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

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

"На правах рукопису"
УДК *004.93*

До захисту допущено
В.о. завідувача кафедри
_____ Олександр ПАВЛОВ
“ ____ ” _____ 20 *20* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за освітньо-професійною програмою

«*Інформаційні управляючі системи та технології*»

зі спеціальності 126 «*Інформаційні системи та технології*»

на тему:

«Інформаційна система з підтримки роботи керівника мобільної групи з питань охорони та безпеки торгівельної мережі»

Виконав:
студент VI курсу, групи **ІС-з91мп**
Уманський Володимир Андрійович _____

Керівник:
Доцент, к.ф.-м.н. доцент
Гавриленко Олена Валеріївна _____

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

Рецензент:
д.т.н., професор,
Стенін Олександр Африканович _____

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

Київ – 2020 року

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

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Рівень вищої освіти – *другий (магістерський)*

Спеціальність – 126 «*Інформаційні системи та технології*»

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

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

«__» _____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Уманському Володимиру Андрійовичу

1. Тема дисертації «Інформаційна система з підтримки роботи керівника мобільної групи з питань охорони та безпеки торгівельної мережі», науковий керівник дисертації Гавриленко Олена Валеріївна, доцент, к.ф.-м.н. доцент, затверджені наказом по університету від «26» жовтня 2020 р. № 3133-с
2. Строк подання студентом дисертації “ *2* ” *12* 20*20* р.

3. Об’єкт дослідження: торгівля, процес роботи керівника служби безпеки та охорони.4. Перелік завдань, які потрібно розробити

5. Орієнтовний перелік графічного (ілюстративного) матеріалу

5.1. Плакат 1 Діаграма класів

5.2. Плакат 2 Діаграма варіантів використання

5.3. Плакат 3 Перегляд карти, режим з супутника

5.4. Плакат 4 Перегляд вулиць у додатку

5.5. Плакат 5 Схема стуктурна компонентів проекту toptw

6. Орієнтовний перелік публікацій

6.1 Уманський В. А. ІНФОРМАЦІЙНА СИСТЕМА З ПІДТРИМКИ РОБОТИ КЕРІВНИКА МОБІЛЬНОЇ ГРУПИ З ПИТАНЬ ОХОРОНИ ТА БЕЗПЕКИ ТОРГІВЕЛЬНОЇ МЕРЕЖІ [Електронний ресурс] / В. А. Уманський, О. В. Гавриленко. – 2020. – Режим доступу до

ресурсу: ІСТУ-2020_осень.pdf.

6.2 Уманський В. А. РОЗВ`ЯЗУВАННЯ ЗАДАЧІ ПІДТРИМКИ ПРАВЦІВНИКА МОБІЛЬНОЇ ГРУПИ З УРАХУВАННЯМ ЧАСОВИХ ВІКОН [Електронний ресурс] / В. А. Уманський, О. В. Гавриленко. – 2020. – Режим доступу до ресурсу: static/ua-articles-mods-2020.html.

7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Жданова Олена Григорівна		
2	Жданова Олена Григорівна		
3	Гавриленко Олена Валеріївна		
4	Гавриленко Олена Валеріївна		

8. Дата видачі завдання “ *1* ” *вересня* 20 *20* р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	*Систематизація результатів огляду літератури*	*30.09*	
2	*Порівняльний аналіз існуючих методів розв’язання задачі*	*15.10*	
3	*Постановка та формалізація математичної моделі задачі*	*30.10*	
4	*Модифікація існуючих методів розв’язання задачі*	*05.11*	
5	*Розробка інформаційного та програмного забезпечення*	*17.11*	
7	*Проведення експериментальних досліджень розроблених алгоритмів*	*18.11*	
8	*Оформлення документації*	*19.11*	
9	*Подання роботи на попередній захист*	*20.11*	
10	*Подання роботи на основний захист*	*02.12*	

Студент _____ *Володимир УМАНСЬКИЙ*

Науковий керівник _____ *Олена ГАВРИЛЕНКО*

РЕФЕРАТ

Актуальність. Для холдингу Fozzy дуже важливою компонентою є ефективна робота працівників, особливо тих, які працюють поза межами офісу. Департаменту дуже важливо розуміти, як працює ця категорія людей, які поставлені перед ними задачі, і як вони з ними справляються. Тому, використання інформаційних технологій з метою організації часу мобільної групи є актуальною задачею на сьогоднішній день, тому що це не тільки допоможе працівникам у роботі, а і надасть можливість керівництву чітко усвідомлювати ті задачі, які поставлені перед працівником і відслідковувати їх виконання та завантаженість кожного із них. Саме тому і набули широкого розповсюдження персоналізовані електронні органайзери (Microsoft Outlook), у функціонал яких покладено задачу організації робочого часу. Математична модель може дещо змінюватись у зв'язку зі зміною умов досліджуваної області. Математичною моделлю цієї роботи є задача Організації робочого часу Куратора СБ з використанням часових вікон (Security Leader Problem with Time Windows, SLPTW).

Мета роботи і задачі дослідження. Метою є підвищення ефективності організації процесу відвідування філій та вирішення задач працівником мобільної групи. Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналітику відомих методів вирішення задачі SLPTW;
- удосконалити існуючий метод розв'язання задачі за допомогою технологій паралельного програмування;
- програмно реалізувати алгоритм SLPTW;
- вирішити задачу алгоритмічної реалізації алгоритму;
- провести дослідження на ефективність реалізованого алгоритму.[]

Об'єкт дослідження – є процес прокладання маршрутів керівнику мобільної групи з питань охорони та безпеки торгівельної мережі.

Предмет дослідження – задача підвищення ефективності організації

робочого часу працівника мобільної групи.

Методи дослідження, використані в роботі, відносять до класу алгоритмів метаевристики.

Наукова новизна отриманих результатів базується на вдосконаленні алгоритму ILS (повторюваного локального пошуку) та у порівнянні його з алгоритмом SA (імітаційного відпалу), застосування паралельної форми обчислень паралельного у програмуванні з метою модифікації алгоритму повторюваного локального пошуку для вирішення проблематики задачі TOPTW.

Зв'язок роботи з науковими програмами, планами, темами. Робота реалізовувалась на кафедрі АСОІУ факультету ФІОТ Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» .

КЛЮЧОВІ СЛОВА : ДЕТЕРМІНОВАНИЙ ЛОКАЛЬНИЙ ПОШУК, ПОВТОРЮВАНИЙ ЛОКАЛЬНИЙ ПОШУК, АЛГОРИТМ ІМІТАЦІЇНОГО ВІДПАЛУ, ЗАДАЧА КОМАНДНОГО СПОРТИВНОГО ОРІЄНТУВАННЯ З ЧАСОВИМИ ВІКНАМИ, ЗАДАЧА ПОБУДОВИ МАРШРУТІВ, ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ У ПРОГРАМУВАННІ, АЛГОРИТМИ МЕТАЕВРИСТИКИ

ABSTRACT

For the Fozzy group holding, a very important component is the efficient work of employees, especially those who work outside the office. It is very important for the department to understand how this category of people who are tasked with it and how they handle it. Therefore, the use of information technology to organize the time of the mobile group is an urgent task today, because it will not only help workers in the work, but also allow management to clearly understand the tasks that are set before the employee and monitor their performance, and the workload of each of them.

In this regard, personalized electronic organizers (Microsoft Outlook), which include the functionality of working time, have become widespread. When solving it, the mathematical model may differ depending on which domain conditions are taken into account. In this paper, the mathematical model is the task of organizing the Security Leader Problem with Time Windows (SLPTW).

Since response time for software is an important feature, developing an efficient algorithm for the task at hand is an up-to-date task. Therefore, this work is dedicated to the research and refinement of SLPTW.

Purpose and tasks of the study. The goal is to maximize the aggregate value of visiting affiliates and solving problems with a mobile group employee.

To achieve this goal it is necessary to solve the following tasks:

- analyze known results of solving the SLPTW task;
- to develop a method (modification of an existing method) of solving a problem using parallel programming technologies;
- develop algorithmic support for the SLPTW task;
- to develop software implementation of algorithm (s);

Conduct research on the effectiveness of the developed algorithmic support.

The object of study is the process of drawing routes to a mobile group employee.

The subject of the study - the task of improving the organization of working hours of a mobile group employee.

The research methods used in the work are based on metaheuristic algorithms.

The scientific novelty of the obtained results is to modify the algorithm of repetitive local search, to compare it with the algorithm of imitative annealing, to use the parallel programming technologies to modify the algorithms of repetitive local search, and to simulate annealing algorithm to solve the problem of the SLPTW problem.

Relationship with working with scientific programs, plans, topics. The work was performed in the branch of the Department of Automated Information Processing and Control Systems of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky ».

KEY WORDS : DETERMINISTIC LOCAL SEARCH, ITERATED LOCAL SEARCH, SIMULATED ANNEALING ALGORITHM, TOURIST TRIP DESIGN PROBLEM, TEAM ORIENTEERING PROBLEM WITH TIME WINDOWS, PARALLEL PROGRAMMING, METAHEURISTIC ALGORITHMS

ЗМІСТ

ВСТУП

Торгівля в Україні є важливою галуззю економіки. За останні 5 років частка продукції АПК у структурі експортної виручки України зросла з 31% у 2014 році до 39,3% у 2018. Проте варто зазначити, що основу аграрного експорту все ще становить експорт сировини, а саме продукція рослинного походження – пшениця, кукурудза, ячмінь та соєві боби. Частка цієї продукції в структурі становить близько 55%. На сьогоднішній день в секторі торгівлі України працює близько 35% населення. Торгівля перетинається з діяльністю 40 галузей української економіки та входить до переліку п’яти галузей України, що приносять найвищі доходи.

Проте формування привабливого торгівельного іміджу країни може бути ускладнено політичними, соціально-економічними факторами, відсутністю достовірної та актуальної інформації щодо торгівельних ресурсів та послуг. Торгівля, як і всі інші форми господарської діяльності, потребує використання принципів маркетингу. Згідно з французькими науковцями Ланкаром та Ольє, маркетинг являє собою серію методів і прийомів, вироблених для дослідження, аналізу та вирішення поставлених завдань щодо найповнішого задоволення потреб покупців, а також визначення економічно раціональних способів ведення справ в галузі торгівлі. Основним фокусом маркетингу згідно даних Всесвітньої організації туризму (World Retail Organization, WRO) є впровадження нововведень в торгівлю [69]. Тому, використання інформаційних технологій з метою розвитку торгівлі, є актуальною задачею на сьогоднішній день. У зв’язку з цим, розробляється велика кількість мобільних і Web-застосувань для торгівлі. Наприклад, широкого розповсюдження набули персоналізовані електронні органайзери для рітейлу (Personalized ElectronicRetailOutlook, PERO), до функціональності яких входить кластеризація задач. Дана функціональність сприяє розповсюдженню актуальної інформації щодо торгівельних потужностей та ресурсів у доступному всім споживачам форматі. PETs дозволяють

користувачам будувати маршрути враховуючи їх пріорітети. При їх створені виникає проблема побудови маршрутів. Процес побудови цих маршрутів є об’єктом дослідження роботи. Формально описати дану проблему можна в термінах задачі побудови маршрутів з часовими вікнами, що є предметом дослідження. Метою є максимізація ефективності побудованих маршрутів заданої тривалості з врахуванням часових періодів відвідування необхідних філій. Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз відомих результатів розв’язування задачі SLPTW;
- розробити метод (модифікацію існуючого методу) розв’язання задачі з використанням технологій паралельного програмування;
- розробити алгоритмічне забезпечення задачі SLPTW;
- розробити програмну реалізацію алгоритму (ів);
- провести дослідження ефективності розробленого алгоритмічного забезпечення.[85]

З огляду на складність задачі SLPTW методами дослідження обрані метаевристичні алгоритми: повторюваний локальний пошук і метод імітаційного відпалу.

Науковою новизною в задачі є використання технологій паралельного програмування для зменшення часу роботи алгоритмів, аналіз роботи імплементованих алгоритмів, імплементація модифікації повторюваного локального пошуку. Дослідження проведені у напрямку пошуку ефективних методів розв’язування задачі SLPTW необхідні для конструювання якісного алгоритмічного забезпечення для PETs. Тому, дана робота присвячена дослідженню та удосконаленню розв’язування задачі SLPTW.

1 ПРОЕКТНІ РІШЕННЯ З РОЗРОБКИ СИСТЕМИ

1.1 Опис бізнес процесів

1.1.1 Опис процесу діяльності

Торгівля в Україні є важливою галуззю економіки. За останні 5 років частка продукції АПК у структурі експортної виручки України зросла з 31% у 2014 році до 39,3% у 2018. Проте варто зазначити, що основу аграрного експорту все ще становить експорт сировини, а саме продукція рослинного походження – пшениця, кукурудза, ячмінь та соєві боби. Частка цієї продукції в структурі становить близько 55%. На сьогоднішній день в секторі торгівлі України працює близько 35% населення. Торгівля перетинається з діяльністю 40 галузей української економіки та входить до переліку п’яти галузей України, що приносять найвищі доходи.

Проте формування привабливого торгівельного іміджу країни може бути ускладнено політичними, соціально-економічними факторами, відсутністю достовірної та актуальної інформації щодо торгівельних ресурсів та послуг. Торгівля, як і всі інші форми господарської діяльності, потребує використання принципів маркетингу. Згідно з французькими науковцями Ланкаром та Ольє, маркетинг являє собою серію методів і прийомів, вироблених для дослідження, аналізу та вирішення поставлених завдань щодо найповнішого задоволення потреб покупців, а також визначення економічно раціональних способів ведення справ в галузі торгівлі. Основним фокусом маркетингу згідно даних Всесвітньої організації туризму (World Retail Organization, WRO) є впровадження нововведень в торгівлю [69]. Тому, використання інформаційних технологій з метою розвитку торгівлі, є актуальною задачею на сьогоднішній день. У зв’язку з цим, розробляється велика кількість мобільних і Web-застосувань для торгівлі. Наприклад, широкого розповсюдження набули персоналізовані електронні органайзери для рітейлу (Personalized ElectronicRetailOutlook, PERO), до функціональності яких входить кластеризація задач. Дана функціональність сприяє розповсюдженню актуальної інформації щодо торгівельних потужностей та ресурсів у доступному всім споживачам форматі. PETs дозволяють

користувачам будувати маршрути враховуючи їх пріорітети. При їх створені виникає проблема побудови маршрутів. Процес побудови цих маршрутів є об’єктом дослідження роботи. Формально описати дану проблему можна в термінах задачі побудови маршрутів з часовими вікнами, що є предметом дослідження. Метою є максимізація ефективності побудованих маршрутів заданої тривалості з врахуванням часових періодів відвідування необхідних філій.

Також для створення комфортних умов праці необхідно, з огляду на обов’язки акторів системи, інтегрувати можливий функціонал у інформаційну систему, що розробляється, а саме: календар, графіки та табличку з достовірною інформацією по усім критичним показникам для куратора та інспектора, таких як графік втрат, таблична інформація, показники стосовно динаміки показників втрат, продажів, заповненості штату і некоректно проведених переобліків. А також спроектувати графік для порівняння відсотка втрат на минулий та поточний рік. Варто додати календар, для проектування та планування зустрічей та фіксування планів на тиждень, місяць, а може і на рік. У систему також інтегрований чек-лист служби безпеки, для полегшення роботи куратора та інспектора. Адже з огляду на завантаженість співробітників служби безпеки необхідно завжди мати у вільному доступі консолідовану інформацію стосовно усіх критичних показників, необхідних для роботи та оперативного реагування на зміни будь-якого із показників. Календар необхідно впровадити у систему, тому що вище керівництво, а також аналітики консолідованої інформації майже щоденно запрошують певну інформацію. Дану інформацію, як правило необхідно зібрати за лічені дні, а оскільки звіти надходять майже щоденно, тому куратору (а, як правило, досить часто куратор може делегувати збір частини інформації інспектору служби охорони) потрібно додати календар та нагадування про термін здачі певного звіту, так як тримати усі необхідні для виконання задачі у голові досить складно та проблематично.

1.1.2 Актори та функції

Акторами у системі є:

- куратор служби безпеки та охорони;
- інспектор служби безпеки та охорони.

Куторатор має право переглядати інформацію стосовно своїх магазинів, переглядати інформацію по втратах у розрізі кожного магазину, своїх магазинів у цілому, у розрізі відділів, макрогруп та артикулів. Також куратор має право заповнювати чек-лист, переглядати список своїх магазинів, будувати маршрути собі на день, переглядати інформацію щодо протяжності маршруту та послідовності відвідування своїх магазинів. Також має право створювати зустрічі та події у календарі, планувати свій робочий час, проводити оцінювання персоналу магазину та якості обслуговування.

Побудова маршрутів куратору необхідна для того, щоб підвищити ефективність своєї роботи, а також організувати свій час для якісних перевірок, навчання та оцінки персоналу, а також для запобігання усіх надзвичайних подій, перевіряючи роботоспроможність камер відеоспостереження, встигаючи провести усі корегуючі переобліки, а також провести інвентаризацію усіх матеріальних цінностей.

Інспектор служби охорони має право переглядати інформацію стосовно своїх магазинів, переглядати інформацію по втратах у розрізі кожного магазину, своєї магазинів у цілому, у розрізі відділів, макрогруп та артикулів. Також інспектор має право заповнювати чек-лист, переглядати список своїх магазинів, собі на день, переглядати інформацію щодо протяжності маршруту та послідовності відвідування своїх магазинів. Також має право створювати зустрічі та події у календарі, планувати свій робочий час, проводити оцінювання персоналу магазину та якості обслуговування.

1.1.3 Стукрура бізнес процесів

В інформаційній системі існують наступні бізнес процеси:

– підтримка процесу планування робочого часу з допомогою календаря;
– побудова маршрутів за допомогою алгоритму touristOrientingProblem;
– проведення переобліку;
– візуалізація інформації по втратах та поточних показниках торгівельної мережі.

Бізнес процес проведення чек-листа можна описати наступним чином: куратору або інспектору необхідно перейти на сторінку чек-листа, потім обрати магазин, у якому вони проводять чек-лист, працівнику відкриються пункти чек-листа, а потім куратор чи інспектор має можливість відмічати галочками пункти чек-листа, які, на їхню думку, магазин пройшов. Після проведення чек-листа працівникам доступна кнопка «Провести чек-лист». Після завершення чек-листа куратору доступні результати проходження чек-листа магазином у відсотковому графіку. А також у розробці відображення статистики по проходженню чек-листів по всіх магазинах куратора, щоб було легше виявити слабке місце, над яким необхідно попрацювати. А також перегляд статистики за рік з проходження чек-листа певним магазином.

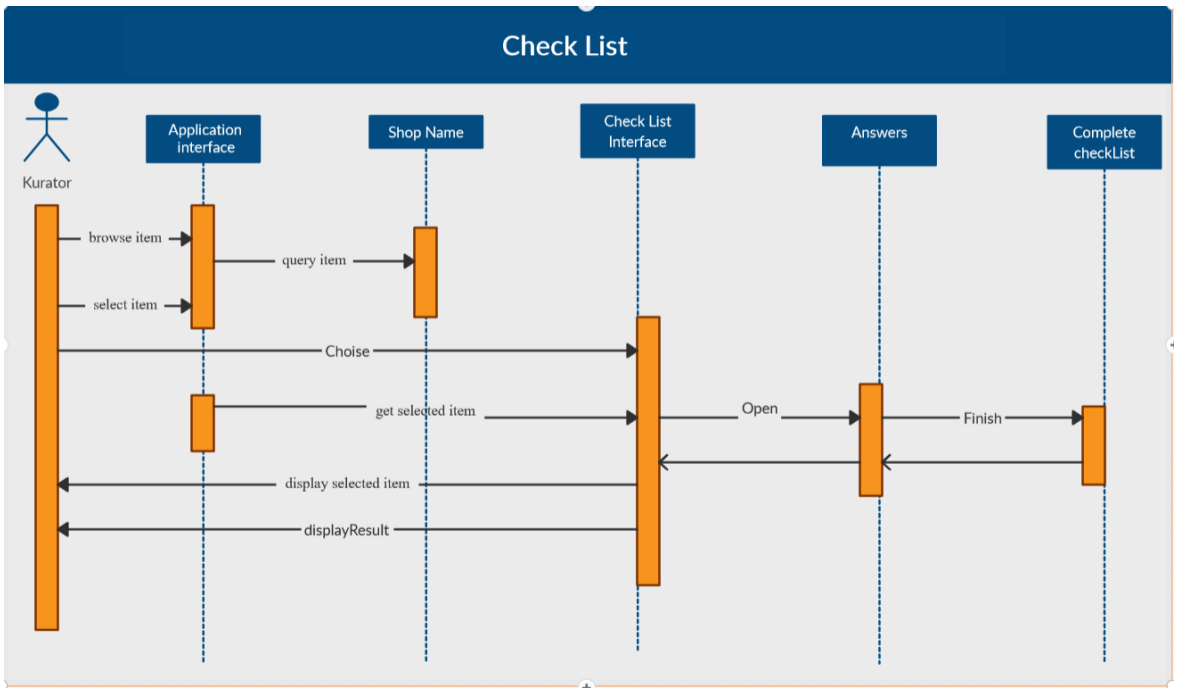


Рисунок 1.1.3.1 – Стуктура бізнес процесу «Проведення чек-листа»

Бізнес процес побудови маршрутів описується наступним чином: куратор

повинен перейти або відкрити основну сторінку додатку. Оновити обов'язково інформацію по втратах, які є одним із основних складових розрахунку рейтингу локацій. Опуститись вниз сторінки, натиснути кнопку «Відправити запит» та очікувати. У свою чергу в програмному забезпеченні оновлюються вхідні дані: оновити показниики втрат, передати в програмний код вхідні дані вигляду: локація широта та довгота, показники втрат, дистанції. Відбувається розрахунок маршрутів, потім дані передаються в в скрипт JavaScript, відображаються на мапі граф маршруту з використанням GoogleMapsApi, а також розраховується відстань між точками у кілометрах, а наступний крок – всі дані передаються в інтерфейс та відображаються користувачу. Бізнес процес зображено на рисунку 1.1.3.2.

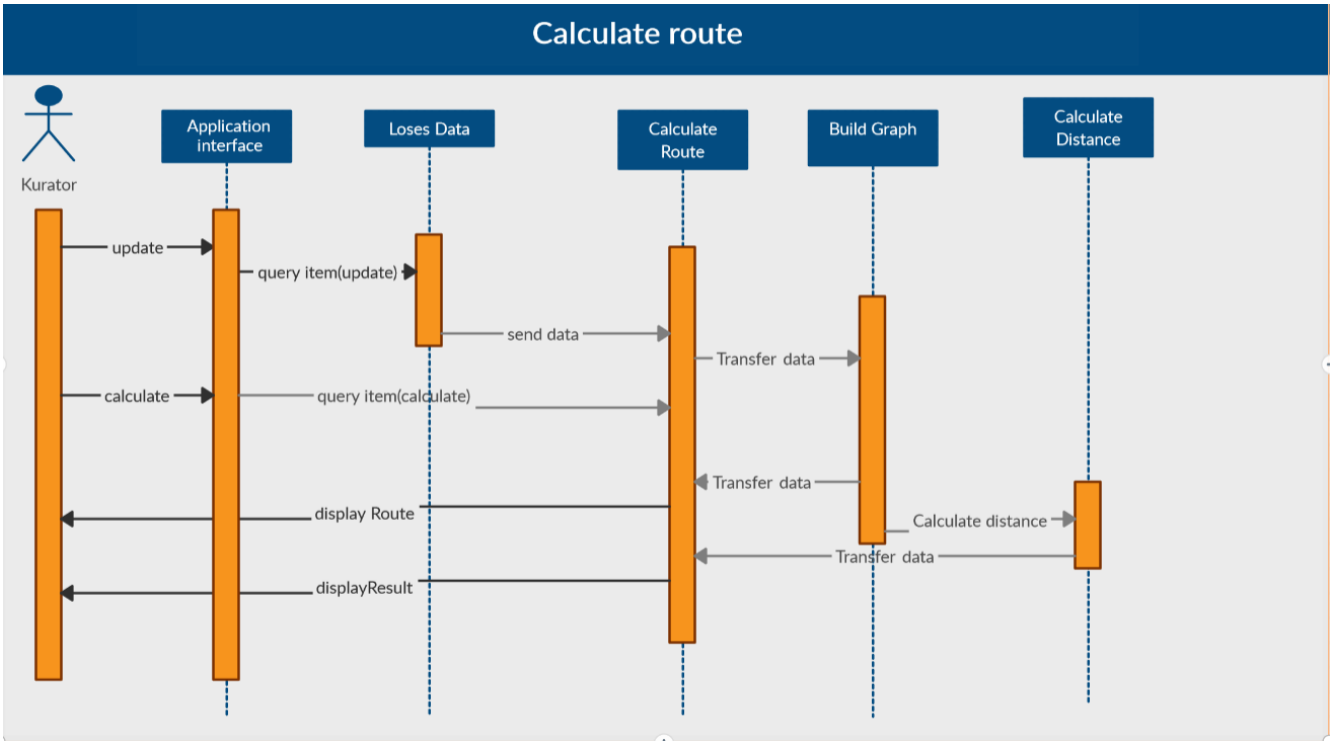


Рисунок 1.1.3.2 – Стуктура бізнес-процесу побудови маршруту

Бізнес процес проведення корегуючого переобліку можна описати наступним чином: куратор чи інспектор служби охорони перевіряє інформацію по втратах або користується наданою аналітиком інформацією, створює комісію у складі себе, якщо це інспектор, то повідомляє куратора про переоблік, керуючого та заступника керуючого магазином. Створює відомості переобліку.

Та разом із комісією проводить переоблік по стореній відомості. Спочатку комісія повинна зняти дані про продажі, тоді необхідно перерахувати товар на складі. Внести у відомість. Потім потрібно перерахувати товар у торгівельній залі. Після цього ще раз зняти дані про продажі перераховуваних артикулів. Якщо щось продалось, обов'язково помітити собі. І знову обновити дані по залишкам. Обрахувати сальдо фактично порахованого товару із залишком із бази даних.

Після цього внести результати переобліку в базу даних. Переобліки поділяються на види: плановий, корегуючий та частковий. Куратор чи інспектор служби охорони мають право проводити часткові переобліки. Охоронець разом з інспкетором проводить частковий переоблік по певним довіреним відділам. А при плановому переобліку куратор і інспектор обов'язково мають бути присутні на довірених їм магазинах та контролювати корректність проведених переобліків.

Бізнес процес планування робочого часу, за допомогою календаря, необхідний куратору служби охорони для того, щоб фіксувати перелік необхідних до виконання задач та кінцеві терміни їх виконання. Також для внесення дат об'їздів усіх магазинів, планування сумісних об'їздів та контролю з іншими кураторами, планування усіх мірроприємств та зустрічей, на яких обов'язково повинен бути присутнім. І на мою думку, комбінація усіх складових у моєму додатку необхідна для зосередження усієї інформації в одному місці, для комфортної роботи співробітників офісу служби безпеки та охорони.

1.2 Опис постановки задачі

Задача складання маршрутів була сформульована у [70]. Дана задача ставить на меті сформувати робочі маршрути на один чи декілька днів, що обмежені у часі. Маршрути повинні максимізувати сатисфакцію від відвідування туристичних об'єктів. Дана задача виникла як спосіб автоматизації процесу складання туристичних маршрутів у реальному часі, оскільки на час написання роботи (2007 рік) більшість користувалася газетами, журналами, інтернет

виданнями про туризм для планування поїздок. Тому автори запропонували концепцію мобільного застосування, що розв'язує дану проблему. В їх роботі математичною постановкою служить проблема спортивного орієнтування (Orienteering Problem, OP). OP являє собою задачу з області Дослідження операцій [64]. В такому формулюванні TTDP на виході має один маршрут. Подальшу роботу над даною проблемою вони опублікували у [59]. Детальніше розглянуто проблему визначення необхідності відвідати філію за допомогою векторної моделі [13] і запропоновано метод розв'язання OP – керований локальний пошук [6].

Необхідно зазначити, що формулювання TTDP у термінах OP обмежує можливості користувачів застосування, що описане у вищерозглянутих роботах, адже, зазвичай тури включають декілька днів, а математична постановка у термінах OP дозволяє лише один маршрут. У подальших роботах дослідників, що працюють над тематикою, представлені інші підходи до формування математичної постановки для TTDP з одним маршрутом. Необхідно зазначити, що вагома частина пізніших робіт присвячена TTDP з багатьма маршрутами.

Обширний огляд з даної проблеми проведено у [5]. Розглянуто такі аспекти, що пов'язані з TTDP:

- постановка задачі TTDP і її варіації, що виникають від того, в якій мірі враховуються реальні умови предметної області;
- основний функціонал систем, що розв'язують TTDP;
- математичні постановки, в термінах яких може бути представлена TTDP;
- перспективні напрямки досліджень.

До додаткової функціональності можна віднести:

- візуалізація на карті;
- навігація.

