Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Перший (бакалаврський)
(рівень вищої освіти)

Розроблення програмного забезпечення для формування маршруту переміщення свердла для верстатів з числовим програмним керуванням
(тема)

Виконав:
студент _4_ курсу, групи АКТСІ-20-2
Ключник Є.С.
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Системна інженерія
(повна назва освітньої програми)
Керівник доц. Бронніков А.І.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАМ _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики та комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти Перший (бакалаврський)
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентці Ключник Євгенії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного забезпечення для формування маршруту переміщення свердла для верстатів з числовим програмним керуванням
Затверджена наказом по університету від 03.06.2024 № 545 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024
3. Вихідні дані до роботи Координати переміщення свердла, алгоритми пошуку маршруту переміщення, задача комівояжера

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз сучасного стану розробок в області INDUSTRY 4.0 та числового програмного керування
4.3.Вибір і обґрунтування технічних засобів для створення програмного забезпечення формування маршруту переміщення свердла
4.4 Розроблення програмного забезпечення для формулювання маршруту переміщення свердла для верстатів з числовим програмним керуванням
4.5 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайди у форматі Power Point у кількості 11 слайдів з розширенням .pptx

6. Консультанти розділів роботи

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	Приміт-ка
1	Аналіз технічного завдання	09.04.2024	виконано
2	Опрацювання літератури за темою роботи.	10.04.2024	виконано
3	Виконання розділу 1 Аналіз сучасного стану розробок в області INDUSTRY 4.0 та числового програмного керування	12.04.2024	виконано
4	Виконання розділу 2 Вибір і обґрунтування технічних засобів для створення програмного забезпечення формування маршруту переміщення свердла	15.05.2024	виконано
5	Виконання розділу 3 Розроблення програмного забезпечення для формулювання маршруту переміщення свердла для верстатів з числовим програмним керуванням	18.05.2024	виконано
6	Оформлення пояснювальної записки	25.05.2024	виконано
7	Оформлення презентації	28.05.2024	виконано
8	Подання роботи на рецензію	16.06.2024	виконано
9	Подання роботи на підпис зав. кафедри	18.06.2024	виконано
10	Подання атестаційної роботи в ЕК	20.06.2024	

Дата видачі завдання 09.04.2024

Студент _____ _____
(підпис) (прізвище, ініціали)

Керівник роботи _____ доц. Бронніков А.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота містить: 66 с., 21 рис., 3 таблиці, 27 джерел, 2 додатки.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАРШРУТ, ЗАДАЧА КОМІВОЯЖЕРА, АЛГОРИТМ ЗБЕРЕЖЕННЯ, OR-TOOLS, PYTHON

Мета кваліфікаційної роботи – автоматизація процесу планування траєкторій свердління,що забезпечить зменшення часу налаштування обладнання та підвищення точності обробки деталей.

Об’єкт розробки – технологічні операції верстатів з числовим програмним забезпеченням.

Предмет розробки – операція свердління за допомогою верстата з числовим програмним забезпеченням.

В кваліфікаційній роботі розглянуто актуальні питання за темою, запропоновані рішення з пошуку маршруту переміщення свердла в процесі свердління за допомогою верстату з числовим програмним керуванням на сучасному виробництві, розроблено програмне забезпечення пошуку найкоротшого маршруту переміщення свердла.

За результатами роботи опубліковано статтю у збірнику ADED-2023 (2) та Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті».

ABSTRACT

The qualification work contains: 66 pages, 21 figures, 3 tables, 27 sources, 2 appendices.

SOFTWARE, ROUTE, TRAVELER'S PROBLEM, SAVING ALGORITHM, OR-TOOLS, PYTHON

The purpose of the qualification work is to develop software for the formation of the shortest drill movement route for numerically controlled machines by solving the salesman's problem.

The object of development is technological operations of machines with numerical software.

The subject of development is a drilling operation using a machine tool with numerical software.

In the qualification paper, topical questions on the topic were considered, solutions were proposed for finding the route of moving the drill in the drilling process using a machine with numerical software control in modern production, software for finding the shortest route of moving the drill was developed.

Based on the results of the work, an article was published in the collection ADED-2023 (2) and International forum of young scientists «Radio electronics and youth in the xxi century».

ЗМІСТ

Перелік скорочень і термнів 7
Вступ 8
1 Аналіз сучасного стану розробок в області INDUSTRY 4.0 та числового програмного керування 10
1.1 Industry 4.0 та її верстати 10
1.1.1 Поняття та концепція Industry 4.0 10
1.1.2 Верстати Industry 4.0 14
1.2 Операція свердління за допомогою ЧПК 22
1.3 Вирішення задачі маршрутизації транспортних засобів 25
1.3.1 Проблема маршрутизації транспортних засобів 25
1.3.2 Різні методи вирішення задачі комівояжера 27
1.3.3 Стратегії першого рішення 30
2 Вибір і обґрунтування технічних засобів для створення програмного забезпечення формування маршруту переміщення свердла 32
2.1 Вибір мови програмування для створення програмного забезпечення ... 32
2.2 Бібліотека OR-tools та завдання пошуку маршруту 36
2.3 Розрахунок завдання пошуку найкоротшого маршруту руху свердла ... 39
3 Розроблення програмного забезпечення для формулювання маршруту переміщення свердла для верстатів з числовим програмним керуванням 47
3.1 Розроблення блок-схеми алгоритму роботи програми 47
3.2 Розроблення програмного забезпечення 48
3.3 Охорона праці при роботі з електрообладнанням 59
Висновки 63
Перелік джерел посилання 64
Додаток А Текст програми 67
Додаток Б Демонстраційний матеріал 73

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМНІВ

АОМК – алгоритм оптимізації мурашиної колонії;

АОЧК – алгоритм оптимізації частинок колонії;

ДО – дослідження операцій;

ЗК – задача комівояжера;

ІКТ – інформаційно-комунікаційні технології;

КФС – кіберфізичні системи;

ЛМІ – людино-машинні інтерфейси;

МНВ – місцева найдешевша вставка;

МТЗ – маршрутизації транспортних засобів;

НОДШ – найбільш обмежена дуга шляху;

НШ – найдешевший шлях;

ООП – об’єктно-орієнтоване програмування;

ПНВ – паралельна найдешевша вставка;

ПрК – проблема комівояжера;

ЧПК – числове програмне керування;

CAD – система автоматизованого проєктування;

CAM – автоматизована система технологічної підготоовки виробництва;

IoS – інтернет послуг;

IoT – інтернет речей;

RIFD – радіочастотна ідентифікація.

ВСТУП

Industry 4.0 обіцяє кардинально трансформувати виробництво та виробничу інфраструктуру. Його назва походить від його потенціалу розпочати четверту промислову революцію – сміливу мету, якщо взяти до уваги масштаби революцій у паровій енергії, конвеєрному виробництві та комп'ютерній автоматизації, які їй передували.

Четверта промислова революція, Індустрія 4.0, характеризується впровадженням нових концепцій у виробничі системи. Багато віртуальних систем дозволяють передбачати погані умови, економити енергію, вивчати особливі випадки тощо, але вони потребують впровадження нових цифрових інструментів, які дозволяють краще розвивати виробничий процес. Таким чином, запрограмовані алгоритми можна оцінити, щоб знати продуктивність виробничого процесу.

У наш час важливо, де на друкованій платі починається свердління і який маршрут проходить свердло у процесі для зменшення часу виробництва.

Таким чином, актуальним завданням у наш час є забезпечення виконання виробничих функцій для підвищення ефективності виробництва за рахунок забезпечення швидкої роботи верстатів з числовим програмним керуванням, зокрема для операцій свердління.

Метою кваліфікаційної роботи є автоматизація процесу планування траєкторій свердління, що забезпечить зменшення часу налаштування обладнання та підвищення точності обробки деталей.

Об'єкт розробки – технологічні операції верстатів з числовим програмним забезпеченням.

Предмет розробки – операція свердління за допомогою верстата з числовим програмним забезпеченням.

Задачами, що потрібно вирішити у атестаційній роботі є:

– провести аналіз сучасного стану розробок в області Industry 4.0 та числового програмного керування;

– провести вибір і обґрунтування технічних засобів для створення програмного забезпечення формування маршруту переміщення свердла;

– провести розрахунок шляху свердла для п’яти точок;

– розробити блок-схему алгоритму для створення програмного забезпечення;

– розробити програмне забезпечення для формування найкоротшого маршруту переміщення свердла;

– розглянути питання охорони праці;

– оформити пояснювальну записку згідно [1] та [2].

Дана атестаційна робота пройшла апробацію у журналі ADED-2023 (2) [3]та Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» [4].

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗРОБОК В ОБЛАСТІ INDUSTRY 4.0 ТА ЧИСЛОВОГО ПРОГРАМНОГО КЕРУВАННЯ

1.1 Industry 4.0 та її верстати

1.1.1 Поняття та концепція Industry 4.0

В останні десятиліття розвиток інформаційно-комунікаційних технологій (ІКТ) та їх інтеграція у виробничі процеси принесла переваги всьому ланцюжку створення вартості. Еволюція можливостей цих технологій сприяє підвищенню промислової продуктивності, зниженню виробничих витрат і забезпеченню ефективних рішень для якісного, швидкого та рентабельного обслуговування клієнтів [5]. Зіткнувшись із цими останніми технологічними розробками та сценарієм, у якому зростає попит на індивідуальні продукти, більшу складність, вищу якість і знижені витрати; поява нової індустріальної моделі обговорюється в усьому світі під темою Industry 4.0.

Вважаючись деякими науковцями та підприємцями 4 промисловою революцією, Industry 4.0 є одним із термінів, які використовуються для опису стратегії високих технологій. Маючи вплив на всіх рівнях виробничих систем, вона охоплює набір найсучасніших технологій, пов’язаних з Інтернетом, щоб зробити їх більш гнучкими та зручними для співпраці. У цьому контексті такі технології, як кіберфізичні системи (КФС) [6-7], що самоорганізуються, контролюють процеси та створюють віртуальну копію реального світу. Інтернет речей (IoT) з’єднує машини, об’єкти та людей у режимі реального часу, а хмарні обчислення пропонують рішення для зберігання, а також обмін інформацією та керування нею, що дозволяє поєднувати виробничі та бізнес-процеси для створення цінності для організацій. Ці нові виробничі структури, оснащені розумними пристроями, підключеними до мережі, де продукти, машини та виробничі системи отримують комунікаційні можливості, є

інтелектуальними фабриками майбутнього та є ключем до досягнення рівня гнучкості для подолання сучасних викликів. варіативність, налаштування та скорочення життєвих циклів продукту.

У минулих століттях промисловість пройшла через масштабні зміни, які революціонізували спосіб виробництва продукції та принесли різноманітні переваги, головним чином пов'язані зі збільшенням продуктивності. Сьогодні, після трьох промислових революцій, поєднання передових технологій та Інтернету знову трансформує промисловий ландшафт і називається 4 промисловою революцією або Industry 4.0.

Концепція враховує руйнівний потенціал інтеграції фізичних об'єктів в інформаційну мережу, яка революціонізує традиційну індустрію трансформації [8]. У цілісному розумінні підхід Industry 4.0 являє собою зміну парадигми у виробничих процесах і бізнес-моделях, встановлюючи новий рівень розвитку та управління для організацій.

У контексті Industry 4.0 КФС складається з розумних об'єктів (наприклад, машин, продуктів або пристроїв), які автономно обмінюються інформацією, працюючи у співпраці з фізичним світом навколо них. Розумні продукти, ідентифіковані за допомогою тегів радіочастотної ідентифікації (RIFD), надають інформацію про їх місцезнаходження, історію, статус і маршрути. Ця інформація дозволяє робочим станціям знати, які етапи виробництва виконуються для кожного продукту та потрібна адаптація. для виконання конкретного завдання. IoT полегшує весь цей процес. IoT підключає всі ці пристрої до мережі Інтернет, дозволяючи збирати складні та великі набори даних і обмінюватися інформацією в режимі реального часу. Наприклад, через IoT це можна виконувати моніторинг продуктивності промислового обладнання. З інформації, яку генерує обладнання, можна виявити невидимі проблеми, такі як деградація машини або знос компонентів. Інтернет послуг (IoS) представляє подібний підхід. За допомогою таких технологій, як великі дані та хмарні обчислення, можна забезпечити масштабовану обчислювальну потужність для зберігання та аналізу даних із

багатьох джерел і клієнтів для підтримки прийняття рішень, оптимізації операцій, економії енергії та покращення продуктивності системи, від проектування до розподіл. Крім того, дозволити співробітникам отримати доступ до них з будь-якого місця, через планшет або смартфон.

Розумна фабрика – це провідне в галузі рішення для досягнення гнучких виробничих цілей, таких як вимоги до виходу на ринок, цільові обсяги виробництва та стратегія економії [9] у динамічному виробничому середовищі та зростаючій складності. Завдяки взаємодії вищезгаданих технологій можна буде перейти від централізованої моделі виробництва до децентралізованої моделі, де матеріали та обладнання можуть спілкуватися один з одним і приймати автономні рішення. Результатом стануть розумні мережі створення вартості, здатні майже автоматично реагувати на зміни ринку. На цих підключених фабриках майбутнього фізичні прототипи будуть менш важливими. Удосконалення обчислювального потенціалу (кластери та хмара) збільшує потенціал підходів моделювання як важливих інструментів аналізу та зворотного зв'язку для підтримки прийняття рішень у реальному часі. Завдяки загальнозаводському датчику можна буде використовувати дані в реальному часі, щоб побудувати фізичний світ у віртуальну модель, яка може включати машини, продукти та людей. Це дозволить операторам тестувати та оптимізувати конфігурації машини для наступного поточного продукту у віртуальному світі до їх фізичного переходу, тим самим скорочуючи час конфігурації машини та підвищуючи якість, окрім безпеки виробництва з точки зору часу та витрат. . Однак для досягнення бачення, запропонованого Industry 4.0, необхідно інтегрувати різні етапи виробництва.

