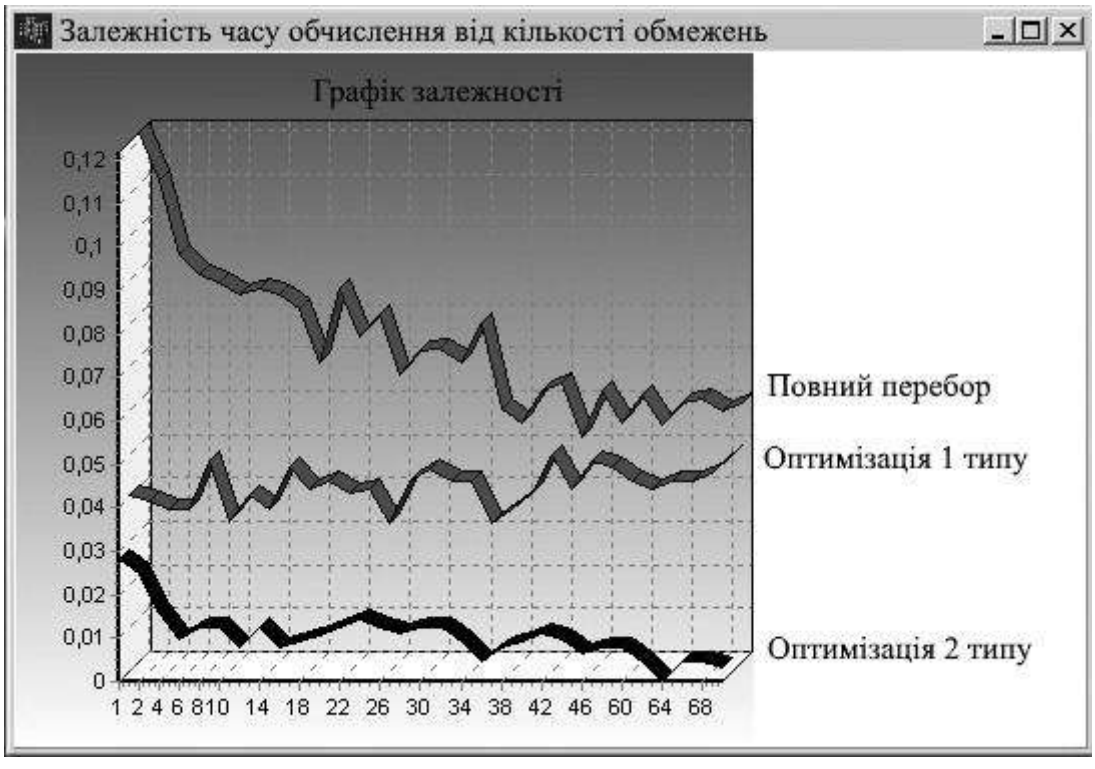

Рисунок 4.8 – Залежність часу обчислення від кількості обмежень

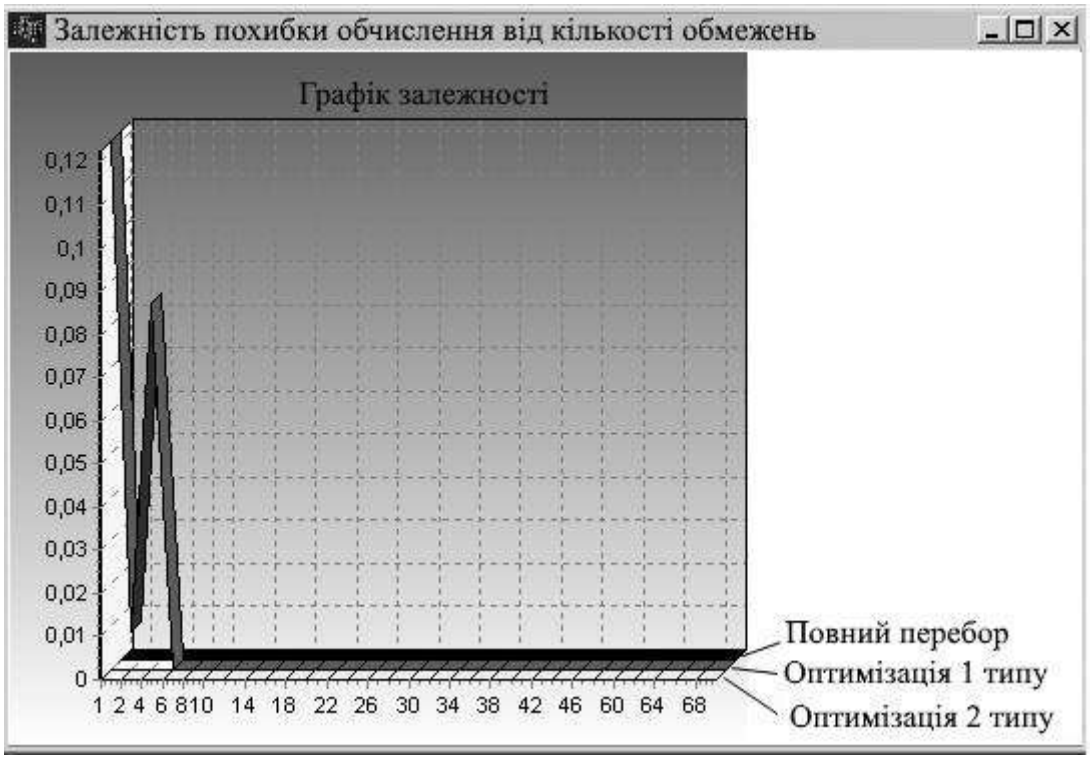

Рисунок 4.9 – Залежність похибки обчислення від кількості обмежень

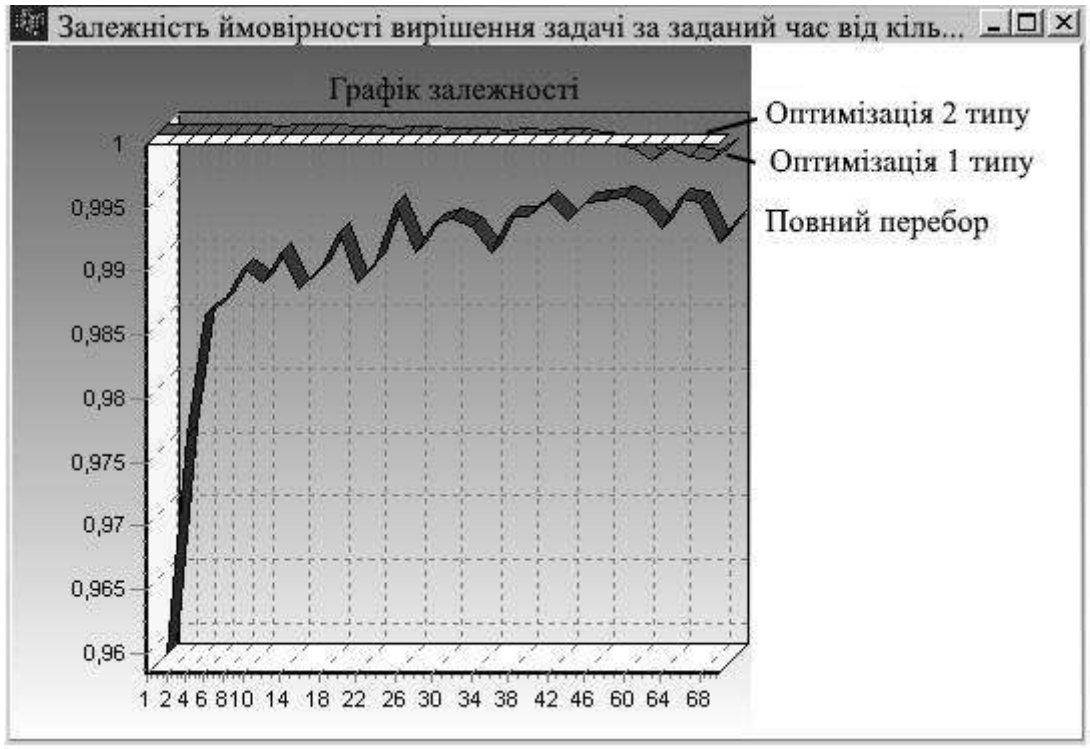

Рисунок 4.10 – Аналіз ймовірності виконання завдання у надані часові