У літературі можна знайти різні назви для застосувань (здебільшого мобільних застосувань), що вирішують проблему TTDP. Наприклад, у [5]

застосовується назва «Персоналізовані електронні путівники» (Personalized Electronic Tourist guides, PETs). Згідно з [58], PETs включають таку основну функцінальність:

- генерування списку філій, які має відвідати працівник;
- складання маршрутів;
- функціональність для редагування згенерованих маршрутів.

За генерування списку об'єктів відповідає рекомендаційна система, а складання маршрутів – це алгоритм, який на основі даних, що генеруються рекомендаційною системою, формує розв'язок – філія маршрут(и). Маршрут являє собою послідовність робочих об'єктів. Він обмежений в часі. Оптимальне рішення являє собою маршрут максимальної корисності (сатисфакції).

В багатьох публікаціях підсистеми генерування списку робочих об'єктів і складання маршрутів розглядаються в комплексі, бо перша генерує вхідні дані для другої. Огляд відносно рекомендаційних систем для TTDP наведено у [31].

В літературі для TTDP наведено ряд задач в рамках яких наведено математичну постановку. Це спричинено тим, що при розв'язанні проблеми врахувати всі обмеження предметної області практично неможливо, оскільки їх надзвичайно багато, наприклад, деякі з них:

- різний час роботи філій (з 7 до 18; з 8 до 23 та цілодобово);
- різний час на рух у різних напрямках між двома пунктами;
- різний час на проходження однієї ділянки, в залежності від часу доби.

Крім того, об'єкти можуть бути представлені як видовими маршрутами, так і окремими пунктами; кількість маршрутів може бути різною. Тому дослідники, зазвичай, обирають декілька аспектів, на яких вони концентруються. Проте, мінімальний набір вхідних даних для будь-якої математичної постановки такий:

- набір ТО.
- для кожного ТО можуть бути надані певні атрибути (місцезнаходження, робочі години, тип і т. д.);

- час на переміщення між ТО;
- корисність кожного ТО, що генерується рекомендаційною системою;
- кількість маршрутів, що повинна бути згенерована;
- час на відвідування ТО;
- час відведений для кожного маршруту.

Перелік задач в рамках яких наводиться математична постановка задачі TTDP наведено в таблиці 1.2.1.

Таблиця 1.2.1 – Математичні постановки TTDP

Назва	Абревіатура	Опис
Chinese postman problem	CPP	Обхід всіх ребер графа
Rural postman problem	RPP	Обхід певних ребер графа
Capacitated arc routing problem	CARP	Обхід ребер декількома транспортними засобами
Maximum benefit chinese postman problem	MBCPP	Обхід ребер графа з метою максимізації корисності від відвідування ребер
Prize-Collecting Rural Postman Problem	PCRPP	Побудова маршруту максимальної корисності, по ребрам, щоб разово отримати користь від їх відвудування
Maximum benefit chinese postman problem	MBCPP	Обхід ребер графа з метою максимізації корисності від відвідування ребер
Prize-Collecting Rural Postman Problem	PCRPP	Побудова маршруту максимальної корисності, по ребрам, щоб разово отримати користь від їх відвідування.
Orienteering Problem	OP	Маршрут обмежений у часі максимальної корисності корисності, коли корисність задана для вершин
Vehicle routing problem	VRP	Маршрути мінімальної вартості, що задовольняють нормам, встановленим для кожної вершини
Team Orienteering Problem	TOP	Маршрут обмежений у часі максимальної корисності, коли корисність задана для вершин
Arc Orienteering Problem	AOP	Маршрут обмежений у часі максимальної корисності, коли корисність задана для ребер
Team Orienteering Arc Routing Problem	TOARP	Маршрути обмежений у часі максимальної корисності

Продовження таблиці 1.2.1

Team Orienteering Problem with Time Windows	TOPTW	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задані часові вікна і корисність
Mixed Team Orienteering Problem with Time Windows	MTOPTW	Маршрути обмежений у часі максимальної корисності коли для певних вершин і ребер, що включені і мають корисність, задані часові вікна
Mixed Team Orienteering Problem with Time Windows	MTOPTW	Маршрути обмежений у часі максимальної корисності коли для певних вершин і ребер, що включені і мають корисність, задані часові вікна
The Chinese Postman Problem with load-dependent costs	CPP-LC	Обхід всіх ребер графа, де вартість відповідних ребер змінна
Time-dependent Team orienteering problem	TDTOP	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задана корисність. Час проходження по ребрах залежить від години доби.
Time-dependent Team Orienteering Problem with Time Window	TDTOPTW	Маршрути максимальної корисності, що обмежені у часі, коли для вершин задані часові вікна і корисність. Час проходження по ребрах залежить від години доби.
The Stacker Crane Problem	SCP	Маршрут на змішаному графі мфнімальної вартості, що включає всі дуги

1.3 Огляд постановок задачі TTDP

1.3.1 Паралельні обчислення – перспективний напрямок в розробці алгоритмічних методів для TTDP

Одна з основних задач в розробці алгоритмічних методів для TTDP – це швидка реакція на запити користувача. Паралельні обчислення можуть стати механізмом для досягнення хороших результатів у цьому напрямку. Враховуючи вищерозглянуті методи розв’язання задачі, евристичні та метаевристичні методи найкраще піддаються для розпаралелювання, оскільки простір рішень надає багато варіацій для паралельних обчислень. Наприклад, згідно [36] можна представити паралельну версію локального пошуку за допомогою розбиття

простору рішень на підмножини. Таким чином, можна запустити евристику на кожній підмножині.

Інший варіант – запустити евристику чи різні евристики на загальному просторі рішень починаючи з одного чи різних початкових розв'язків. В ході роботи потоки можуть обмінюватись результатами відносно кращих знайдених розв'язків на даний момент. Цікавий аспект цих підходів полягає у тому, що вони також можуть забезпечити генерування нових евристик з якіснішими розв'язками, оскільки вони можуть шукати в просторі рішення і поєднувати рішення так, що це було б дуже дорого імітувати за послідовної реалізації. Хоча паралельна евристика були запропоновані в літературі для VRP і TSP [23, 25, 26, 56, 4] паралельні рішення для TTDP відсутні (або нечисленні, тому й не охоплені під час огляду літератури), і розробка нової паралельної евристики для TTDP може зменшити час реакції застосувань на запит користувачів.

1.3.2 Практичні застосування задачі SLPTW

Окрім TTDP, задача SLPTW має інші практичні застосування:

- задача краудсорсингу;
- задача розподілу медичного персоналу для домашніх візитів;
- задача створення туристичних маршрутів.

Дані проблеми можна розглянути у рамках задачі маршрутизації транспортних потоків (Vehicle routing problem, VRP), але формулювання у термінах ОР дозволяє краще врахувати специфіку предметних областей даних задач.

Задача краудсорсингу, коли існує набір робіт для краудсорсингу і набір працівників, що можуть виконувати різні роботи, була розглянута у [67] як задача класу ОР. Зазвичай застосування для краудсорсингу, для формування набору задач роблять відбір на основі територіального положення працівника. Обираються роботи, що лежать в певному радіусі, що охоплює поточне положення працівника. Підходи у [51, 66] направлені на те, щоб запропонувати більш гнучкий підхід до даної проблеми. Пункти, що рекомендуються працівнику

обираються на основі його траєкторії і корисності робіт. Оцінка корисність робіт може надаватись залежно від таких факторів як оплата за роботу, навички працівника, вподобання працівника і т.д.

Задача розподілу персоналу була розглянута в літературі для індустрії надання медичних послуг в домашніх умовах. Коли дану задачу сформувати у термінах VRP, то дана вона постає як проблема задоволення попиту клієнтів. Коли задача формується у термінах OP, то вона розглядається з точки зору компанії, що надає послуги. Постановка у термінах OP дозволяє врахувати ситуацію, коли кількість клієнтів така велика, що перевищує можливості персоналу надати медичні послуги. Візитам до пацієнта має певну корисність, щоб максимізувати сумарну сатисфакцію обирається максимальний набір клієнтів, що зможе обслужити компанія за певний проміжок часу.

У [60] представлено задачу призначення медичних візитів в домашніх умовах з врахуванням випадковості у часі обслуговування і переміщення. Було запропоновано методи розв’язку даної проблеми, що тестувались на синтетичних наборах даних. Були сформовані набори як для детермінованих, так і стохастичних випадків.

У [30] задачу сформовано у термінах TOP with soft Time Windows and Variable Profit, TOPsTWVP. На меті ставиться максимізація сатисфакції пацієнтів, тому кожен візит має корисність. Передбачається, що візити обмежені у часі, але можливі також і візити з запізненням. Запізнення призводять до зниження корисності візиту. Також враховується, що пацієнти очікують певного лікаря, тому, якщо візит заплановано не з цим лікарем, то корисність знижується.

Треба зазначити, що проблеми класу OP добре підходять для формулювання задач побудови маршрутів для розподілу персоналу у різних предметних областях, не лише для області надання медичних послуг.

Якщо сформувати дану задачу у термінах TOPTW, то можна врахувати такі фактори:

- попит на послуги більший за пропозицію, тому не всі клієнти можуть бути обслуговані за виділений час;

- обслуговування відбувається у рамах певних часових вікон;
- щоб надати послуги наступному клієнту необхідно витратити час на переміщення до нього;
- обслуговування клієнта займає певний час;
- обслуговування клієнтів може мати різну корисність.

1.4 Постановка задачі дослідження

Мета дисертаційної роботи – є максимізація сумарної корисності від відвідування філій та вирішення задач працівником мобільної групи.

Для досягнення поставленої мети, необхідно вирішити такі **задачі**:

- провести аналіз відомих результатів у розв'язанні задачі SLPTW;
- розробити метод розв'язання задачі з застосуванням паралельних обчислень;
- розробити алгоритмічне забезпечення задачі;
- розробити програмну реалізацію алгоритму;
- провести дослідження ефективності розробленого алгоритму.

Об'єкт дослідження – процес складання туристичних маршрутів.

Предмет дослідження – задача командного орієнтування з врахуванням часових вікон.

Методи дослідження, застосовані в роботі, базуються на методах дослідження операцій, паралельного програмування, метаевристичних та евристичних алгоритмах.

1.5 Висновки до розділу

Було наведено огляд існуючих задач у термінах яких можна сформулювати задачу складання туристичних маршрутів, в результаті чого, було обрано задачу командного спортивного орієнтування з врахуванням часових вікон. Для неї було проведено огляд існуючих методів розв'язання і виявлено, що тема використання паралельних обчислень слабкодосліджена.

З метою пошуку ефективного розв'язання задачу командного спортивного

орієнтування з врахуванням часових вікон, була сформульована відповідна постановка задачі та мета дослідження.

Слід зазначити, що дослідження направлені на розв’язання задачі командного спортивного орієнтування з врахуванням часових вікон мають практичне значення не лише для проблеми складання туристичних маршрутів, а й для ряду задач, що пов’язані з побудовою маршрутів для надання різноманітних послуг клієнтам. Було наведено короткий огляд цих задач. Левова частка програмного забезпечення, що створюється для автоматизації даних процесів, є мобільними застосуваннями, тому розробка ефективних і швидких алгоритмів для розв’язання задач класу ОР є актуальною темою наразі.

2 МОДЕЛІ ТА МЕТОДИ

2.1 Змістовна постановка задачі

Необхідно побудувати набір робочих маршрутів, щоб відвідати місця певної місцевості. Куратори перебувають на певному об’єкті декілька годин. Час, що виділяється на відвідування місць ними є обмеженим кожен день. В результаті може виникнути ситуація, що вони не можуть відвідати всі місця за час їх візиту. Тому для місць задається оцінка, що позначає задоволеність туристів від їх відвідування. Таким чином, можна розставити пріоритети і формувати набір маршрутів з огляду них.

При побудові набору маршрутів вважається, що всі вони починаються і закінчуються в одному місці. Кінцевий і початковий пункти призначення не обов’язково повинні співпадати.

Необхідно при формуванні маршрутів враховувати ряд обмежень, що викликані часовими рамками. Не всі місця можна відвідати цілодобово. На переміщення між місцями і їх відвідування повинно відводитись достатньо часу. Також, може виникнути ситуація, коли треба витратити час на очікування відкриття певного місця. Для відвідування кінцевого і початкового пунктів час не відводиться. Підсумовуючи, основною метою є побудова маршрутів на кожен день, що максимізується загальна сумарну задоволеність кураторів від повного набору і враховуються всі вищенаведені обмеження. В частині графічного матеріалу змістовну постановку задачі представлено на прикладі.

2.2 Математична поставновка задачі

Математичним формулюванням наведеної змістовної постановки є задача Security Lead Problem with Time Windows (SLPTW). SLPTW – задача максимізації сумарної корисності набору маршрутів, що обмежені у часі.

Пункти (місця), що треба відвідати, можна представити як вершини орієнтовного повного зваженого графу. Граф містить n вершин. Необхідно побудувати m маршрутів у цьому графі. Кожен маршрут починається з вершини $i = 1$, а закінчується в вершині $i = n$.

Вхідні дані:

n – кількість пунктів;

m – кількість маршрутів, що треба побудувати;

R_i – корисність від відвідування пункту $i = \overline{1,n}$

O_i – початок часового вікна, коли можна відвідати пункт $i = \overline{1,n}$

C_i – кінець часового вікна, коли можна відвідувати пункт $i = \overline{1,n}$

T_i – час необхідний на відвідування пункту $i = \overline{1,n}$

T_{max} – максимальна тривалість кожного з маршрутів;

M – константа, значення якої велике відносно задіяних даних;

c_{ij} – час, що необхідний на переміщення між пунктами

Змінними є:

S_{id} – час початку візиту до пункту $i = \overline{1,n}$ на маршруті $d = \overline{1,m}$ змінна, якій присвоюється значення 1, тільки тоді, коли на маршруті $d = \overline{1,m}$ пункт $j = \overline{1,n}$ відвідується відразу після пункту $i = \overline{1,n}$.

Вихідні дані:

$$\mathbf{y_{id}} = \begin{cases} \mathbf{1}, \text{ якщо на маршруті } \mathbf{d} \in \overline{\mathbf{1,m}} \text{ відвідується пункт } \mathbf{i} \in \overline{\mathbf{1,n}} \\ \mathbf{0}, \text{ у іншому випадку} \end{cases}$$

Цільова функція:

Визначається формулою $\sum_{d=1}^{m} \sum_{i=2}^{n-1} R_i y_{id} \longrightarrow \max$,означає,що ефективність від відвідування всіх пунктів, що включаються в маршрути максимізується.

Обмеження:

$\sum_{j=2}^{n-1} \sum_{d=1}^{m} x_{1id} = \sum_{i=2}^{n-1} \sum_{d=1}^{m} x_{ind} = m$ - початковий і кінцевий пункти повинні бути відвідані рівно m разів;

$\sum_{i=1}^{n-1} x_{ikd} = \sum_{j=2}^{n} x_{kjd} = y_{kd}, k = \overline{2, n-1}, d = \overline{1,m}$ – правило збереження потоку;

$s_{id} + T_i + c_{ij} - s_{jd} \le M(1 - x_{ijd}), i = \overline{1,n},\ j = \overline{1,n}, d = \overline{1,m}$ – забезпечує, що відрізок часу між двома пунктами, що відвідуються постлідовно достатній, щоб виділити час не тільки на відвідування першого, а і переміщення між двома послідовними пунктами;

$\sum_{d=1}^{m} y_{kd} \le 1, k = 2, \overline{n-1}$ – забезпечує, що пункт не буде відвідано більше разу на всіх маршрутах;

$\sum_{i=1}^{n-1}(T_I y_{id} + \sum_{j=2}^{n} c_{ij} x_{ijd}) \le T_{\max}, d = \overline{1,m}$ – забезпечує, що загальна тривалість кожного маршруту не більше T_{max};

$O_i \le S_{id}, i = \overline{1,n}, d = \overline{1,m}$ та $S_{id} = C_i, i = \overline{1,n}, d = \overline{1,m}$ – відвідування пункту повинно початись в рамках заданого часового вікна;

$C_n = T_{\max}$ – кінцевий пункт можна відвідати не пізніше Tmax;

$T_1 = T_n = 0$ – кінцевий і початковий пункти не потребують часу для відвідування.

Математична постановка задач має вигляд:

$$\sum_{d=1}^{m}\sum_{i=2}^{n-1} R_i y_{id} \longrightarrow \max \qquad (2.1)$$

$$\sum_{j=2}^{n-1}\sum_{d=1}^{m} x_{1id} = \sum_{i=2}^{n-1}\sum_{d=1}^{m} x_{ind} = m \qquad (2.2)$$

$$\sum_{i=1}^{n-1} x_{ikd} = \sum_{j=2}^{n} x_{kjd} = y_{kd}, k = \overline{2,n-1}, d = \overline{1,m} \qquad (2.3)$$

$$s_{id} + T_i + c_{ij} - s_{jd} \le M(1 - x_{ijd}), i = \overline{1,n},\ j = \overline{1,n}, d = \overline{1,m} \qquad (2.4)$$

$$\sum_{d=1}^{m} y_{kd} \le 1, k = 2, \overline{n-1} \qquad (2.5)$$

$$\sum_{i=1}^{n-1}(T_I y_{id} + \sum_{j=2}^{n} c_{ij} x_{ijd}) \le T_{\max}, d = \overline{1,m} \qquad (2.6)$$

$$O_i \le S_{id}, i = \overline{1,n}, d = \overline{1,m} \qquad (2.7)$$

$$S_{id} = C_i, i = \overline{1,n}, d = \overline{1,m} \quad (2.8)$$

$$x_{ijd}, y_{id} \in \{0,1\}, i = \overline{1,n}, d = \overline{1,m} \quad (2.9)$$

$$C_n = T_{\max} \quad (2.10)$$

$$T_1 = T_n = 0 \quad (2.11)$$

Розв'язком задачі є:

- розбиття множини вершин графу на підмножини (маршрути);
- задання порядку обходу на кожній підмножині.

Розв'язок є прийнятним (допустимим), якщо всі маршрути задовольняють оюмеженням задачі.

2.3 Огляд методів розв'язання

Алгоритми детермінованого локального пошуку (ДЛП) – це розділ ітераційних алгоритмів, принцип роботи яких заснований на частковому переборі варіантів на кожному кроці ітерації між точками околу поточної вершини, тобто усіх точок, що є сісудніми до поточної. В методах та алгоритмах цього класу застосовується направлений локальній перебір у підмножинах сусідніх точок, що носять назву – околи, замість повного Звідси і випливає назва даного класу алгоритмів – алгоритми локального пошуку. У сфері комбінаторної оптимізації (КО) саме клас алгоритмів локального пошкуку мають великий попит через свою високу ефективність та наочність. Загальна схема алгоритма локального пошуку може бути пояснена наступною логікою:

КРОК 1. Підбір або генерація допустимого можливого рішення *x*, який і обирається як поточний варіант розв'язку.

КРОК 2. Наступна ітерація .

Формується окіл O(x) поточного варіанта та точно або наближено знаходиться елемент $y \in 0(x)$, який є наближено оптимальним розв'язком у обраному околі. Якщо $y \neq x$, то обрана точка оголошується черговим поточним варіантом (здійснюється переприсвоєння x←y) і починається чергова ітерація, інакше – повернення на п. 3.

КРОК 3. Завершення роботи алгоритму: x – локальний розв'язок, якщо на останній ітерації здійснюється вичерпний пошук в околі [81].

Отже, результатом роботи алгоритму локального пошуку із заданим початковим наближенням x^0 створюється масив точок x^1, … , x^m , що мають наступні характеристики:

- розмірність m послідовності є невідомою заздалегідь;
- кожен слідуючий розв'язок в послідовності належить околу попереднього;
- значення цільової функції кожного наступного розв'язку краще за попереднє;
- x^0 є локальним розв'язком із зони притягання x^i.

Під терміном «зона притягання» розуміється така частина простору допустимих розв'язків, у якій цільова функціє є унімодальною. Локальний пошук подібний простому алгоритму спуску з тією відмінністю, що околи поточного розв'язку досліджуються систематично замість безладного блукання та пошук в околі повторюється до тих пір, поки не буде знайдений локально оптимальний розв'язок.

Відомою задачею з класу ARP є Задача (китайського) листоноші (Chinese postman problem або Route inspection problem) [46] метою якої є пошук найкоротшого циклу в графі, що включає всі ребра. Існують варіанти задачі для орієнтованих та неорієнтованих графів та для графів, частина ребер яких орієнтована, а частина – ні.

Задача отримала назву завдяки Алану Голдману з NIST, що назвав її так, бо першим дослідником цієї задачі був китайський математик Мей-Ку Кван.

Задача визначена на графі G = (V, E ∪ A), де V являє собою множину вершин, а E – множину ребер, а A – дуг. Граф G називається орієнтовним, якщо множина E порожня, неорієнтованим, якщо A порожня, і змішаним, якщо множини E і A не порожні. В відповідності до цього розрізняють орієнтовану, неорієнтовану і змішану задачі китайського листоноші. Узагальненням цих трьох

задач є Windy Postman Problem. Вона визначена на ненаправленому графі, де ребро може мати різну вартість обходу у різних напрямках.

Мета: знаючи вартість обходу c_{ij} ребер або/та дуг (v_i, v_j) необхідно побудувати маршрут найменшої вартості T представлений вектором виду $(v_1, v_2, \dots, v_n)$, де (v_i, v_{i+1}) належить $E \cup A$ при $i = 1, \dots, n-1$ та $v_n = v_1$. Даний маршрут повинен включати всі ребра з $E \cup A$.

2.3.1 Задача Rural postman problem

Задача сільського листоноші (Rural postman problem) є узагальненням CPP. Задано множину $S \subset E \cup A$ і необхідно побудувати замкнений маршрут мінімальної вартості через ребра, що містяться у S. Щоб звести RPP до CPP треба множину S визначити такою, що вона включає всі ребра з $E \cup A$.

Мета: знаючи вартість обходу c_{ij} ребер або/та дуг (v_i, v_j) необхідно побудувати маршрут найменшої вартості T представлений вектором виду $(v_1, v_2, \dots, v_n)$, де (v_i, v_{i+1}) належить $E \cup A$ при $i = 1, \dots, n-1$ та $v_n = v_1$. Даний маршрут повинен включати всі ребра з S.

2.3.2 Задача Capacitated arc routing problem

Задача Capacitated arc routing problem є розширенням задачі RPP. Вводяться додаткові обмежень пропускної здатності на ребрах (дугах). Є m однакових транспортних засобів, кожен з яких має ємність Q. Задається вершина графу що являє собою депо, і кожне необхідне ребро має невід'ємний попит. Маршрут допустимий, якщо він містить депо і якщо сумарний попит не перевищує місткість Q. Число m транспортних засобів може бути задане апріорі або може бути змінною для якої приймається рішення.

Мета: знайти множину допустимих маршрутів для транспортних засобів, що мають сумарну мінімальну вартість, а кожне необхідне ребро обслуговується рівно одним транспортним засобом [9].

2.3.3 Задача Maximum benefit chinese postman problem

Вперше дана задача була сформульована Маландракі і Даскіном [46] у 1993 році і визначена на неорієнтовному графі.

Для кожного ребра e ∈ E ∪ A визначено прибуток від проходження $p^i(i_e)$ = 1,2. . . n_e. Цю величину можна трактувати як потребу у сервісі певного ребра. З кожним проходом і по e прибуток p_e^i зменшується, а після того, як через ребро e пройдено n_e разів, він дорівнює нулю. Це означає, що дане ребро більше не потребує сервісу і подальші проходження по цьому ребру можна вважати збитковими. Витрати c^i i = 1,2…n на проходження по ребру e незмінні, а коли по ребру пройдено n_e разів, витрати для проходження цього ребра зростають. Результуюча корисність проходження по певному ребру визначається як $p^k - c^k$, де k позначає скільки разів ребро e зустрічається у маршруті.

Мета: знайти маршрут для якого результуюча сумарна корисність ребер буде максимальною.

2.3.4 Задача Prize-Collecting Rural Postman Problem

Задача була сформульована вперше Араосом, Фернандезом і Золтаном у 2006 році. Вона являє собою частковий випадок задачі MBCPP, коли кожен прибуток з ребра можна отримати лише при першому проходженні, тобто [11].

Мета: знайти цикл через депо для якого результуюча сумарна корисність ребер (дуг) буде максимальною.

2.3.5 Orienteering Problem

Задача вперше була представлена у [37]. Вона отримала свою назву, бо її математична постановка якнайкраще моделює гру спортивного орієнтування. Суть гри в тому, що спортсмени за обмежений час відвідують пункти в певній місцевості. Всі вони починають і повинні закінчити в наперед визначених пунктах. За відвідування інших пунктів нараховуються бали, причому, кількість балів різна для кожного пункту. Перемагає той спортсмен, маршрут якого матиме найбільшу сумарну кількість балів.

В літературі дана задача зустрічається і під іншими назвами: Selective Traveling Salesperson Problem (STSP) [38], Maximum Collection Problem (MCP) [39] and Bank Robber Problem (BRP) [12,40].

Задано початкову і кінцеву вершини. Для кожної вершини визначено корисність c_{ij} від проходження по ній. Більш того, лише перше проходженняприносить корисність. Задано величину t_{ij} для дуг (i, j), що зазвичай трактується як час потрібний на проходження дуги. На час проходження для всього побудованого маршруту задане обмеження величиною T.

Мета: знайти цикл через початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

2.3.6 Arc Orienteering Problem

Дана задача визначена на направленому графі [57]. Задано початкову вершину з якої треба побудувати маршрут. Для кожної дуги визначено корисність c_{ij} від проходження по ній. Більш того, лише перше проходження дугою приносить корисність. Задано величину t_{ij} для дуг (i, j), що зазвичай трактується як час потрібний на проходження дуги. На час проходження для всього побудованого маршруту задане обмеження величиною T.

Мета: знайти цикл через початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

2.3.7 Задача маршрутизації транспортних засобів

Формально класичну задачу маршрутизації ТЗ (Vehicle routing problem) можна представити так: дано граф, де множина вершин визначає клієнтів, яких треба відвідати, і депо, з якого починаються і яким закінчуються всі маршрути [34]. Кожен клієнт характеризується певним попитом . Дуги, що поєднують вершини зважені. Вагою дуги є час переїзду ТЗ від клієнта до клієнта , цей час включає в себе час обслуговування клієнта. Кожен клієнт обслуговується тільки одним транспортним засобом і тільки один раз. Транспортний засіб не може

обслужити більше клієнтів, ніж дозволяє його вантажопідйомність. Кожен автомобіль покидає депо і прибуває туди один раз. На рисунку 2.3.7.1 схематично зображено можливий розв’язок задачі.

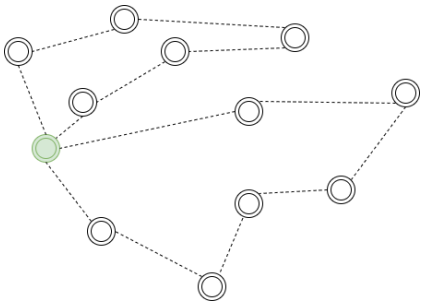


Рисунок 2.3.7.1 – Схематичне зображення можливого розв’язку задачі VRP

Мета: побудувати маршрути, що являють собою розбиття множини вершин графу, такі, що сумарний витрачений час мінімальний, а всі потреби клієнтів задоволено.

2.3.8 Team Orienteering Problem

Задача спортивного командного орієнтування (TOP) була вперше представлена у [50]. Вона являє собою розширення задачі OP до випадку з багатьма маршрутами. В літературі її можна також знайти під назвою Multiple Tour Maximum Collection Problem (MTMCP) [20].

На рисунку 2.3.7.1 схематично зображено можливий розв’язок задачі, коли необхідно побудувати 2 маршрути. Початкова вершина зеленого кольору.Той факт, що не всі вершини включено, є основною відмінністю даної задачі від VRP.

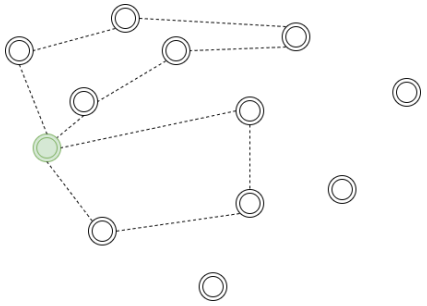


Рисунок 2.3.8.1 – Схематичне зображення можливого розв’язку задачі TOP

Мета: знайти k початкову вершину максимальної прибутковості, щоб час проходження ним не перевищував T.

2.3.9 Team Orienteering Arc Routing Problem

Задача Team Orienteering Arc Routing Problem являє собою розширення AOP. Задано декілька транспортних засобів для яких потрібно побудувати маршрути, що не перетинаються один з одним.

Одним із застосувань TOARP є транспортування вантажними автомобілями. Деякі клієнти повинні обслуговуватися, тоді як інші можуть бути відкладені або взагалі не обслуговуватися. Наприклад, обслуговування найменш прибуткових клієнтів може бути передане на аутсорсинг.

Мета: Побудувати набір маршрутів таких, що час на проходження кожного не перевищує T і всі обов'язкові дуги пройдено. Сумарна корисність по всім маршрутам повинна бути максимальною [2].

2.3.10 Team orienteering problem with time windows

Дана задача є розширенням TORP, у тому сенсі, що кожна дуга має час візиту і час відвідування. Таким чином, прибуток з відвідування дуги можна отримати лише в певному часовому вікні. Вперше дана задача була представлена у [70].