Таким чином, це сприяє скороченню розриву між різними етапами виробництва, оскільки збір інформації про продукт протягом його життєвого циклу є важливим для підтримки персоналізованих вимог клієнтів.

З дуже амбітним потенціалом Industry 4.0 обіцяє підвищену гнучкість, масове налаштування, підвищену швидкість, кращу якість і підвищену продуктивність у виробництві [10], що дозволяє компаніям справлятися з

глобальними викликами та персоналізованими потребами та залишатися прибутковими. Останні тенденції, такі як глобалізація ринків і потреби задовольнити потреби клієнтів, вимагають виробничого обладнання та процесів, здатних адаптуватися до нових продуктів і варіантів, щоб отримати конкурентоспроможність. У цьому контексті нові інформаційні та комунікаційні технології, які стануть розумною фабрикою майбутнього, можуть допомогти запровадити відповідний режим виробництва для досягнення цих цілей. Наприклад, оскільки датчики, комп'ютери та мережеві машини можуть легко спілкуватися один з одним і зі своїми користувачами в режимі реального часу, виробничі процеси стануть більш видимими та доступними для контролю, зменшуючи кількість відмов, що призводить до підвищення якості та ефективності. Інша тенденція полягає в тому, що коли виробничі компанії та постачальники послуг тісніше співпрацюватимуть, поєднуючи продукти та послуги, вони зможуть продавати свої ноу-хау чи інші послуги замість продажу кінцевих продуктів, дозволяючи іншим компаніям або партнерам використовувати свої навички та знання як послуга для розробки власного продукту. Деякі головні результати цієї нової хвилі промислового виробництва включають:

– масову адаптацію, коли виробництво має бути здатним виробляти дуже невеликі або навіть одиничні партії для задоволення потреб кожного клієнта, покращуючи конкурентоспроможність;

– більшу гнучкість, що дозволяє виробничому ланцюжку швидко реагувати на зміни в вимогах;

– контроль якості, завдяки наскрізній видимості можна буде оптимізувати процеси, запаси, ресурси та, як наслідок, скоротити витрати;

– нові можливості з появою інноваційних бізнес-моделей і послуг, які сприяють новим формам взаємодії ланцюжка вартості (наприклад, цифрові платформи);

– оптимізовані прийняття рішень завдяки використанню розумних продуктів і пристроїв, здатних знати та оновлювати їх статус, дозволяючи відстежувати виробництво в режимі реального часу;

– покращені людино-машинні інтерфейси (ЛМІ), які включають співіснування з роботами та нові способи взаємодії та роботи на фабриці, що дозволяє працівникам мати більш гнучкою та диверсифікованою кар'єрою, щоб вони довше залишалися продуктивними.

1.1.2 Верстати Industry 4.0

Верстати є частиною Industry 4.0, що впливають на зміни в ній. Ці зміни в першу чергу стосуються [11]:

– профілактичного обслуговування;

– покращеного використання;

– економії енергії;

– уникнення неналежного використання;

– покращеного забезпечення якості та зміни в житті людей.

Однією з основних характеристик Industry 4.0 є збір даних. Верстати мають датчики, які збиратють багато різних типів даних, включаючи дані про те, скільки машина працювала, умови, в яких вона працювала, і стан компонентів машини. Збираючи та аналізуючи ці дані, машини можуть оцінити, коли компонент потребує заміни. Прогнозне технічне обслуговування може забезпечити більш ефективну роботу машин і запобігти простою.

Дослідження, проведені за останні кілька років, показали той факт, що на промислових підприємствах верстати використовуються неефективно. У загальному часі використання верстатів обробка займає менше 40% (в окремих випадках менше 25%) часу.

Покращене використання інструментальних верстатів можливе шляхом збору та аналізу даних про такі речі, як зміни інструментів, зупинки програми

та утримання подачі. Енергозбереження є важливою характеристикою нових інтелектуальних верстатів, які також можуть збирати дані про споживання енергії, щоб допомогти компаніям заощадити гроші. Industry 4.0 також допоможе визначити, коли машини використовуються неналежним чином. Машина може надіслати сповіщення, якщо виявить умови, що виходять за межі нормальної роботи. Це може допомогти запобігти серйозним помилкам, спричиненим людською помилкою або несправністю, що може уникнути простою, а також травмування працівників. Однією з найважливіших цілей Industry 4.0 у сфері верстатів є підвищення якості. У новому верстаті дуже важливе використання автоматизації, яка також покращує якість, зменшуючи природні коливання, пов'язані з діями людини. Концепція Industry 4.0 змінить основну роль людей-операторів. Багато робіт перемістяться від фізичної роботи до керування ними через комп'ютери, моніторингу даних і забезпечення нагляду до автоматизованих операцій. Це вимагатиме значної перепідготовки людських працівників. Однією з переваг Industry 4.0 для працівників є покращена безпека. Автоматизація дозволяє працівникам здійснювати спостереження, тим самим зменшуючи ризик людської помилки.. Основні цілі, яких слід досягти за допомогою принципів Industry 4.0, це краща інформація – краще прийняття рішень і кращі досягнуті результати.

IoT присутній у різних сферах застосування, таких як [12]:

– розумна їжа/управління водними ресурсами: виявлення якості води, витоків води, річкових паводків і вжиття запобіжних заходів; управління водою в будинках; контроль ланцюга постачання у виробництві води загалом тощо;

– розумне здоров'я: моніторинг фізичної активності, особливо для спортсменів та людей з особливими потребами, таких як інваліди, люди похилого віку; пристрої, що запобігають падінню; контроль температури тощо медичних холодильників; моніторинг хронічних захворювань, спостереження за пацієнтами тощо;

– розумне життя: програми для розумних покупок, за допомогою яких відстежуються звички клієнтів, стан здоров’я, як-от алергія, і термін придатності продукту; метеостанції, які можуть надати прогноз погоди; діяльність, пов’язана з повсякденним життям, наприклад контроль споживання енергії та води тощо;

– інтелектуальний моніторинг навколишнього середовища: виявлення лісових пожеж, контроль забруднення повітря, моніторинг вологості ґрунту, твердої вібрації тощо та запобігання зсувів і лавин; спостереження за тваринами та захист дикої природи; відстеження вантажних транспортних засобів та перевірка вантажів бензину/небезпечних відходів у морях і на узбережжі; і контроль трафіку для навколишнього середовища тощо;

– розумна енергія: розумні мережі, за допомогою яких можна відстежувати споживання енергії; керування системами водного транспорту, відстеження вітрових турбін, вимірювання рівнів радіації, особливо для ядерних установок тощо;

– розумні будівлі: контроль доступу та безпека; кондиціонування повітря в приміщенні (вологість, температура, CO2, освітлення); спостереження за умовами навколишнього середовища в спеціальних місцях, таких як музеї; інтелектуальна пожежна сигналізація, зрошення саду тощо;

– розумне транспортування та мобільність: оплата громадського транспорту та паркування, відстеження умов завантаження, особливо для робіт із особливими вимогами, таких як холодовий ланцюг, визначення розташування товарів у таких зонах, як склади і порти, відстеження парків, керування транспортними засобами, сплата дорожніх зборів, скорочення часу транспортування та викидів CO2 за допомогою відстеження тощо;

– розумна промисловість: відстеження рівня резервуарів для зберігання, розрахунок запасів для силосів, виявлення вибухонебезпечних газів, планування робіт з технічного обслуговування/ремонту шляхом прогнозування погіршення стану, якості повітря на підприємствах,

використання на верстатах з числовим програмним керуванням (ЧПК), що є одним з найактуальніших видів використання, тощо;

– розумні міста: моніторинг місць для паркування та систем направлення/бронювання в режимі реального часу; відстеження міцності конструкції для розумних парків, будівель, мостів тощо; шумове картографування; розумне освітлення; відстеження та управління відходами; розумний полив; інтелектуальний трафік, здатний відстежувати умови клімату, аварії, щільність тощо; програми для смартфонів; розумний туризм за допомогою картографування тощо.

Сила верстатів з ЧПК полягає в їх здатності обробляти різні типи матеріалів з високою точністю [13]. На відміну від інших машин, машини з цифровим керуванням довели свою ефективність у роботі з товстими матеріалами та можуть легко проникати в них. Верстати з ЧПК також можуть обробляти дрібні деталі, наприклад, вирізати букву V може бути непросто для інших верстатів, щоб мати справу з такою роботою, яка вимагає точності, але цей розріз буде легким, якщо використовувати ЧПК. Верстати з цифровим керуванням мають багато переваг і справляються з різними такі матеріали, як деревина та інші матеріали, і ця універсальність призвела до існування більш ніж одного типу верстатів з ЧПК:

– фрезерні машини з ЧПК ріжуть деревину, пластик, а також листовий метал і мають тривимірні осі при виготовленні великогабаритних виробів і складних виробів, використовують для різання різних матеріалів уздовж шляху (рис. 1.1);

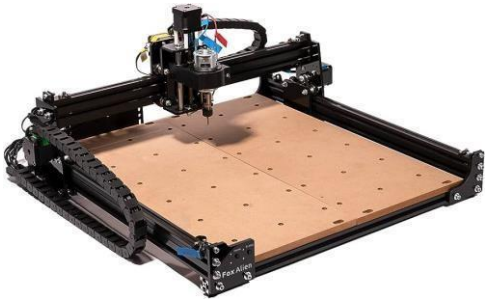

Рисунок 1.1 – Фрезерні машини з ЧПК

– різання струменем води – система різання, яка використовується у верстатах з ЧПК, система холодного різання, яка не забруднює навколишнє середовище, ріжеться за допомогою прокачування води та використовує цю техніку з будь-яким м’яким або жорстким матеріалом, таким як метал, скло та кераміка (рис. 1.2);

Рисунок 1.2 – Система різання струменем води

– тривимірний друк – це один із сучасних методів виробництва (додаткове виробництво), при якому тривимірний виріб можна виготовити стереоскопічним і відчутним шляхом проектування на комп’ютері, а потім надрукувати (виготовити) за допомогою 3D-принтера. Процес друку здійснюється шляхом накладання шарів матеріалу один на одного, доки не буде завершено потрібний об’єкт (рис 1.3);

– токарні верстати – один із типів верстатів з ЧПК, які використовуються для наповнення та виробництва металів шляхом обертання артефактів, які потрібно сформувати, надаючи їм бажану форму (рис. 1.4);

– фрезерний верстат – цей тип верстатів з ЧПК використовує ротаційні різці для різання різних матеріалів із готових матеріалів. Коли вона працювала вручну (рис. 1.5);

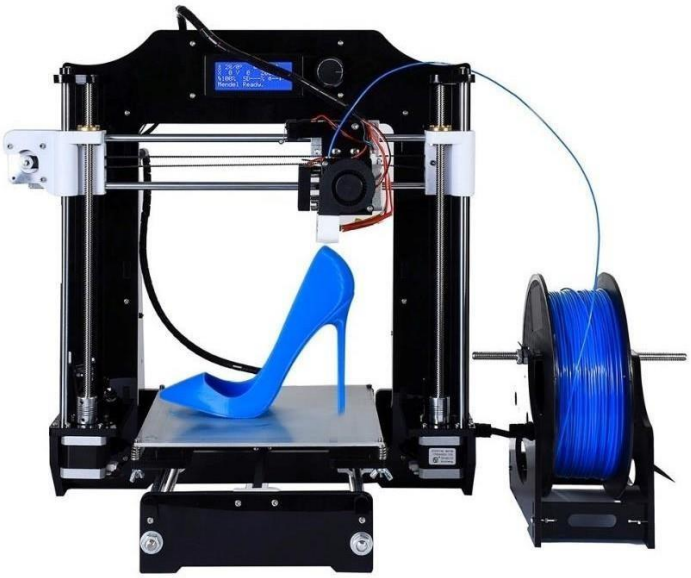

Рисунок 1.3 – 3D-принтер

Рисунок 1.4 – Токарний верстат

– плазмові різаки Цей тип верстатів з ЧПК розрізає сталь та інші матеріали за допомогою плазми, а процес різання виконується за допомогою плазмових ножиць (рис. 1.6).