терміни

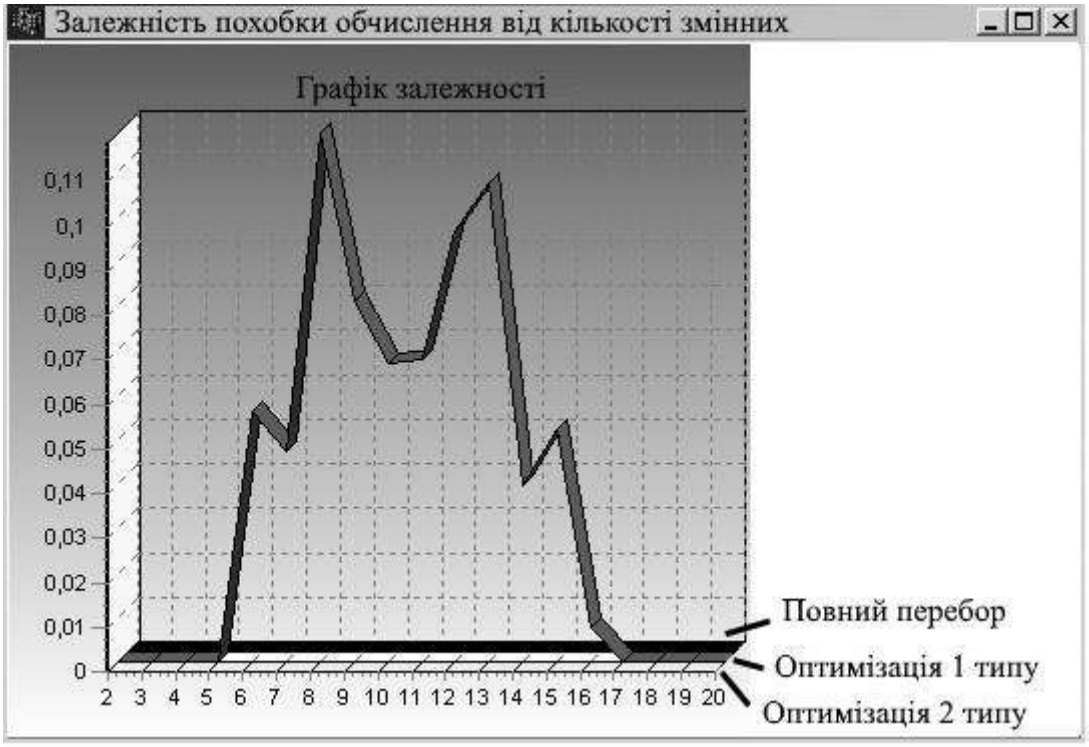

Рисунок 4.11 – Залежність похибки обчислення від кількості змінних

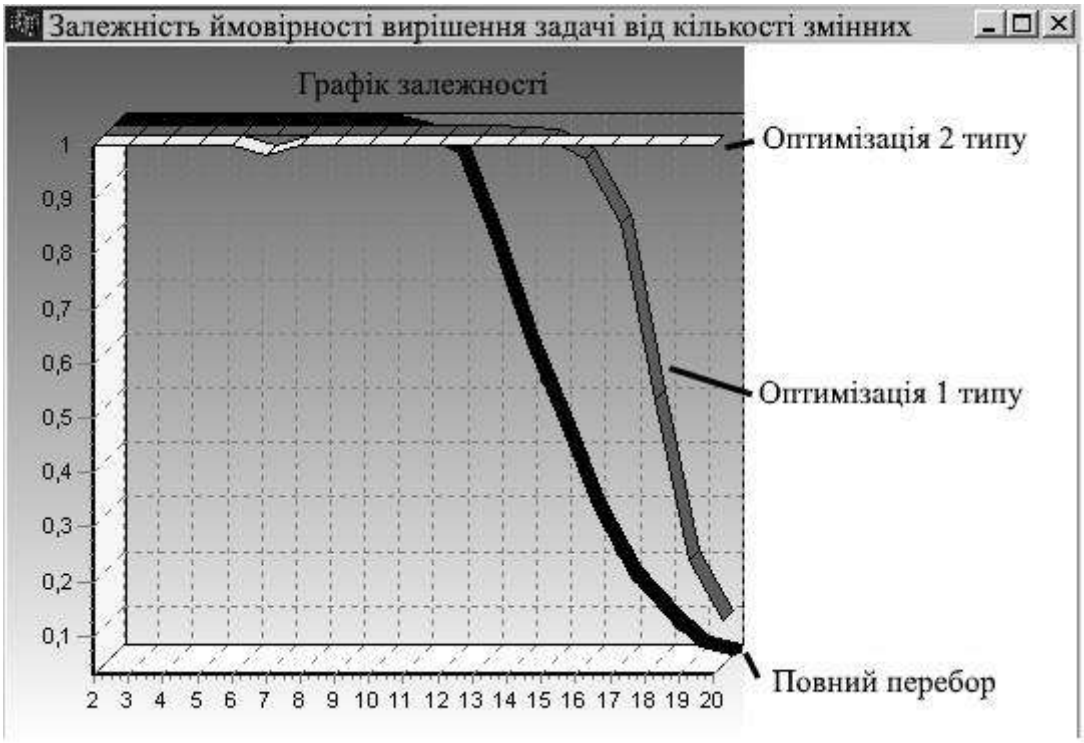

Рисунок 4.12 – Залежність ймовірності виконання завдання від

кількості змінних

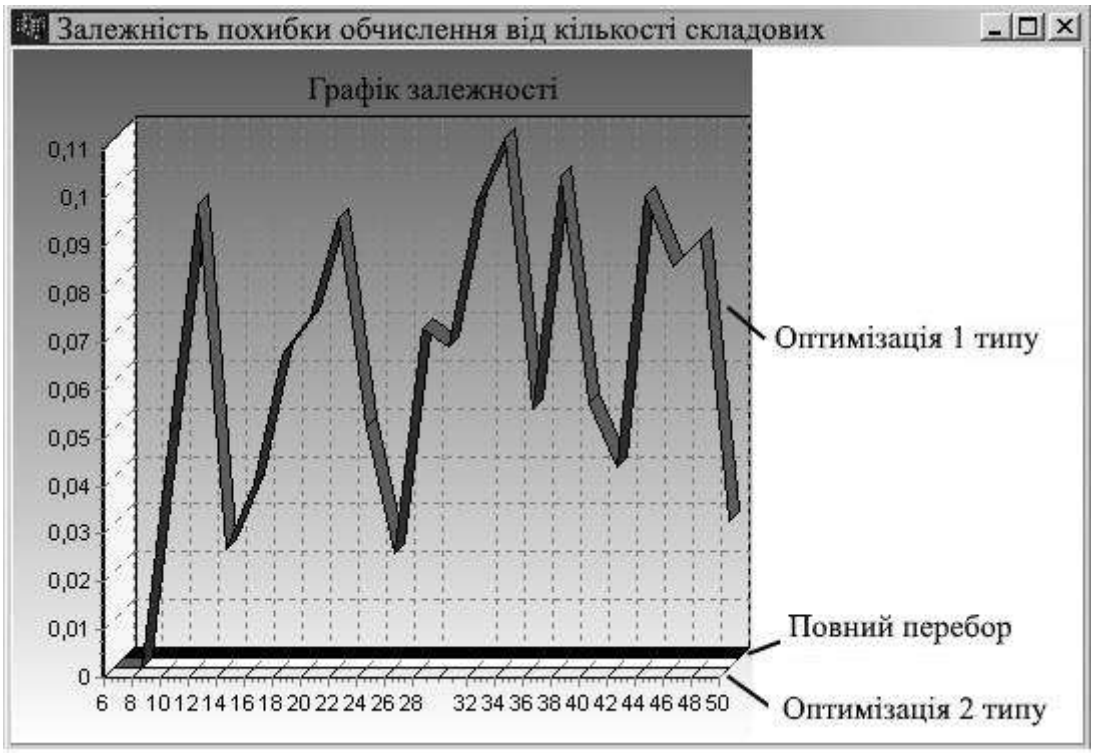

Рисунок 4.13 – Залежність похибки обчислення від кількості складових

Експериментальні дослідження проводились з різною кількістю обмежень, котра змінювалося від 5 до 42, кількість змінних варіювалося від 2 до 20, кількість доданків в функціоналі і обмеженнях змінювалося від 3 до 30. Коефіцієнти функціоналу та обмежень змінювалися у відповідності рівномірному закону розподілу.

ВИСНОВКИ

1 Проведений аналіз сучасних методів планування та виконання завдань в розподіленій обчислювальній системі. Показані переваги використання Grid технологій, визначена послідовність обробки запитів в кластері та наведені характеристики даного процесу.