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

2.3.11 Time-dependent Team orienteering problem

Дана проблема є розширенням TOP. Задача ускладнюється тим, що в різні проміжки часу проходження по певній дузі може бути швидшим або довшим. Наприклад, якщо розглянути випадок коли маршрут частково проходиться пічки, а частково проїжджається громадським транспортом. Знаючи розклад громадського транспорту, можна оцінити скільки займе маршрут. На загальний час маршруту буде впливати скільки займає пройти певні відстані пішки, час очікування громадського транспорту і час переїздів. Якщо пішохід обиратиме

маршрут, що дозволить йому підлаштуватися під розклад громадського транспорту, це дозволить скоротити час переміщення між певними ділянками.

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

2.3.12 Time-dependent Team Orienteering Problem with Time Window

TDTOPTW – це проблема, яка добре моделює складні та реалістичні вимоги TTDP. TDTOPTW є особливо складною, оскільки він додає часові залежності, що присутні у TOPTW. У [7979] описано проблему, яка відноситься до TDTOPTW, і ROSE – мобільний додаток, що допомагає знаходити події та місцеположення, переміщаючись через зв'язки громадського транспорту.

Мета: Побудувати набір маршрутів, таких, що кожен з них починається і завершується в одній вершині, час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

2.3.13 Mixed Team Orienteering Problem with Time Windows

Задача TOAP має аналог для вершин. Умови залишаються незмінними, але корисність накопичується за проходження вершин, а не ребер. Задача має назву Team Orienteering Problem (TOP).

Задача Mixed Team Orienteering Problem with Time Windows є комбінацією TOP і TOAP, причому задані часові вікна. Тобто, корисність задається ребрам і вершинам [28].

Мета: Побудувати набір маршрутів, таких, що час на проходження кожного не перевищує T. Сумарна корисність по всім маршрутам повинна бути максимальною.

2.3.14 The Chinese Postman Problem with load-dependent costs

Задача буле предсталена у 2016 році як розширення CPP [62]. Сформульована вона з метою побудови маршрутів, спланованих таким чином,

щоб викиди вуглекислого газу спричинені транспортним засобом зменшувались.

Задача визначена на зв'язаному неорієнтовному графі. Задана вершина, яка являє собою депо. Для кожного ребра $e \in E$ визначено попит $q_e \geq 0$ і довжину $d_e \geq 0$ ребра. Попит можна трактувати як вагу солі, що необхідно відвантажити на ребрі. Вага транспортного засобу складається з спорядженої маси і маси, що навантажена на транспортний засіб $\sum q_e$. Коли транспортний засіб проходить по ребру e він відвантажує на ньому вагу q_e. Таким чином, його сумаран вага зменшується. $q_e = 0$ для ребер, що вже були пройдені. Витрати на проходження ребра визначаються як добуток повної маси транспортного засобу і довжини ребра. Результатом того, що повна маса транспортного засобу змінюється під час проходження по ребрам витрати на проходження ребер не є постійними виличинами.

Мета: Побудувати цикл через депо мінімальної вартості, що охоплює всі ребра $e \in E$.

2.3.15 The Stacker Crane Problem

Задача Stacker Crane Problem [64] отримала свою назву від застосування до практичної проблеми експлуатації крана. Кран, починаючи з вихідної вершини, повинен виконати набір рухів, і повернутися в початкове положення.

SCP була запропонована Фредеріксоном, Хехтом, і Кімом, які описали три версії задачі в залежності від кінцевого положення крана. В першій версії кран повинен повернутися в точку з якої починав. В другій, кран зупиняється на певній вершині, не обов'язково тій, де починав, в той час як для третьої варіації дозволяється закінчити в будь-якій вершині. Перша версія може легко бути приведена до другої.

Задача визначена на мішаному графі $V \in (E \cup A)$, де V являє собою множину вершин, а E – множину ребер, а A – дуг, множина E не порожня і A не порожня. Кожна ланка (дуга або ребро) графа має пов'язану з ним невід'ємну вартість.

Мета: знайти тур мінімальної вартості, знаючи початкову вершини, такий,

що всі елементи множини A включено у маршрут.

2.4 Огляд існуючих методів розв’язання задачі SLPTW

За своєю складністю задачі SLPTW не менш складна, ніж TOP. В свою чергу, TOP є розширенням задачі OP (OP є частковим випадком TOP) тому, як і OP, TOP і TOPTW є NP-складними задачами. Доведення того, що OP є NP-складною задачею наведено у [29].

SLPTW може бути оптимально розв’язана точними методами. Втім, такі методи можуть використовуватись лише для розв’язку задач із невеликою кількістю вузлів. До цих методів відносяться динамічне програмування і метод гілок та меж. Наприклад, у [61] SLPTW розв’язано точними методами для графу з 30 вершинами.

Зважаючи на складність проблеми, переважна більшість літератури присвячена розв’язанню проблеми неточними методами з застосуванням евристичних та метаевристичних методів.

Евристичні методи – сукупність прийомів в пошуку розв’язання задачі, які дозволяють обмежити перебір, тобто проводиться відносно обмежений пошук по простору рішень, і зазвичай знаходять розв’язок за прийнятний час. Недолік даних методів в тому, що вони є приблизними, розв’язок, отриманий даними методами, може бути далеким від оптимального. Перевага в тому, що вони дозволяють знайти прийнятний розв’язок NP-повних задач великої розмірності за прийнятний час. Евристичні методи часто протиставляються формальним методам розв’язання, які спираються, наприклад, на точні математичні алгоритми.

Метаевристики – загальна назва для будь-якого стохастичного алгоритму оптимізації, який використовується в якості «останньої надії» на шляху до вирішення завдання з використанням випадкового пошуку або повного перебору. Тобто це такий метод вирішення широкого класу обчислювальних завдань шляхом комбінування існуючих процедур з відкритим інтерфейсом і закритою реалізацією, яке призводить до максимально ефективного вирішення. В метаевристичних методах акцент робиться на ретельне вивчення найбільш

перспективних частин простору рішень. Якість одержуваних рішень виходить вищою, ніж у отриманих класичними евристиками.

Здебільшого, існуючі методи для розв’язання SLPTW являють собою метаевристики, що включають:

- крок вставки нового пункту у одному з маршрутів, що ітеративно відбувається, доки не знайдено допустиме рішення;
- крок диверсифікації для уникнення локального оптимуму.

Ці два кроки повторюються, доки критерій виходу не задовольняється. Залежно від критерію вставки, існуючі методи розроблені як детерміновані (ті, що завжди продукують однаковий розв’язок для певних вхідних даних), стохастичні чи імовірносні (ті, що включають ступінь випадковості в генерацію розв’язку). Вважається, що застосування імовірносних методів приводить до генерації розв’язків високої якості, бо проводиться більш обширний пошук у просторі розв’язків, за рахунок підвищення часу виконання.

У [65] запропоновано алгоритм локального пошуку SLPTW зі змінними околами. У ході виконання алгоритм намагається замінити сегмент шляху вершинами, що не включені у маршрут, та приносять більшу корисність. Для цього вирішується задача про призначення, що відноситься до SLPTW, і на основі рішення вставляються нові дуги у маршрут.

У [43] запропоновано алгоритм на основі метаевристики імітації відпалу для SLPTW. На кожній ітерації з околу поточного розв’язку генерується новий шляхом перестановок, інверсій чи вставок з однаковою імовірністю. Отриманий розв’язок стає новим, якщо він кращий за попередній (корисність більша), в протилежному випадку він приймається з певною імовірністю, що зменшується відповідно втраті корисності. Після певної кількості ітерацій цього алгоритму застосовуються локальний пошук.

У [43] запропоновано алгоритм на основі ітеративного локального пошуку, що застосовує кроки «струсу» і «вставки». Крок вставки відповідальний за додавання послідовності вершин, таким чином, щоб розв’язок залишався

допустимим (кожна вершина повинна відвідуватись у своєму часовому вікні). Для кожної вершини, що може бути вставлена, визначається відношення корисності до затримки у часі, що спричиняє вставка. Обирається вершина з найбільшим значенням даного відношення. Крок збурення застосовується для уникнення локального оптимуму. На кожному кроці заміняться для кожного маршруту одна чи більше вершин на ті, що не включені в розв'язок, незважаючи на їх корисність.

Алгоритм мурашиної колонії запропонований для розв'язку задачі у [72]. Комбінацію жадібного рандомізованого адаптивного пошуку і еволюційного локального пошуку застосовано у [35, 35].

2.4.1 Ключові аспекти реалізації детермінованого локального пошуку

Ключовими моментами для реаліалізації алгоритму локального пошуку є:

- визначення околів;
- створення чергової точки;
- критерій завершення перегляду точок у поточному околі та переходу до наступного;
- метод обрахунку значення величини змінної цільової функці при переході до нового кращого розв'язку;
- критерій завершення;
- формувати початкове наближення.

Ефективність роботи алгоритму локального пошуку залежить від обраного типу початкового околу $O(x)$. А чим більший окіл, тим більша імовірність отримати кращий результат, але практика розширення околів стає делалі непрактичнішою. Утім для багатьох задач КО алгоритми з околами розміром більше, ніж $O(n^2)$, де n позначає розмірність задачі, стають неефективними й тому досить рідко використовуються на практиці. Найчастіше за все використовуються метричні околи чи околи, які були утворені алгоритмічним методом (приклад : Окіл замін Ліна).

Можливі наступні методи вибору точки в околі:

- до першого покращення значення цільової функції;
- до кращого значення.

Таким чином пошук всередині околу може здійснюватись або до першого кращого варіанта цільової функції, або методом повного перебору усіх сусідніх точок. У деяких модифікаціях методів локального пошуку використовуються околи з радіусом, що є неконстантним. Іншим моментом, що має вплив на ефективність роботи алгоритму локального пошуку, є організація перебору точок в околах. Можливі два варіанти організації перебору точок в околі при переході між ітераціями:

- лінійний генератор – при зміні поточного варіанта в новому околі має початок від породження елементів околу;
- кільцевий генератор – точки, що містяться в околі є певним чином упорядковані і уже після переходу до наступної ітерації продовжується перебір точок, починаючи з наступної у відповідністю до зазначеного упорядкування.

У цих випадках чи програмно, чи явно визначається упорядкування операторів зсуву, що утворюють точки в околі, і послідовно формуються такі точки. Однак у випадку номер один після того, як був обраний кращий варіант перебір у новому околі починається з початкового оператора зсуву, а в другому – продовжується з поточного оператора зсуву. Відповідно ці способи дістали назву лінійного та кільцевого генератора. Перевагою алгоритму локального пошуку над іншими алгоритмами евристики є те, що масив усіх можливих варіантів буде досліджено досить ефективно: замість обчислення значення цільової функції для $y \in O(x)$, достатньо обрахувати тільки сальдо використавши властивості прикладних задач, щоб уникнути явного обчислення і перевірити, чи є величина додатною.

Інакше кажучи, алгоритм ЛП може опрацьовувати рішення у просторі пошуку за той же час, за який, наприклад, еволюційний алгоритм може оцінити

тольки одне окреме рішення. ЛП використовує цей плюс – як і інші алгоритми пошуку в околах, як і ті, що описані нижче.

Критерієм завершення роботи алгоритми вважають наступні умови:

- відсутність кращої точки в околі поточного розв'язку, тобто перегляд усіх точок околу не був закінчений з покращенням;
- вихід за межі заданого часу;
- проведення уже заданої кількості ітерацій.

Як початковий варіан розв'язку найчастіше обирається довільний припустимий розв'язок. Для його знаходження часто використовують випадкове генерування варіантів розв'язку з подальшою перевіркою їх на припустимість – інколи таких варіантів генерується кілька, а з них вибирається найкращий. Також слід зазначити, що використовуються розв'язки, які знайшли алгоритми евристики; при цьому максимально враховується специфіка задач.

Необхідно вказати, що усі обмеження задачі логічно враховуються при пошуку чергового варіанта рішення, при перевірці його на допустимість. У комбінаторних просторах окіл часто визначається з використанням метрик [81]. У деяких застосуваннях використовуються спеціальні метрики, які враховують особливості задач. Наприклад, для ЗК поняття сусідства може вводитися на основі 2-замін Ліна чи вставлення фрагментів.

До методів локального пошуку передусім належать:

- пошук зі змінними околами;
- табу-пошук;
- керований локальний пошук.

2.4.2 Специфіка детермінованого локального пошуку для SLPTW

Далі під локальним пошуком (ЛП) будемо вважати детермнінований локальний пошук. Наведемо основні ознаки ЛП для SLPTW.

Окіл поточного розв'язку формується з розв'язків, що утворюються шляхом додавання однієї вершини, що не включається в розв'язок. Окіл включає всі допустимі варіанти розміщення цієї вершини. Початковим розв'язком є множина

маршрутів, що складаються з початкової і і кінцевої вершин.

Чергова точка генерується шляхом включення однієї вершини в «найвигіднішу» позицію. До значення цільової функції додається корисність вставленої вершини. Після включення нової вершини новий розв’язок стає поточним і розглядається окіл нового розв’язку.

ЛП завершується, коли не можливо додати нову вершину в маршрути через часові обмеження на тривалість маршрутів та часових вікон.

Перехід в околі відбувається коли знаходиться «найвигідніша» позиція. Детальніше це описано при описі алгоритму вставки нової вершини. Перебір точок відбувається за принципом лінійного генератору.

2.5 Метод повторюваного локального пошуку

У якості методу розв’язання було обрано повторюваний локальний пошук (Iterated local search, ILS). Основною ідеєю, що лежить в основі ILS, є зосередження пошуку не на всьому просторі пошуку, а на розв’язках, що формуються деяким алгоритмом, що лежить в основі. Як правило, вбудованим алгоритмом є локальний пошук (Local search, LS). Отриману поведінку ILS можна охарактеризувати як ітеративну побудова ланцюга рішень вбудованого алгоритму. Результатом є концептуально проста метаевристика, яка, тим не менш, призвела до появи сучасних ефективних алгоритмів для багатьох складних обчислювальних задач. Зазвичай, дуже хороша продуктивність часто досяжна навіть при досить простих реалізаціях метаевристики [44].

Крім того, модульна архітектура ILS робить його дуже підходящим для інженерного підходу у побудові алгоритму, коли продуктивність алгоритму поступово може бути додатково оптимізована.

Проста ідея, що лежить в основі ILS мала багато інтерпретацій в літературі, тому можна зустріти різні назви, такі як Iterated descent [14, 15], Large-step Markov chains [48], Iterated Lin-Kernighan [36], Chained local optimization [47] і їх комбінації [10]. Детальний огляд історії ILS наведено у [37].

2.6 Загальна схема алгоритму ILS

Для розгляду загальної схеми ILS вважатимемо, що вбудованим алгоритмом є LS. Нехай f – цільова функція, задачею комбінаторної оптимізації є мінімізація цільової функції. Допустимий розв'язок задачі – s, множина всіх допустимих розв'язків – S. Нехай в результаті роботу LS для s генерується детермінованим чином розв'язок s^*, причому, значення $f(s^*) \leq f(s)$. Розв'язки, що генеруються процедурою LS формують множину S^*. На рисунку 2.1 наведено розподіл щільності ймовірностей для $f(s^*)$ та f(s) для комбінаторних задач. Дисперсія та математичне сподівання для $f(s^*)$ менші ніж для f(s). Тому проводити пошук множині S^* ефективніше, ніж в S. Елементи S^* не відомі, тому визначити окіл і застосовувати LS на множині нема можливості. Замість цього, використовується алгоритм ILS. Будується послідовність значень, що належать S^*. Спочатку послідовність має тільки одне значення – s^*, тож саме воно є поточним на першій ітерації. Щоб отримати наступне значення послідовності, маючи поточне значення, виконується збурення s^*, внаслідок цього формується розв'язок s', що належить S. З нього генерується новий розв'язок $s^{*\prime}$, що належить S^*. Якщо даний розв'язок задовольняє певному критерію прийнятності, то він стає поточним значенням послідовності і буде використаний для пошуку наступного значення послідовності. У протилежному випадку послідовність не змінюється, процедура знову повторюється для s^*. Важливо, щоб збурення, що вносяться в поточний елемент послідовності були вагомими. Якщо зміни не вагомі, то ILS може згенерувати той самий поточний елемент, до якого вносились зміни. Занадто радикальні зміни можуть призводили до того, що ILS матиме природу випадкового пошуку. Тобто, коли випадковим чином генеруються значення з S для кожного з них виконується локальний пошук і обирається найкращий розв'язок.

Оскільки детерміновані збурення можуть призводити до циклів, то необхідно робити їх адаптованими до зациклення або рандомізованими. Якщо перестановки залежать від попередніх значень з S^*, то формується послідовність у S^* з пам'яттю цільової функції Вищезгаданий критерій прийнятності визначає

природу і ефективність.

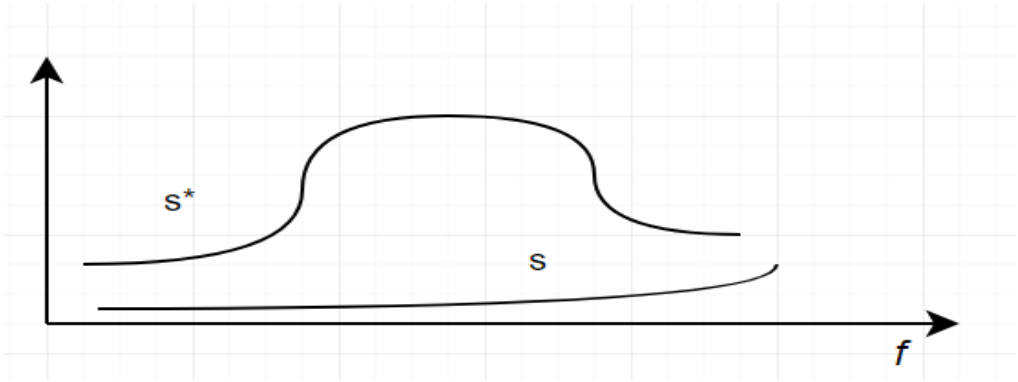


Рисунок 2.6.1 – Щільність розподілу значень

Даний критерій регулює баланс між інтенсифікацією та диверсифікацією пошуку. Наприклад, при ILS без пам’яті для інтенсифікації пошуку застосовний критерій, що приймає новий розв’язок лише коли він призводить до покращення значення цільової функції. Для інтенсифікації пошуку використовується критерій, що завжди приймає розв’язок.

Існують інші критерії, що ставлять на меті встановлення балансу між інтенсифікацією та диверсифікацією. Наприклад, розв’язки, що призводять до покращення значення цільової функції приймаються завжди, а всі інші – з деякою ймовірністю. Інший можливий критерій з застосуванням пам’яті – критерій з перезапуском. У випадку, коли певна кількість ітерацій ILS не призводить до покращення, то пошук починається заново з нового початкового розв’язку. Вся послідовність розв’язків в S^* конструюється заново. Псевдокод алгоритму наведено на рисунку 2.2, де змінними є:

s^0 – початковий розв’язок;

s^* – розв’язок, що генерується вбудованим алгоритмом і додається до послідовності розв’язків в S^*;

s' – розв’язок, що отримується з s^* після збурення;

s' – розв’язок, що генерується вбудованим алгоритмом з s';

history – змінні, що служать за пам’ять алгоритму.

Функціями є:

GenerateInitialSolution – генерує початковий розв’язок;

LocalSearch – вбудований алгоритм, що проектує значення множини S в S^*;

AcceptanceCriterion – визначає чи буде додано розв’язок до послідовності розв’язків в S^*

1: $s_0 = \text{GenerateInitialSolution}$
2: $s^* = \text{LocalSearch}(s_0)$
3: **repeat**
4: $s' = \text{Perturbation}(s^*, history)$
5: $s^{*\prime} = \text{LocalSearch}(s')$
6: $s^* = \text{AcceptanceCriterion}(s^*, s^{*\prime}, history)$
7: **until** termination condition met

Рисунок 2.6.2 – Узагальнений алгоритм ILS

2.7 Специфікація методу повторюваного локального пошуку для SLPTW

Блок-схема алгоритму повторюваного локального пошуку наведена в частині графічного матеріалу.

Для розв’язування задачі SLPTW застосовується спеціальний вбудований алгоритм і алгоритм збурення розв’язку. Новий розв’язок приймається, якщо значення цільової функції краще за попереднє.

Крім змінних, що наведені у математичні постановці задачі, вводяться додаткові змінні:

$$a_i = s_j + T_j + c_{ji}, i, j = \overline{1,n} \quad (2.12)$$

$$\text{Wait}_i = \max\{0,\ O_i - a_i\}, i = \overline{1,n} \quad (2.13)$$

$$MaxShift_i = \min\{MaxShift_{i+1} + Wait_{i+1}, c_j - s_j, i = \overline{1,n}\} \quad (2.14)$$

$$Shift_j = c_{ij} + c_{jk} - c_{ij} + Wait_j + T_j, i, j, k = \overline{1,n} \quad (2.15)$$

$$Ratio_i = \frac{R_i^2}{Shift_j}, i = \overline{1,n} \quad (2.16)$$

$$s_i = a_i + Wait_i \quad (2.17)$$

Змінна a_i позначає час прибуття до пункту (вершини) $i \in \overline{1,n}$. Її значення визначається за формулою (2.12), передбачається, що пункт j, $j = \overline{1,n}$

розташований безпосередньо перед пунктом. Оскільки прибуття в пункт може відбутись перед відкриттям, вводиться змінна $Wait_i$, що позначає час очікування у пункті. Значення, що приймає змінна визначається за формулою (2.13).

Змінна $MaxShift_i$ позначає те, наскільки візит до пункту може буте відкладений, щоб такий зсув не привів до того, що час затрачений на весь маршрут не перевищував T_{max}. Згідно формули (2.14), значення змінної дорівнює часу, тому наскільки візит до пункту, що розташований за даним у маршруті, може бути відкладений і часу, що треба очікувати перед початком наступного маршруту. В формулі також враховується те, що початок візиту завжди повинен знаходитись у рамках часового вікна, тому величина $MaxShift_i$ не повинна перевищувати $C_i - s_i$. Опис змінної $Ratio_i$, що визначається за формулою (2.16), буде наведено нижче.

2.7.1 Побудова початкового розв'язку

Побудова початкового розв'язку починається з того, що всі маршрути містять тільки початкову і кінцеві вершини. Потім крок за кроком, додаються по одній новій вершині. Для вибору нової вершини j, що буде потенційно додана у маршрут, обчислюється мінімальне значення змінної $Shift_j$. Перевіряється кожна позиція, де може бути додана нова вершина у всіх маршрутах. По формулі (2.15) обчислюється час, що буде затрачено, якщо між вершиною i та k буде додано нову вершину j. Щоб забезпечити допустимість отримуваних розв'язків, положення для яких значення $Shift_j$ більше за $MaxShift_i$ вважаються недопустимими. Таким чином, серед всіх допустимих положень обирається одне, де значення $Shift_j$ найменше. Після цього, за формулою (2.16) розраховується $Ratio_i$. В результаті обирається вершина у якої $Ratio_i$ найменше, вона буде додана до маршруту у положення, з найменшим зсувом маршруту у часі. Такі вставки у маршрут продовжуються поки можна знайти хоч одну вершину для якої є позиція, така що $Shift_j \leq MaxShift_j$.

2.7.2 Алгоритм вставки нової вершини

КРОК 1. Для кожної вершини j, що не належить жодному маршруту:

1.1 визначити позицію з найменшим $Shift_j$;

1.2 розрахувати $Ratio_j$.

КРОК 2. Вставити вершину з найбільшим $Ratio_j$ в маршрут.

КРОК 3. Розрахувати $Wait_j$, a_j, s_j для вставленої вершини.

КРОК 4. Для кожної вершини i , що слідує за вставленою:

4.1 Поки $Shift_i > 0$

4.2 оновити величини $Wait_i$, a_i, $MaxShift_i$, s_i.

КРОК 5. Оновити $MaxShift_j$ вставленої вершини.

КРОК 6. Для всіх вершин i перед вставленою:

6.1 оновити $MaxShift_i$

Для оновлення значень після вставки використовуються формули (2.15, 2.18-2.22). В даних формулах застосовується така нотація: нова вершина j вставляється між вершинами i та k. Для розрахування того, наскільки візит в вершину j збільшить загальну тривалість маршруту використовується формула (2.15). Отримується величина $Shift_j$, за цим значенням новий час очікування $Wait_{k*}$ для вершини (пункту) k розраховується за формулою (2.18) і новий час прибуття a_{k*} розраховується за формулою (2.19). $Shift_k$, що розраховується за формулою (2.20) використовується для розрахування нових значень часу візиту і $MaxShift_{k*}$ в вершині k за формулами (2.21) і (2.22) відповідно. Формули (2.18-2.22) також використовуються для оновлення змінних для вершин після k.

$$Wait_{k*} = \max(0, Wait_k - Shift_j) \quad (2.18)$$

$$a_{k*} = a_k + Shift_j \quad (2.19)$$

$$s_{k*} = s_k + Shift_k \quad (2.20)$$

$$MaxShift_{k*} = MaxShift_k - Shift_k \quad (2.21)$$

Описаний алгоритм застосовується не лише для побудови початкового

розв’язку, він також використовується на кожній ітерації повторюваного локального пошуку для конструювання нового розв’язку.

2.7.3 Збурення розв’язку

Коли алгоритм вставки вершини завершує свою роботу, то ймовірно, що отриманий розв’язок є локальним оптимумом, тому необхідно провести модифікацію отриманого розв’язку, щоб мати можливість покращити знайдений розв’язок. Модифікація розв’язку відбувається шляхом вилучення R ∈ N вершин на позиції $S \in N$ у кожному маршруті. При кожній ітерації повторюваного локального пошуку позиція S зсувається вперед. Коли S досягає кінця найкоротшого маршруту, то S поміщується у початок всіх маршрутів.

Після видалення вершини, всі візити для наступних вершин зсуваються у часі до вершини на позиції S − 1. Для вершини після видалених треба оновити Wait, a, s, MaxShift. Для вершин, що передують видалим необхідно оновити MaxShift.

Алгоритм збурення розв’язку

КРОК 1. Для кожного маршруту:

1.1 видалити R вершин починаючи з позиції S;

1.2 розрахувати Shift.

КРОК 2. Для кожної вершини i, що слідує за видаленою(ими):

2.1 поки $Shift_i > 0$

2.2 оновити величини $Wait_i$, a_i, $MaxShift_i$, s_i.

КРОК 3. Для всіх вершин і перед видаленими:

3.1 оновити $MaxShift_i$

2.8 Модифікація алгоритму ILS

Модифікація оригінального алгоритму [35] полягає у тому, що змінюється критерій прийняття нового розв’язку. Замість того, щоб приймати всі розв’язки, приймаються тільки не гірші.

Збереження попереднього розв’язку достатньо ресурсозатратне, тому що необхідно зберігати значення всіх змінних, що обчислені для вершин у маршрутах. Замість цього зберігаються вершини, що видалені на кроці збурення і ті, що додані на кроці вставки. Якщо розв’язок гірший, то тоді необхідно реконструювати старі маршрути. Спочатку видаляються вершини, що були вставлені при роботі ЛП (крок вставки). Перераховуються змінні, додаються видалені вершини і знову перераховуються значення змінних.

Блок-схема модифікованого алгоритму повторюваного локального пошуку наведена в частині графічного матеріалу.

2.9 Паралельне обчислення околу

На кроці вставки нової вершини, у розв’язок послідовно для кожної вершини, що не входить до маршрутів, визначається найкраща позиція для вставки. Цю процедуру можна виконувати паралельно на загальній пам’яті, адже при ній не може виникнути умови змагання (race condition). Кожній вершині можна виділити окремий процес. Йому треба модифікувати тільки «свої» ділянки пам’яті. Якщо він звертається до інших комірок пам’яті, то тільки для зчитування.

2.10 Приклад розв’язання задачі

Наведемо приклад побудови початкового розв’язку для конкретної задачі. Необхідно побудувати 2 маршрути з $T_{max} = 170$. Дано 5 пунктів. З них 1 являє собою пункт в якому закінчуються і починаються всі маршрути. Даний пункт матиме номер 0, крім того додається дві копії даного пункту. Вони служитимуть кінцевим пунктом для маршрутів і матимуть номери 5 та 6. Вхідні дані наведено у таблиці 2.10.1.

Таблиця 2.10.1 – Вхідні дані

Номер вершини i	Час на відвідування верши ни T_i	Корисність від відвудування R_i	Початок часового вікна, O_i	Кінець часового вікна C_i	Координата X	Координата Y
0	0	0	1	180	0	0
1	5	30	7	165	65	34
2	12	30	47	120	1	-46
3	10	10	2	150	-23	45
4	4	1	1	170	62	22
5	0	0	0	170	0	0
6	0	0	1	170	0	0

Час на переміщення між пунктами не задано, в даній задачі він прирівнюється до евклідових відстаней між вершинами. Розраховані значення наведено в таблиці 2.10.2.