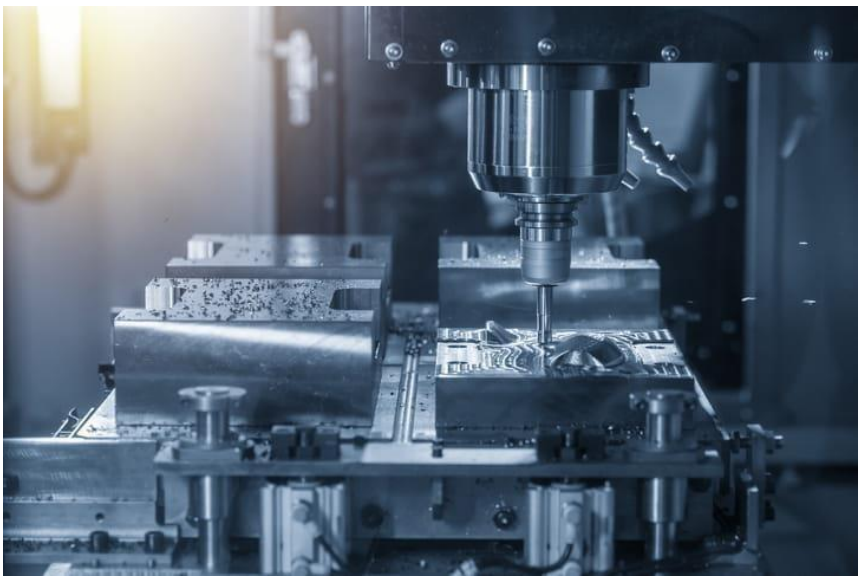

Рисунок 1.5 – Фрезерний верстат

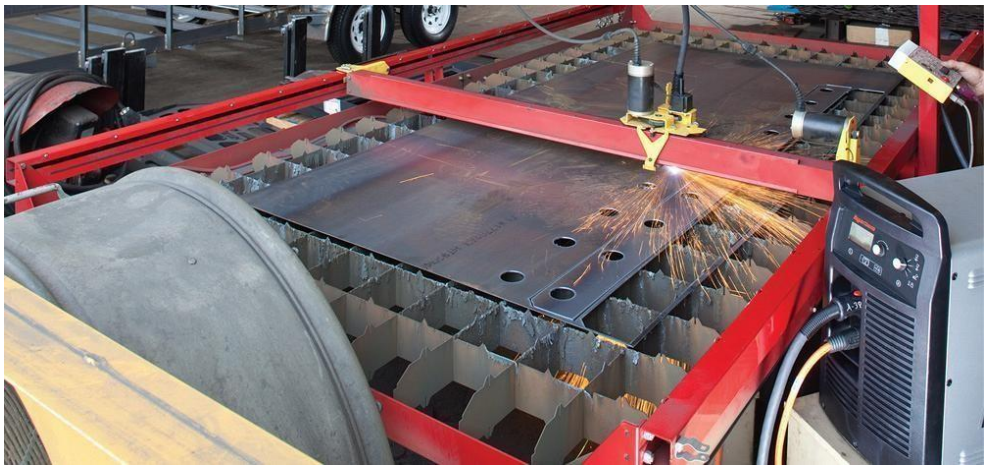

Рисунок 1.6 – Плазмовий верстат

Основою життя будь-якої країни-виробника є промисловість. Чим більше галузь покладається на машини, тим успішніша вона. Чим точніше ці машини працюють, тим швидше вони працюють, якщо вчені почнуть розвивати використання електронних систем у галузі механічних машин. і працював над підключенням цих машин до комп’ютера для посилення контролю. Переваги ЧПК:

– програми можна вводити безпосередньо з верстата та зберігати їх у пам'яті комп'ютера, підключеного до верстата;

– легкість виправлення та перегляду програми;

– вхідних даних менше, а виробництво продукції відбувається швидше;

– ця машина безпеки операторів. активність робіт висока, тому що системна машина дозволяє працівникам виконувати інші роботи одночасно;

– виготовлення високоточних виробів для всіх деталей за рахунок використання однієї програми;

– виробництво складної продукції, яку важко виготовити на традиційній машині;

– скоротіть загальний час виробництва, тому що потрібно налаштувати та оснастити машину лише на першій операції;

– зниження собівартості продукції. тому що немає збоїв у виробництві;

– економія витрат часу на виробництві порівняно з традиційними машинами;

– забезпечує високу точність виготовлення деталей;

– можливість повторити велику кількість виробів з однаковою точністю для всіх деталей;

– можливість контролювати відповідні умови різання;

– легкість переміщення між різними формами продукту.

У світі немає досконалого винаходу, який має переваги, це означає, що він має недоліки, але в хорошого винаходу переваги є більшими, ніж недоліки:

– висока вартість верстата;

– оператор повинен мати найвищий рівень підготовки для роботи з цим типом машин;

– дефіцит спеціалістів з обслуговування даного типу машин;

– надання спеціальних інструментів для дистанційного керування цією машиною та її безперервної роботи.

1.2 Операція свердління за допомогою ЧПК

У сфері комп'ютеризації свердління деталей зробило обов'язковим виготовлення деталей та його складання. Свердління проводиться за кількома матеріалами, такими як дерево, камінь, сталь, прес, роздільники, пластик, композитні матеріали окремо. На сучасному заводі, якщо ми використовуємо ручне буріння, будуть потрібні інвестиції для виконання кількох завдань у короткий термін. Щоб це виконати, потрібні додаткові можливості, які можна зробити за короткий час та з гарною підготовкою (рис. 1.7).

Рисунок 1.7 – Операція свердління

Використовуючи верстати з ЧПК, можна ефективно виконати завдання. Механічні технології викликали ентузіазм у людей загалом і в деяких підприємств, пов'язаних із свердлінням. Складну частину, яка фізично

неможлива, виконує верстат з ЧПК. Механічні інновації сподіваються, що вони усунуть фізичний та розумовий занепокоєння, пов'язане з роботою.

Верстат має базову структуру програмування. Швидкість подачі та швидкість свердління повинні бути відомими параметрами під час автоматичного свердління. Після завершення робіт з свердління виконуються подальші процеси, такі як розточування, розгортання, зенковка, розточування тощо. З іншого боку, у масовому виробництві впроваджується комп'ютеризований верстат свердлильний (ЧПК).

Чинником, що впливає на процес, є час, необхідний для зміни вартості, тому для швидкості зміна діаметра отвору має бути зведена до мінімуму. Тому верстат з ЧПК оснащено декількома шпинделями в револьверних головках різного діаметра, які використовуються для встановлення під час свердління. Задля більшої економічної ефективності необхідно вибрати відповідний тип ЧПК для конкретної геометрії. Для робіт із невеликим обсягом виробництва може бути достатньо ручного або напівавтоматичного буріння. Програмування верстату з ЧПК необхідно вивчити для написання відповідного коду конкретної геометрії. Для цього потрібний досвідчений програміст. А ось у автоматичному свердлінні є вірогідність полегшити завдання –легко розташувати свердло у певній точці роботи та виконати заплановану операцію [14].

В найпростішому випадку такий верстат з ЧПК може складатися з наступних елементів [15].

Для створення недорогого верстата з ЧПК можна використати три крокові двигуни та відповідні драйвери крокових двигунів, деталі рами з полівінілхлору, приводні дроти, направляючі стрижні, а також електронні пристрої та аксесуари.

Технічні деталі верстата з ЧПК:

– переміщення по осі X, осі Y і осі Z, що відповідає вимогам. Крокові двигуни можуть бути обрані довільно;

– ходовий гвинт з нержавіючої сталі з мідною гайкою;

– двигун шпинделя;

– базове джерело живлення змінного струму;

– драйвери крокових двигунів.

Крокові двигуни рухаються за частки оберту, які можна змінювати залежно від вимоги, щоб генерувати настільки плавний рух, наскільки це необхідно, з урахуванням вимог дослідження щодо точності. Це відомо як мікростепінг. Драйвер має шість ступінчастих перемикачів. Керування може забуватися на Arduino або Raspberry Pi. Після підключення всіх електронних компонентів двигун шпинделя встановлюється на осі z. Шпиндельний двигун постійного струму, живиться постійним струмом через імпульсне джерело живлення і встановлений на осі z верстата. Особливість цього двигуна полягає в тому, що він безщітковий. Безщіточні двигуни мають високе співвідношення потужності до ваги, високу швидкість, хороше електронне керування та низькі витрати на обслуговування.

Для створення маршруту руху свердла можна використовувати будь-який графічний редактор з можливістю отримувати координати точок переміщення свердла.

Усі параметри, пов’язані з свердлінням, такі як швидкість, тип інструменту та орієнтація, надсилаються відповідним чином. Створюється шлях інструмента, а потім генерується G-код.

Після завершення моделювання G-код надсилається до верстату з ЧПК через програмне забезпечення.

Сам верстат з ЧПК використовує це мікропрограмне забезпечення, яке може інтерпретувати те, що говорить прикладне програмне забезпечення машини, і, отже, керує кроковими двигунами. Після створення файлів G-Code вони візуалізуються за допомогою програмного забезпечення CAD/CAM, яке потім діє як зв’язок для контролера для наступної обробки. Далі шістнадцятковий файл завантажується на контролер.

Перед початком операції свердління є можливість зробити шлях інструменту більш оптимальним за допомогою вирішення задачі комівояжера

(з використанням алгоритмів пошуку найкоротшого шляху свердла), що буде розглянуто далі.

1.3 Вирішення задачі маршрутизації транспортних засобів

1.3.1 Проблема маршрутизації транспортних засобів

Однією з головних проблем, з якою стикаються люди у своїй повсякденній діяльності, є проблема транспорту. Перевезення товарів і послуг є важливою темою для сучасного суспільства. Щодня витрачається велика сума грошей на паливо, доставку товарів і послуг, обслуговування обладнання тощо. Саме тут знають і використовують методи дослідження операцій (ДО). Якщо доступні ресурси відомі, можна застосувати цю техніку. Використання комп'ютеризованих методів у вирішенні транспортної проблеми найчастіше призводить до приблизно 5% - 20% економії вартості транспортування [16].

Таким чином, планування процесу розповсюдження, дослідження та вивчення операційних методів є доцільним і заощадить певні витрати на транспортування.

Однією з основних сфер транспортування, якій протягом багатьох років приділялося багато уваги і яка стосується ДО, є проблема маршрутизації транспортних засобів (МТЗ).

З моменту впровадження ДО у розв'язанні транспортних проблем було зафіксовано успіх, особливо з використанням методів оптимізації. Кілька методів точних методів і евристик із сильними формулюваннями були запропоновані та розроблені для вирішення МТЗ.

Ця задача узагальнює відому й поширену задачу комівояжера (ЗК), яка є однією з найпростіших проблем маршрутизації. ЗК передбачає пошук оптимального/найкоротшого маршруту, який з'єднує всі маршрути точно один раз і повертається до початкового вузла із заданого набору кінцевих маршрутів, а також вимірювання відстані між ними.

Завдяки набору кількох доступних маршрутів МТЗ є обчислювально-складною проблемою, навіть незважаючи на те, що протягом багатьох років було запропоновано багато алгоритмів (таких як евристичний і точний алгоритм). Складне завдання полягає в тому, як створити рішення, яке буде швидким і надійним. Мета МТЗ полягає в тому, щоб знайти оптимальний маршрут із набору маршрутів, у якому кожен транспортний засіб стартує з депо (скажімо, вузла А) і з'єднує всіх клієнтів, а потім має повернутися до депо.

МТЗ – це задача комбінаторної оптимізації та цілочисельного програмування, яка знаходить оптимальний шлях для доставки товарів і послуг кінцевому набору клієнтів. МТЗ була проблемою протягом кількох десятиліть і однією з найбільш досліджуваних проблем у інженерії логістики, прикладній математиці та інформатиці, яка описується як пошук оптимальних маршрутів для парку транспортних засобів для обслуговування деяких розрізнених клієнтів зі складу. Іншими словами, МТЗ включає в себе транспортні засоби, які доставляють товари (та послуги) до кількох клієнтських вузлів, і коли в транспортному засобі закінчуються товари, він повинен повернутися до депорту, щоб вибрати додаткові товари в іншому місці, щоб продовжити постачання клієнтів.

Транспортний засіб відвідає клієнта рівно один раз і повинен повернутися в депо, звідки він стартував.

Проблема полягає у тому, щоб знайти набір маршрутів для деяких транспортних засобів, які доставляють товари або послуги в деякі відомі місця.

Маючи набір міст і відстань між кожною парою з них, проблема комівояжера (ПрК) або маршрутизації транспортних засобів, полягає в тому, щоб знайти найкоротший шлях відвідування всіх міст і повернення до початкової точки. Хоча це твердження просте для формулювання, але його складніше вирішити.

Задача комівояжера є задачею оптимізації, має великий простір пошуку та називається NP-складною, що означає, що її неможливо розв'язати за

поліноміальний час [17]. Це одна з найфундаментальніших проблем сучасної інформатики. ПрК зараз використовується в багатьох сферах. Деякі з його застосувань – маршрутизація транспортних засобів, виробництво мікрочіпів, маршрутизація пакетів у GSM, свердління друкованих плат (про що буде йти мова далі) тощо. Простіше кажучи, скажімо, у нас є набір із n міст, і тоді ми можемо отримати (n - 1)! альтернативні маршрути для покриття (відвідування) всіх міст. Задача комівояжера полягає в тому, щоб знайти маршрут, який має найменшу відстань.

Багато евристичних методів було використано для пошуку ефективного вирішення проблеми, як-от жадібний метод, мурашині алгоритми, моделювання відпалу, пошук табу та генетичні алгоритми. Але зі збільшенням кількості міст обчислення, щоб знайти рішення, стає складним. Незважаючи на складність обчислень, ми можемо використовувати такі методи, як генетичні алгоритми та пошук варіантів вирішення проблеми, які можуть дати майже оптимальне рішення для тисяч міст.

1.3.2 Різні методи вирішення задачі комівояжера

У літературі наводиться цілий ряд методів вирішення задачі комівояжера (ЗК), що відрізняються різними підходами до рішень, ефективністю процедур, а також результатами. Наведемо короткі характеристики найбільш часто використовуваних.

Метод сумарного перерахування. В принципі це комбінаторне рішення. Метод заснований на оцінці всіх потенційних маршрутів (послідовностей) у загальній кількості (n – 1)!. Перевага полягає в тому, що глобальний оптимум завжди знайдений, однак він неможливий, якщо враховувати більшу кількість відвідуваних місць. З кожним доданим елементом (вузлом) кількість можливих рішень зростає експоненціально, і навіть сьогодні ми не маємо достатньо потужних комп’ютерів, щоб забезпечити оптимальне рішення протягом розумного часу [18].

Метод гілок і меж [19]. Цей метод належить до найстаріших і найчастіше використовуваних алгоритмів для рішень ЗК. Перевага методу полягає в поступовому розкладанні можливого набору рішень на ряд взаємно диз'юнктивних підмножин, позначених як гілки. На кожному кроці оцінюється наступне:

– верхня межа цільової функції, яка найчастіше є значенням цільової функції без дотримання обмежень;

– максимальна нижня межа цільової функції прийнятних рішень, які нам відомі в межах крок.

Обидві оцінки можна використовувати для пошуку неперспективних напрямків подальших процедур: якщо для будь-якої гілки перша менше за другу, то даний напрямок можна виключити. Однак цей метод також, особливо для вищих n, занадто трудомісткий і не завжди гарантує оптимальне рішення з першої спроби.

Ефективний алгоритм Кларка і Райта, що буде використано в наступних розділах [20]. Значний прогрес у розв'язках ЗК забезпечив метод Кларка і Райта. Початкова ситуація припускає, що кожне місце надається окремо, і завжди слід повернення на стартову базу. Основна ідея базується на розрахунках економії, досягнутої завдяки інтеграції інших місць у кільцевий маршрут. Беззаперечною перевагою цього алгоритму є його функція поважати подальші обмеження, які часто породжуються практикою, напр. потреба оптимізувати більше орбітальних маршрутів, використовувати більше транспортних засобів, поважаючи їхню різну пропускну здатність тощо.