2 Проведено моделювання процесу планування обробки даних в розподілених обчислювальних системах. Запропонована математична модель даного процесу, що базується на відображенні декартова добутку матриць характеристик завдань на ресурсну матрицю. Також при моделюванні використовувався метод групової вибірки з індивідуальною сегментацією.

3 Удосконалений метод планування завдань в розподіленій обчислювальній системі, за рахунок використання Grid-технології. У методі використаний ранговий підхід до вирішення отриманої задачі нелінійного булевого програмування.

4 Проведено дослідження методу планування обробки даних в розподілених обчислювальних системах. Визначена оцінка часової складності складових методу. Також визначені залежності похибок відповідних алгоритмів та ймовірнісні характеристики від розмірності розв'язуваної задачі булевого програмування, кількості обмежень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A. Kovalenko, R. Miroshnychenko, A. Martyntsov Distributed computing systems based on the use of Grid technologies. Системи управління, навігації та зв'язку. – Полтава : Полтавський національний технічний університет імені Юрія Кондратюка, 2023. – Вип. 1. – С. 101-103. doi: 10.26906/SUNZ.2023.1.101

2. Foster I. and Taylor S. Strand: New Concepts in Parallel Programming. – Prentice-Hall, Inc., 1990. – 378 p.

3. Martin F. Maldonado, Grid Technical Architect, Grid computing in higher education: Trends, values and offerings, December 2004, http://www-1.ibm.com/grid/pdf/grid_computing_in_higher_ed.pdf

4. Moore R. and Baru C., Virtualization Services for Data Grids // Grid Computing: Making the Global Infrastructure a Reality. – John Wiley & Sons Ltd., 2003. – P. 398-410.

5. The Grid Application Development Software (GrADS) Project. Available from <http://hipersoft.cs.rice.edu/grads/>.

6. The Grid2003 Project. The Grid2003 Production Grid: Principles and Practice. iVDGL: Technical Report. – 2004. – 42 p.

7. Ian Foster. What Is The GRID? A Three Point Checklist. GRID Today, July 22, 2002: Vol. 1 No. 6, http://www.gridtoday.com/02/0722/100136.html. http://www.gridclub.ru/library/publication.2004-11-29.5830756248

8. Робота з шаблонами SnapML: виявлення об’єктів [Електронний ресурс] – Режим доступу до ресурсу: https://heartbeat.fritz.ai/working-with-snapml-templates-object-detection-a5760d2fe241

9. Object Detection – робота з шаблонами [Електронний ресурс] – Режим доступу до ресурсу: https://lensstudio.snapchat.com/templates/ml/object-detection/

10. SDK доповноеної реальності [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ru/articles/web-ar-tools-overview.html

11. Огляд Vuforia SDK [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/2017.2/Documentation/Manual/vuforia-sdk-overview.html

12. Object & Scene Tracking Augmented Reality [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikitude.com/augmented-reality-object-scene-recognition/

13. Що таке Colaboratory [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com/notebooks/welcome.ipynb?hl=ru#scrollTo=5fCEDCU_qrC0

14. Google Colab - ваш робочий простір на Python в хмарному середовищі [Електронний ресурс] – Режим доступу до ресурсу: https://www.machinelearningmastery.ru/google-colab-your-python-workspace-on-cloud-c3aed424de0d/

15. Просте введення в Pytorch для нейронних мереж [Електронний ресурс] – Режим доступу до ресурсу: https://www.machinelearningmastery.ru/an-easy-introduction-to-pytorch-for-neural-networks-3ea08516bff2/

16. Що таке TensorFlow і як це використовується? [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerworld.ru/articles/Chto-takoe-TensorFlow-i-kak-eto-ispolzuetsya

17. Початок роботи з набором даних COCO [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/getting-started-with-coco-dataset-82def99fa0b8

18. Архітектура IT рішень [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/347204/

19. Довідник UML. Об'єктно-орієнтоване проектування [Електронний ресурс] – Режим доступу до ресурсу: https://openu.ru/Books/UML/Use_case.asp

20. Діаграми пакетів (package diagrams) [Електронний ресурс] – Режим доступу до ресурсу: https://itteach.ru/rational-rose/diagrammi-paketov-komponentov-i-razmescheniya

21. Валідація та верифікація [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.com.ua/baza-znaniy/verifikatsiya-i-validatsiya/