Таблиця 2.10.2 – Час на переміщення між вершинами

№ вершини	0	1	2	3	4	5
0	0	73.36	46.01	88.32	25.05	0
1	73.35	0	102.45	141.42	54.34	73.35
2	46.01	102.4	0	119.2	68.88	46.01
3	88.32	141.4	119.2	0	90.95	88.32
4	25.05	14.34	68.88	90.95	0	25.05
5	0	73.36	46.01	88.32	25.05	0
6	0	73.36	46.01	88.32	25.05	0

Конструювання маршрутів починається з того, що в маршрути додаються початкова і кінцева вершини. Для кожної вершини розраховується змінні $Wait_i$, a_i, $MaxShift_i$, s_i. На даному етапі розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 5$

Маршрут 2: $0 \rightarrow 6$

Значення змінних для вершини 0:

– $a=0$;

– $s=0$;

– Wait = 0;

– MaxShift = 0. Значення змінних для вершин 5 і 6:

– a=0;

– s=0;

– Wait = 0;

– MaxShift = 170.

Крок 1

Необхідно визначити вершину, яка буде вставлена в один з маршрутів наступною. Вона обирається з тих вершин, що поки не належать жодному з маршрутів: 1, 2, 3, 4. Вибір робиться на основі величини Shift, що розраховується по формулам (2.14, 2.15). Результати розрахунків наведено у таблиці 2.3. Номер позиції – потенційне місце вставки нової вершини. Наприклад, якщо в таблиці № позиції дорівнює 1, то це означає, що вставка відбудеться після першої по порядку вершини у маршруті. Перша вершина у маршруті має номер 0. Пусті місці у таблиці означають, що дану вершину неможливо вставити в вказану позицію маршруту. Така ситуація може виникнути з двох причин: прибуття у пункт відбувається після закриття; вставка неможлива, бо час необхідний для цього більше ніж MaxShift, іншими словами, зсув призведе до того, що розклад стане недопустимим. Розклад недопустимий, якщо пункти відвідуються не в рамках своїх часових вікон, чи загальна тривалість маршруту перевищує T_{max}.

Таблиця 2.10.3 – Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 1

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
1	1	1	151
1	2	1	151
2	1	1	125
2	2	1	105
3	1	1	-
3	2	1	-
4	1	1	34
4	2	1	51

Після даних обчислень для кожної вершини запам’ятовується найменше значення змінної Shift і відповідні значення позиції і маршруту вставки. Збережені дані відображено в таблиці 2.10.4. На наступному кроці серед даних вершин буде обрана одна, що має найбільше значення Ratio. Дані значення обчислюються згідно формули 2.16. Результати обчислень наведено в таблиці 2.10.4.

Таблиця 2.10.4 – Результати обчислень Ratio на кроці 1

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
1	1	1	151	5.96
2	1	1	105	8.57
3	-	-	-	-
4	1	1	54	0.018

З таблиці 2.10.4 видно, що найбільше значення Ratio вершини 2. Тому, вершина 2 буде вставлена у перший маршрут і стане другою вершиною по порядку в ньому. Після вставки для вершини 2 обчислюються значення змінних Wait, a, s за формулами (2.12, 2.13, 2.17). Далі за формулами (2.18-2.22) для вершини 5 оновлюються значення змінних Wait, a, s, MaxShift. Тепер можна проводити обчислення MaxShift для вершини 2.14.

В результаті розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 6$

Значення змінних для вершин 0 та 6 не змінюються. Значення змінних для вершини 2:

- a = 46;
- s=47;
- Wait = 1;
- MaxShift = 66. Значення змінних для вершини 5:
- a = 105;
- s =105;
- Wait = 0;

– MaxShift = 65.

Тепер вершину 2 не можна вставити повторно в маршрут. Оскільки на даному етапі було додано вершину у один з маршрутів, пошук продовжується. Коли на поточному кроці не додано жодної вершини, то пошук завершується.

Крок 2

Необхідно визначити вершину, яка буде вставлена в один з маршрутів наступною. Вибір робиться поміж вершин 1, 3, 4 на основі величини Shift. Результати розрахунків наведено у таблиці 2.10.5.

Таблиця 2.10.5 – Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 2

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
1	1	1	-
1	1	2	-
1	2	1	151
3	1	1	-
3	1	2	-
3	2	1	-
4	1	1	52
4	1	2	52
4	2	1	54

В таблиці 2.10.6 наведено розрахунки, на основі яких обиратиметься вершина для вставки.

Таблиця 2.10.6 – Результати обчислень Ratio на кроці 2

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
1	2	1	151	5.9
3	-	-	-	-
4	1	1	52	0.02

Згідно розрахунків у таблиці 2.10.6 видно, що найбільше значення Ratio вершини 1. Тому, вершина 1 буде вставлена у другий маршрут і стане другою вершиною по порядку в ньому. Значення змінних обчислюються за таким же принципом, як на кроці 1.

В результаті розв'язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 1 \rightarrow 6$

Значення змінних для вершин 0, 2, 5 не змінюються. Значення змінних для вершини 1:

- Wait = 0;
- MaxShift= 19;
- S = 73.

Значення змінних для вершини 6:

- Wait = 0;
- MaxShift = 19;
- S = 151;
- A = 151.

Тепер вершину 1 не можна вставити повторно в маршрут. На даному етапі було додано вершину у один з маршрутів, тому пошук продовжується.

Крок 3

Вершина 3 або 4 потенційно може бути вставлена на даному кроці у один з маршрутів, вибір родиться на основі розрахунків, що наведені в таблицях 2.7, 2.8.

Таблиця 2.10.7 – Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 3

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
3	1	1	-
3	1	2	-
3	2	1	-
3	2	2	-
4	1	1	52
4	1	2	52
4	2	1	10

Таблиця 2.10.8 – Результати обчислень Ratio на кроці 3

№ вершини, що вставляється	№ маршруту	№ позиції	Мінімальне значення Shift	Значення Ratio
3	-	-	-	-
4	2	1	10	0.1

Згідно розрахунків у таблиці 2.10.8 видно, що можливо вставити лише вершину 4. Вона буде вставлена у другий маршрут і стане другою вершиною по порядку в ньому. Значення змінних обчислюються за таким же принципом, як на кроці 1.

В результаті розв’язок має такий вигляд:

Маршрут 1: $0 \rightarrow 2 \rightarrow 5$

Маршрут 2: $0 \rightarrow 4 \rightarrow 1 \rightarrow 6$

Значення змінних для вершин 0, 2, 5 не змінюються. Значення змінних для вершини 4:

- Wait = 0;
- MaxShift = 9. Значення змінних для вершини 1:
- a=25;
- s=25;

– Wait = 0;

– MaxShift = 9. Значення змінних для вершини 6:

– a=83;

– s=83;

– Wait = 0;

– MaxShift = 9;

– a=161;

– s=161.

На даному етапі було додано вершину у один з маршрутів, тому пошук продовжується.

Крок 4

На даному кроці можливо вставити лише вершину 3. Розрахунки найкоротшого часу вставки наведено у таблиці 2.10.9.

З таблиці видно, що неможливо вставити вершину 3 у будь-який маршрут, бо маршрут стане недопустимим після цього. Жодної вершини не вставлено, тому алгоритм завершує свою роботу. Початковий розв’язок побудовано.

Таблиця 2.10.9 – Розрахунки для визначення найкоротшого часу вставки для кожної вершини на кроці 4

№ вершини, що вставляється	№ маршруту	№ позиції	Shift
3	1	1	-
3	1	2	-
3	2	1	-
3	2	2	-
3	2	3	-

2.11 Алгоритм імітаційного відпалу

Алгоритм імітації відпалу (Simulated Annealing, SA) серед розроблюваних метаевристик був одним з перших методів, що надавали стратегію уникнення локального оптимуму. Метод розроблений на основі алгоритму Метрополіса з

області статистичної механіки. Ідея SA була основана на процесі відпалу металу і скла, коли конфігурування низького рівня споживання енергії забезпечується при правильному графіку охолодження [8]. SA вперше був представлений як алгоритм пошуку для комбінаторних задач в [21, 39]. Збіжність методу доведено в [45].

Фундаментальна ідея уникнення потрапляння в локальні оптимуми полягає в тому, щоб дозволити рухатися до розв'язків, що мають гірші значення цільової функції ніж у поточного розв'язку s. На кожній ітерації формуються елементи околу s` ∈ N(s). Якщо для конкретного розв'язку s′ значення цільової функції краще, ніж для розв'язку s, то s′ стає новим поточним розв'язком. В протилежному випадку s′ приймається з імовірністю, що є функцією від початкової температури (параметр алгоритма) T_k і f(s′) − f(s). Зазвичай, дана імовірність розраховується за законом Больцмана-Ґібса:

$$p(s', s, T) = e^{-\frac{f(s') - f(s)}{T_k}} \qquad (2.23)$$

Динамічний процес SA породжує ланцюг Маркова, оскільки на кожній ітерації вибір нового розв'язку залежить лише від значення попереднього розв'язку. Це означає, що базова процедура SA не має пам'яті. Однак, використання пам'яті при дослідженні простору пошуку може бути застосовано для побудови ефективних методів, що конструюються на основі SA. Приклад застосування SA з пам'яттю наведено у [22].

В структурі алгоритму можна виділити три основні частини:

- побудова початкового розв'язку;
- встановлення початкової температури;
- зміна температури.

В псевдокоді алгоритму, який наведено на рисунку 2.3, цим крокам у відповідність ставляться такі процедури:

- GeneratelnitialSolution();

– SetInitialTemperature();

$s \leftarrow$ GenerateInitialSolution()
$k \leftarrow 0$
$T_k \leftarrow$ SetInitialTemperature()
while termination conditions not met **do**
  $s' \leftarrow$ PickNeighborAtRandom($\mathcal{N}(s)$)
  if ($f(s') < f(s)$) **then**
    $s \leftarrow s'$ {s' replaces s}
  else
    Accept s' as new solution with probability $\mathbf{p}(s' \mid T_k, s)$
  end if
  AdaptTemperature(T_k)
  $k \leftarrow k + 1$
end while
output: best solution found

Рисунок 2.11.1 – Псевдокод алгоритму імітації відпалу [22]

Алгоритм починається з створення початкового розв'язку, який може бути побудовано випадково або евристично. Початкова температура вибирається така, що ймовірність прийняття погіршуючих розв'язків була досить високою на початку роботи алгоритму. Температура T_k змінюється на кожній ітерації по температурі відповідно до графіку охолодження. Графік охолодження визначає значення T_k на кожній такій ітерації k. Вибір підходящого графіку охолодження має вирішальне значення для ефективної роботи алгоритму. Ймовірність прийняття погіршуючих розв'язків повинна зменшуватись при роботі алгоритму, чим досягається немонотонний характер зміни цільової функції.

Теоретично, необхідно обирати графік охолодження таким чином, щоб імовірність знаходження глобального оптимуму при $k \to \infty$ збігалась до одиниці. Така збіжність забезпечується, коли графік охолодження слідує логарифмічному закону. На жаль, графіки охолодження, що забезпечують збіжність до глобального оптимуму, часто не застосовні практично, бо вони занадто повільні. Тому використовуються швидші графіки охолодження. Один з найбільш популярних графіків охолодження – графік, що слідує геометричному закону (2.24).

$$T_k = \alpha T_{k-1}, \alpha \in (0,1) \qquad (2.24)$$

Графік охолодження може використовуватися для балансування між диверсифікацією та інтенсифікацією пошуку. Наприклад, на початку пошуку, T_k може бути константним значенням або лінійно зменшується, щоб дослідити простір пошуку; після T_k може слідувати, наприклад, геометричному закону, щоб забезпечити швидшу збіжність алгоритму до локального оптимуму в кінці пошуку. Більш ефективним варіантом є застосування немонотонного графіку охолодження [45, 55]. Такий графік характеризується чергуванням фаз охолодження та нагрівання, що забезпечує коливальний баланс між диверсифікацією та інтенсифікацією.

Графік охолодження та початкова температура повинні бути адаптовані до конкретної проблеми, що розглядається, оскільки вартість уникнення локальних оптимумів залежить від ландшафту простору пошуку. Простий спосіб емпіричного визначення початкової температури – спочатку випадковим чином згенерувати розв'язки, щоб приблизно оцінити середнє значення та дисперсію значень цільової функції. Заснована на даних розв'язках, початкова температура визначається таким чином, що прийняття погіршуючих розв'язків має високу вірогідність.

У [39] запропоновано обирати початкову температуру як максимальне значення різниці між значеннями цільових функцій будь-яких двох сусідніх розв'язків (2.25).

$$T_0 = \Delta f_{max}. \qquad (2.25)$$

Інша схема, що ґрунтується на більш точній оцінці розподілу запропонована у [7, 77]. Рекомендується обирати значення початкової температури згідно формули (2.26).

$$T_0 = K\sigma^{\prime} . \qquad (2.26)$$

Тут σ^2 – момент другого порядку розподілу значення цільової функції, коли температура наближається до ∞. σ_∞ оцінюється на основі випадкової генерації розв'язків.

Більш класичний та інтуїтивно зрозумілий метод описаний у [39]. Він полягає у обчисленні такої температури, що коефіцієнт приймання

погіршуючого розв'язку приблизно дорівнює заданому значенню χ_0. Спочатку обирається висока початкова температура. Тоді робиться декілька ітерацій з використанням цієї температури.

Імовірність прийнятих переходів порівнюється з χ_0. Якщо вона менша за χ_0, температура збільшується в 2 рази. В протилежному випадку температура зменшується. Процедура триває, доки імовірність переходу не приближується до χ_0.

Інший підхід запропоновано в [16]. Задається додатня початкова температура і на її основі робиться прогін SA для одного чи більше початкових розв'язків $s_n \in S, n \in N^+$. Потім відбираються погіршуючі розв'язки $s' \in S', t \in N^+$. В результаті отримується множина пар розв'язків E, опис якої подається формулою (2.27):

$$E = \{f(s), f(s'), s, s' \in S, f(s) > f(s'), f(s) \to f(s')\} \quad (2.27)$$

Проводиться розрахунок оцінки температури за формулою (2.27).

Процедура закінчується, якщо виконується умова (2.29), де

χ_0 – очікувана ймовірність прийняття погіршуючих розв'язків, ϵ – точність.

В протилежному випадку нова оцінка температури обчислюється за формулою (2.30), де p – константа і проводяться розрахунки для за формулою (2.28).

$$X(T_n) - X_0 \le e \quad (2.28)$$

$$\left|X(T_n) - X_0\right| < e \quad (2.29)$$

$$T_{n+1} = T * \frac{X(T_n)}{\ln(X_0)} \quad (2.30)$$

2.12 Специфіка алгоритму імітаційного відпалу задачі SLPTW

Початкова температура обирається на основі методу з [16], що описаний в кінці попереднього розділу. Процедура визначення початкової температури запускається з такими початковими параметрами:

- $\chi_0 = 0.8$;
- $P = 0.5$;
- $\epsilon = 0.1$

Графік охолодження слідує логарифмічному закону з параметром $\alpha = 0.9$.

Процедури побудови і збурення розв'язку, такі ж як для алгоритму повторюваного локального пошуку. Змінюється критерій прийняття нового розв'язку. Новий розв'язок приймається з ймовірністю, що виначається за формулою (2.23).

2.13 Результати досліджень ефективності розв'язку

Метою досліджень є оцінка ефективності розроблених алгоритмів та пропонованих модифікацій для розв'язування задачі TOPTW; порівняння алгоритмів, виявлення їх недоліків і переваг.

Використовувалось таке апаратне забезпечення: процесор Intel Core i5-3317U, 1.7 GHz з оперативною пам'яттю 4 GB.

Для дослідження ефективності запропонованого алгоритму методу повторюваного локального пошуку проводились експерименти на 27 наборах даних, взятих із [63]. Будувалось три маршрути з часовим обмеженням на роботу алгоритмів у 40 секунд. З такими вхідними даними було запущено модифікований і немодифікований алгоритми локального пошуку, результати наведені у частині графічного матеріалу.

Далі проводилось дослідження ефективності модифікованого алгоритму повторюваного локального пошуку та імітаційного відпалу. Використовувались набори даних сконструйовані на основі наборів для VRP [63]. Результати експериментів наведено в таблиці 2.1. Зафарбовані комірки показують, що алгоритм ILS завершив свою роботу раніше виділеного часу.

Таблиця 2.13.1 – Результати виконання алгоритмів ILS та SA

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t=10с									
c101	100	5.20	570	3.39	10.65	570	3.39	590	0.00
c102	100	5.66	640	1.54	10.30	640	1.54	650	0.00
c103	100	10.71	680	4.23	10.42	650	8.45	710	4.23
c104	100	9.10	700	4.11	10.57	730	0.00	730	-4.11
c105	100	7.93	640	0.00	10.59	640	0.00	640	0.00
c106	100	7.15	620	0.00	10.51	620	0.00	620	0.00
c107	100	10.52	660	0.00	10.52	660	0.00	660	0.00
c108	100	10.01	670	1.47	10.51	670	1.47	680	0.00
c109	100	10.63	690	2.82	10.52	690	2.82	710	0.00
pr01	48	3.83	450	7.02	10.11	442	8.68	484	1.65
pr02	96	11.48	640	6.16	10.52	640	6.16	682	0.00
pr03	144	10.96	643	10.45	10.69	642	10.58	718	0.14
pr04	192	13.20	695	20.21	12.19	708	18.71	871	-1.49
pr05	240	12.91	859	16.03	12.80	859	16.03	1023	0.00
pr06	288	14.16	852	13.06	14.64	852	13.06	980	0.00
pr07	72	10.88	523	6.61	10.60	517	7.68	560	1.07
pr08	144	12.23	644	20.49	11.95	644	20.49	810	0.00
pr09	216	13.20	646	22.82	13.47	646	22.82	837	0.00

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t=10с									
r101	100	6.34	340	2.58	10.96	340	2.58	349	0.00
r102	100	12.15	494	2.76	10.40	494	2.76	508	0.00
r103	100	9.26	513	0.77	11.03	513	0.77	517	0.00
r104	100	11.59	511	5.55	10.71	511	5.55	541	0.00
r105	100	11.65	472	-4.19	10.81	449	0.88	453	5.08
r106	100	11.43	522	0.57	11.06	522	0.57	525	0.00
r107	100	8.08	519	2.99	10.55	519	2.99	535	0.00
r108	100	12.92	526	5.73	11.21	526	5.73	558	0.00
r109	100	10.56	484	2.81	11.20	484	2.81	498	0.00
Кількість маршрутів m=3, обмеження часу виконання t =10с									
c101	100	8.04526	780	3.70	12.0661	780	3.70	810	0.00
c102	100	8.97144	870	3.33	11.0513	860	4.44	900	1.11
c103	100	11.0706	940	2.08	10.3933	940	2.08	960	0.00
c104	100	10.8448	940	5.05	10.712	950	4.04	990	-1.01
c105	100	10.6826	840	2.33	11.5304	810	5.81	860	3.49
c106	100	11.1701	840	2.33	10.3909	830	3.49	860	1.16
c107	100	10.519	850	4.49	10.6453	850	4.49	890	0.00
c108	100	10.4625	890	1.11	11.2609	890	1.11	900	0.00
c109	100	10.442	920	2.13	10.4874	900	4.26	940	2.13
pr01	48	3.99206	580	3.65	10.0942	580	3.65	602	0.00
pr02	96	11.9542	823	7.94	10.7276	823	7.94	894	0.00
pr03	144	12.9435	842	12.56	13.7032	842	12.56	963	0.00
pr04	192	17.7269	1066	11.17	20.2806	1066	11.17	1200	0.00
pr05	240	23.2442	1244	8.80	25.166	1244	8.80	1364	0.00
pr06	288	21.5724	1177	13.58	25.0679	1177	13.58	1362	0.00
pr07	72	10.304	698	3.72	10.5029	677	6.62	725	2.90
pr08	144	11.4885	947	11.82	15.3713	947	11.82	1074	0.00
pr09	216	19.0642	1019	13.28	20.5016	1019	13.28	1175	0.00
r101	100	2.38457	438	9.50	11.1031	438	9.50	484	0.00

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t=10c									
r102	100	10.5227	646	5.83	10.7848	646	5.83	686	0.00
r103	100	11.1175	693	5.07	11.085	693	5.07	730	0.00
r104	100	10.7632	702	7.75	10.4514	702	7.75	761	0.00
r105	100	8.88443	631	-1.94	10.546	631	-1.94	619	0.00
r106	100	11.1931	728	-1.82	11.1987	674	5.73	715	7.55
r107	100	12.1705	692	7.11	11.2187	692	7.11	745	0.00
r108	100	12.459	724	6.94	10.8471	724	6.94	778	0.00
r109	100	12.2556	634	9.69	10.3406	625	10.97	702	1.28
Кількість маршрутів m=2, обмеження часу виконання t =20c									
c101	100	4.98194	570	3.39	20.291	570	3.39	590	0.00
c102	100	5.47878	640	1.54	20.3962	630	3.08	650	1.54
c103	100	12.3183	680	4.23	20.3081	660	7.04	710	2.82
c104	100	8.34995	700	4.11	20.8211	700	4.11	730	0.00
c105	100	7.26274	640	0.00	20.4093	640	0.00	640	0.00
c106	100	9.20604	620	0.00	20.5463	620	0.00	620	0.00
c107	100	14.1443	660	0.00	20.4715	630	4.55	660	4.55
c108	100	11.8131	670	1.47	20.3337	670	1.47	680	0.00
c109	100	10.7699	690	2.82	20.8828	700	1.41	710	-1.41
pr01	48	4.99885	450	7.02	20.1142	438	9.50	484	2.48
pr02	96	22.59	655	3.96	21.5911	652	4.40	682	0.44
pr03	144	21.8718	649	9.61	20.6946	643	10.45	718	0.84
pr04	192	25.3429	722	17.11	21.3924	707	18.83	871	1.72
pr05	240	26.62	859	16.03	28.101	859	16.03	1023	0.00
pr06	288	31.2305	852	13.06	31.1972	852	13.06	980	0.00
pr07	72	15.4457	523	6.61	21.2709	523	6.61	560	0.00
pr08	144	21.3254	653	19.38	22.3905	654	19.26	810	-0.12
pr09	216	23.7174	686	18.04	26.9255	646	22.82	837	4.78
r101	100	3.43922	340	2.58	21.4911	340	2.58	349	0.00

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t =20с									
r102	100	13.656	494	2.76	20.5374	489	3.74	508	0.98
r103	100	15.0161	513	0.77	21.0423	513	0.77	517	0.00
r104	100	15.6933	511	5.55	21.5526	511	5.55	541	0.00
r105	100	14.5062	478	-5.52	20.9089	412	9.05	453	14.57
r106	100	11.4099	522	0.57	21.1925	522	0.57	525	0.00
r107	100	18.7168	519	2.99	20.517	527	1.50	535	-1.50
r108	100	19.297	526	5.73	20.3915	526	5.73	558	0.00
r109	100	16.601	484	2.81	20.469	427	14.26	498	11.45
Кількість маршрутів m=3, обмеження часу виконання t =20с									
c101	100	9.07432	780	3.70	20.5093	780	3.70	810	0.00
c102	100	11.2409	870	3.33	20.4902	880	2.22	900	-1.11
c103	100	18.6001	940	2.08	20.2895	920	4.17	960	2.08
c104	100	19.0346	940	5.05	20.8657	960	3.03	990	-2.02
c105	100	15.4703	840	2.33	21.2214	840	2.33	860	0.00
c106	100	16.7319	840	2.33	21.2151	830	3.49	860	1.16
c107	100	20.3872	860	3.37	20.8258	860	3.37	890	0.00
c108	100	21.2946	890	1.11	20.3831	890	1.11	900	0.00
c109	100	21.5562	920	2.13	20.7299	930	1.06	940	-1.06
pr01	48	5.79311	580	3.65	20.219	577	4.15	602	0.50
pr02	96	22.364	868	2.91	20.8547	843	5.70	894	2.80
pr03	144	22.6746	842	12.56	23.3112	842	12.56	963	0.00
pr04	192	22.0462	1066	11.17	22.7909	1078	10.17	1200	-1.00
pr05	240	41.1855	1244	8.80	36.5613	1244	8.80	1364	0.00
pr06	288	42.2009	1177	13.58	53.5806	1177	13.58	1362	0.00
pr07	72	20.3966	709	2.21	20.6886	667	8.00	725	5.79
pr08	144	23.3895	947	11.82	20.7222	948	11.73	1074	-0.09
pr09	216	29.3671	1019	13.28	34.7676	1019	13.28	1175	0.00
r101	100	5.66315	438	9.50	20.7263	464	4.13	484	-5.37

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t =20с									
r102	100	17.8757	646	5.83	21.0709	646	5.83	686	0.00
r103	100	21.0006	693	5.07	20.6237	704	3.56	730	-1.51
r104	100	20.5995	702	7.75	20.4109	707	7.10	761	-0.66
r105	100	12.4442	631	-1.94	20.7205	562	9.21	619	11.15
r106	100	21.4909	757	-5.87	21.2239	678	5.17	715	11.05
r107	100	17.1512	692	7.11	20.9595	711	4.56	745	-2.55
r108	100	20.8815	724	6.94	20.3157	731	6.04	778	-0.90
r109	100	21.0063	637	9.26	21.4826	623	11.25	702	1.99
Кількість маршрутів m=2, обмеження часу виконання t =40с									
c101	100	4.9708	570	3.39	41.25	570	3.39	590	0.00
c102	100	5.97947	640	1.54	40.62	630	3.08	650	1.54
c103	100	14.4166	680	4.23	40.88	710	0.00	710	-4.23
c104	100	9.95855	700	4.11	41.25	730	0.00	730	-4.11
c105	100	7.61226	640	0.00	40.74	640	0.00	640	0.00
c106	100	9.20955	620	0.00	40.36	620	0.00	620	0.00
c107	100	13.5904	660	0.00	40.56	660	0.00	660	0.00
c108	100	11.4933	670	1.47	40.34	670	1.47	680	0.00
c109	100	12.0195	690	2.82	40.81	690	2.82	710	0.00
pr01	48	4.84563	450	7.02	40.21	484	0.00	484	-7.02
pr02	96	41.0094	676	0.88	41.03	682	0.00	682	-0.88
pr03	144	41.6802	680	5.29	41.22	687	4.32	718	-0.97
pr04	192	41.6143	738	15.27	43.40	763	12.40	871	-2.87
pr05	240	42.7625	893	12.71	42.86	896	12.41	1023	-0.29
pr06	288	43.2829	860	12.24	43.86	861	12.14	980	-0.10
pr07	72	12.2225	523	6.61	40.54	560	0.00	560	-6.61
pr08	144	42.6629	695	14.20	41.32	716	11.60	810	-2.59
pr09	216	43.1534	697	16.73	42.28	655	21.74	837	5.02
r101	100	2.69258	340	2.58	40.39	340	2.58	349	0.00

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=2, обмеження часу виконання t =40с									
r102	100	11.5199	494	2.76	40.50	494	2.76	508	0.00
r103	100	12.8544	513	0.77	40.56	513	0.77	517	0.00
r104	100	15.3709	511	5.55	40.57	511	5.55	541	0.00
r105	100	15.2195	478	-5.52	41.40	453	0.00	453	5.52
r106	100	11.7993	522	0.57	40.65	522	0.57	525	0.00
r107	100	14.9367	519	2.99	40.86	519	2.99	535	0.00
r108	100	18.2739	526	5.73	40.65	526	5.73	558	0.00
r109	100	15.1	484	2.81	40.57	472	5.22	498	2.41
Кількість маршрутів m=3, обмеження часу виконання t = 40с									
c101	100	10.9885	780	3.70	43.14	780	3.70	810	0.00
c102	100	17.9966	870	3.33	41.64	890	1.11	900	-2.22
c103	100	29.6285	940	2.08	42.61	940	2.08	960	0.00
c104	100	24.3113	940	5.05	43.92	970	2.02	990	-3.03
c105	100	15.1197	840	2.33	41.45	840	2.33	860	0.00
c106	100	15.1272	840	2.33	41.36	820	4.65	860	2.33
c107	100	38.8412	890	0.00	40.65	880	1.12	890	1.12
c108	100	19.3506	890	1.11	42.49	890	1.11	900	0.00
c109	100	38.1773	940	0.00	44.65	940	0.00	940	0.00
pr01	48	5.2616	580	3.65	40.52	580	3.65	602	0.00
pr02	96	43.6293	878	1.79	45.91	849	5.03	894	3.24
pr03	144	41.3597	891	7.48	49.71	841	12.67	963	5.19
pr04	192	44.5807	1093	8.92	47.61	1078	10.17	1200	1.25
pr05	240	45.5188	1244	8.80	76.07	1244	8.80	1364	0.00
pr06	288	47.6005	1177	13.58	77.76	1177	13.58	1362	0.00
pr07	72	25.8317	721	0.55	41.85	676	6.76	725	6.21
pr08	144	41.7968	961	10.52	47.79	961	10.52	1074	0.00
pr09	216	45.9012	1095	6.81	43.20	1019	13.28	1175	6.47
r101	100	4.60142	438	9.50	42.60	438	9.50	484	0.00

Продовження таблиці 2.13.1

Назва файлу	Кількість вершин	Час роботи ILS, с	Значення ЦФ ILS	Похибка ILS,%	Час роботи SA , с	Знач. ЦФ SA	Похибка SA, %	Краще відоме знач. ЦФ	Різниця похибок, %
Кількість маршрутів m=3, обмеження часу виконання t = 40с									
r102	100	19.3809	646	5.83	41.96	646	5.83	686	0.00
r103	100	19.2088	693	5.07	41.96	693	5.07	730	0.00
r104	100	22.6753	702	7.75	43.24	702	7.75	761	0.00
r105	100	11.2694	631	-1.94	41.60	556	10.18	619	12.12
r106	100	24.0777	762	-6.57	40.99	683	4.48	715	11.05
r107	100	19.565	692	7.11	42.73	692	7.11	745	0.00
r108	100	23.4561	724	6.94	40.83	724	6.94	778	0.00
r109	100	21.6781	637	9.26	41.82	639	8.97	702	-0.28

Було проведено 162 прогони для кожного з алгоритмів. Було використано 27 наборів вхідних даних на яких конструювалось m ∈ {2,3} маршрути. При кожному запуску задавалось обмеження на час роботи алгоритму time ∈ {10, 20, 40}. Наступна ітерація алгоритму не починається, якщо алгоритм виконується довше, ніж time. В результаті, для 44 прогонів алгоритм ILS знаходив кращі розв’язки ніж SA, що складає 27% всіх прогонів. Для 29 прогонів, навпаки, SA давав кращі результати, що складає 18% від усіх прогонів. В таблиці 5.2 наведено результати порівняння ефективності роботи алгоритмів для різних використаних конфігурацій.