Комп'ютерне моделювання [21]. Розвиток імітаційних моделей, їх програмна підтримка та збільшення обчислювальної потужності викликали спроби використання методів моделювання для розв'язування великих ЗК. Їх перевага полягає у випадковій вибірці у великому масштабі, яка пізніше оцінюється відповідно до вибраної цільової функції. Незважаючи на те, що рішення не гарантує глобального оптимуму, достатньо велика кількість

симуляцій буде видаватися для досягнення найкращого можливого рішення, значення якого буде близьким до оптимального.

Алгоритм оптимізації мурашиної колонії (АОМК) є одним із метаевристичних методів розв'язання ЗК [22]. Спостереження за справжніми мурахами, які, знайшовши їжу, повертаються до своєї колонії, прокладаючи сліди феромонів. Якщо інші мурахи знайдуть такий шлях, вони, швидше за все, не продовжуватимуть мандрувати навмання, а натомість підуть по сліду, повертаючись і зміцнюючи його, якщо врешті-решт знайдуть їжу. Існує більша ймовірність того, що буде обраний слід з більшою концентрацією феромонів. Феромонний слід дозволяє мурахам знайти дорогу назад до джерела їжі та навпаки. Стежка використовується іншими мурахами, щоб знайти джерело їжі, яке виявила будь-яка мураха. Коли від гнізда до джерела їжі існує кілька шляхів, колонія мурашок може використати феромонний слід, залишений окремими членами колонії, щоб знайти найкоротший шлях від гнізда до джерела їжі та назад. Чим більше мурах вибирають шлях, яким слідувати, феромони на шляху накопичуються, що робить його більш привабливим для інших мурах

Алгоритм оптимізації частинок колонії (АОЧК) виходить із соціальної поведінки організмів, наприклад зграї птахів і рибальства [23]. Завдяки співпраці між окремими особами група часто може ефективно та результативно досягти своєї мети. АОЧК моделює цю соціальну поведінку як інструмент оптимізації для вирішення деяких проблем оптимізації. Кожна частинка летить у просторі пошуку зі швидкістю, яка динамічно регулюється на основі її власного досвіду польоту та досвіду польоту її супутників. Іншими словами, кожна частинка буде використовувати як поточну власну інформацію про найкраще положення, так і глобальну інформацію про найкраще положення, яку рій шукав до цього часу, щоб змінити свою швидкість і, таким чином, прибути в нове положення.

Генетичні алгоритми [24]. В останні роки були спроби використовувати так звані генетичні алгоритми для рішень ЗК. Простіше кажучи, генетичні

алгоритми переносять принципи еволюції живих організмів у інтелектуальний пошук і оптимізацію моделей в інших областях. Біологічна термінологія також застосована до цього самого опису алгоритму. Генетичний алгоритм, як і природа, працює з популяцією особин (P), що визначається одним або кількома математичними генами – хромосомами (тобто послідовностями чисел у двійковій системі числення).

1.3.3 Стратегії першого рішення

Стратегії першого рішення – це евристичні методи, які використовуються для пошуку можливого початкового рішення для екземпляра ЗК. Це початкове можливе рішення можна вдосконалити іншими методами [25].

Стратегії першого рішення сьогодні використовуються рідше, ніж у минулому, оскільки метаевристичні методи в даний час зазвичай достатньо надійні, щоб не потребувати можливого рішення для початку пошуку; замість цього буде достатньо будь-якого випадкового рішення. Результати оказують, що стратегії першого рішення мають значні відмінності у своїй продуктивності з алгоритмом заощаджень. Метод Кларка і Райта, як правило, є найефективнішим.

Послідовна вставка. При цьому методі маршрути будуються одним транспортним засобом. Першим кроком методу є ініціалізація маршруту з випадковим немаршрутованим вузлом, таким чином, щоб транспортний засіб прямував від депо до вузла і назад. На другому кроці в маршрут вставляється ще один немаршрутизований вузол. Це робиться шляхом вибору вузла та його позиції вставки в маршруті таким чином, щоб збільшення довжини маршруту було мінімізоване та щоб додавання було можливим відповідно до обмежень проблеми. Потім другий крок повторюється, доки до маршруту не можна буде додати додаткові вузли, після чого новий маршрут (для нового транспортного засобу) ініціалізується відповідно до першого кроку, або всі немаршрутизовані вузли будуть включені в маршрути.

Паралельна вставка. Цей метод є модифікацією послідовного вставлення, при цьому маршрути будуються паралельно. Крок перший полягає в ініціалізації можливих маршрутів (як у послідовному вставленні) від найближчих немаршрутизованих вузлів. Крок другий полягає в тому, щоб вибрати випадковий немаршрутизований вузол і вставити його в найкраще місце серед усіх маршрутів, з найкращим можливим значенням і найменшим збільшенням довжини маршруту. Потім другий крок повторюється, доки жоден вузол не може бути реально доданий до маршрутів, після чого ініціалізується новий маршрут або всі вузли не будуть включені в маршрути.

Найближчий сусід. У цьому методі маршрути будуються по одному. На першому кроці вибирається найближчий немаршрутизований вузол до депо та ініціалізується один маршрут, що йде від депо до вузла і назад. На другому кроці найближчий немаршрутизований вузол до попередньо доданого вставляється в кінець маршруту, якщо це можливо. Крок другий повторюється, доки нові вузли не будуть додані в кінці маршруту. Потім ініціалізується новий маршрут, якщо не всі вузли були включені в маршрути.

Місцева найдешевша вставка (МНВ). Цей спосіб по суті рівноцінний до описаного вище методу послідовного вставлення. Вартість вставки базується на функції вартості дуги, тобто часі подорожі між вузлами.

Паралельна найдешевша вставка (ПНВ). Цей спосіб по суті рівноцінний до методу паралельного введення, описаного вище. Вартість вставки базується на функції вартості дуги, тобто часі подорожі між вузлами.

Найдешевший шлях (НШ). Цей метод по суті еквівалентний методу найближчого сусіда, описаного вище.

Найбільш обмежена дуга шляху (НОДШ). Цей метод схожий на НШ, але вузол, вибраний для додавання, не є найближчим, а натомість найбільш обмеженим (точніше, найбільш обмеженою дугою, з’єднанням між двома вузлами), на основі обмежень, визначених екземпляром проблеми, таких як часові вікна, вимоги вузла, самовивіз та доставка тощо.

2 ВИБІР І ОБҐРУНТУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ФОРМУВАННЯ МАРШРУТУ ПЕРЕМІЩЕННЯ СВЕРДЛА

2.1 Вибір мови програмування для створення програмного забезпечення

В процесі навчання було вивчено дві основні мови програмування, такі як C++ та Python, отже проведемо їх порівняння.

Python і C++ – це дві різні мови, які мають різні функції та різну поведінку. Обидві ці мови мають одну спільну рису, тобто потужну підтримку об’єктно-орієнтованого програмування.

Основні функції мови C++, що вивчались:

а) компільована мова;

б) мова, чутлива до регістру;

в) незалежна від комп'ютера та модульна;

г) швидка;

д) чутлива до синтаксису;

е) використовує покажчики та має величезну бібліотеку функцій;

ж) об'єктно-орієнтована мова програмування, що підтримує такий функціонал ООП як:

– класи та об'єкти;

– абстракція;

– інкапсуляція;

– поліморфізм;

– спадкування.

Основні функції мови Python, що вивчались:

– мова проста у вивченні та має достатньо зрозумілий синтаксис;

– мова може бути розширена за рахунок підключення модулів;

– Python є безкоштовним, із відкритим вихідним кодом і є кросплатформним;

– також є об'єктно-орієнтованою мовою програмування з високою читабельністю та надійністю;

– можна використовувати для створення прототипів і тестування коду (бібліотеки для тестування), який пізніше можна використовувати для розробки повноцінної програми з використанням інших мов вищого рівня;

– поставляється з величезною стандартною бібліотекою, яка складається з аналізаторів XML, інтерфейсу Excel тощо;

– побудована на мові C++ та її бібліотеках.

Компіляція. C++ є скомпільованою мовою. Компілятор C++ генерує об’єктний код із вихідного коду C++, а потім виконується для отримання результату. Python є інтерпретованою мовою. Код Python не потребує компіляції. Ми можемо безпосередньо передати його інтерпретатору Python і створити результат.

Використання. C++ має багато функцій, а також має порівняно складний синтаксис. Написати код на C++ не так просто. Python легко писати та має чіткий синтаксис. Тому писати програми на Python набагато легше порівняно з C++.

Природа мови. C++ є статично типізованою мовою, тобто оголошення змінної, тип даних змінних тощо перевіряються під час компіляції. Це зберігає вихідний код без помилок під час виконання. Python, з іншого боку, не є статично типізованим. Під час компіляції перевірка типу не виконується. Отже, код схильний до помилок.

Портативність. C++ не є портативним, тобто нам потрібно перекомпілювати код на кожній платформі. C++ – це в основному Напишіть один раз, компілюйте будь-де. Python портативний. Він також є кросплатформним, і ми можемо виконувати програми на будь-якій платформі.

Збирання сміття/керування пам'яттю. У C++ керування пам'яттю здійснюється вручну. C++ не підтримує автоматичний збір ресурсів.

Python, з іншого боку, має функцію автоматичного збирання сміття. Його управління пам'яттю контролюється системою.

Швидке створення прототипів. Ми не можемо створювати швидкі прототипи за допомогою C++. Використовуючи Python, ми можемо створювати швидкі прототипи коду, щоб потім його можна було використовувати для створення програм з використанням мов вищого рівня.

Область змінних. C++ має код, позначений блоками за допомогою фігурних дужок ({}) і циклів. Область дії змінних обмежена цими блоками та циклами, позначеними {}. Область змінних, що використовуються в Python, не обмежується блоками або циклами. Змінні доступні навіть поза фігурними дужками.

Встановлення. C++ можна легко встановити на Windows. Python, однак, важко встановити.

Типи. У C++ типи даних прив'язані до імен і перевіряються під час компіляції. Це зменшує ймовірність будь-яких помилок під час виконання.У Python типи даних прив'язані до значень і перевіряються під час виконання. Код може бути більш схильним до помилок під час виконання, оскільки ми не виявляємо ці помилки під час компіляції.

Функції – це блоки кодів з одним або кількома параметрами та значенням, що повертається. Кожен із параметрів і значення, що повертається, має тип. У C++ типи параметрів і тип повернення під час виклику функції мають відповідати визначенню функції. У Python немає такого обмеження на типи параметрів і повернення.

Ефективність. Код C++ важко підтримувати, оскільки його може стати складніше для читання, коли рішення стають більшими. Python, з іншого боку, має чистий код і простий синтаксис. Вихідний код для Python легше підтримувати.

Складність синтаксису. У C++ існує чітке розмежування коду за допомогою блоків, укладених у {}, крапки з комою, що вказує на кінець оператора тощо. Таким чином у C++ синтаксис добре організований. У Python немає блоків і крапок з комою. Натомість Python використовує відступи.

Швидкість виконання. Що стосується швидкості виконання, програми на C++ працюють швидше. Насправді C++ відомий і широко використовується в програмах, які повинні працювати швидше, як ігрові платформи. З іншого боку, Python працює повільно. Крім того, програми Python працюють повільніше, ніж програми Java. Тому ми використовуємо Python спеціально для додатків, які можуть знизити швидкість.

Продуктивність. C++ є статично типізованою мовою, тому ми маємо менше помилок, про які потрібно піклуватися під час виконання. C++ також створює більш надійний і швидший код виконання. Це робить C++ мовою з високою продуктивністю. Python є динамічним, тому під час виконання можуть виникнути деякі помилки або небажана ситуація. Отже, що стосується продуктивності, Python відстає від C++. Але коли справа доходить до машинного навчання, Python є тим, хто має перевагу.

Популярність. Python простий у вивченні та застосуванні на практиці порівняно з C++, який стає складнішим у міру просування його можливостей. Ще однією перевагою Python є його бібліотеки, які дозволяють нам писати будь-які функції, особливо аналіз даних і машинне навчання. Тож за популярністю Python перевершує C++. Особливо для розробки програм машинного навчання, це вибір номер один для програмістів.

Простота та зручність використання. Python із його простотою та простими у використанні функціями дозволяє нам писати стислий код, який легко читається тощо. Це корисно, коли ми розробляємо складні програми для машинного навчання, оскільки нам не потрібно боротися з мовою програмування. По-друге, Python легко вивчити і це проста мова. Цього не можна сказати про C++. C++ – це мова більш низького рівня, яка легша для комп’ютерів, ніж для людей.

Таким чином, за цими параметрами Python є більш доцільним для створення додатків машинного навчання, таке як пошук маршруту переміщення свердла верстату з ЧПК.

Відмінності цих мов програмування запишемо у табл. 2.1.

Таблиця 2.1 – Відмінності C++ та Python

Параметр	C++	Python
Компіляція	Скомпільована	Інтерпретована
Використання	Складно писати код	Писати код значно легше
Характер мови	Статично типізована	Динамічно типізована
Портативність	Не є портативною	Портативна
Прибирання сміття	Не підтримується	Підтримується
Встановлення	Легко	Важко
Типи даних	Типи даних, пов’язані з іменами, перевіреними під час компіляції	Прив’язується до значень, перевіряється під час виконання
Область змінних	Обмежений циклами або блоками	Доступні за межами циклів або блоків
Швидке прототипування	Неможливе	Можливе
Функції	Обмеження на тип параметрів або значення, що повертається	Немає обмежень щодо типу параметрів або значення, що повертається
Ефективність	Важко підтримувати	Дуже легко підтримувати
Складність синтаксису	Використовує блоки та крапки з комою.	Не використовує блоки та крапки з комами
Швидкість виконання	Висока	Низька
Продуктивність	Висока	Низька
Популярність	Більш популярний для вбудованих або корпоративних програм	Найпопулярніший для машинного навчання, комп’ютерного зору тощо
Простота і зручність використання	Важко освоїти і використовується в програмі низького рівня	Проста і використовується для машинного навчання або веб-додатків

2.2 Бібліотека OR-tools та завдання пошуку маршруту

Бібліотека or-tools – це набір інструментів дослідження операцій, написаних на C++ у Google.

Основними інструментами є:

– вирішувач програмування обмежень;

– простий і уніфікований інтерфейс для декількох розв’язувачів лінійного програмування та змішаного цілочисельного програмування (CBC, CLP, GLOP, GLPK, Gurobi, SCIP і Sulum);

– ранцеві алгоритми (комбінаторна оптимізація);

– алгоритми графів (найкоротші шляхи, мінімальний потік витрат, максимальний потік, призначення лінійної суми);

– підтримка FlatZinc (мова низького рівня моделювання для проблем обмежень. Він розроблений таким чином, щоб легко підключатися до програм вирішення обмежень (наприклад, Gecode)).

Бібліотека or-tools:

– є безкоштовною та з відкритим вихідним кодом Усе, включаючи приклади, реалізацію алгоритмів, різноманітну документацію, ліцензовано згідно з ліцензією Apache 2.0 і доступне для завантаження;

– активно підтримується, оновлення та вдосконалення вносяться майже щодня;

– має багато прикладів, написаних на C++, Python, Java та C#;

– переносна – оскільки код створено Google, він суворо відповідає стилям кодування Google. Підтримуються як 32-бітна, так і 64-бітна архітектури, хоча код оптимізовано для роботи в 64-бітному режимі.

– ефективна – підтримуються всі додатки від Google.

– доступність – усе закодовано на C++, але доступне через SWIG у Python, Java та .NET (з використанням Mono на платформах, відмінних від Windows);

– зручність для користувача – код максимально простий у використанні (особливо на Python і C#);

– достовірність використання завдяки рокам підтримки від користувачів.

Основний функціонал бібліотеки OR-tools, що буде використано для створення програмного забезпечення буде наступним.

По-перше – це клас RoutingIndexManager. Менеджер для будь-якого перетворення індексу змінної NodeIndex <->. Вирішувач маршрутизації використовує змінні індекси всередині та через свій API. Цими індексами змінних складно керувати напряму, оскільки один вузол може відповідати багатьом змінним, залежно від того, скільки разів вони з'являються в моделі та чи використовуються вони як початкові та/або кінцеві точки. Цей клас має на меті спростити використання змінного індексу, дозволяючи користувачам використовувати замість нього NodeIndex.

Зіставлення між індексами вузлів та індексами змінних може змінюватися, тому не слід робити жодних припущень. Єдина гарантія полягає в тому, що індекси коливаються від 0 до n-1, де n = кількість транспортних засобів * 2 (для початкових і кінцевих вузлів) + кількість непочаткових або кінцевих вузлів.

Наступний клас це RoutingModel, що використовується для створення об'єктів для вирішення проблеми переміщення свердла. В ньому використано метод DefaultRoutingSearchParameters для завдання параметрів пошуку.

Також буде використано клас FirstSolutionStrategy – перші стратегії вирішення, використані як відправна точка локального пошуку з заданими опціями. Опції, що будуть використані представлені в табл. 2.2.

Таблиця 2.2 – Обрані стратегії для рішення завдання

Опції	Пояснення
PATH_CHEAPEST_ARC	Починаючи з вузла «початок» маршруту, з'єднайте його з вузлом, який створює найдешевший сегмент маршруту, а потім розширте маршрут, повторюючи останній вузол, доданий до маршруту.

Продовження таблиці 2.2

Опції	Пояснення
SAVINGS	Алгоритм заощаджень (Кларк і Райт). Докладніше про нього у наступному підрозділі.
PARALLEL_CHEAPEST_INSERTION	Ітеративно будуйте рішення, вставляючи найдешевший вузол у його найдешевшу позицію; вартість вставки базується на функції вартості дуги.
FIRST_UNBOUND_MIN_VALUE	Виберіть перший вузол із незв’язаним наступником і підключіть його до першого доступного вузла.

2.3 Розрахунок завдання пошуку найкоротшого маршруту руху свердла

Нехай буде зображення, на якому є 5 точок, в яких необхідно зробити отвори за допомогою свердла (рис. 2.1).

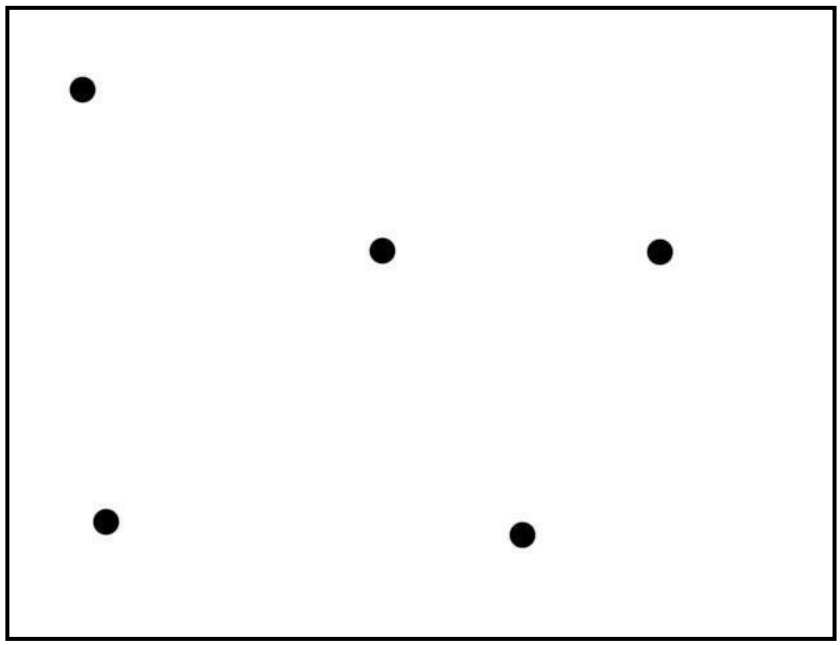

Рисунок 2.1 – Отвори, в яких повинно працювати свердло

Позначимо ці точки та їх координати на зображенні (рис. 2.2).

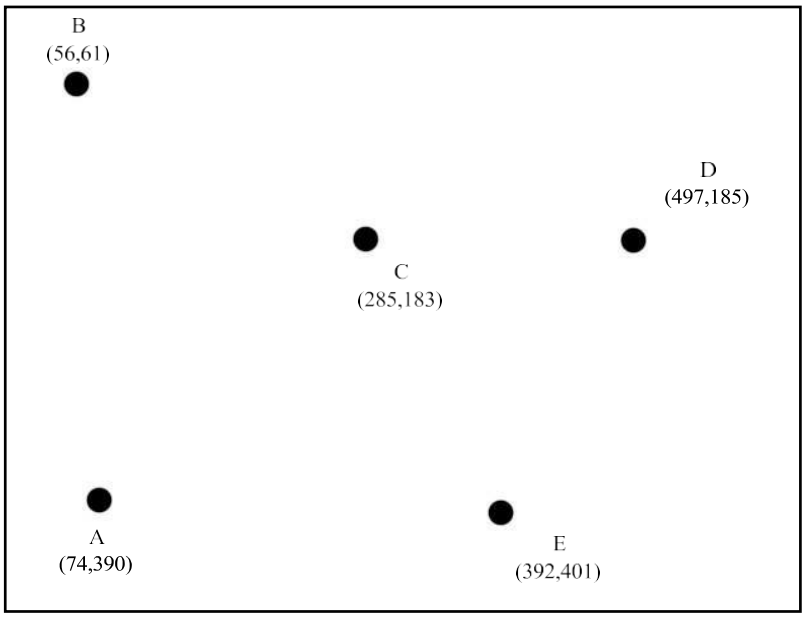

Рисунок 2.2 – Точки та їх координати

Розрахуємо Евклідові відстані між усіма точками зображення за формулою:

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

де (x_1, y_1) – координати першої точки переміщення,

(x_2, y_2) – координати другої точки переміщення.

Проведемо розрахунки:

$$L_{AB} = \sqrt{(56 - 74)^2 + (61 - 390)^2} = 330 \text{ пт}$$
$$L_{AC} = \sqrt{(285 - 74)^2 + (183 - 390)^2} = 296 \text{ пт}$$
$$L_{AD} = \sqrt{(497 - 74)^2 + (185 - 390)^2} = 470 \text{ пт}$$
$$L_{AE} = \sqrt{(392 - 74)^2 + (401 - 390)^2} = 318 \text{ пт}$$
$$L_{BC} = \sqrt{(285 - 56)^2 + (183 - 61)^2} = 260 \text{ пт}$$
$$L_{BD} = \sqrt{(497 - 56)^2 + (185 - 61)^2} = 438 \text{ пт}$$
$$L_{BE} = \sqrt{(392 - 56)^2 + (401 - 61)^2} = 478 \text{ пт}$$

$$L_{CD} = \sqrt{(497-285)^2 + (185-183)^2} = 212 \text{ пт}$$

$$L_{CE} = \sqrt{(392-285)^2 + (401-183)^2} = 243 \text{ пт}$$

$$L_{DE} = \sqrt{(392-497)^2 + (401-183)^2} = 242 \text{ пт}$$

Позначимо всі отримані результати (рис. 2.3).

Зробимо табл. 2.3 для більш наглядного представлення та внесемо всі отримані раніше результати.

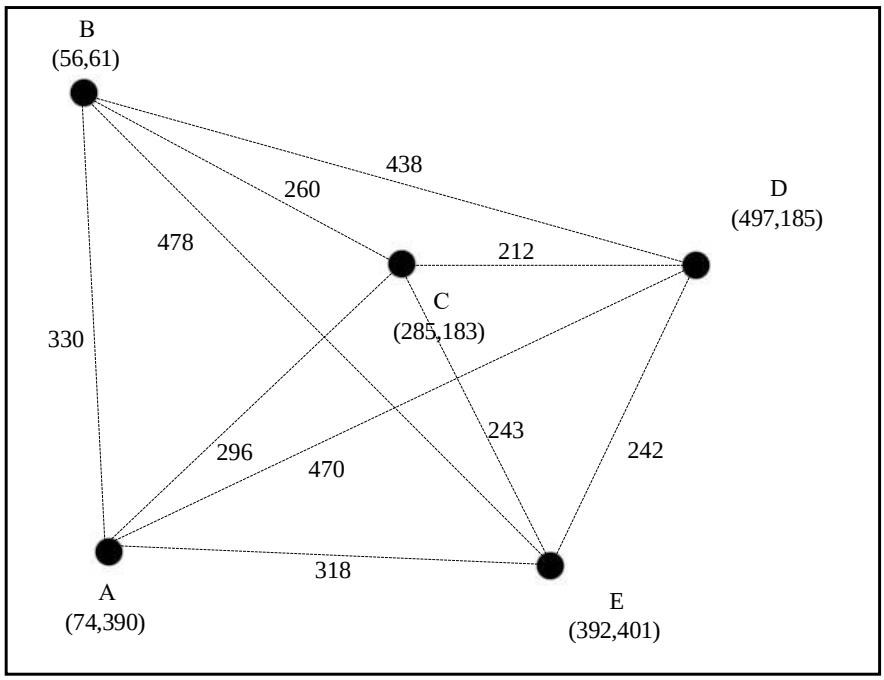

Рисунок 2.3 – Евклідові відстані між точками

Таблиця 2.3 – Результати переміщень (еквклідові відстані)

Точки	A	B	C	D	E
A	0	330	296	470	318
B	330	0	260	438	478
C	296	260	0	212	243
D	470	438	212	0	242
E	318	478	243	242	0

Для побудови та знаходження маршруту буде використано алгоритм заощадження (savings), знайдений Кларком та Райтом [26]. Він полягає у наступному. Задача маршрутизації транспортного засобу, для якої розроблено алгоритм, характеризується наступним чином. Зі складу товари повинні бути доставлені в заданих кількостях даним клієнтам. Для перевезення вантажів доступна низка транспортних засобів, кожна з яких має певну місткість щодо кількості. Кожен транспортний засіб, який використовується в рішенні, повинен пройти маршрут, який починається і закінчується в депо, на якому товари доставляються одному або кільком клієнтам.

Проблема полягає в тому, щоб визначити розподіл клієнтів між маршрутами, послідовність відвідування клієнтів на маршруті та який транспортний засіб має пройти маршрут.

Мета полягає в тому, щоб знайти рішення, яке мінімізує загальні витрати на транспортування. Крім того, рішення має задовольняти обмеження, згідно з якими кожен клієнт відвідується точно один раз, куди доставляються потрібні обсяги, а загальний попит на кожному маршруті має бути в межах потужності транспортного засобу.

Транспортні витрати вказуються як вартість проїзду з будь-якої точки в будь-яку іншу точку. Витрати не обов'язково ідентичні в двох напрямках між двома заданими точками.

Основою цього алгоритму є заощадження (рис. 2.4).

а) Маршрут через депо б)Маршрут без депо

Рисунок 2.4 – Ілюстрація алгоритму заощаджень

Спочатку на рисунку клієнти i та j відвідуються різними маршрутами. Альтернативою цьому є відвідування двох клієнтів одним маршрутом, наприклад, у послідовності i-j, як показано праворуч. Оскільки вказані витрати на транспортування, можна розрахувати економію, яка є результатом проїзду маршрутом на рис. 2.4(б) замість двох маршрутів на малюнку 2.4(а). Позначивши вартість транспортування між двома даними точками i та j через c_{ij}, загальна вартість транспортування D_a на малюнку 2.4(а) дорівнює:

$$D_a = c_{0i} + c_{i0} + c_{0j} + c_{j0} \tag{2.1}$$

Вартість транспортування D_b тоді буде дорівнювати:

$$D_b = c_{0i} + c_{ij} + c_{j0} \tag{2.2}$$

Поєднуючи два маршрути, можна заощадити S_{ij}:

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij} \tag{2.3}$$

Розрахуємо маршрути та знайдемо найбільші заощадження.