Якщо в комірці таблиці наведено назву алгоритму, то це означає, що даний алгоритм для даної конфігурації згенерував розв’язок з кращим значенням цільової функцій. Кольором виділені комірки, де алгоритм ILS завершив роботу раніше виділеного часу, іншими словами, знайшов локальний оптимум в множині розв’язків, що генерує локальний пошук.

Таблиця 2.13.2 – Порівняння ефективності роботи алгоритмів

	m=2				m=3		
	time=10s	time=20s	time=40s		time=10s	time=20s	time=40s
c101				c101			
c102		ILS	ILS	c102	ILS	SA	SA
c103	ILS	ILS	SA	c103		ILS	
c104	SA		SA	c104	SA	SA	SA
c105				c105	ILS		
c106				c106	ILS	ILS	ILS
c107		ILS		c107			ILS
c108				c108			
c109		SA		c109	ILS	SA	
pr01	ILS	ILS	SA	pr01		ILS	
pr02		ILS	SA	pr02		ILS	ILS
pr03	ILS	ILS	SA	pr03			ILS
pr04	SA	ILS	SA	pr04		SA	ILS
pr05			SA	pr05			
pr06			SA	pr06			
pr07	ILS		SA	pr07	ILS	ILS	ILS
pr08		SA	SA	pr08		SA	
pr09		ILS	ILS	pr09			ILS
r101				r101		SA	
r102		ILS		r102			
r103				r103		SA	
r104				r104		SA	
r105		ILS	ILS	r105		ILS	ILS
r106				r106		ILS	ILS
r107		SA		r107		SA	
r108				r108		SA	
r109		ILS	ILS	r109		ILS	SA

В результаті, загальний час роботи алгоритму ILS на всіх прогонах склав 2774 с, коли для SA цей показник складає 4200 с. Хоч обмеження на час виконання алгоритмів задавались однакові, алгоритм ILS в багатьох випадках закінчує свою роботу раніше. Наприклад, при побудові 2 маршрутів і обмеженні у 10с співвідношення часу роботи алгоритмів показано на рисунку 2.13.1.

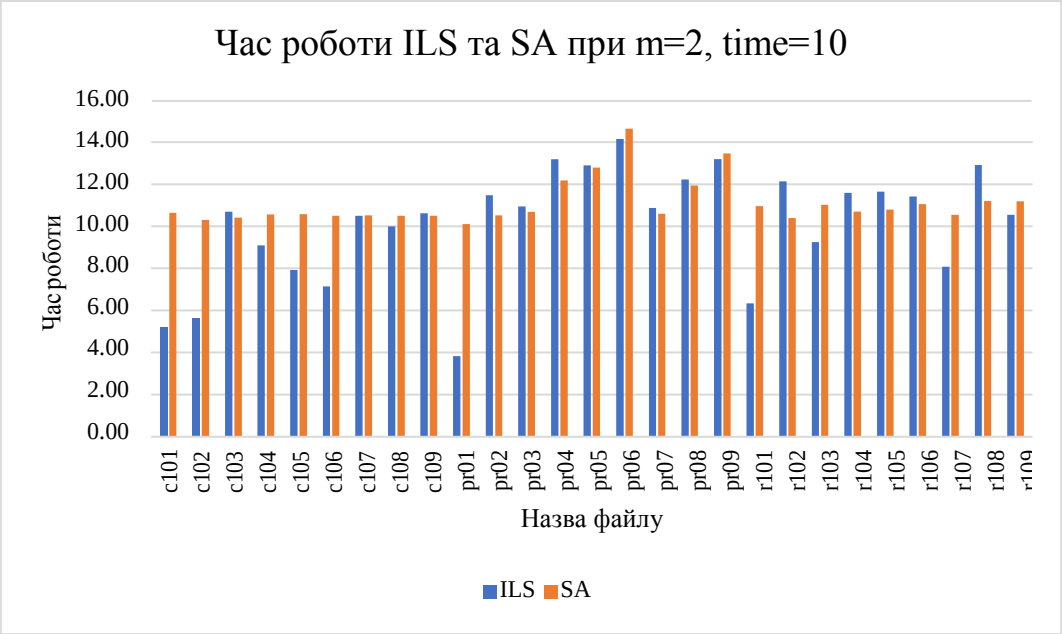


Рисунок 2.13.1 – Співвідношення часу роботи алгоритмів ILS та SA

В основу даних алгоритмів лягла процедура локального пошуку з паралельним обчисленням околу. Щоб оцінити зміну часу роботи порівняно з послідовним обчисленням околу було проведено експерименти на наборах даних pr01- pr09.

Випробування відбувались з такими параметрами:

– будується 2 маршрути;

– виконується 20 ітерацій алгоритму.

Для кожного набору даних проводилось по 5 прогонів з паралельним і послідовним обчисленням околу. Усереднені результати наведено в таблиці 2.13.2. На рисунку наведено навантаження на процесори при роботі алгоритму ILS з використанням послідовного ЛП, а на рисунку 5.3 наведено роботу ILS з паралельним ЛП.

Таблиця 2.13.2 – Усереднений час роботи алгоритму повторюваного ЛП

Кількість вершин	48	72	96	144	160	192	216	240	288
Час роботи послідовного ЛП, с	4.97	21.45	27.8	47.72	57.77	78.72	89.272	156.9	167.48
Час роботи паралельного ЛП, с	3.23	12.3	21.03	39.56	44.43	69.33	70.8	108.8	127.7
Покращення,%	35	42.6	24.34	17.09	23.09	11.92	20.69	30.61	23.75

В частині графічного матеріалу наведено порівняння роботи послідовної і паралельної реалізацій модифікованого ILS при зміні конфігурації технічного забезпечення на таку: 2.2GHz, Intel Core i7, 16GB RAM.

2.14 Аналіз отриманих результатів

Для проведених прогонів модифікованого повторюваного локального пошуку загальний час роботи складає 716 с, а для немодифікованого – 1101 с, що на 35% довше. Хоч алгоритм модифікованого повторюваного локального пошуку не приносить результатів у напрямку покращення якості знайденого розв’язку, проте час роботи скорочується. Це досягається тому, що модифікований алгоритм має високу ступінь інтенсифікації пошуку у просторі розв’язків ЛП, в той час як для оригінального алгоритму характер пошукув просторі розв’язків ЛП ближче до випадкового. З огляду на це, доречно використовувати таку модифікацію. Далі модифікований ILS порівнюється з SA. В основі алгоритмів SA та ILS лежить однакова процедура локального пошуку. Судячи з результатів в таблиці 2.13.2, час роботи для паралельної реалізації у середньому на 25% краще за послідовну версію. Крім того, з рисунків 2.13.1 видно, що балансування навантаження на процесори краще. Тож дану модифікацію доцільно запровадити для подальшої роботи алгоритмів. Крім того, дане покращення у швидкодії буде збільшуватись зі збільшенням кількості задіяних процессорів, але ідеальної завнтаженості

процессорів для алгоритмів досягнути не вдасться, адже велика доля обчислень у SA та ILS виконується послідовно. Варто зазначити, що коли кількість процесорів зростатиме більше ніж n/m, то пришвидшення не зростатиме, бо процесори не будуть використані при обчисленні околу (n – кількість вершин, m − кількість маршрутів). В результаті проведених експериментів, видно, що основною перевагою ILS перед алгоритмом SA є його швидкодія. У частині графічних матеріалів наведені графіки часу роботи алгоритмів SA та ILS. При однакових обмеженнях на час виконання для вісіх прогонів для алгоритму ILS булово витрачено на 35%менше часу. Слід зазначити, що з таблиці видно, що з часом SA має тенденцію знаходити кращі розв'язки. З огляду на специфіку задачі TOPTW можна виділити ряд переваг і недоліків реалізованих алгоритмів у порівнянні один з одним.

Перевагою модифікованого ILS є те, що через високу інтенсифікацію пошуку, час роботи алгоритму меншше. Він знаходить кращі розв'язки швидше за SA.

Недоліком ILS є той факт, що через присутність детермінованого критерію виходу і інтенсифікації імовірність потрапити у локальний оптимум в просторі розв'язків ЛП висока і подальші покращення розв'язку після цього неможливі. Перевагою SA є те, що завдяки диверсифікації пошуку, з часом алгоритм здатен знаходити кращі розв'язки ніж модифікований ILS.

Недоліки SA:

- необхідно підбирати вхідні параметри, для регулювання температурного розкладу;
- необхідно генерувати початкову температуру, що потребує додаткового часу роботи, адже містить процедуру генерування ряду розв'язків;
- на загал довший час роботи.

2.15 Висновки до розділу

Було подано математичну і змістовну постановки задачі slptw та методи її розв'язання. Для обраних методів було наведено узагальнену схему алгоритму і

описано специфіку реалізації даних методів для задачі. Був наведений приклад розв’язання задачі slptw алгоритмом, що лежить в основі обраних методів. Було запропоновано модифікацію, що може лежати в основі обох алгоритмів. Проведено аналіз розроблених модифікацій алгоритму повторюваного локального пошуку і амітаційного відпалу. За результатами експериментальних досліджень встановлено, що запропоновані модифікації покращують швидкодію алгоритмів. Було порівняно алгоритми і наведено їх недоліки і переваги. Як наслідок, була досягнута мета досліджень.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Вхідні данні

Вхідні дані – (input data) це дані, що подаються на оюробку програмним забезпеченням. Вхідні данні надаються користуваачем або потрапляють в систему через зчитування з файлу. Приклад файлу з вхідними даними для роботи алгоритмів наведено на рисунку 3.1.

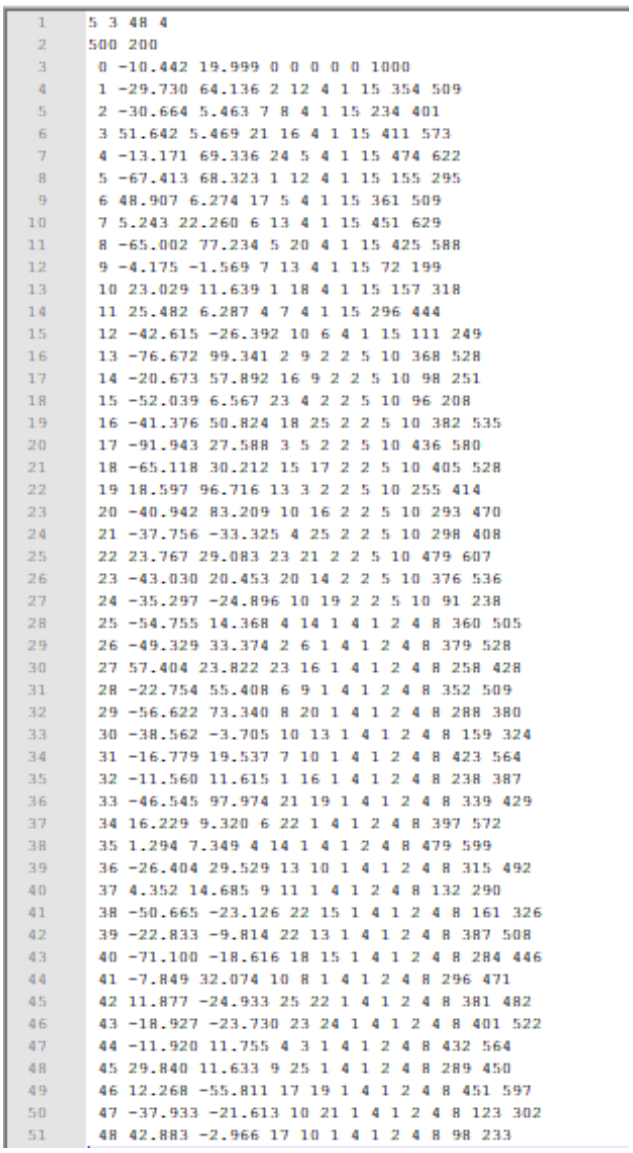
Для формування вхіідних даних задачі користувач вводить в систему наступні параметри:

- префікс, на його основі буде форммуватись назва файлу для зчитування;
- інформацію про розташування кожної локації (вершинни графу);
- час, на проходження маршруту (програмного зазначено 9 годин);
- про кожну локацію вводяться наступні дані:
- координати локації;
- необхідність відвідуваня локації (корисність);
- час, що витрачається на відвідування локації;
- час відкриття та закриття локації (час роботи локації);
- кількість локацій в турі (щоденному маршруті);
- ID локації.

Номери пунктів і їх кількість визначаються автоматично.

Для того, щоб запустити алгоритми необхідно надати такі вхідні дані:

- кількість локацій;
- обмеження на час роботи;
- назву файлу з вхідними даними.



```
5 3 48 4
500 200
 0 -10.442 19.999 0 0 0 0 0 1000
 1 -29.730 64.136 2 12 4 1 15 354 509
 2 -30.664 5.463 7 8 4 1 15 234 401
 3 51.642 5.469 21 16 4 1 15 411 573
 4 -13.171 69.336 24 5 4 1 15 474 622
 5 -67.413 68.323 1 12 4 1 15 155 295
 6 48.907 6.274 17 5 4 1 15 361 509
 7 5.243 22.260 6 13 4 1 15 451 629
 8 -65.002 77.234 5 20 4 1 15 425 588
 9 -4.175 -1.569 7 13 4 1 15 72 199
 10 23.029 11.639 1 18 4 1 15 157 318
 11 25.482 6.287 4 7 4 1 15 296 444
 12 -42.615 -26.392 10 6 4 1 15 111 249
 13 -76.672 99.341 2 9 2 2 5 10 368 528
 14 -20.673 57.892 16 9 2 2 5 10 98 251
 15 -52.039 6.567 23 4 2 2 5 10 96 208
 16 -41.376 50.824 18 25 2 2 5 10 382 535
 17 -91.943 27.588 3 5 2 2 5 10 436 580
 18 -65.118 30.212 15 17 2 2 5 10 405 528
 19 18.597 96.716 13 3 2 2 5 10 255 414
 20 -40.942 83.209 10 16 2 2 5 10 293 470
 21 -37.756 -33.325 4 25 2 2 5 10 298 408
 22 23.767 29.083 23 21 2 2 5 10 479 607
 23 -43.030 20.453 20 14 2 2 5 10 376 536
 24 -35.297 -24.896 10 19 2 2 5 10 91 238
 25 -54.755 14.368 4 14 1 4 1 2 4 8 360 505
 26 -49.329 33.374 2 6 1 4 1 2 4 8 379 528
 27 57.404 23.822 23 16 1 4 1 2 4 8 258 428
 28 -22.754 55.408 6 9 1 4 1 2 4 8 352 509
 29 -56.622 73.340 8 20 1 4 1 2 4 8 288 380
 30 -38.562 -3.705 10 13 1 4 1 2 4 8 159 324
 31 -16.779 19.537 7 10 1 4 1 2 4 8 423 564
 32 -11.560 11.615 1 16 1 4 1 2 4 8 238 387
 33 -46.545 97.974 21 19 1 4 1 2 4 8 339 429
 34 16.229 9.320 6 22 1 4 1 2 4 8 397 572
 35 1.294 7.349 4 14 1 4 1 2 4 8 479 599
 36 -26.404 29.529 13 10 1 4 1 2 4 8 315 492
 37 4.352 14.685 9 11 1 4 1 2 4 8 132 290
 38 -50.665 -23.126 22 15 1 4 1 2 4 8 161 326
 39 -22.833 -9.814 22 13 1 4 1 2 4 8 387 508
 40 -71.100 -18.616 18 15 1 4 1 2 4 8 284 446
 41 -7.849 32.074 10 8 1 4 1 2 4 8 296 471
 42 11.877 -24.933 25 22 1 4 1 2 4 8 381 482
 43 -18.927 -23.730 23 24 1 4 1 2 4 8 401 522
 44 -11.920 11.755 4 3 1 4 1 2 4 8 432 564
 45 29.840 11.633 9 25 1 4 1 2 4 8 289 450
 46 12.268 -55.811 17 19 1 4 1 2 4 8 451 597
 47 -37.933 -21.613 10 21 1 4 1 2 4 8 123 302
 48 42.883 -2.966 17 10 1 4 1 2 4 8 98 233
```

Рисунок 3.1.1 – Приклад файлу вхідних даних

3.2 Вихідні данні

При роботі з ПЗ формуються такі вихідні дані:

– текстові файли з даними для роботи алгоритмів;

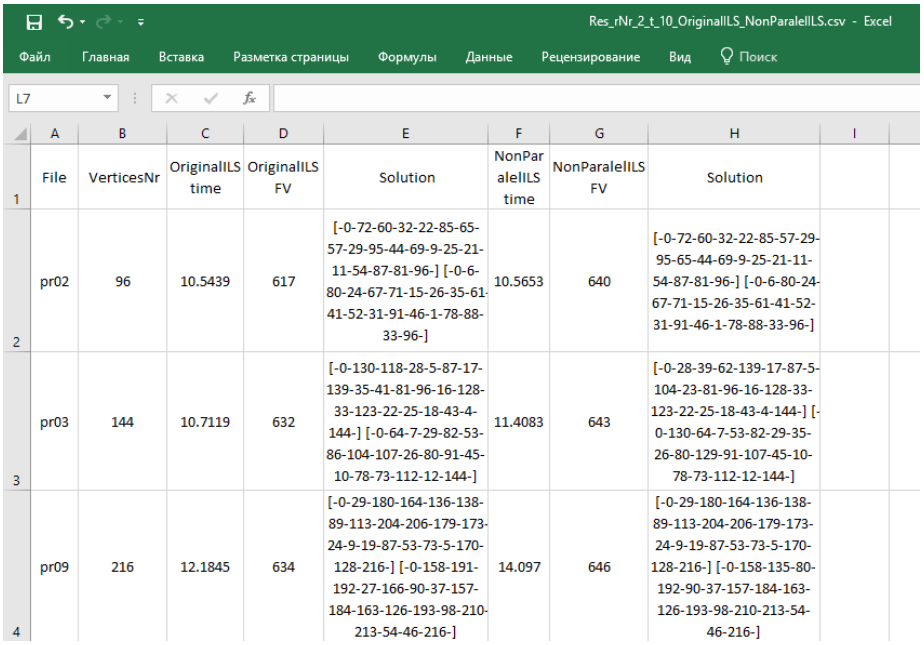
– Excel файли з результатами роботи алгоритмів.

Опис файлів з даними для роботи алгоритмів наведено у [53].

Результати роботи алгоритмів записуються в файл Excel. На рисунку 3.2.1 представлений приклад файлу вихідних даних, що сформовано ПЗ після роботи алгоритмів послідовного модифікованого повторюваного локального пошуку і немодифікованого повторюваного локального пошуку. В першій колонці наведено назву файлу вхідних даних на яких запусались алгоритми. В другій

колонці наведено кількість вершин. В наступних колонках для кожного алгоритму записано час роботи, обчислене значення цільової функції і сформовані маршрути.

Назва Excel файлу вихідних даних формується з таких даних: кількість маршрутів, обмеження на час роботи, назви алгоритмів.



	A	B	C	D	E	F	G	H	I
1	File	VerticesNr	OriginalILS time	OriginalILS FV	Solution	NonParalelILS time	NonParalelILS FV	Solution	
2	pr02	96	10.5439	617	[-0-72-60-32-22-85-65-57-29-95-44-69-9-25-21-11-54-87-81-96-] [-0-6-80-24-67-71-15-26-35-61-41-52-31-91-46-1-78-88-33-96-]	10.5653	640	[-0-72-60-32-22-85-57-29-95-65-44-69-9-25-21-11-54-87-81-96-] [-0-6-80-24-67-71-15-26-35-61-41-52-31-91-46-1-78-88-33-96-]	
3	pr03	144	10.7119	632	[-0-130-118-28-5-87-17-139-35-41-81-96-16-128-33-123-22-25-18-43-4-144-] [-0-64-7-29-82-53-86-104-107-26-80-91-45-10-78-73-112-12-144-]	11.4083	643	[-0-28-39-62-139-17-87-5-104-23-81-96-16-128-33-123-22-25-18-43-4-144-] [-0-130-64-7-53-82-29-35-26-80-129-91-107-45-10-78-73-112-12-144-]	
4	pr09	216	12.1845	634	[-0-29-180-164-136-138-89-113-204-206-179-173-24-9-19-87-53-73-5-170-128-216-] [-0-158-191-192-27-166-90-37-157-184-163-126-193-98-210-213-54-46-216-]	14.097	646	[-0-29-180-164-136-138-89-113-204-206-179-173-24-9-19-87-53-73-5-170-128-216-] [-0-158-135-80-192-90-37-157-184-163-126-193-98-210-213-54-46-216-]	

Рисунок 3.2.1 – Файл результатів роботи алгоритмів

3.3 Засоби розробки

Для написання програмного забезпечення використовувалось середовище розробки, що надається Visual Studio Community 2017.

Python – інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі вбудовані структури даних у поєднанні з динамічним набором тексту та динамічним прив’язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання в якості мови сценаріїв або склеювання для з’єднання існуючих компонентів разом. Простий, легкий у вивченні синтаксис Python підкреслює читабельність і, отже, зменшує витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python та велика стандартна бібліотека

доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно розповсюджуватися.

Часто програмісти закохуються в Python через підвищену продуктивність, яку він забезпечує. Оскільки не існує кроку компіляції, цикл редагування-тестування-налагодження неймовірно швидкий. Налагодження програм Python дуже просто: помилка або неправильний ввід ніколи не спричинить помилку сегментації. Натомість, коли інтерпретатор виявляє помилку, виникає виняток. Коли програма не вловлює виняток, інтерпретатор друкує трасування стека. Налагоджувач рівня джерела дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки зупинки, переходити через код за рядком за один раз тощо. Налагоджувач написаний на самому Python, що свідчить про інтроспективну силу Python. З іншого боку, найчастіше найшвидшим способом налагодження програми є додавання кількох операторів друку до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

Python – це зрозуміла і потужна об’єктно-орієнтована мова програмування, порівнянна з Perl, Ruby, Scheme або Java.

Деякі помітні особливості Python:

Використовує елегантний синтаксис, що полегшує читання програм, які ви пишете.

Це проста у використанні мова, яка спрощує роботу вашої програми. Це робить Python ідеальним для розробки прототипів та інших спеціальних завдань програмування без шкоди для ремонтопридатності.

Поставляється з великою стандартною бібліотекою, яка підтримує багато загальних завдань програмування, таких як підключення до веб-серверів, пошук тексту за допомогою регулярних виразів, читання та зміна файлів.

Інтерактивний режим Python дозволяє легко тестувати короткі фрагменти коду. Існує також комплексне середовище розробки під назвою IDLE.

Легко розширюється додаванням нових модулів, реалізованих компільованою мовою, такою як C або C ++.

Також може бути вбудований у програму для забезпечення програмованого інтерфейсу.

Працює де завгодно, включаючи Mac OS X, Windows, Linux та Unix, неофіційні збірки також доступні для Android та iOS.

Це безкоштовне програмне забезпечення у двох сенсах. Не потрібно нічого завантажувати чи використовувати Python або включати його у свою програму. Python також можна вільно модифікувати та розповсюджувати, оскільки, хоча мова захищена авторським правом, вона доступна за ліцензією з відкритим кодом.

Деякі особливості мови програмування Python:

Доступні різноманітні основні типи даних: числа (з плаваючою точкою, складні та довгі цілі числа необмеженої довжини), рядки (як ASCII, так і Unicode), списки та словники.

Python підтримує об’єктно-орієнтоване програмування з класами та кількома спадкоємствами.

Код можна згрупувати в модулі та пакети.

Мова підтримує створення та ловлення винятків, що призводить до більш чистого оброблення помилок.

Типи даних набираються сильно та динамічно. Змішування несумісних типів (наприклад, спроба додати рядок і число) призводить до виникнення винятку, тому помилки виявляються раніше.

Python містить розширені функції програмування, такі як генератори та розуміння списків.

Автоматичне управління пам’яттю Python позбавляє вас від необхідності розподіляти та звільняти пам’ять у коді вручну.

Дивіться колекцію коротких програм SimplePrograms, яка поступово збільшується в довжині, що демонструє синтаксис та читабельність Python.

Eclipse – це безкоштовна платформа для розробки на основі Java, відома своїми плагінами, які дозволяють розробникам розробляти та тестувати код, написаний іншими мовами програмування. Eclipse випускається на умовах публічної ліцензії Eclipse.

Eclipse почав свою діяльність у 2001 році, коли IBM подарувала три мільйони рядків коду зі своїх інструментів Java для розробки інтегрованого середовища розробки з відкритим кодом (IDE). Спочатку IDE контролювався консорціумом постачальників програмного забезпечення, що прагнув створити та сприяти створенню нової спільноти, яка доповнювала б спільноту з відкритим кодом Apache. Подейкують, що назва платформи виникла з другорядної мети, яка полягала в затьмаренні популярної IDE від Microsoft, Visual Studio.Сьогодні Eclipse управляється Eclipse Foundation, неприбутковою корпорацією, стратегічними членами якої є CA Technologies, IBM, Oracle та SAP. Фонд, який був створений у 2004 році, підтримує проекти Eclipse з чітко визначеним процесом розробки, що оцінює якість, стабільність інтерфейсу прикладного програмування (API) та послідовні графіки випусків. Фонд надає послуги з управління інфраструктурою та інтелектуальною власністю спільноті Eclipse та допомагає членам спільноти продавати та просувати комерційні програмні продукти, що базуються на Eclipse.

Spyder – це потужне наукове середовище, написане на Python, для Python, і розроблене науковцями, інженерами та аналітиками даних. Він має унікальну комбінацію розширених функцій редагування, аналізу, налагодження та профілювання всебічного інструменту розробки з дослідженням даних, інтерактивним виконанням, глибоким оглядом та прекрасними можливостями візуалізації наукового пакету.

Редактор

Ефективно працюйте в багатомовному редакторі з браузером функцій / класів, засобами аналізу коду, автоматичним заповненням коду, горизонтальним / вертикальним розбиттям та переходом до визначення.

Консоль IPython

Використовуйте потужність будь-якої кількості консолей IPython, скільки завгодно, завдяки гнучкості повного інтерфейсу графічного інтерфейсу; запустити код за рядком, коміркою або файлом; і надавати графіки прямо в рядку.

Провідник змінних

Взаємодійте та змінюйте змінні на льоту: складайте гістограму чи часові ряди, редагуйте датафрейм або масив Numpy, сортуйте колекцію, копайте вкладені об’єкти тощо! Графіки Переглядайте, копіюйте та зберігайте малюнки та зображення, створені під час виконання коду.

3.4 Архітектура програмного забезпечення

Структура проекту ПЗ наведена на рисунку 3.4.1 Основними компонентами проекту є:

- компонент TOPTW, що являє собою динамічну бібліотеку;
- компонент TOPTWLib, що являє собою статичну бібліотеку яка містить алгоритмічек забезпечення для ПЗ;
- компонент UnitTests для тестування алгоритмічного забезпечення;
- компонент WindowsFormApp, що відповідає за графічний інтерфейс користувача.