Маршрут ABC:

COST1 = cab+ cba + cac +cca = 330+330+296+296=1252

COST2 = cab+cbc+cca = 330 + 260+296=886

S_{bc}= COST1-COST2 = 1252-886=366.

Маршрут ABD:

COST1 = cab+ cba + cad +cda = 330+330+470+470=1600

COST2 = cab+cbd+cda = 330 + 458+470=1258

S_{bd}= COST1-COST2 = 1600-1046=342.

Маршрут ABE:

COST1 = cab+ cba + cae +cea = 330+330+318+318=1296

COST2 = cab+cbe+cea = 330+478+318=1126

S_{be}= COST1-COST2 = 1296-1126=170.

Маршрут ACD:

COST1 = cac+ cca + cad +cda = 296+296+470+470=1532

COST2 = cac+ccd+cda = 296+212+470=711

S_{cd}= = COST1-COST2 = 1532-711=821.

Маршрут ACE:

COST1 = cac+ cca + cae +cea = 296+296+318+318=1228

COST2 = cac+cce+cea = 296+243+318=857

S_{ce}= COST1-COST2 = 1228-857=371.

Маршрут ADE:

COST1 = cad+ cda + cae +cea = 470+470+318+318=1576

COST2 = cad+cde+cea = 470+242+318=1030

S_{de}= = COST1-COST2 = 1576-1030=546.

Маршрут BDC:

COST1 = cbd+ cdb + cbc +ccb = 438+438+260+260=1396

COST2 = cbd+cdc+ccb = 438+212+260=486

S_{dc}= COST1-COST2 = 1436-930=506.

Маршрут BDE:

COST1 = cbd+ cdb + cbe +ceb = 438+438+478+478=1832

COST2 = cbd+cde+ceb = 438+242+478=1158

S_{de}= COST1-COST2 = 1448-966=674.

Маршрут CDE:

COST1 = ccd+ cdc + cde +ced = 212+212+242+242=908

COST2 = ccd+cde+cec = 212+242+243=697

S_{de}= COST1-COST2 = 211.

Для роботи алгоритму обираємо лише ті маршрути, що мають найбільші заощадження, а це маршрути ABC, ACD, ADE, BDE.

Також, треба пам’ятати, що повертатися у депо (точка А) не потрібно для кожного маршруту, а лише в самому кінці. Отже, маршрут матиме вигляд A-B-C-D-E-A (рис. 2.5).

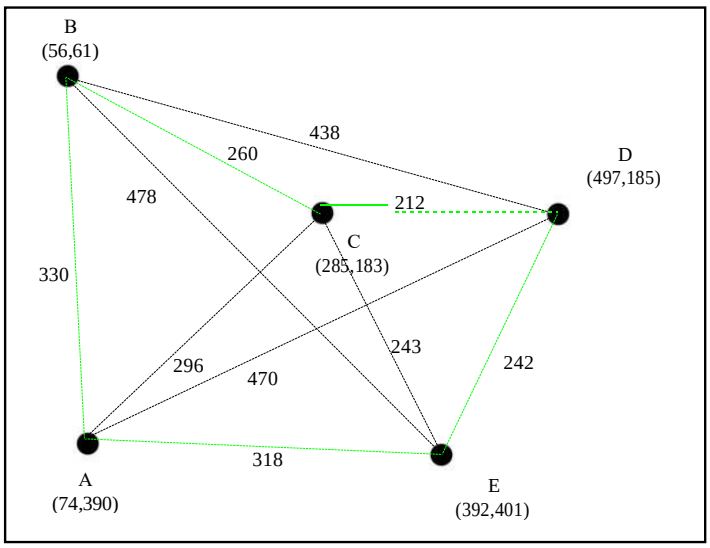

Рисунок 2.5 – Маршрут для руху 5 точками

Розрахуємо цей маршрут:

Маршрут = 330+260+212+242+318=1362

Використовуючи програмне забезпечення маршрут матиме наступний результат (рис. 2.6).

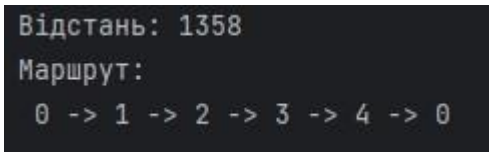

Рисунок 2.6 – Результат програмної реалізації

Похибка в результатах пов’язана з тим, що округлення всіх маршрутів між точками (евклідовими відстанями) було у більшу сторону.

3 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУЛЮВАННЯ МАРШРУТУ ПЕРЕМІЩЕННЯ СВЕРДЛА ДЛЯ ВЕРСТАТІВ З ЧИСЛОВИМ ПРОГРАМНИМ КЕРУВАННЯМ

3.1 Розроблення блок-схеми алгоритму роботи програми

Блок-схема алгоритму роботи програми представлена на рис. 3.1.

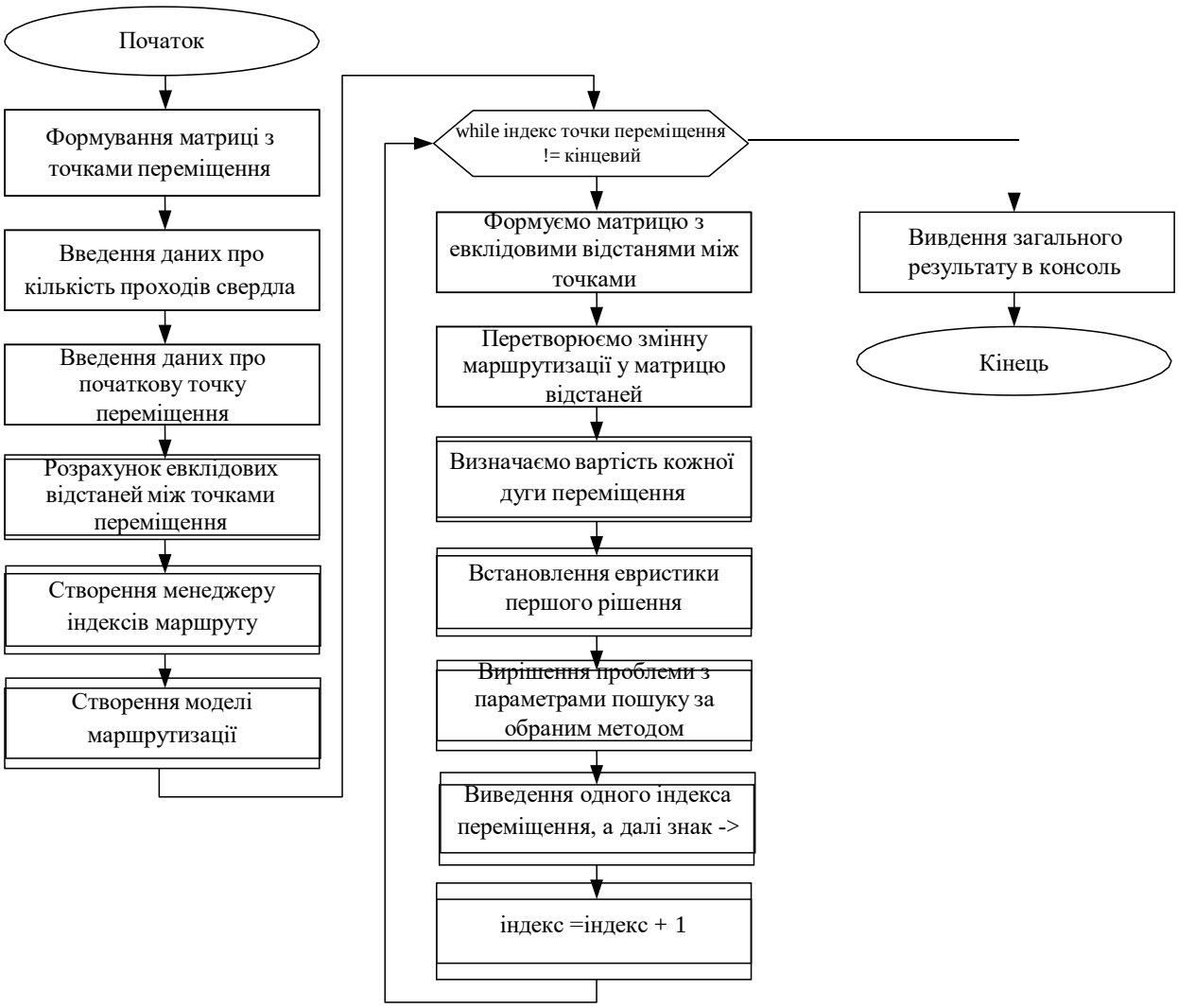

Рисунок 3.1 – Блок-схема роботи програми

3.2 Розроблення програмного забезпечення

Для створення програмного забезпечення потрібно підключити бібліотеки:

```
import math
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp
import numpy as np
import matplotlib.pyplot as plt.
```

Перша бібліотека (math) використовується для використання математичних функцій.

Бібліотека ortools використовується для маршрутизації транспортних засобів, в нашому випадку – для переміщення свердла. Модуль routing_enums_pb2 потрібен для використання алгоритму першої стратегії рішення. Модуль pywrapcp використовується для створення моделі маршрутизації.

Бібліотека numpy потрібна для математичних операцій над масивами (в нашому випадку точок переміщення свердла).

Бібліотека matplotlib використовується для виведення масиву цих точок на графік.

Далі потрібно створити модель для пошуку маршруту, а саме функцію, що зберігатиме дані для маршруту у вигляді кортежу, що має в своєму складі масив точок переміщення у довільному порядку, кількість транспортних засобів (або проходів свердла), а також початкову і кінцеві точки переміщення інструменту:

```
def create_data_model(): # функція створення моделі для пошуку маршруту
```

```
        """Зберігає дані"""
        data = {}
        # Розташування в блоках
        data["locations"] = [ # масив точок
            # початок точок АКТСІ-20-2
            (36, 454), (45, 430), (54, 399), (62, 379), (73, 347), (82, 316), (91, 288),
(106, 255), (121, 275), (124, 286),
            (136, 338), (143, 340), (148, 360), (152, 379), (163, 403), (169, 425),
(180, 454), (78, 381), (95, 381),
            (113, 381), (136, 381), (235, 253), (235, 286), (235, 353), (235, 400),
(236, 427), (235, 454), (246, 364),
            (263, 349), (278, 325), (302, 299), (320, 279), (335, 262), (342, 253),
(270, 340), (285, 360), (296, 377),
            (313, 401), (328, 421), (337, 438), (348, 454), (370, 262), (398, 262),
(431, 262), (475, 262), (497, 262),
            (516, 262), (442, 454), (442, 417), (442, 375), (442, 327), (442, 290),
(678, 436), (651, 449), (606, 445),
            (575, 429), (555, 399), (542, 369), (549, 318), (561, 281), (595, 266),
(629, 259), (658, 262), (678, 266),
            (738, 253), (728, 294), (728, 358), (728, 395), (728, 427), (728, 454),
(844, 371), (820, 373), (791, 371),
            (772, 373), (876, 281), (907, 268), (935, 266), (959, 277), (976, 310),
(968, 344), (950, 369), (935, 388),
            (911, 410), (894, 425), (878, 443), (992, 445), (968, 445), (939, 445),
(911, 443), (894, 443), (1140, 368),
            (1132, 327), (1127, 303), (1110, 273), (1084, 268), (1062, 273), (1042,
303), (1033, 336), (1029, 368),
            (1036, 405), (1055, 434), (1094, 447), (1118, 432), (1134, 405), (1140,
379), (1173, 371), (1197, 371),
```

```
        (1223, 373), (1243, 371), (1391, 443), (1363, 445), (1350, 443), (1310,
441), (1284, 443), (1267, 445),
        (1278, 283), (1302, 270), (1336, 264), (1369, 286), (1374, 310), (1367,
345), (1345, 377), (1315, 403),
        (1297, 423)
        # кінець точок АКТСІ-20-2
    ]
    data["num_vehicles"] = 1 # кількість проходів
    data["depot"] = 0 # з якої точки починаємо рух
    return data.
```

В якості зображення було обрано наступну траєкторію (рис. 3.2).

Рисунок 3.2 – Зображення для свердління

Програмно це зображення задано у вигляді пар точок, що зберігають координати (x,y) місць, де потрібно буде проводити операцію свердління. Зображення цих точок приведено на рис. 3.3-3.4.