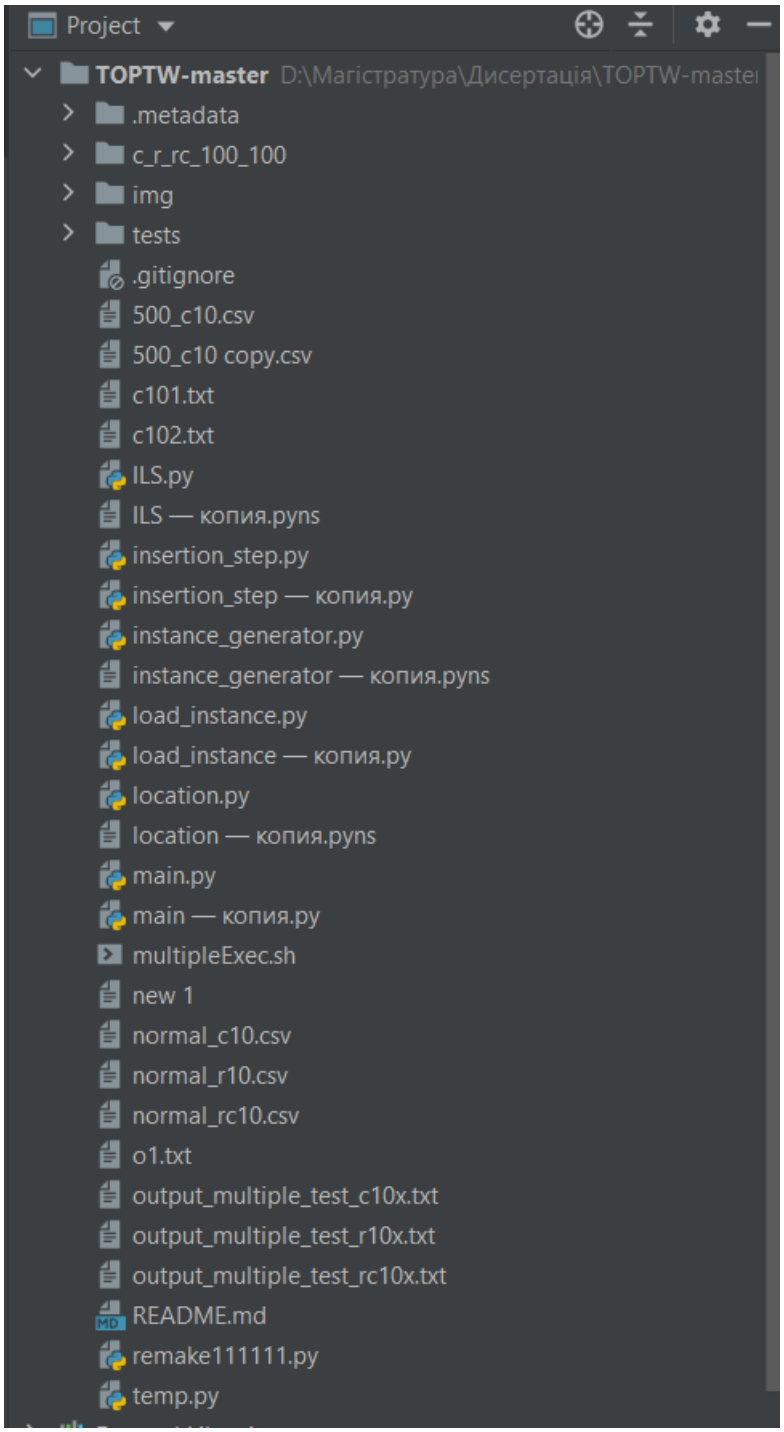


Рисунок 3.4.1 – Структура проекту ПЗ

Точкою входу є компонент main. Коли користувач хоче ініціювати роботу алгоритму, то main втикликає функції, що містятьться в динамічній бібліотеці TOPTW. При компілюванні і збірці TOPTW до неї включається весь вміст MultiExec. Тому, при виконанні main потребує лише TOPTW бібліотеку. Компонент main тестує функціональність алгоритму та всіх вхідних даних.

3.4.1 Опис класів

Основна функціональність проекту сконцентрована в TOPTW. Вона реалізується засобами таких класів:

- Main;
- Location;
- Load Instance;
- ILS;
- Insertion Step;
- Instance Generator;
- FLASK (interface)

Клас Insetion Step призначений длля обрахунку усіх параметреві точок маршрутів.

Вхідні дані зберігається у текстовому вайлі та передаються у клас Locations.

Instance Generator – застосовується, якщо параметри вхідного файлу не визначені. Тоді клас генерує певну кількість вершин, із заданою кількістю.

ILS – клас у якому виконується сама робота алгоритму ILS.

Клас MAIN – виконує запис даних у файл та відображення даних у консоль.

Клас FLASK – виконує передачу даних в веб інттерфейс та відображає стуктуру усіх класів та у цьому класі відбувається рендеринг данних у веб.

У частині графічного метеріалу представлена схема структурна класів бібліотеки TOPW.

Частина графічних матеріалів та аналізу вибірки даних міститься на платформі **DataBricks.** Для візувлізації реалізовано базу даних та на основі отриманих вихідних даних роботи алгоритму побудовано графіки та достілджено точність відпрацювання алгоритму.

3.4.2 Специфікація функцій

Таблиця 3.4.2.1 – Опис специфікації функцій бібліотеки TOPTW

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення функції
update_stuff	локація, час початку руху, час	Локації	Розрахунок доціальності включення локації в мааршрут на початковому етапі руху
wait	Час прибуття, час відкриття локації	Час очікування	Розрахунок витраченого часу на очікування працівником відкриття локації
estimateArrival	час, час початку маршруту, координати судуніх локацій	Час на пересування між 2-ма локаціями	Розрахунок часу витраченого на пересумання між двома локаціями на маршруті
getLeaveTime			Отримання часу відбуття з локації
ratio			Розрахунок цінності відвідування локації
Shift			Отримання загальної вартістості часу включення місця між location_i та location_k
update_locations			Ітерація з другого місця до передостаннього місця
set_at_better_position			Після функції оновлення встановляємо найкращу локація з максимальною корисністю відвідування
validate_time_windows			Визначаємо часові вікна
select_potential_location			Визначенн япотенційних локацій для відвідування у цьому турі
update_ratio			Оновлення цінності
update_shift			Оновлення загалььної вартості часу
update_wait			Оновлення часу очікування
update_arrival			Оновлення часу прибуття
update_leave			Оновлення часу від уття

Специфікацію функцій для класів TOPTWLib наведено в таблицях 3.4.2.2 – 3.4.2.8.

Таблиця 3.4.2.2 – Опис специфікації функцій класу Instance generator

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення функції
EuclideanDistance			Розрахунок евклідоової дистанції
generate_times_for_instances	Час прибуття, час відкриття локації	Час очікування	Генерування часу для пересування між дистанцями
generate			Генерація парамметрів локацій
load_instance			Завантаження даних з файлу

Таблиця 3.4.2.3 – Опис специфікації функцій класу Main

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
Print_Locations()	-	-	Відповідає за відображення даних у консолі
Write_file		-	Відповідає за запис даних у файл
Get tour ratio		-	Визначає максимальну користість побудованих маршрутів
Complete Tour		-	Створює масив маршрутів, які перетворює в тур
shake	-		Повертає остаток маршрутів, які не увійшли в тур, і збільшує їх пріорітет у відвідуванні.
main	-		Точка входу в програму

Таблиця 3.4.2.4 – Insertion step

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
Wait()	-	Поточний час, час відкриття магазину	Повертає інформації про очікування відкриття.
Get leave time ();	-	Час відбуття з магазину	Повретає час закінчення роботи на філії
int GetClosingTime();	-	Час закриття	Повертає значення, що відповідає останньому моменту, коли можна почати відвідування вершини
ratio		-	Обраховується корисність відвудування вершини
Update locations ()	-		Повертає оновлене значення параметрів пунктів
Select to insert	Час	-	Встановлює які вершини необхідно включити у маршрут
Update ratio ()	-	Час	Повертає оновлену корисність
Update arrival ()	-	Локація	Повертає оновлене значення прибуття
Update leave	-	Координата	Повертає оновлене значення відбуття
Set as better positions		-	Встановлює найкращы значення пунктів
void SetVariables()	-	-	Встановлює початкові значення для змінних
int GetArrivalTime()	-	Момент прибуття	Повертає значення часу прибуття в пункт
int EuclDist(Node*)	Адреса вершини	Відстань	Розраховує евклідову відстань між двома вершинами

Таблиця 3.4.2.5 – Опис специфікації функцій класу InputData

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
InputData(int verticesNr, Node ** nodesArr)	Номер вершини; масив адрес вершин	-	На основі вхідних даних створює об’єкт , що містить необхідні вхідні дані для роботи алгоритмів
InputData(const char fPath[])	Повний шлях до файлу	-	Зчитує дані необхідні для роботи алгоритмів з файлу
Node **const GetVertices()	-	Вершина	Викликається щоб отримати положення масиву вершин
const int *const GetVerticesNr()	-	Кількість вершин	Викликається щоб отримати адресу змінної зі значенням кількості вершин
void ReadData()	-	-	Допоміжна функція для зчитування з файлу

Таблиця 3.4.2.6 – Опис специфікації функцій класу ILS

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
ILS(InputData *inputData, const int *rtNr,int nrOfShakes=5, bool stopOnTime = false, int maxTimeSec = 0)	Вхідні дні; кількість маршрутів; кількість ітерацій; критерій завершення; очікуваний час роботи	-	Створює об’єкт, що будує розв’язок задачі TOPTW за даними, що подаються на вхід. Встановлюється критерій виходу для ILS
void Initialize()	-	-	Допоміжна функція для конструювання нового об’єкту
virtual void Run()	-	-	Запускає робуту модифікованого алгоритму повторюваного локального пошуку
Int GetFunctionValue()	-	Значення цільової функції	Повертає значення цільової функції після роботи алгоритму

Продовження таблиці 3.4.2.6

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
void Shake()	-	-	Запускається для модифікації отриманого розв’язку
bool TimeElapsed (time_point startTm)	Час початку роботи алгоритму	Значення предикат	Викликається, щоб перевірити чи не перевищує час виконання алгоритму встановлену межу
int CalculateShift (int rIndx, int i, Node*insNd)	Номер маршруту; позиція в маршруті; адреса нової вершини	Збільшення тривалості маршруту спричинене додаванням нової вершини	Викликається щоб визначити величину на яку збільшиться тривалості маршруту через додаванням нової вершини в певну позицію
int EuclidDist (Node *v1, Node *v2)	Адреса вершини; адреса вершини	Евклідова відстань	Повертає збережену евклідову відстань між заданими вершинами
virtual string ConsructedRoutesToString()	-	Побудовані маршрути	Повертає рядок, що містить послідовності вершин, що містять побудовані маршрути
void CalculateEuclidDistses()	-	-	Розраховує евклідові відстані між всіма вершинами і зберігає їх
virtual void InsertVertice (int routeIndx, int insPos,Node* insNd,int shift, int freeNdIndx)	Номер маршруту; позиція в маршруті; адреса нової вершини; зміна часу; позиція вершини	-	Викликається щоб додати нову вершину у маршрут в певну позицію
virtual void RemoveFromRoute (int * removePos, int removedVsNr, int m, bool saveHist = true)	Позиція у маршруті; кількість вершин;номер маршруту; показник збереження даних	-	Викликається щоб видалити певну кількість вершин на маршруті починаючи з визначеної позиціє. Дозволяє зберігати дані про те, які вершини і де було видалено.
void UpdateAfterRemoval (Node * leftVx, Node * rightVx,int rNdIndex)	Адреса вершини; адреса вершини; номер маршруту	-	Перераховує значення змінник після видалення вершин з маршруту.

Продовження таблиці 3.4.2.6

Сигнатура функції	Опис вхідних параметрів	Опис вихідних параметрів	Призначення та/чи опис функції
void ChangeShakeParams()	-	-	Регулює параметри збурення розв’язку
virtual void LS(double degree=2)	Степінь, що впливає на значимість корисності	-	Запускає алгоритм локального пошуку
void EvaluateNode (Node* prevNode, Node* crntNode)	Адреса вершини; адреса вершини; номер маршруту	-	Розраховує значення змінних, що була вставлена у маршрут
int PossibleWaitTime (Node *v1, Node *v2)	Адреса 1-ої вершини; адреса 2-ої вершини	Час очікування	Розраховує час очікування перед відвідуванням 2-ої вершини, при умові, що спочатку відвідується 1-а.
void EvaluateNode (Node* prevNode, Node* crntNode)	Адреса вершини; адреса вершини; номер маршруту	-	Розраховує значення змінних, що була вставлена у маршрут
void UpdateMaxShift (vector<Node *>& route, int NdPos)	Адреса маршруту; позиція в маршруті;	-	Оновлює змінну MaxShift для вставленої вершини і вершин, що передують їй
int EvaluateSubsequentNodes (vector<Node*>*route, int insertedNdPos, int insertedNdShift)	Адреса маршруту; позиція в маршруті; зсув у часі	Позиція, де оновлення завершилось	Оновлює значення змінних для вершин, що слідують вставленій. Оновлення завершується, коли зустрічається вершина на яку вставка не вплинула
int EvaluateSubsequentNodesAfter Removal (vector<Node *> *route, int crntNd, int shift)	Адреса маршруту; адреса вершини; час на який змінилась тривалість маршруту	Позиція вершини, де оновлення завершилось	Оновлює значення змінних для вершин, що слідують видаленій(им). Оновлення завершується, коли зустрічається вершина на яку зміни не вплинули

3.5 Технологічний розділ

Вимоги до технічного забезпечення:

Для роботи із застосуванням необхілно мати будь який встановлений веб-браузер, операційну систему Windows, процесор – Intel Core I5, оперативна пам’ять 64 Мб, пристої для взаємодії з користувачем – монітор, клавіатура та миша.

Керівництво користувача:

Головна сторінка застосування містить коротку інформацію про поточний відсоток втрат, поточний фактор плинності персоналу, місячні продажі та некоректно проведені переобліки.

А також інформацію, представлену у табличному викляді для порівняння показників втрат минулого та поточного місяця. Також можна переглянути усі магазини куратора із випадаючого списку, а також переглядати карту з створеними маршрутами для переміщення куратора.

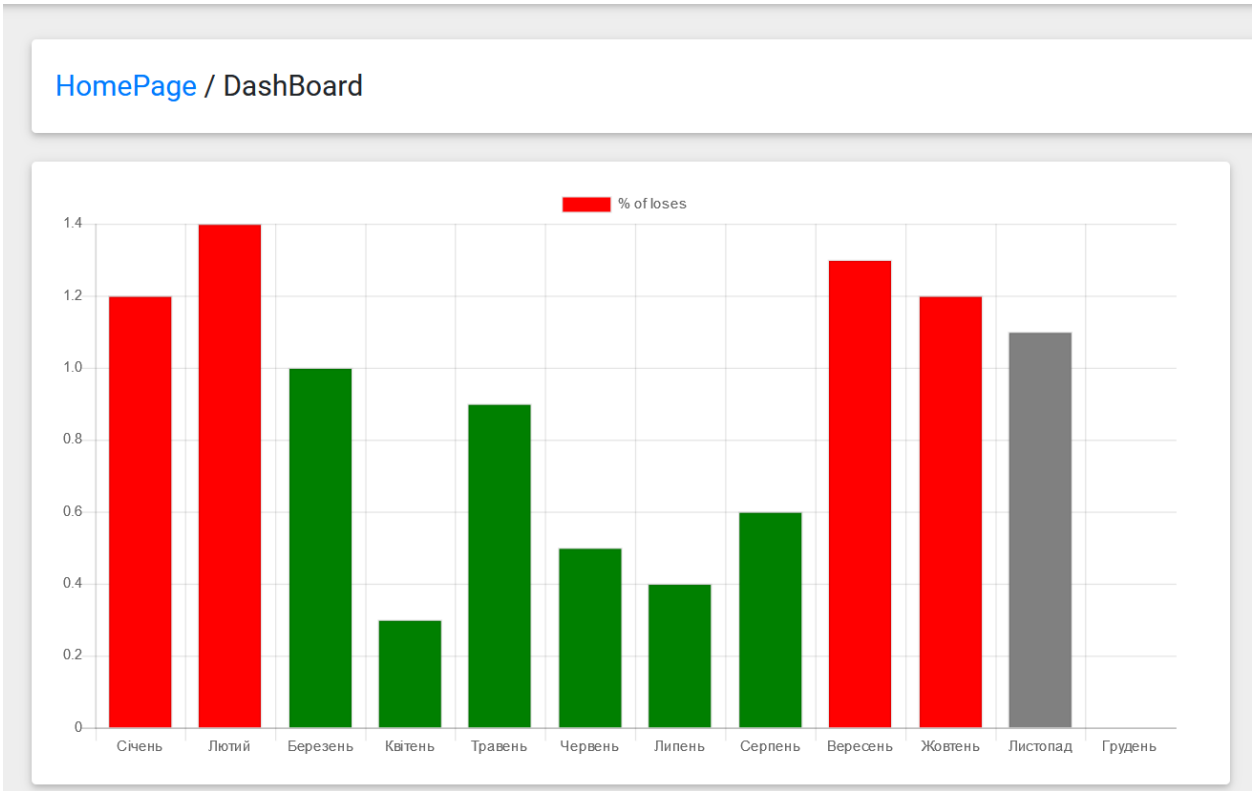


Рисунок 3.5.1 – Графік втрат з головної сторінки

Жовтень	Продажі	Списання	Недостача	Втрати
Ентузіастів 7	125 553	-2 205	-500	-2 705
Тичини 15	245 450	-3 251	-1 021	-4 271
Березняківська 36	125 553	-2 205	-500	-2 705

Листопад	Продажі	Списання	Недостача	Втрати
Ентузіастів 7	129 553	-1 205	-500	-1 705
Тичини 15	260 450	-2 251	-1 021	-3 271
Березняківська 36	134 553	-2 105	-400	-2 505

Рисунок 3.5.2 – Порівняння ключових показників попереднього та поточного місяців

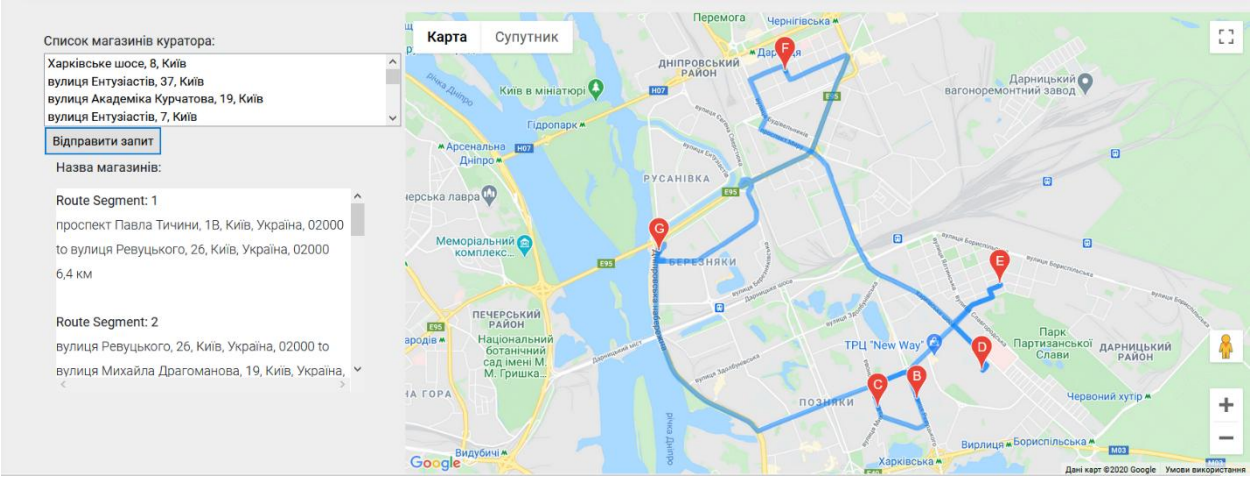


Рисунок 3.5.3. – Перегляд випадаючого списку магазиниів кураторів і карти з розрахованим маршрутом

На наступній сторінці – Таблиці, можна ознайомитись із інформацією по річним втратам, а також переглянути інформативний графік – порівняння з минулим роком.

Місяць	Січень	Лютий	Березень	Квітень	Травень	Червень	Липень	Серпень	Вересень	Жовтень	Листопад	Грудень
Продажі	151 438,00	123 584,00	153 515,00	153 138,00	235 556,00	123 584,00	153 515,00	123 155,00	123 518,00	123 551,00	153 115,00	242 365,00
Списання	-1 532,00	-1 351,00	-1 531,00	-3 512,00	-4 582,00	-1 531,00	-3 251,00	-1 223,00	-1 241,00	-1 215,00	-3 210,00	-3 214,00
Недостача	-632,00	-321,00	-423,00	-154,00	-325,00	-485,00	-751,00	-854,00	-632,00	-573,00	-987,00	-961,00
Втрати	-2 164,00	-1 672,00	-1 954,00	-3 666,005	-4 907,00	-2 016,00	-4 002,00	-2 077,00	-1 873,00	-1 788,00	-4 197,00	-4 175,00
% Втрат	-1,43%	-1,35%	-1,27%	-2,39%	-2,08%	-1,63%	-2,61%	-1,69%	-1,52%	-1,45%	-2,74%	-1,72%

Рисунок 3.5.4 – Табличне представлення інформації по втратам за рік

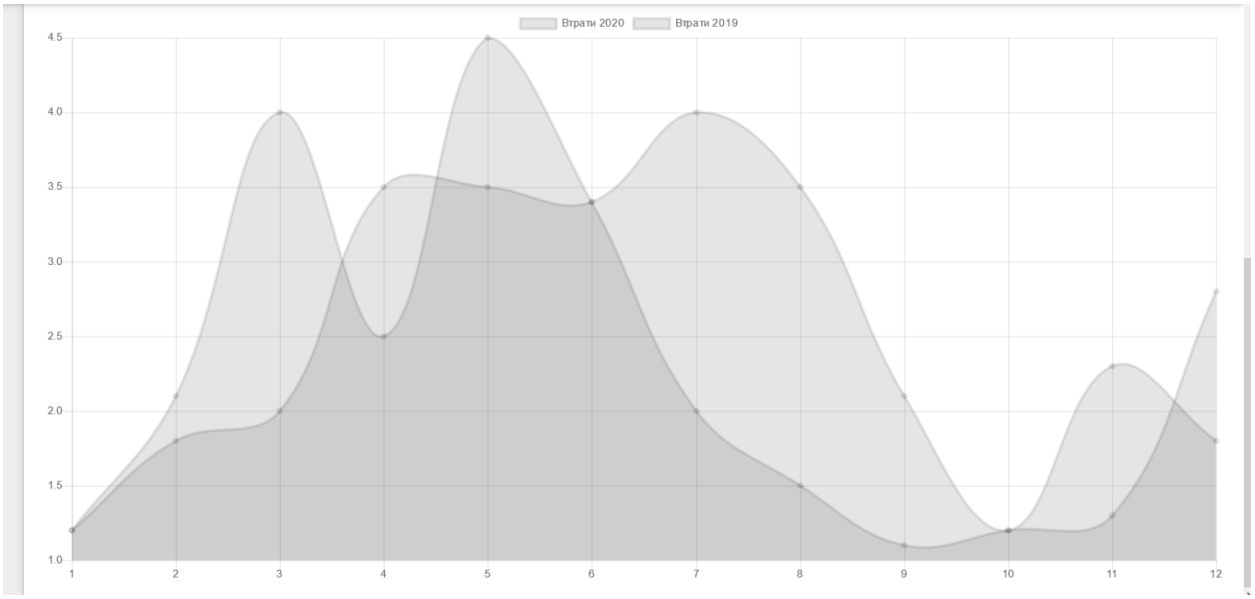


Рисунок 3.5.5 – Графік порівняння показників з минулим періодом

На вкладці «Чек-лист» можна ознайомитись з розкладом куратора, через написаний календар а також заповнити інформацію по чек-листу.

Також через календар можна створювати повторювану зустріч, встановлювати нагадування,планувати подію на весь день, а також обирати комфортний час.

Виконано/Не виконано	Пункт чек-листа	Пояснення до кожного пункту
☐	Перегляд камер відео спостереження	Пеглянь камери відеоспостереження на предмет виносу товару
☑	Перевірка списання	Перевір списання, що готуються викидати
☐	Проведення корегуючого переобліку	Перерахуй товари групи ризику, або ті, по яким критичні показники втрат
☑	Наявність форми у охорони	Пам'ятай, охоронець у формі - обличчя магазину