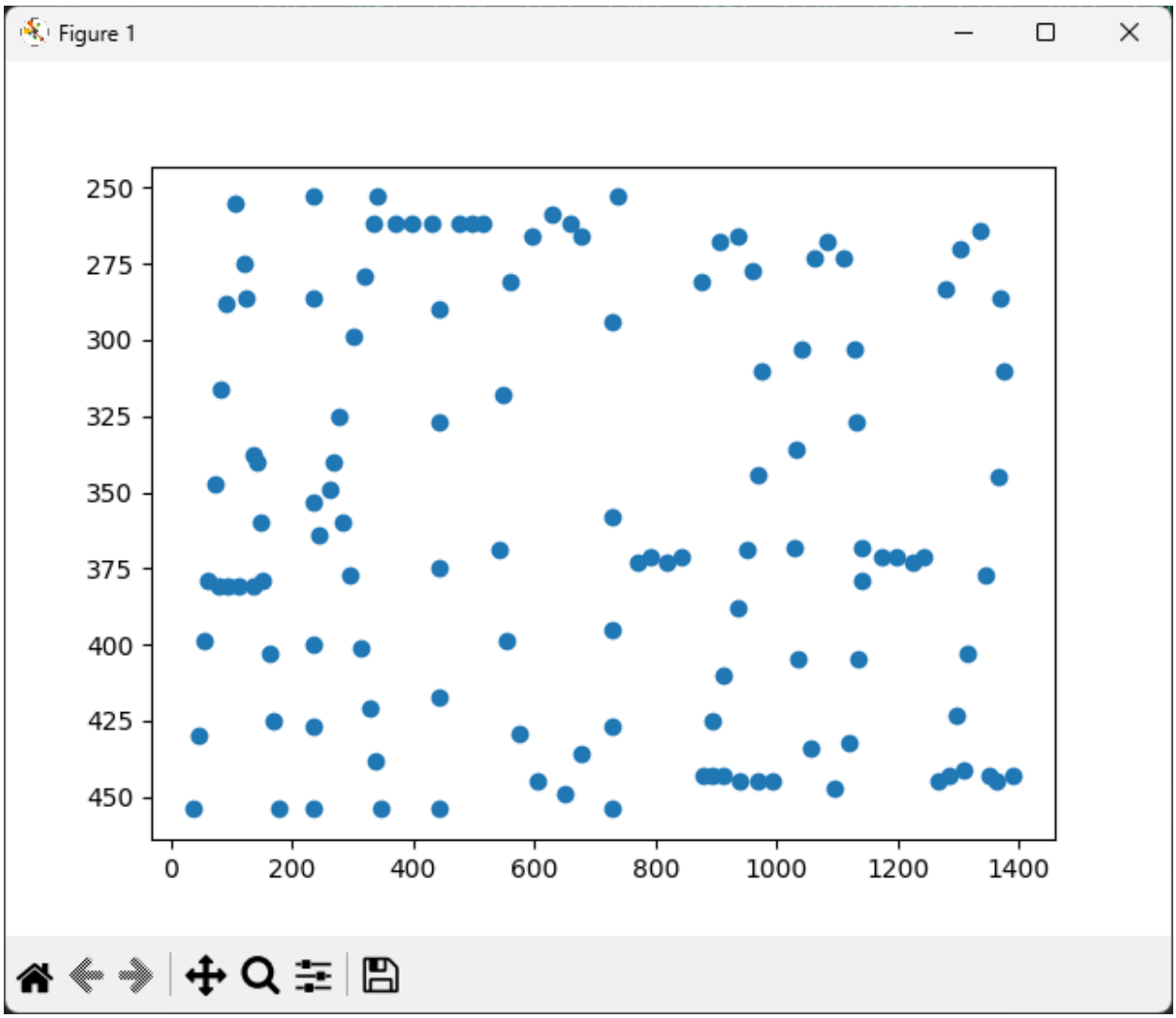

Рисунок 3.3 – Збільшене зображення точок для свердління

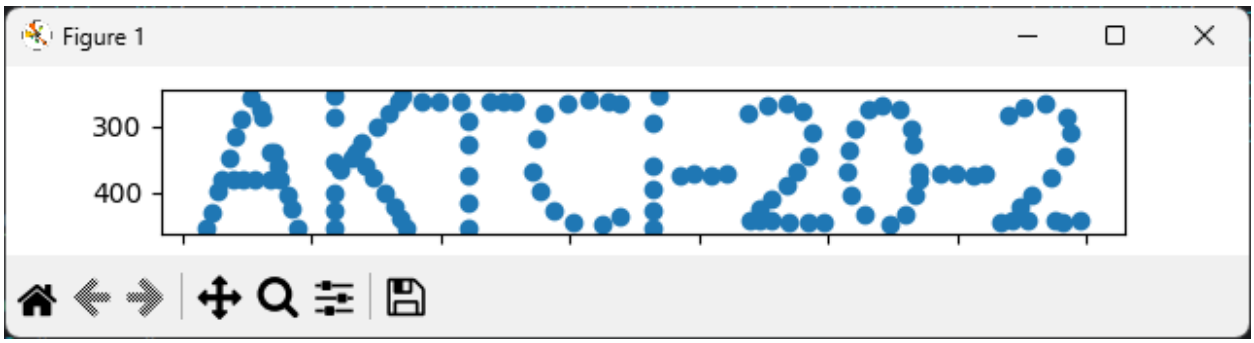

Рисунок 3.4 – Зменшене зображення точок для свердління

Програмний код побудови цих рисунків має наступний вигляд:

```
data2=np.array(data["locations"])
```

```
x, y = data2.T
plt.scatter(x, y)
plt.gca().invert_yaxis()
plt.show().
```

Наступною функцією програми є функція розрахунку евклідових відстаней між усіма точками з масиву і має наступний вигляд:

```
def compute_euclidean_distance_matrix(locations): # розрахунок евклідової відстані між точками
"""Створює функцію зворотнього виклику для повернення відстані між точками."""
distances = {} # Створюємо пустий словник для збереження відстаней
for from_counter, from_node in enumerate(locations):
  distances[from_counter] = {}
  for to_counter, to_node in enumerate(locations):
    if from_counter == to_counter:
      distances[from_counter][to_counter] = 0
    else:
      # Розраховуємо Евклідову відстань
      distances[from_counter][to_counter] = int(
        math.hypot((from_node[0] - to_node[0]), (from_node[1] - to_node[1])) #статичний метод повертає квадратний корінь із суми квадратів своїх аргументів
        )
return distances.
```

Евклідова відстань визначається як $L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, де (x_1, y_1)– координати початкової точки переміщення, а (x_2, y_2) – координати кінцевої точки переміщення.

Наступною функцією є виведення результату в консоль. В якості аргументів функції використано manager – менеджер індексів маршрутів (список індексів), routing – початковий індекс переміщення (зазвичай 0), solution – ціна проходження маршруту за обраним алгоритмом:

```
def print_solution(manager, routing, solution): # функція виведення даних в консоль
    """Виведення рішення в консоль"""
    print(f"Відстань: {solution.ObjectiveValue()}") # Ціна проходження маршруту за обраним алгоритмом
    index = routing.Start(0) # початковий індекс переміщення
    plan_output = "Маршрут:\n"
    route_distance = 0 # змінна для збереження відстані маршруту

    while not routing.IsEnd(index): # цикл перебирання всіх точок переміщення і їх індексів до кінцевої (якщо не кінцевий індекс)
        plan_output += f" {manager.IndexToNode(index)} ->" # виведення одного індекса переміщення, а далі знак ->
        previous_index = index
        index = solution.Value(routing.NextVar(index))
        # Повертає вартість дуги переміщення між двома вузлами.
        # Вхід – змінні індекси вузла. Це повертає 0, якщо транспортний засіб < 0.
        route_distance += routing.GetArcCostForVehicle(previous_index, index, 0)
    plan_output += f" {manager.IndexToNode(index)}\n"
    print(plan_output)
    plan_output += f"Маршрут: {route_distance}m\n".
```

В якості вхідної точки програми використано функцію main(), в якій додано наступні команди.

Визначаємо проблему з даними, для цього створимо об’єкт, що зберігатиме результат виконання функції розрахунку евклідової відстані між точками:

```
data = create_data_model().
```

Далі створюємо менеджер індексів маршруту (для керування маршрутом), який має у складі масив точок переміщення, кількість проходів (транспортних засобів або свердло), а також початкову і кінцеву точку переміщення свердла:

```
manager = pywrapcp.RoutingIndexManager(
    len(data["locations"]), data["num_vehicles"], data["depot"]
  ).
```

Створюємо модель маршрутизації, а також присвоюємо значення Евклідової відстані точок:

```
routing = pywrapcp.RoutingModel(manager)
distance_matrix = compute_euclidean_distance_matrix(data["locations"]).
```

Всередині створюємо ще одну локальну функцію, що перетворює змінну маршрутизації Index на матрицю NodeIndex відстані:

```
def distance_callback(from_index, to_index):
    """Повертає відстань між двома вузлами"""
    # Перетворення змінної маршрутизації Index на матрицю NodeIndex
відстані
```

```
        from_node = manager.IndexToNode(from_index)
        to_node = manager.IndexToNode(to_index)
        return distance_matrix[from_node][to_node].
```

Реєструємо зворотній виклик переміщення (в початкову точку переміщення):

transit_callback_index = routing.RegisterTransitCallback(distance_callback).

Визначаємо вартість кожної дуги переміщення:

routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index).

Встановлюємо евристики першого рішення:

```
search_parameters = pywrapcp.DefaultRoutingSearchParameters()
   search_parameters.first_solution_strategy = (
      routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC # метод для визначення евристики
      ).
```

В якості стратегії першого рішення було обрано PATH_CHEAPEST_ARC – що означає лише отримання мінімальної вартості, знайденої в першому процесі.

Далі знаходимо вирішення проблеми (переміщення свердла в точки) з параметрами пошуку за обраним методом (PATH_CHEAPEST_ARC):

solution = routing.SolveWithParameters(search_parameters).

Якщо отримаємо результат, то виводимо в консоль дані вирішення:

```
if solution:
    print_solution(manager, routing, solution).
```

В результаті виконання програми отримаємо такий алгоритм переміщення за індексами (рис. 3.5), а також довжину маршруту:

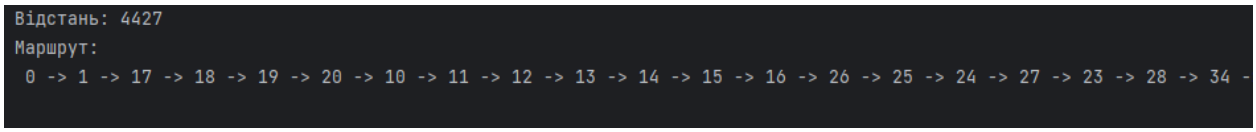

Рисунок 3.5 – Виведення результатів в консоль

Результат формування маршруту буде таким:

0 -> 1 -> 17 -> 18 -> 19 -> 20 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 26 -> 25 -> 24 -> 27 -> 23 -> 28 -> 34 -> 35 -> 36 -> 37 -> 38 -> 39 -> 40 -> 47 -> 48 -> 49 -> 50 -> 58 -> 57 -> 56 -> 55 -> 54 -> 53 -> 52 -> 69 -> 68 -> 67 -> 66 -> 73 -> 72 -> 71 -> 70 -> 82 -> 83 -> 84 -> 89 -> 88 -> 87 -> 86 -> 85 -> 81 -> 80 -> 79 -> 98 -> 99 -> 100 -> 101 -> 102 -> 103 -> 104 -> 90 -> 105 -> 106 -> 107 -> 108 -> 122 -> 123 -> 114 -> 113 -> 112 -> 111 -> 110 -> 109 -> 121 -> 120 -> 119 -> 118 -> 117 -> 116 -> 115 -> 91 -> 92 -> 93 -> 94 -> 95 -> 96 -> 97 -> 78 -> 77 -> 76 -> 75 -> 74 -> 64 -> 65 -> 63 -> 62 -> 61 -> 60 -> 59 -> 46 -> 45 -> 44 -> 51 -> 43 -> 42 -> 41 -> 33 -> 32 -> 31 -> 30 -> 29 -> 22 -> 21 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 0

Його довжина має 4427 пікселів.

При зміні стратегії (еврістики) першого рішення з PATH_CHEAPEST_ARC на FIRST_UNBOUND_MIN_VALUE (Виберіть перший вузол із незв’язаним наступним і підключіть його до першого

доступного вузла) маршрут буде змінено і його довжина також зменшиться (рис. 3.6):

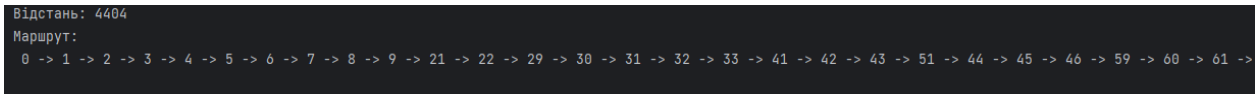

Рисунок 3.6 – Виведення результатів зі стратегією FIRST_UNBOUND_MIN_VALUE

Маршрут матиме такий вигляд:

0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 21 -> 22 -> 29 -> 30 -> 31 -> 32 -> 33 -> 41 -> 42 -> 43 -> 51 -> 44 -> 45 -> 46 -> 59 -> 60 -> 61 -> 62 -> 63 -> 65 -> 64 -> 74 -> 75 -> 76 -> 77 -> 78 -> 79 -> 80 -> 81 -> 98 -> 97 -> 96 -> 95 -> 94 -> 93 -> 92 -> 91 -> 115 -> 116 -> 117 -> 118 -> 119 -> 120 -> 121 -> 109 -> 110 -> 111 -> 112 -> 113 -> 114 -> 123 -> 122 -> 108 -> 107 -> 106 -> 105 -> 90 -> 104 -> 103 -> 102 -> 101 -> 100 -> 99 -> 85 -> 86 -> 87 -> 88 -> 89 -> 84 -> 83 -> 82 -> 70 -> 71 -> 72 -> 73 -> 66 -> 67 -> 68 -> 69 -> 52 -> 53 -> 54 -> 55 -> 56 -> 57 -> 58 -> 50 -> 49 -> 48 -> 47 -> 40 -> 39 -> 38 -> 37 -> 36 -> 35 -> 34 -> 28 -> 23 -> 27 -> 24 -> 25 -> 26 -> 16 -> 15 -> 14 -> 13 -> 12 -> 11 -> 10 -> 20 -> 19 -> 18 -> 17 -> 0

Довжина цього шляху складатиме 4404 пікселя.