Рисунок 3.5.6 – Чек-лист куратора

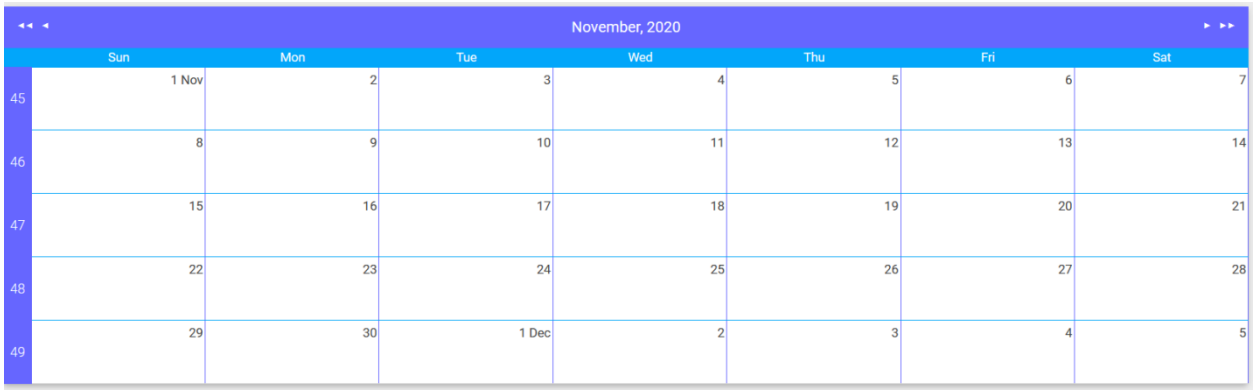


Рисунок 3.5.7 – візуалізація календаря

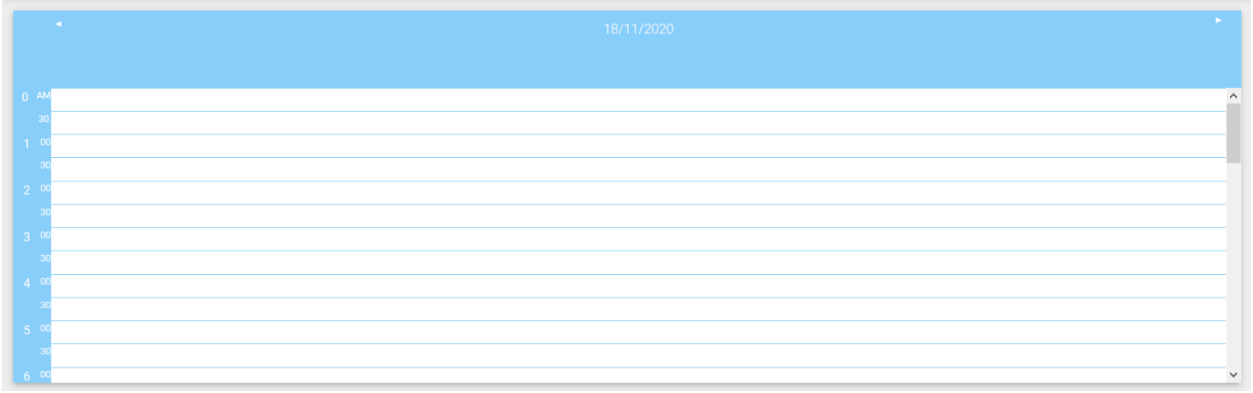


Рисунок 3.5.8 – Вибір часу для створення події

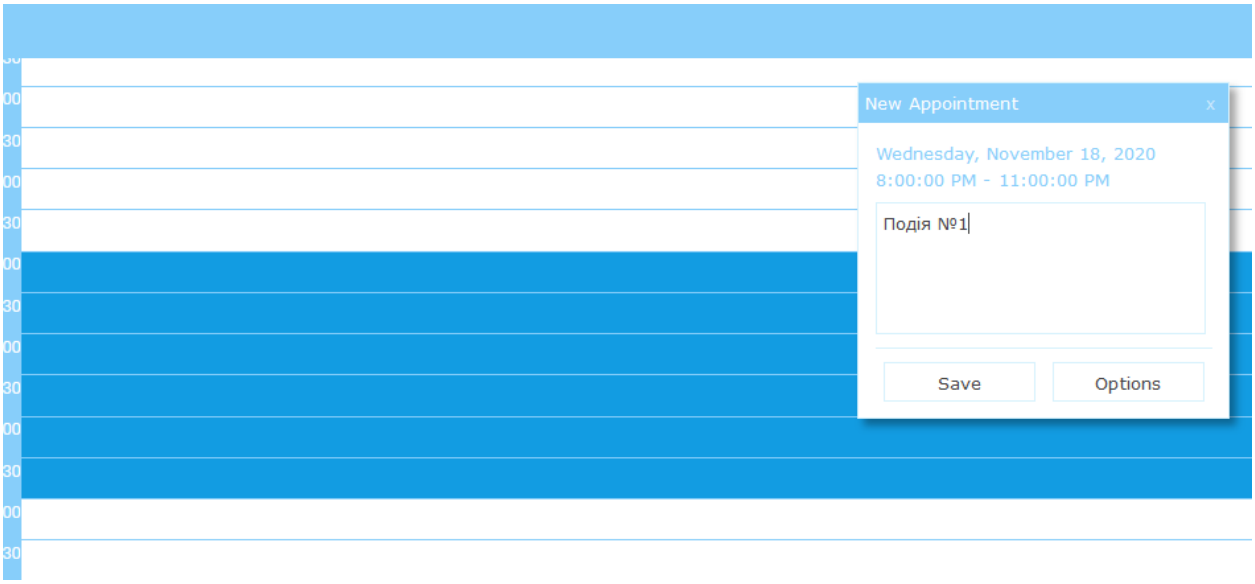


Рисунок 3.5.9 – Створення події

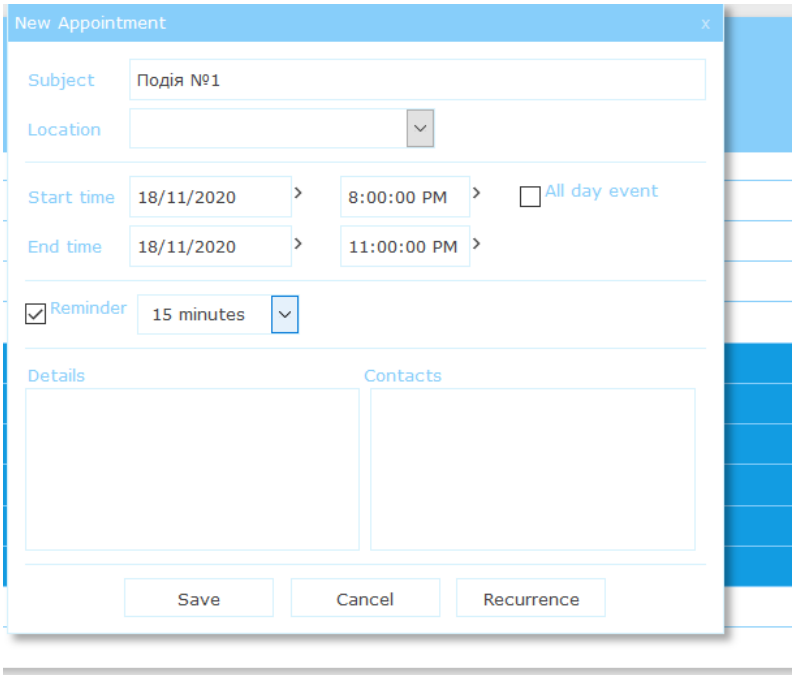


Рисунок 3.5.10 – Вибір нагадування

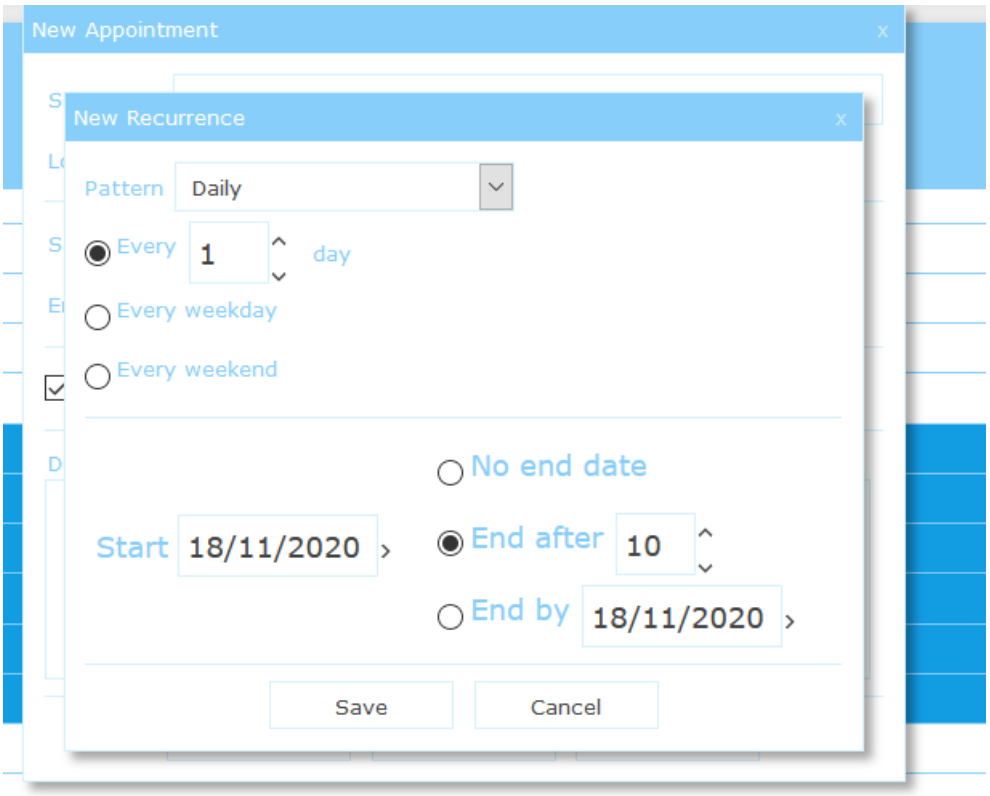


Рисунок 3.5.11 – Вибір повторень

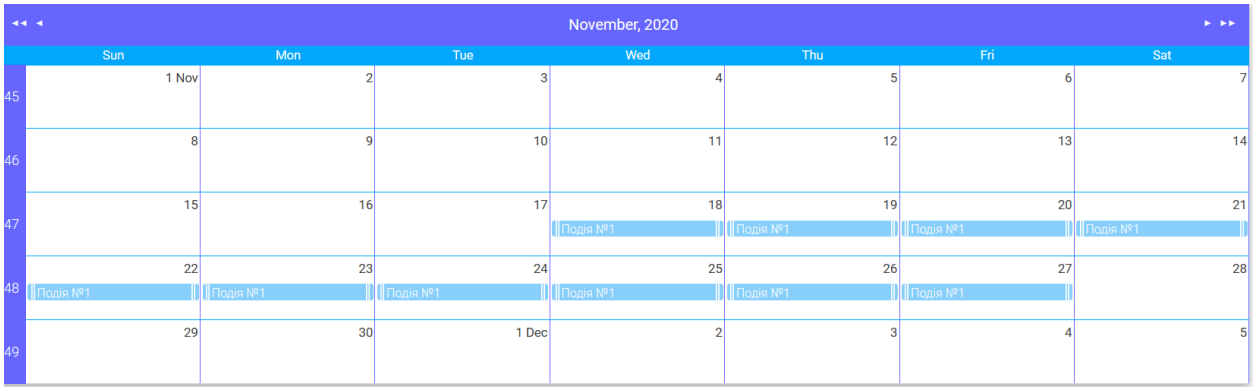


Рисунок 3.5.12 – Подію створено

3.6 Висновки до розділу

Даний розділ присвячено опису програмного забезпечення, в ньому наведено опис вхідних та вихідних даних, надано обґрунтування вибору засобів розробки і їх основні характеристики. Описано архітектуру програмного забезпечення на рівні компонентів і класів. Надано мотивіцію зробленим архітектурним рішенням. Наведено специфікацію функцій.

4 РОЗРОБЛЕННЯ СТАРТАП ПРОЄКТУ

4.1 Опис ідеї проєкту

З метою поліпшення якості виконання куратором своїх посадових обов'язків та організації свого робочого часу мною, керівництву служби безпеки було запропонована ідея розробки інструменту який допоможе організувати підлеглим свій час, а також дасть нам інструмент контролю та звітності.

Призначенням проєкту є реалізація програми органайзера, що допоможе не тільки притримуватись плану виконання потавлених завдач та завдань на день, а також, на основі закріплених об'єктів за кожним із кураторів будувати їм маршрути та розплановувати маршрути по дням.

Сутністю розробки проєкту є реалізація програми органайзера, що допоможе не тільки притримуватись плану виконання потавлених завдач та завдань на день, а також, на основі закріплених об'єктів за кожним із кураторів будувати їм маршрути та розплановувати маршрути по дням.

Цільовою аудиторією є куратори магазинів, в першу чергу куратори служби безпеки та охоони холдингу та їх керівництво, також аналітики, які будуть дану інформацію відслідковувати.

Основною вигодою використання цього продукту є мінімізація витрат часу на виконання рутинних задач, та мінімізація витрат часу на дорогу, а також збільшення корисної дії від кожного з праниців та зростання якості їхньої роботи.

Для того, щоб мати цілісне уявлення про зміст ідеї, а також можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів, опишемо зміст ідеї стартап проєкту, напрямки можливого застосування, а також вигоди для користувача у таблиці 4.1.1.

Таблиця 4.1.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Реалізація системи органайзера, що може будувати маршрути, та будувати розклад проходження того чи іншого маршруту у окремий день тижня.	Торгівля, Бізнес	Організація робочого часу, завжди можна зробити більше роботи, иникнути критичних ситуацій, через організацію власної роботи, а також можливості навчати персонал.

Отже, ідея створення системи автоматизованого аналізу тексту на предмет застосування прийомів психологічного впливу на приймачів інформації є досить актуальної з огляду на те що прийоми маніпулювання людською думкою завжди були неймовірно потужним інструментом у руках керівників держав чи організацій.

З огляду на те що з кожним роком люди сприймають все більше інформації з електронних джерел, розроблена система може бути інтегрована з найпопулярнішими інструментами що використовують потенційні користувачі для сприйняття інформації.

Наступним кроком буде проведення аналізу потенційних техніко-економічних переваг ідеї, у порівнянні із тим, що пропонують конкуренти: визначити перелік техніко-економічних властивостей та характеристик ідеї; визначити попереднє коло конкурентів (проєктів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та провести збір інформації щодо значень техніко-економічних показників для ідеї власного проєкту та проєктів-конкурентів відповідно до переліку, визначеного вище; провести порівняльний аналіз показників: для власної ідеї визначити показники, що мають

а) гірші значення (W, слабкі);

б) аналогічні (N, нейтральні) значення;

в) кращі значення (S, сильні) (табл. 5.2).

Після детального пошуку у мережі інтернет інструментів подібних до запропонованої системи був знайдений один конкурент – український стартап leegle.

У таблиці 4.1.2 наведемо порівняльний аналіз сильних, слабких та нейтральних характеристик ідеї проєкти у порівнянні із зазначеними конкурентами.

Таблиця 4.1.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

Ідея	**товари/концепції конкурентів**		**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
	Мій проєкт	**Tawagoto**			
Реалізація алгоримту локального пошуку з часовими вікнами для найбільш корректної побудови графіків маршрутів кожного із кураторів.	+	-			+
Передбачення та написання розкладу	+	-			+
Обмеження за обсягом введених даних	+	+		+	
Інтеграція системи з різними браузерами	+	+		+	
Інтеграція системи з різними операційними системами(Android, iOS, Smart TV)	-	+	-	+	
Можливість отримання консолідованого звіту по кожному прцівнику, якщо ти керівник, та кожен працінник може переглянути світ про совю діяльність протягом тижня та місяця	+	-			+

На основі визначених сильних, слабких та нейтральних сторін ідеї, можна провести аналіз та зрозуміти її конкурентоспроможність.

Отже, згідно наведеної таблиці можна зазначити, що розроблювальний проєкт має деякі переваги над системами-аналогами (своїми конкурентами), а саме: аналіз популярних електронних ресурсів на предмет отримання звітності, можививомстиі збору та обробки велиикх даних, та відсутність обмеження за обсягом введених даних.

Втім, на даному етапі розробки не планується включати у систему велику кількість інтеграцій з різними системами через брак часу. Але при необхідності інтеграції можуть бути реалізовані. Також, через певні обмеження використаних підходів неможливо виділяти конкретні ділянки тексту що відовідають патернам психологічного впливу. Але даний функціонал також не реалізований у конкурента.

4.2. Технічний аудит ідеї проєкту

В рамках даного розділу проведемо аудит технологій, за допомогою яких можна реалізувати ідею проєкту [11]. Співвідношення ідеї проєкти із технологією реалізації наведено у таблиці 4.2.1.

Таблиця 4.2.1 – Технологічна здійсненність ідеї проєкту

Ідея проєкту	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
Алгоритм SLWPLY	GoogleMapsAPI	Наявна	Доступна безкоштовно
Створення веб серверу що дозволить інтегрувати систему у інші продукти	REST API	Наявна	Доступна безкоштовно
Аналіз популярних електронних ресурсів на предмет використання патернів психологічного впливу	Мова програмування python та бібліотека для краулінгу сайтів scrapy	Наявна	Доступна безкоштовно
Розробка браузерного розширення що забезпечує функціональність системи	Мова програмування JavaScript	Наявна	Доступна безкоштовно

Як можна судити із наведеної таблиці, розроблювальний проєкт не потребує розробки нових технологій, адже використовує ті, вже наявні що є безкоштовними.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Визначимо ті ринкові можливості, які можна використати під час виходу проєкту на ринок, а також такі загрози, що можуть зашкодити підприємству.

Це дозволить спланувати можливі напрямки розвитку проєкту, з урахуванням того стану, у якому знаходиться ринкове середовище. Результати наведемо у таблиці 4.3.1.

Таблиця 4.3.1 – Попередня характеристика потенційного ринку стартап-проєкту

Номер пункту	Показники	Характеристика
1	Кількість головних гравців	1
2	Загальний обсяг продаж, грн/ум.од	Використання користувачами що сприймають інформації з електронних ресурсів
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Можливі у певних країнах світу
6	Середня норма рентабельності в галузі (або по ринку), %	Невідома

Оцінка потенційного ринку розродлювального стартап-проєкту показує, що умови для входу до ринку є досить сприятливими, особливо враховуючи що на ринку поки наявні фірми, які нещодавно на ньому перебувають та чиї системи поки не використовуються дуже успішно.

4.4 Розробка маркетингової програми стартапу

У цьому розділі сформуємо маркетингову концепцію товару, що отримає кінцевий споживач.

Кінцевий споживач може отримати доступ як до веб версії продукту, так і

до розширення для таких браузерів, як Google Chrome, Mozilla Firefox, Opera та Internet Explorer. Для нових користувачів надається тріал-версія продукту сроком до 1 місяця, для подальшого використання необхідно внести щомісячну сплату. Клієнт може підв'язати свою кредитну картку, і плата буде зніматися автоматично.

Для початку визначимо основні переваги продукту, що розробляється у таблиці 4.4.1.

Таблиця 4.4.1 – Основні переваги використання розроблювального продукту

Потреба	Вигода від використання продукту	Ключові переваги перед конкурентами
Перевірка інтернет-контенту на наявність спеціальних лінгвістичних конструкцій, що сприяють некритичному сприйняттю інформації	Класифікація інтернет-контенту на наявність пропаганди, захист користувача від маніпулятивних технік	Зручний інтерфейс, можливість використовувати плагін для браузерів, автоматична класифікація новин за допомогою штучного інтелекту

Таким чином можна бачити, що продукт має явні переваги над конкурентами, та відповідає потребам користувачів. Наступним кроком розробимо оберемо систему збуту. Інформацію надамо у таблиці 4.4.2.

Таблиця 4.4.2 – Система збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Підпис	Основні функції постачальника: ● Розробка ● Тестування ● Вдосконалення ● Збут та підтримка	Перший рівень	Пряма, із пошуком клієнтури

Також розробимо стратегію маркетингових комунікацій, та візьмемо за її основу специфіку поведінки клієнта, та стратегії позиціонування. Стратегія наведена у таблиці 4.4.3.

Таблиця 4.4.3 – Стратегія маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікації	Ключові позиції для позиціювання	Концепція рекламного звернення
Покращення якості наданих послуг, зменшення їх вартості	Інтернет ресурси, контекстна реклама, прес-релізи	Канал першого рівня	Покращення якості продукту, зручний інтерфейс

4.5 Розробка ринкової стратегії стартапу

Для опису ринкової стратегії, необхідно визначити групи потенційних користувачів (таблиця 4.5.1).

Таблиця 4.5.1 – Групи потенційних користувачів продукту

Назва групи	Готовність користування проєктом	Приблизний попит в межах групи	Інтенсивність конкуренції	Складність входу до сегменту
Журналісти, працівники сфери медіа	Готові	Високий	Середня	Середня
Аудиторія Інтернету	Готові	Середній	Середня	Середня

Згідно з таблицею, основними користувачами даного продукту можуть стати журналісти та працівники медіа сфери, оскільки вони мають справу з інтернет-виданнями та аналізують багато інформації, отриманої з джерел Інтернету. Також можливими користувачами можуть стати звичайні користувачі Інтернету, які хочуть перевіряти інформацію на наявність пропаганди. Саме цим сегментам варто пропонувати системи автоматичної класифікації текстових даних на наявність пропаганди, а працюючи із кожним сегментом, варто розробляти план впливу на нього окремо.

Сформуємо базову стратегію розвитку у цільових сегментах та наведемо її у таблиці 4.5.2.

Таблиця 4.5.2 – Визначення базової стратегії розвитку

Альтернатива розвитку продукту	Стратегія охоплення ринку	Конкурентноспроможні позиції відповідно до альтернативи	Базова стратегія розвитку
Стратегія спеціалізації	Пропонування продукту потенційним клієнтам, можлива модифікація ПО під потреби конкретного споживача	Довготривалі стосунки із клієнтами	Стратегія диференціації

У якості базової стратегії розвитку оберемо стратегію диференціації – стратегію, у якій орієнтування йде на потреби користувача. У разі провалу такої ідеї, стратегію можна змінити на стратегію спеціалізації – тобто орієнтування на конкретну цільову групу.

Наступним кроком визначимо стратегію конкурентної поведінки на ринку, та наведемо її у таблиці 4.5.3.

Таблиця 4.5.3 – Стратегія конкурентної поведінки на ринку

Чи є проєкт “першопрохідцем” на цільовому ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Так	Шукати нових споживачів	Так як продукт є “першопрохідцем” на ринку, всі характеристики товару будуть створюватися з нуля під потреби користувача.	Стратегія заняття конкурентної ніші

Отже, обраною стратегією є зайняття конкурентної ніші з охопленням декількох сегментів споживачів, а оскільки продукт не є першопрохідцем на ринку, то споживачів потрібно у більшості випадків забирати у існуючих конкурентів.

Визначена стратегія позиціонування продукту на ринку наведена у таблиці 4.5.4.

Таблиця 4.5.4 – Стратегія позиціонування продукт

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту
• Зменшення часу обробки тексту алгоритмом; • Виокремлення окремих фраз, які є потенційно маніпулятивними; • Робота з текстами не лише англійською, а і українською мовою; • Формування списку потенційно “небезпечних” сайтів.	Диференціації	Висока якість, налаштування на довготривалі стосунки із клієнтами	Швидкодія Клієнторієнтованість Точність класифікації

4.6 Висновки до розділу

Метою цього розділу є формування ідеї для розробки стартап проєкту. Спочатку було проаналізовано існуючий аналог системи автоматичної класифікації текстових даних на наявність пропаганди та проведений порівняльний аналіз з розроблювальним продуктом. Єдиний аналог має схожий функціонал, але було визначено переваги розроблювального алгоритма перед вже існуючим.

Була проведена експертиза можливості втілення проєкту з технічної точки зору, під час якої не було виявлено жодного ризику, оскільки для розробки треба використати ті інструменти, які вже існують та є безкоштовними.

Також був побудований маркетинговий план та визначені стратегії розвитку продукту на ринку.

ЗАГАЛЬНІ ВИСНОВКИ

В результаті огляду математичних постановок задачі побудови туристичних маршрутів предметом дослідження було обрано задачу TOPTW. Були наведені існуючі підходи до розв'язання TOPTW, описано практичні застосуваннння даної задачі.

Сформульована змістовна і математична постановка задачі; описано обрані методи розв'язання і їх специфіку відносно задачі TOPTW; створено алгоритмічне забезпечення для розв'язання задачі TOPTW на основі методів повторюваного локального пошуку та імітаційного відпалу; запропоновано модифікацію алгоритму повторюваного локального пошуку і покращено швидкодію використаних методів засобами паралельного програмування.

Було запрограмоване ПЗ яке реалізує алгоритмічне забезпечення для розв'язанння TOPTW на мові C++ і користувацький інтерфейс на мові C#. Специфіка спроектованої архітектури і простий користувацький інтерфейс дають можливість легко розширювати функціональність.

Для дослідження ефективності розроблених алгоритмів було проведено експерименти і встановлено, що розроблено модифікації алгоритму повторюваного локального пошуку з виграшем у швидкодії.

Було вирішено всі задачі для досягнення поставленої мети роботи. В результаті роботи автоматизовано процес складання туристичних маршрутів, з умовою, що максимізується сумарна корисность побудованих туристичних маршрутів заданої тривалості з врахуванням часових періодів відвідування туристичних місць.

Список літератури

1. The orienteering problem [Електронний ресурс] – Режим доступу до ресурсу: 1. Vansteenwegen P. The orienteering problem: A survey / P. Vansteenwegen, D. Oudheusdena, W. Souffriau. // 209. – 2011. – №1. – С. 1–10.
2. New Formulations for the Orienteering Problem [Електронний ресурс] – Режим доступу до ресурсу: 2. Kara I., Derya T., Bicakci P. New Formulations for the Orienteering Problem // Procedia Economics and Finance. – 2016. – №39. – С. 849–854..
3. A survey on algorithmic approaches for solving tourist trip design problems [Електронний ресурс] – Режим доступу до ресурсу: 3. Gavalas D., Mastakas K., Charalampos K., Pantziou G. A survey on algorithmic approaches for solving tourist trip design problems // Journal of Heuristics. – 2014. – №20. – С. 291–328..
4. Tackling large-scale home health care delivery problem with uncertainty [Електронний ресурс] – Режим доступу до ресурсу: 4. Tackling large-scale home health care delivery problem with uncertainty. / C.Chen, Z. Rubinstein, S. Smith, H. Lau. // Twenty-Seventh International Conference on Automated Planning and Scheduling. – 2017. – №27. – С. 18–23
5. A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing [Електронний ресурс] – Режим доступу до ресурсу: 5. TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing / [C. Chen, S. Cheng, A. Gunawan та ін.]. // HCOMP. – 2014.
6. The orienteering problem [Електронний ресурс] – Режим доступу до ресурсу: 6. Golden L. The orienteering problem / L. Golden, R. Vohra, L. Levy. // Naval Research Logistics. – 1987. – №34. – С. 307–318...
7. A Branch-and-Price Approach for the Team Orienteering Problem with Time Windows [Електронний ресурс] – Режим доступу до ресурсу: 7. Kim B. A Branch-and-Price Approach for the Team Orienteering Problem with Time Windows / B. Kim, H. Tae. // The International Journal of Industrial Engineering: Theory, Applications and Practice. – 2015. – №22. – С. 243–251.
8. Iterated local search heuristic for the team orienteering problem with time windows [Електронний ресурс] – Режим доступу до ресурсу: 8. Iterated local search heuristic for the team orienteering problem with time windows / P.Vansteenwegen, W. Souffriau, G. Berghe, D. Oudheusden. // Computers & Operations Research. – 2009. – №36. – С. 3281–3290..
9. Computing the Initial Temperature of Simulated Annealing [Електронний ресурс] – Режим доступу до ресурсу: 9. Walid B. Computing the Initial Temperature of Simulated Annealing / Ben-Ameur Walid. // Computational Optimization and Applications. – 2004. – №29. – С. 369–385.
10. An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem [Електронний ресурс] – Режим доступу до ресурсу: 10. An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem / Z.Luo, B. Cheang, A. Lim, W. Zhu. // European Journal of Operational Research. – 2013. – №3. – С. 673 – 682.
11. Applegate D. Chained Lin-Kernighan for large traveling salesman problems [Електронний ресурс] – Режим доступу до ресурсу: 11. Applegate D. Chained Lin-Kernighan for large traveling salesman problems / D. Applegate, A. Rohe, W. Cook. // INFORMS Journal on Computing. – 2003. – С. 82 – 92.

12. Privatized rural postman problems [Електронний ресурс] – Режим доступу до ресурсу: 12. Araoz J. Privatized rural postman problems / J. Araoz, E. Fernandez, C. Zoltan. // Computers & Operations Research. – 2016. – №33. – С. 3432–3449.
13. Resource-constrained geometric network optimization [Електронний ресурс] – Режим доступу до ресурсу: Applegate D. Chained Lin-Kernighan for large traveling salesman problems [Електронний ресурс] – Режим доступу до ресурсу: 11. Applegate D. Chained Lin-Kernighan for large traveling salesman problems / D. Applegate, A. Rohe, W. Cook. // INFORMS Journal on Computing. – 2003. – С. 82 – 92
14. Modern Information Retrieva [Електронний ресурс] – Режим доступу до ресурсу: 14. Baeza R. Modern Information Retrieval / R. Baeza , B. Ribeiro. – Нью-Йорк: Addison-Wesley, 1999. – 501 с...
15. A better algorithm for local search in combinatorial optimization problems [Електронний ресурс] – Режим доступу до ресурсу: 15. Baum E. Iterated descent: A better algorithm for local search in combinatorial optimization problems / Baum // Technical report Caltech / Baum. – Pasadena, 1986..
16. Towards practical “neural” computation for combinatorial optimization problems. [Електронний ресурс] – Режим доступу до ресурсу: 16. Baum E. Towards practical “neural” computation for combinatorial optimization problems. / Baum. // AIP conference. – 1986. – С. 53–64
17. Computing the Initial Temperature of Simulated Annealing [Електронний ресурс] – Режим доступу до ресурсу: 17. Ben-Ameur W. Computing the Initial Temperature of Simulated Annealing / WALID Ben-Ameur. // Computational Optimization and Applications. – 2004. – №29. – С. 369–385.
18. Best-in-class tools for any developer [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.
19. A heuristic for the multiple tour maximum collection problem [Електронний ресурс] – Режим доступу до ресурсу: 19. Butt S. A heuristic for the multiple tour maximum collection problem / S. Butt, T. Cavalier. // Computers & Operations Research. – 1994. – №21. – С. 101 – 111.
20. C Sharp [Електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.
21. C++ [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/C%2B%2Bhttps://uk.wikipedia.org/wiki/C_Sharp.
22. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm [Електронний ресурс] – Режим доступу до ресурсу: 22. Cerny V. A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm / Cerny. // Journal of Optimization Theory and Applications. – 1985. – №45. – С. 41–51.
23. A powerful improving concept for simulated annealing algorithms [Електронний ресурс] – Режим доступу до ресурсу: 23. Chardaire P. Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms. / P. Chardaire, J. Lutton, A. Sutter. // European Journal of Operational Research. – 1995. – №86. – С. 565–579
24. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem [Електронний ресурс] – Режим доступу до ресурсу: 24. Chen L. A parallel ant colony algorithm on massively parallel processors and its convergence analysis for the travelling salesman problem / L. Chen, H. Sun, S. Wang. // Information Sciences. – 2012. – №199. – С. 31 – 42.

25. Common Language Runtime (CLR) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/standard/clr.
26. A parallel iterated tabu search heuristic for vehicle routing problems [Електронний ресурс] – Режим доступу до ресурсу: 26. Cordeau J. A parallel iterated tabu search heuristic for vehicle routing problems / J. Cordeau, M. Maischbergerb. // Computers & Operations Research. – 2012. – №39. – С. 2033–2050.
27. Solution Methods for Vehicle Routing Problems [Електронний ресурс] – Режим доступу до ресурсу: 27. Crainic T. Parallel Solution Methods for Vehicle Routing Problems / Crainic // The Vehicle Routing Problem: Latest Advances and New Challenges / Crainic., 2008. – (Science and Business Media). – С. 497–542 Dynamic-link library [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Dynamic-link_library.
28. Efficient Metaheuristics for the Mixed Team Orienteering Problem with Time Windows [Електронний ресурс] – Режим доступу до ресурсу: 28. Efficient Metaheuristics for the Mixed Team Orienteering Problem with Time Windows / [D. Gavalas, C. Konstantopoulos, K. Mastakas та ін.]. // Algorithms. – 2016..
29. The orienteering problem [Електронний ресурс] – Режим доступу до ресурсу: 29. Golden L. The orienteering problem / L. Golden, R. Vohra, L. Levy. // Naval Research Logistics. – 1987. – №34. – С. 307–318
30. Home Health Care Delivery Problem [Електронний ресурс] – Режим доступу до ресурсу: 30. Gunawan A. Home Health Care Delivery Problem / A. Gunawan, L. Hoong, K. Kun. // Proceedings of the the 8th Multidisciplinary International Scheduling Conference.
31. Mobile Tourist Recommendation Systems Based On Tourist Trip Design Problem For Indonesia Domestic Tourist, An Exploratory Study [Електронний ресурс] – Режим доступу до ресурсу: 31. Hapsari I. Mobile Tourist Recommendation Systems Based On Tourist Trip Design Problem For Indonesia Domestic Tourist, An Exploratory Study / Indri Hapsari. // Widyatama International Seminar on Sustainability. – 2016. – №8
32. Graph Theory, Combinatorics and Algorithms [Електронний ресурс] – Режим доступу до ресурсу: 32. Hertz A. Recent Trends in Arc Routing / Hertz. // Graph Theory, Combinatorics and Algorithms. – 2005. – №34. – С. 215–236
33. The team orienteering problem with capacity constraint and time window [Електронний ресурс] – Режим доступу до ресурсу: 33. Hu X. The team orienteering problem with capacity constraint and time window / X. Hu, Z. Li. // The 10th International Symposium on Operations Research and its Applications (ISORA 2011). – 2011. – С. 157–163.
34. HCNA Networking Study Guide [Електронний ресурс] – Режим доступу до ресурсу: 34. Huawei. VRP Basics / Huawei // HCNA Networking Study Guide / Huawei., 2016. С. 978–981.
35. Iterated local search for the team orienteering problem with time windows [Електронний ресурс] – Режим доступу до ресурсу: 35. Iterated local search for the team orienteering problem with time windows / P.Vansteenwegena, W. Souffriaua, G. Bergheb, D. Oudheusdena. // Computers & Operations Research. – 2009. – №36. – С. 3281–3290.
36. Local optimization and the travelling salesman problem [Електронний ресурс] – Режим доступу до ресурсу: 36. Johnson D. Local optimization and the travelling salesman problem / Johnson. // In: Proceedings of the 17th Colloquium on Automata,

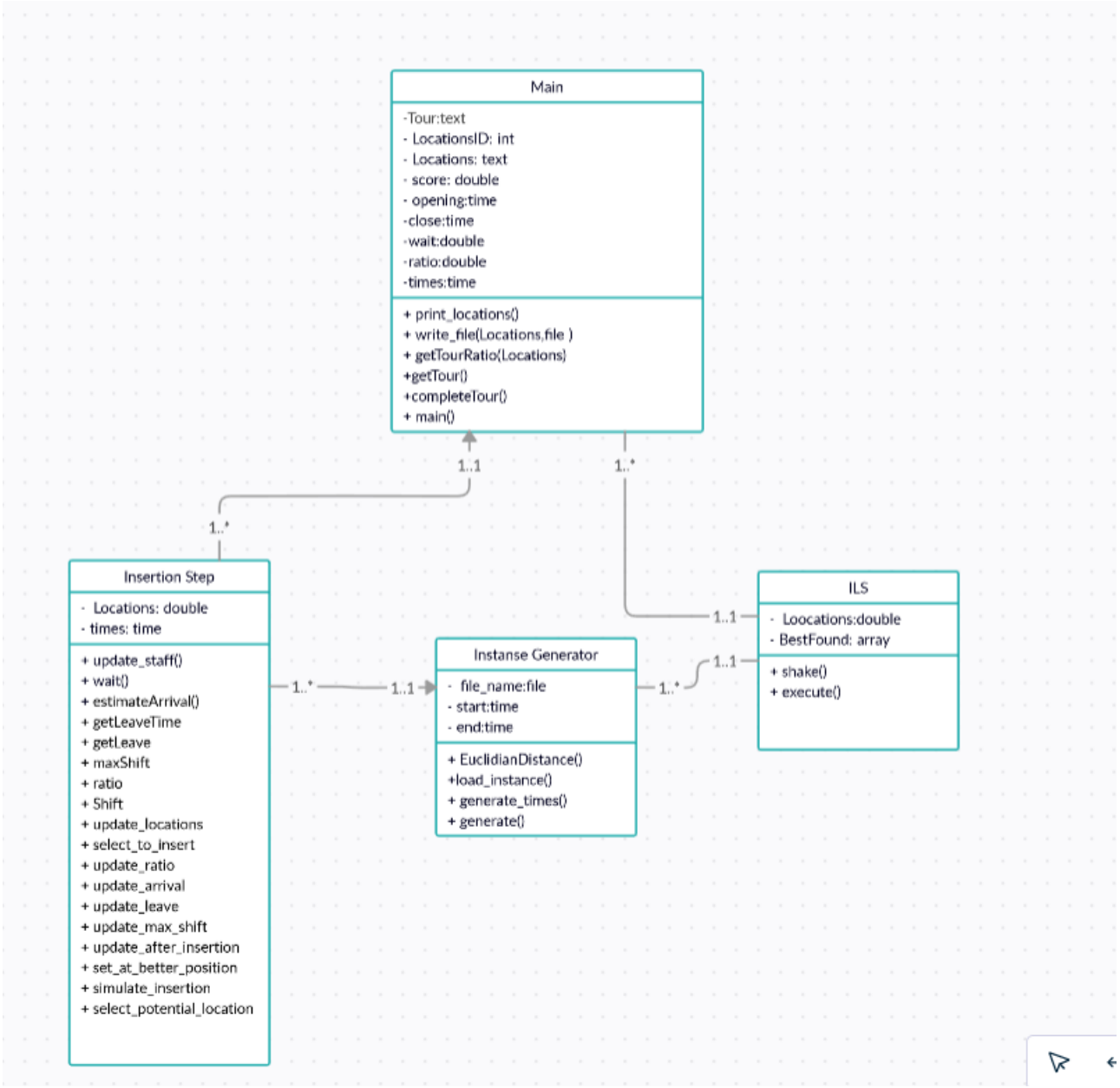
Languages, and Programming. Lecture Notes in Computer Science. – 1990. – №443. – С. 446–461.
37. The Traveling Salesman Problem [Електронний ресурс] – Режим доступу до ресурсу: 37. Johnson D. The Traveling Salesman Problem: A Case Study in Local Optimization /.
38. Local Search in Combinatorial Optimization [Електронний ресурс] – Режим доступу до ресурсу: 38. D. Johnson, L. McGeoch // Local Search in Combinatorial Optimization / D. Johnson, L. McGeoch. – London: John Wiley and Sons, 1997. – С. 215–310
39. An algorithm for single constraint maximum collection problem [Електронний ресурс] – Режим доступу до ресурсу: 39. Kataoka S. An algorithm for single constraint maximum collection problem / S. Kataoka, S. Morito. // Journal of the Operations Research Society of Japan. – 1988. №31. – С. 515–530.
40. Optimization by simulated annealing [Електронний ресурс] – Режим доступу до ресурсу: 40. Kirkpatrick S. Optimization by simulated annealing / S. Kirkpatrick, C. Gelatt, M. Vecchi. // Science. – 1983. – С. 671–689.
41. An Effective Hybrid Evolutionary Local Search for Orienteering and Team Orienteering Problems with Time Windows [Електронний ресурс] – Режим доступу до ресурсу: 41. Labadi N N. An Effective Hybrid Evolutionary Local Search for Orienteering and Team Orienteering Problems with Time Windows / N. Labadi N, J. Melechovsky, R. Calvo. // Parallel Problem Solving from Nature, PPSN XI. PPSN 2010. Lecture Notes in Computer Science. – 2010. – № 6239. – С. 219–228.
42. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows [Електронний ресурс] – Режим доступу до ресурсу: 42. Labadi N N. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows / N. Labadi N, J. Melechovsky, R. Calvo. // Journal of Heuristics. – 2011. – №17. – С. 729–753
43. The selective travelling salesman problem [Електронний ресурс] – Режим доступу до ресурсу: 43. Laporte G. The selective travelling salesman problem / G. Laporte, S. Martello. // Discrete Applied Mathematics. – 1990. – №26. – С. 193 – 207
44. A simulated annealing heuristic for the team orienteering problem with time windows [Електронний ресурс] – Режим доступу до ресурсу: 44. Lin S. A simulated annealing heuristic for the team orienteering problem with time windows / S. Lin, V. Yu. // European Journal of Operational Research. – 2012. – №217. – С. 94 – 107.
45. Local Search: Framework and Applications [Електронний ресурс] – Режим доступу до ресурсу: 45. Lourenço H. Iterated Local Search: Framework and Applications / H. Lourenço, O. Martin, T. Stützle // Handbook of Metaheuristics / H. Lourenço, O. Martin, T. Stützle., 2014. – (2). – (International Series in Operations Research & Management Science; кн. 146). – С. 363–397.
46. Lundy M. Convergence of an annealing algorithm / M. Lundy, A. Mees. // Mathematical Programming. – 1986. – №34. – С. 111–124. Convergence of an annealing algorithm [Електронний ресурс] – Режим доступу до ресурсу: 46. Lundy M. Convergence of an annealing algorithm / M. Lundy, A. Mees. // Mathematical Programming. – 1986. – №34. – С. 111–124.
47. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem [Електронний ресурс] – Режим доступу до ресурсу: 47. Malandraki C. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem / C. Malandraki, M. Daskin. // European Journal of Operational Research. – 1093. – №65. – С. 218–234.

48. Combining simulated annealing with local search heuristics [Електронний ресурс] – Режим доступу до ресурсу: 48. Martin O. Combining simulated annealing with local search heuristics / O. Martin, S. Otto. // Ann. Oper. Res.. – 1996. – №63. – С. 57–75.
49. Large-step Markov chains for the traveling salesman problem [Електронний ресурс] – Режим доступу до ресурсу: 49. Martin O. Large-step Markov chains for the traveling salesman problem / O. Martin, S. Otto, E. Felten. // Complex Syst.. – 1991. – №5. – С. 299–326.
50. Microsoft Visual Studio Community 2017 [Електронний ресурс] – Режим доступу до ресурсу:https://www.visualstudio.com/license-terms/mlt553321/?rr=https%3A%2F%2Fwww.google.de%2F.
51. An ant colony system for team orienteering problems with time windows [Електронний ресурс] – Режим доступу до ресурсу: 51. Montemanni R. An ant colony system for team orienteering problems with time windows / R. Montemanni, L. Gambardella. // Foundations of Computing and Decision Sciences. – 2009. – №34. – С. 287–306.
52. Multi-Agent task assignment for mobile crowdsourcing under trajectory uncertainties (extended abstract) [Електронний ресурс] – Режим доступу до ресурсу: Multi-Agent task assignment for mobile crowdsourcing under trajectory uncertainties (extended abstract) / C.Chen, S. Cheng, A. Misra, H. Lau. // 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015). – 2015. –
53. С. 4–8Niazi A. 10 advantages of C# programming language [Електронний ресурс] / ASAD Niazi – Режим доступу до ресурсу: http://proprogrammershub.blogspot.de/2016/04/top-10-advantages-of-c.html.
54. OP_FORMAT [Електронний ресурс] – Режим доступу до ресурсу: https://www.mech.kuleuven.be/en/cib/op/instances/OP_format/view.
55. OpenMp [Електронний ресурс] – Режим доступу до ресурсу: http://www.openmp.org/.
56. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem [Електронний ресурс] – Режим доступу до ресурсу: 56. Osman I. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. / Osman. // Analysisof Operations Research. – 1993. – №41. – С. 421–451.
57. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints [Електронний ресурс] – Режим доступу до ресурсу: 57. Pang K. An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints / Pang. // Expert Systems with Applications. – 2011. – №38. – С. 11939–11946.
58. Transforming arc routing into node routing problems [Електронний ресурс] – Режим доступу до ресурсу: 58. Pearn W. Transforming arc routing into node routing problems / W. Pearn, A. Assad, B. Golden. // Computers & Operations Research. – 1987. – №14. – С. 285–288.
59. Personalized Tourist Route Generation [Електронний ресурс] – Режим доступу до ресурсу: Personalized Tourist Route Generation / [A. Garcia, O. Arbelaitz, M. Linaza та ін.]. // ICWE 2010: Current Trends in Web Engineering. – 2010. – №6385. – С. 486–497.
60. A Personalised Tourist Trip Design Algorithmfor Mobile Tourist Guides [Електронний ресурс] – Режим доступу до ресурсу: 60. Souffriau W. A Personalised Tourist Trip Design Algorithmfor Mobile Tourist Guides / W. Souffriau, P. Vansteenwegen, J. Vertommen. – 2008

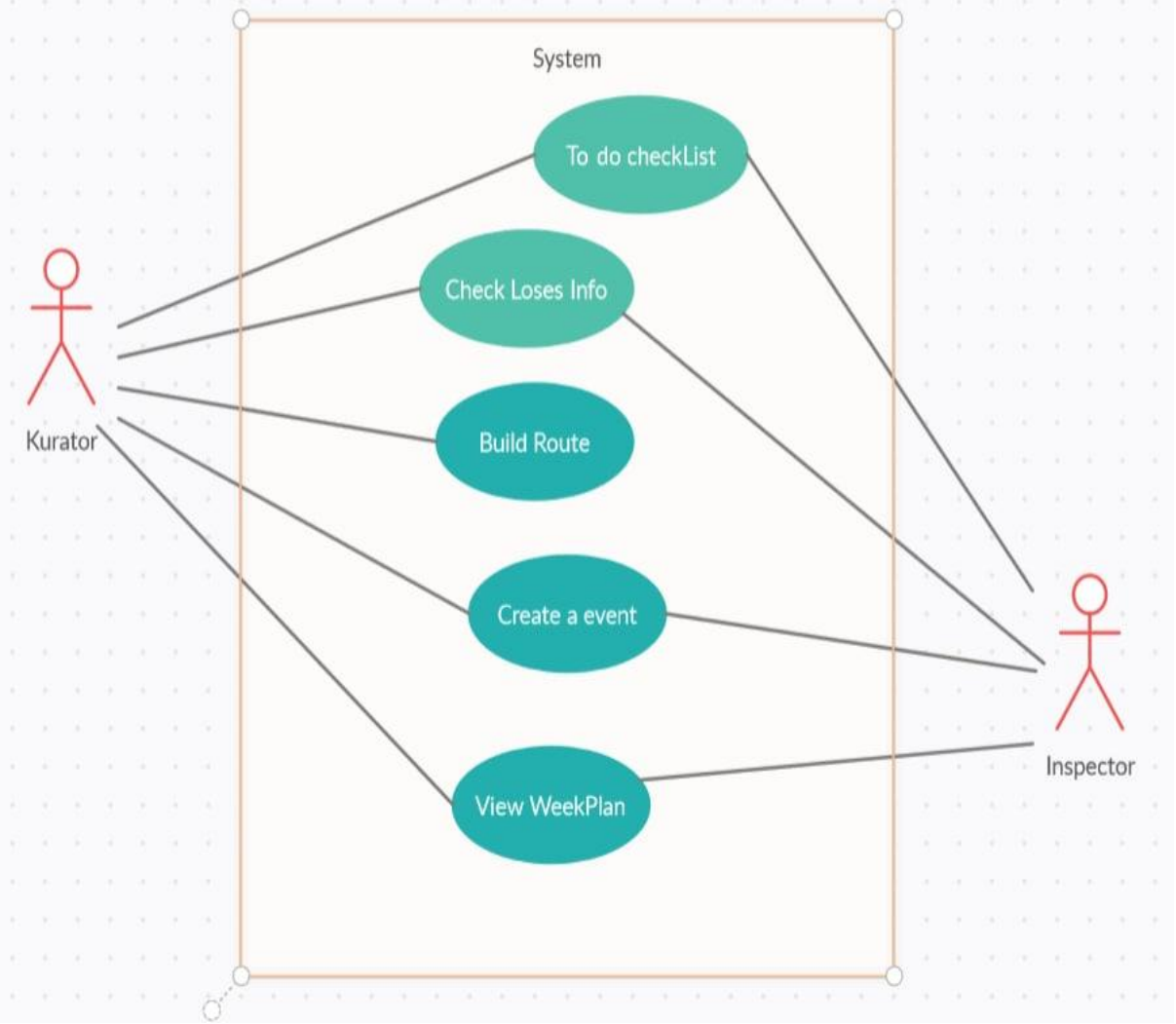
61. Tackling large-scale home health care delivery problem with uncertainty [Електронний ресурс] – Режим доступу до ресурсу: 61. Tackling large-scale home health care delivery problem with uncertainty. / C.Chen,Z. Rubinstein, S. Smith, H. Lau. // Twenty-Seventh International Conference on Automated Planning and Scheduling. – 2017. – №27. – С. 18–23..
62. Approach for the Team Orienteering Problem with Time Windows [Електронний ресурс] – Режим доступу до ресурсу: 62. Tae H. A Branch-and-Price Approach for the Team Orienteering Problem with Time Windows / H. Tae, B. Kim. // The International Journal of Industrial Engineering: Theory, Applications and Practice. – 2015. – №22. – С. 243–251
63. The Chinese Postman Problem with Load-Dependent Costs [Електронний ресурс] – Режим доступу до ресурсу: 63. The Chinese Postman Problem with Load-Dependent Costs / [A. Corberan, E. Gunes, G. Laporte та ін.]. // Transportation Science. – 2018. – №52. – С. 370 – 385.
64. The Orienteering Problem: Test Instances [Електронний ресурс] – Режим доступу до ресурсу: https://www.mech.kuleuven.be/en/cib/op.
65. The Stacker Crane Problem and the directed general routing problem [Електронний ресурс] – Режим доступу до ресурсу: 65. The Stacker Crane Problem and the directed general routing problem / T.Avila, A. Corberan, I. Plana, J. Sanchis. // Networks - An International Journal. – 2015. – №56. – С. 43–55. Tsiligirides T. Heuristic methods applied to orienteering / Tsiligirides. // The Journal of the Operational Research Society
66. The team orienteering problem with time windows: An lp-based granular variable neighborhood search [Електронний ресурс] – Режим доступу до ресурсу: 66. The team orienteering problem with time windows: An lp-based granular variable neighborhood search / N.Labadi, J. Melechovsky, R. Woler Calvo, R. Mansini. // European Journal of Operational Research. – 2012. – №220. – С. 15–27
67. Towards City-scale Mobile Crowdsourcing: Task Recommendations under Trajectory Uncertainties [Електронний ресурс] – Режим доступу до ресурсу: 67. Towards City-scale Mobile Crowdsourcing: Task Recommendations under Trajectory Uncertainties / C.Chen, S. Cheng, H. Lau, A. Misra. // International Joint Conference on Artificial Intelligence (IJCAI-2015). – 2015. – С. 25 – 31.
68. A Framework for Trajectory-Aware Coordinated Urban Crowd- Sourcing [Електронний ресурс] – Режим доступу до ресурсу: 68. TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd- Sourcing / [C. Chen, S. Cheng, A. Gunawan and oth.]. // HCOMP. – 2014.
69. Unmanaged Code [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/uk-ua/library/0e91td57(v=vs.100).aspx.
70. UNWTO [Електронний ресурс] – Режим доступу до ресурсу: http://www2.unwto.org/.
71. Planning in Tourism and Public Transportation - Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling [Електронний ресурс] – Режим доступу до ресурсу: 71. Vansteenwegen P. Planning in Tourism and Public Transportation - Attraction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling / Vansteenwegen. // 4OR-Q J Oper Res. – 2009. – №61. – С. 7–293. Vansteenwegen P. The mobile tourist guide: An or opportunity / P. Vansteenwegen,
72. Operational Research Insight [Електронний ресурс] – Режим доступу до ресурсу: 72. D. Van Oudheusden. // Operational Research Insight. – 2007. – №20. – С. 21–27.

73. The orienteering problem: A survey [Електронний ресурс] – Режим доступу до ресурсу: 73. Vansteenwegen P. The orienteering problem: A survey / P. Vansteenwegen, W. Soffuriau, D. Van Oudheusden. // European Journal of Operational Research. – 2011.
74. Visual Studio IDE overview [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide.
75. The team orienteering problem [Електронний ресурс] – Режим доступу до ресурсу: 75. Wasilc E. The team orienteering problem / E. Wasilc, B. Golden, I. MingChao. // European Journal of Operational Research. – 1996. – №88. – С. 464 – 474.
76. What Is Managed Code? [Електронний ресурс] // What Is Managed Code? – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx.
77. White S. Concepts of scale in simulated annealing [Електронний ресурс] – Режим доступу до ресурсу: White S. Concepts of scale in simulated annealing / White. // IEEE Int. Conference on Computer Design. – 1984.
78. Assisting Pedestrians to Find Preferred Events and Comfortable Public Transport Connections [Електронний ресурс] – Режим доступу до ресурсу: 79. Zenker B. ROSE - Assisting Pedestrians to Find Preferred Events and Comfortable Public Transport Connections / B. Zenker, B. Ludwig. // 6th International Conference on Mobile Technology, Application & Systems. – 2009. – С. 1–6.
79. Windows Forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/winforms/
80. Баран Г. Розвиток туризму в Україні: проблеми та перспективи [Електронний ресурс] / Г. Баран. – 2017. – Режим доступу до ресурсу: http://marker.ua/ua/sotsialnyj- blok/1804-razvitie-turizma-v-ukraine-problemy-i-perspektivy/
81. Гуляницький Л. Ф. Прикладні методи комбінаторної оптимізації / Л. Ф. Гуляницький, О. Ю. Мулеса. – Київ: Видавничо-поліграфічний центр "Київський університет", 2016. – 142 с
82. Прохорова К. Математична постановка та огляд методів розв’язування задачі побудови туристичних маршрутів / К. Прохорова, Л. Гуляницький. // Інформатика та обчислювальна техніка-ІОТ-2017. – 2017. – С. 78-89.
83. Прохорова К. С. Метод Розв’язання Задачі Командного Спортивного Орієнтування з Часовими Вікнами / К. С. Прохорова // Актуальні Питання Сьогодення / К. С. Прохорова. – Обухів: Типография «Друкарик», 2018. – (9). – С. 70–71.
84. Туризм в Україні: проблеми, перспективи розвитку [Електронний ресурс] – Режим доступу до ресурсу: http://skole.com.ua/uk/papers/13-turizm/36-turizmukrproblemi.html
85. Моделі та методи комп'ютерного навчання з урахуванням індивідуальних здібностей користувачів [Електронний ресурс] – Режим доступу до ресурсу: referatu.net.ua/referats/7569/163166.

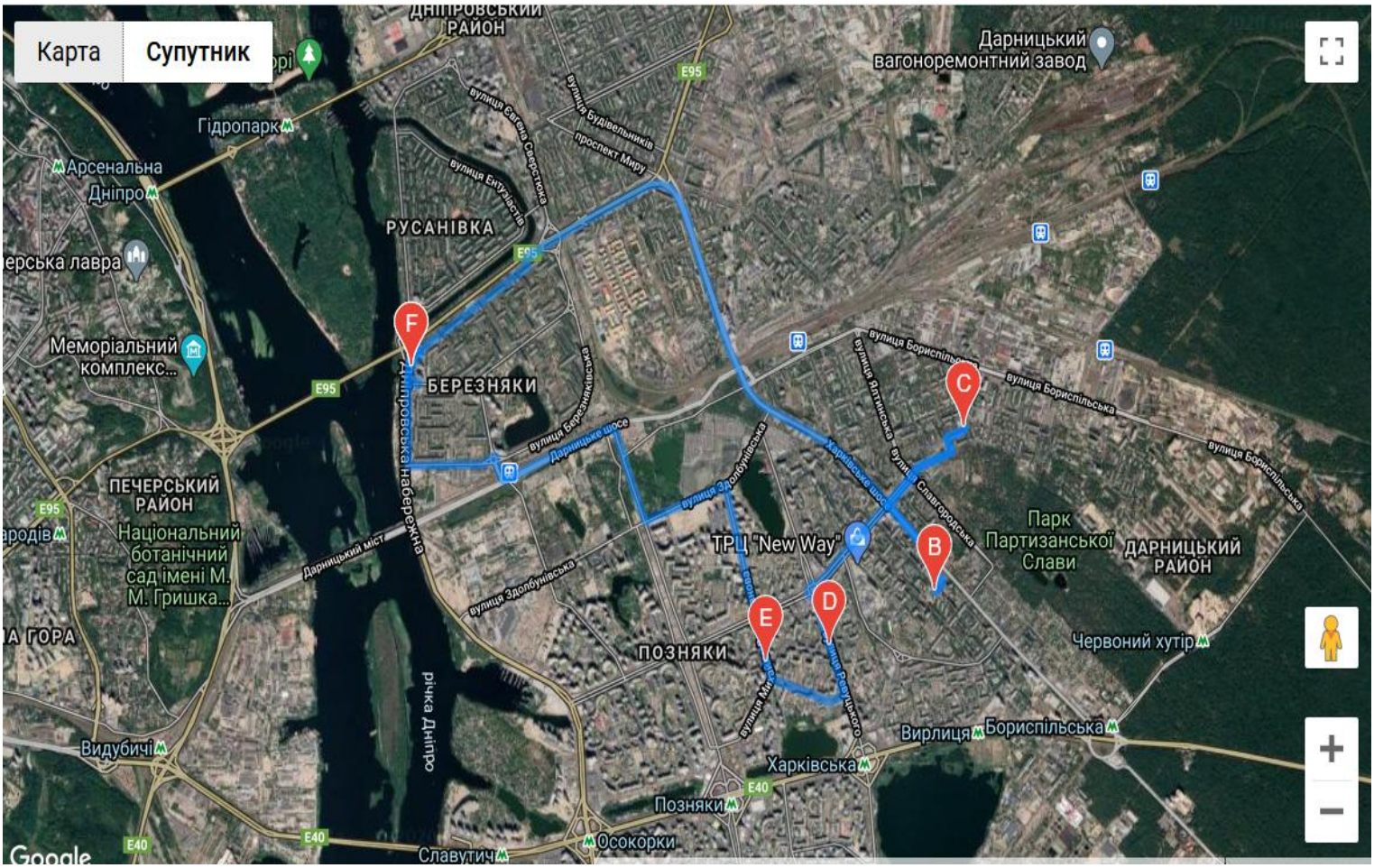
ДОДАТОК А ГРАФІЧНИЙ МАТЕРІАЛ
ПЛАКАТ 1 ДІАГРАМА КЛАСІВ



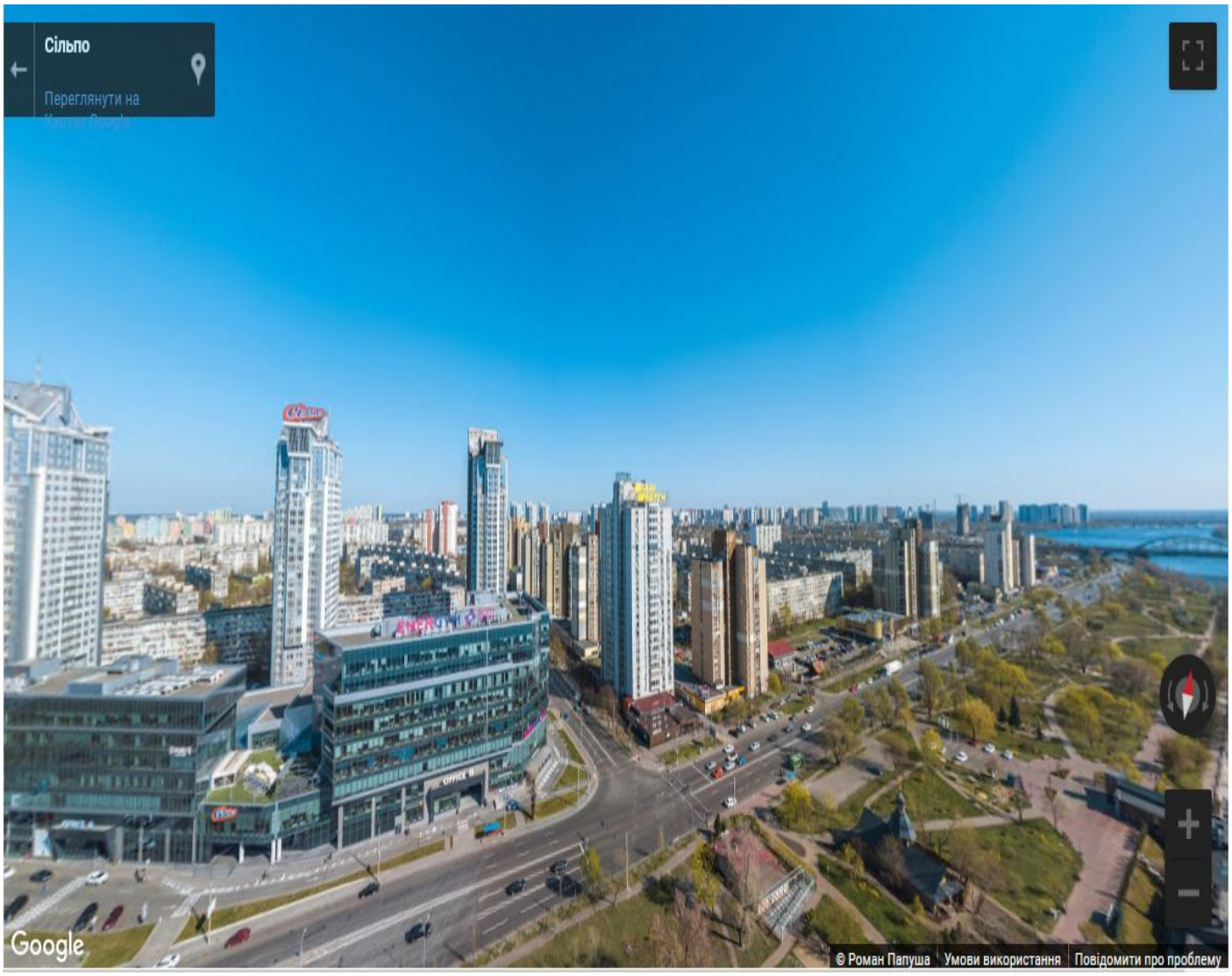
ПЛАКАТ 2 ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ПЛАКАТ 3 ПЕРЕГЛЯД КАРТИ, РЕЖИМ З СУПУТНИКА



ПЛАКАТ 4 ПЕРЕГЛЯД ВУЛИЦЬ У ДОДАТКУ



ПЛАКАТ 5 СХЕМА СТУКТУРНА КОМПОНЕНТІВ ПРОЕКТУ TOPTW