При зміні стратегії (еврістики) першого рішення з FIRST_UNBOUND_MIN_VALUE на PARALLEL_CHEAPEST_INSERTION (ітеративно будує рішення, вставляючи найдешевший вузол у його найдешевшу позицію; вартість вставки базується на функції вартості дуги) маршрут буде змінено і його довжина також зменшиться (рис. 3.7):

Рисунок 3.7 – Виведення результатів зі стратегією PARALLEL_CHEAPEST_INSERTION

Довжина цього шляху складатиме 4500 пікселя і матиме наступний вигляд:

0 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 21 -> 22 -> 29 -> 30 -> 31 -> 32 -> 33 -> 41 -> 42 -> 43 -> 51 -> 44 -> 45 -> 46 -> 59 -> 60 -> 61 -> 62 -> 63 -> 64 -> 65 -> 66 -> 73 -> 72 -> 71 -> 70 -> 74 -> 75 -> 76 -> 77 -> 78 -> 97 -> 96 -> 95 -> 94 -> 93 -> 92 -> 91 -> 115 -> 116 -> 117 -> 118 -> 119 -> 120 -> 121 -> 109 -> 110 -> 111 -> 112 -> 122 -> 123 -> 113 -> 114 -> 108 -> 107 -> 106 -> 105 -> 90 -> 104 -> 103 -> 102 -> 101 -> 100 -> 99 -> 98 -> 79 -> 80 -> 81 -> 85 -> 86 -> 87 -> 88 -> 82 -> 83 -> 89 -> 84 -> 67 -> 68 -> 69 -> 52 -> 53 -> 54 -> 55 -> 56 -> 57 -> 58 -> 50 -> 49 -> 48 -> 47 -> 40 -> 39 -> 38 -> 37 -> 36 -> 35 -> 34 -> 28 -> 23 -> 27 -> 24 -> 25 -> 26 -> 16 -> 15 -> 14 -> 13 -> 12 -> 11 -> 10 -> 20 -> 19 -> 18 -> 17 -> 1 -> 0

При встановленні стратегії (еврістики) першого рішення з PARALLEL_CHEAPEST_INSERTION на SAVINGS (основою алгоритму Clarke and Wright Saving є розрахунок заощаджень, які вимірюються тим, наскільки скорочення пробігу та часу, витраченого на зв’язування вузлів, які існують, і створення маршруту на основі значення заощаджень найбільшого – це відстань між початковими вузлом і вузлом призначення) маршрут буде змінено і його довжина також зменшиться (рис. 3.8):

Рисунок 3.8 – Виведення результатів зі стратегією SAVINGS

А сам маршрут матиме наступний вигляд:

0 -> 17 -> 18 -> 19 -> 20 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 26 -> 25 -> 24 -> 35 -> 36 -> 37 -> 38 -> 39 -> 40 -> 47 -> 48 -> 49 -> 50 -> 58 -> 57 -> 56 -> 55 -> 54 -> 53 -> 52 -> 69 -> 68 -> 67 -> 66 -> 73 -> 72 -> 71 -> 70 -> 80 -> 81 -> 82 -> 83 -> 84 -> 89 -> 88 -> 87 -> 86 -> 85 -> 99 -> 100 -> 101 -> 102 -> 103 -> 114 -> 113 -> 123 -> 122 -> 112 -> 111 -> 110 -> 109 -> 121 -> 120 -> 119 -> 118 -> 117 -> 116 -> 115 -> 108 -> 107 -> 106 -> 105 -> 104 -> 90 -> 91 -> 92 -> 93 -> 94 -> 95 -> 96 -> 97 -> 98 -> 79 -> 78 -> 77 -> 76 -> 75 -> 74 -> 64 -> 65 -> 63 -> 62 -> 61 -> 60 -> 59 -> 46 -> 45 -> 44 -> 51 -> 43 -> 42 -> 41 -> 33 -> 32 -> 31 -> 30 -> 29 -> 34 -> 28 -> 27 -> 23 -> 22 -> 21 -> 9 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 2 -> 1 -> 0

Його довжина має 4441 піксель.

3.3 Охорона праці при роботі з електрообладнанням

Основні небезпеки роботи з електрикою:

– ураження електричним струмом і опіки від контакту з струмоведучими частинами;

– травми внаслідок дії дуги, пожежі через несправне електричне обладнання чи установки;

– вибух, спричинений невідповідним електричним обладнанням або статичною електрикою, що запалює легкозаймисті пари чи пил, наприклад, у камері для фарби.

Ураження електричним струмом також може призвести до інших типів травм, наприклад, через падіння з драбин, риштувань тощо.

Оцінка ризику повинна брати до уваги тип використовуваного електричного обладнання, спосіб його використання та середовище, в якому воно використовується.

У вологому середовищі невідповідне обладнання може потрапити під напругу та оживити навколишнє середовище. Плавкі запобіжники, автоматичні вимикачі та інші пристрої повинні мати правильні номінали для ланцюга, який вони захищають. Роз'єднувачі та коробки запобіжників слід тримати закритими і, по можливості, замкнутими.

Кабелі, вилки, розетки та арматура повинні бути достатньо міцними та належним чином захищеними для робочого середовища. Роботодавці повинні переконатися, що обладнання має доступний вимикач або ізолятор для швидкого відключення електроенергії в екстреній ситуації.

Користувачі електрообладнання, включаючи портативні прилади, повинні проводити візуальний огляд. Роботодавці та працівники повинні негайно вилучити обладнання з експлуатації та перевірити його, відремонтувати або замінити, якщо:

– вилка або роз'єм пошкоджені;

– кабель був відремонтований скотчем, не закріплений, або видно внутрішні дроти тощо;

– наявність опіків або плям (вказують на перегрів).

Ремонт повинен виконувати тільки компетентна особа (той, хто має необхідні навички, знання та досвід для безпечного виконання робіт).

Це гарна практика, якщо роботодавці організовують частіші перевірки предметів, які, швидше за все, будуть пошкоджені (наприклад, портативні електричні інструменти та обладнання, яке регулярно переміщується, або використовується часто чи в складних умовах). Менш часті перевірки необхідні для обладнання, яке менше ймовірно пошкодиться (наприклад, настільні комп’ютери тощо).

Візуальна перевірка зазвичай не потрібна для невеликих предметів, що живляться від батарейок, або обладнання, яке працює від адаптера живлення від мережі (ноутбуки, бездротові телефони тощо). Однак мережевий адаптер для такого обладнання слід перевіряти візуально.

Роботодавці повинні розглянути питання про те, чи слід електричне обладнання, включно з портативними приладами, більш формально перевіряти або перевіряти компетентною особою, враховуючи також інтервали, через які це слід робити.

Необхідно вжити заходів для перевірки та перевірки стаціонарних установок електропроводки, тобто ланцюгів від лічильника та споживача, що живлять вимикачі світла, розетки, під'єднувальне обладнання (наприклад, плити, фени) тощо, які мають проводитися регулярно, щоб уникнути погіршення що призводить до небезпеки. Цю роботу зазвичай має виконувати компетентна особа, зазвичай електрик.

Основні моменти, про які варто пам'ятати:

– переконайтеся, що працівники знають, як безпечно користуватися електрообладнанням;

– переконайтеся, що доступно достатньо розеток. переконайтеся, що розетки не перевантажені, використовуючи адаптери без запобіжників, оскільки це може спричинити пожежу;

– переконайтеся, що немає тягових кабелів, через які люди можуть спіткнутися або впасти;

– перед чищенням або налаштуванням приладів вимикайте та від'єднуйте їх від мережі;

– переконайтесь, що всі шукають електричні дроти, кабелі чи обладнання поблизу місця, де вони збираються працювати, і перевірте наявність знаків, які попереджають про небезпеку від електрики чи будь-яку іншу небезпеку. слід проводити перевірки навколо роботи та пам'ятати, що електричні кабелі можуть знаходитись у стінах, підлозі та стелі тощо (особливо під час свердління в цих місцях);

– переконайтеся, що кожен, хто працює з електрикою, має для цього достатні навички, знання та досвід. неправильне підключення розетки може бути небезпечним і призвести до смертельних нещасних випадків або пожеж;

– негайно припиніть використання обладнання, якщо воно виявилося несправним – віддайте його на перевірку компетентній особі;

– переконайтеся, що будь-яке електрообладнання, яке працівники приносять на роботу, або будь-яке найняте чи позичене, придатне для використання перед його використанням і залишається придатним, якщо його необхідно обслуговувати;

– розгляньте можливість використання пристрою захисного відключення між джерелом електрики та обладнанням, особливо під час роботи на відкритому повітрі або у вологому чи закритому місці.

ВИСНОВКИ

В результаті написання кваліфікаційної роботи було досягнуто мету, а саме створення програмного забезпечення для пошуку маршруту переміщення свердла під час операції свердління, що відбувається з використанням верстатів з числовим програмним керуванням.

Задачі, що вирішені у кваліфікаційній роботі:

– проведено аналіз сучасного стану розробок в області Industry 4.0 та числового програмного керування;

– проведено вибір і обґрунтування технічних засобів для створення програмного забезпечення формування маршруту переміщення свердла, зокрема було обрано мову Python та бібліотеку OR-tools;

– проведено розрахунок найкоротшого маршруту свердла для п’яти точок з використанням алгоритму Кларка і Райта;

– розроблено блок-схему алгоритму для створення програмного забезпечення;

– розроблено програмне забезпечення для формування найкоротшого маршруту переміщення свердла з використанням декількох алгоритмів, таки як: шлях найдешевшої дуги, перше незв'язане мінімальне значення, паралельна найдешевша вставка, збереження ;

– розглянуто питання охорони праці, зокрема електробезпеки в приміщенні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки з підготовки кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва, А.І. Бронніков. – Харків: ХНУРЕ, 2022. – 66 с.

2. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. ДП «УкрНДНЦ», 2016. – 31 с.

3. Є.С. Ключник Аналіз систем автоматизованого свердління у Industry 4.0 / Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2023. – Вип. 2., с. 60-65

4. Ключник Є.С. Автоматизація операцій свердління у епоху Industry 4.0 / 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 2. – Харків: ХНУРЕ. 2024. – с. 25-26.

5. Santos, Beatrice. Industry 4.0: an overview / Beatrice Santos // EMaDeS - Energy, Materials and Sustainable Developmen. - 2018, 6 p.

6. Моделі та методи кіберфізичних виробничих систем в концепції Industry 4.0 : монографія / І. Ш. Невлюдов, В. В. Євсєєв, А. О. Андрусевич, С. С. Максимова ; – Oktan Print – Prague. 2023. – 321 с.

7. Igor Nevliudov, Vladyslav Yevsieiev, Murad Omarov, Artem Bronnikov and Viacheslav Liashenko. Method of Algorithms for Cyber-Physical Production Systems Functioning Synthesis //International Journal of Emerging Trends in Engineering Research (IJETER), Volume 8 No.10 (October 2020), pp: 7465 – 7473.

8. Nasution, Mahyuddin. Industry 4.0. / Mahyuddin Nasution // IOP Conference Series: Materials Science and Engineering, 2020, 9 p.

9. Soori, Mohsen. Internet of Things for Smart Factories in Industry 4.0 / Mohsen Soori, Behrooz Arezoo, Roza Dastres // Elsevier B.V. on behalf of KeAi Communications Co, 2023, pp. 192-204.

10. Valle Enrique. Industry 4.0 enabling manufacturing flexibility: technology contributions to individual resource and shop floor flexibility / Daisy Enrique Valle, Érico Marcon, Fernando Santos Charrua, Alejandro Frank, // Journal of Manufacturing Technology Management, 2022, 20 p.

11. Zivanovic, Sasa Machine tools and Industry 4.0 – trends of development / Sasa Zivanovic, Slobodan Tabaković, Milan Zeljković // COMETa, 2018, 19 p.

12. Özenir, İpek. Industry 4.0 in Manufacturing / İpek Özenir, Gülsün Nakiboğlu // New Trends in Management Studies, 2019, pp.277-296.

13. Oyoun, Louie. Computer numerical control(CNC) / Louie Oyoun // Content uploaded by Louie Oyoun, 2020, 9 p.

14. Giridharan, P. Combined Drilling and Reaming Operation in Robot and CNC / . P Giridharan, Stalin John // IOP Conference Series: Materials Science and Engineering, 2020, 6 p.

15. Praveen, B. Industry 4.0 Researchers Computer Numerical Control Machine Tool to Manufacture Calligraphy Board / B Praveen,. S Abhishek, P Shetty, J. Reddy // Emerging Research in Computing, Information, Communicationand Applications, Lecture Notes in Electrical Engineering, 2021, pp. 197-205.

16. Ibrahim A.A, Capacitated vehicle routing problem / A.A. Ibrahim, N. Lo, R.O. Abdulaziz, J.A. // Ishaya International Journal of Research - Granthaalayah, 7(3), 2019, pp. 310-327.

17. Sangwan, Shabnam. Literature Review on Travelling Salesman Problem / Shabnam Sangwan // International Journal of Research. 5, pp. 1152-1155.

18. Total Population Sampling [Електронний ресурс]. – Режим доступу: https://www.statisticshowto.com/total-population-sampling.

19. Branch and bound [Електронний ресурс]. – Режим доступу: https://www.javatpoint.com/branch-and-bound.

20. Kalatzantonakis, Panagiotis. On a Cooperative VNS Parallelization Strategy for the Capacitated Vehicle Routing Problem / Panagiotis Kalatzantonakis, Angelo Sifaleras, Nikolaos Samaras // Learning and Intelligent Optimization,: Springer, Cham, 2020, pp.231-239.,

21. Solving the Traveling Salesman Problem Using Quantum Computer [Електронний ресурс]. – Режим доступу: https://medium.com/@michal.stechly/solving-the-traveling-salesman-problem-using-quantum-computer-bb00438de223.

22. Li, Shugang. A New Fast Ant Colony Optimization Algorithm: The Saltatory Evolution Ant Colony Optimization Algorithm / Shugang Li, , Wei Yanfang, Liu Xin, Zhu He, Yu Zhaoxu //Mathematics 10, no. 6:, 2022, pp.925-935.

23. A Gentle Introduction to Particle Swarm Optimization [Електронний ресурс]. – Режим доступу: https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization.

24. Katoch, S. A review on genetic algorithm: past, present, and future / S.Katoch, , S.S. Chauhan, V. Kumar // Multimed Tools Appl 80, 2021, pp. 8091–8126 ().

25. Petri Maatta. Comparison of solution methods for the dynamic pickup and delivery problem with time windows / Maatta Petri // Aalto University School of Science, 2021, 58 p.

26. Clarke & Wright's Savings Algorithm [Електронний ресурс]. – Режим доступу:http://52.56.215.113/legacy/maxoptra-scheduler/attachments/6979866/6979871.pdf.

27. Методичні вказівки до виконання розділу "Охорона праці" у випускних роботах ОКР "бакалавр" усіх форм навчання / упоряд.: В. А. Айвазов. Т. Є. Стиценко., Н. Л. Березуцька ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2018. – 28 с. – 1,81.