ДОДАТОК А

Слайди презентації

Атестаційна робота магістра

Дослідження генетичних алгоритмів для реалізації підтримки метромарафонів в системі “Transit challenge”

Виконав: ст. гр. ПЗСм-18-1 Тарас Гордієнко

Керівник: доц. О. О. Мазурова

1

Рисунок А.1 – Титульний слайд

Актуальність проблемної області

Метромарафон – змагання, суть якого полягає в відвідуванні всіх станцій метрополітена за найкоротший час.

- проводиться з 1966 року по всьому світі;
- ніколи не проводився в Україні.

Є NP-повною задачею

Метро	Кількість станцій	Кількість можливих маршрутів
Харківське	30	2.6525286×10^{32}
Київське	52	8.0658175×10^{67}

Для рішення застосовують евристичні методи:

- мурашиний алгоритм
- **генетичні алгоритми**
- нейронні мережі
- та інші

2

Рисунок А.2 – Слайд «Актуальність»

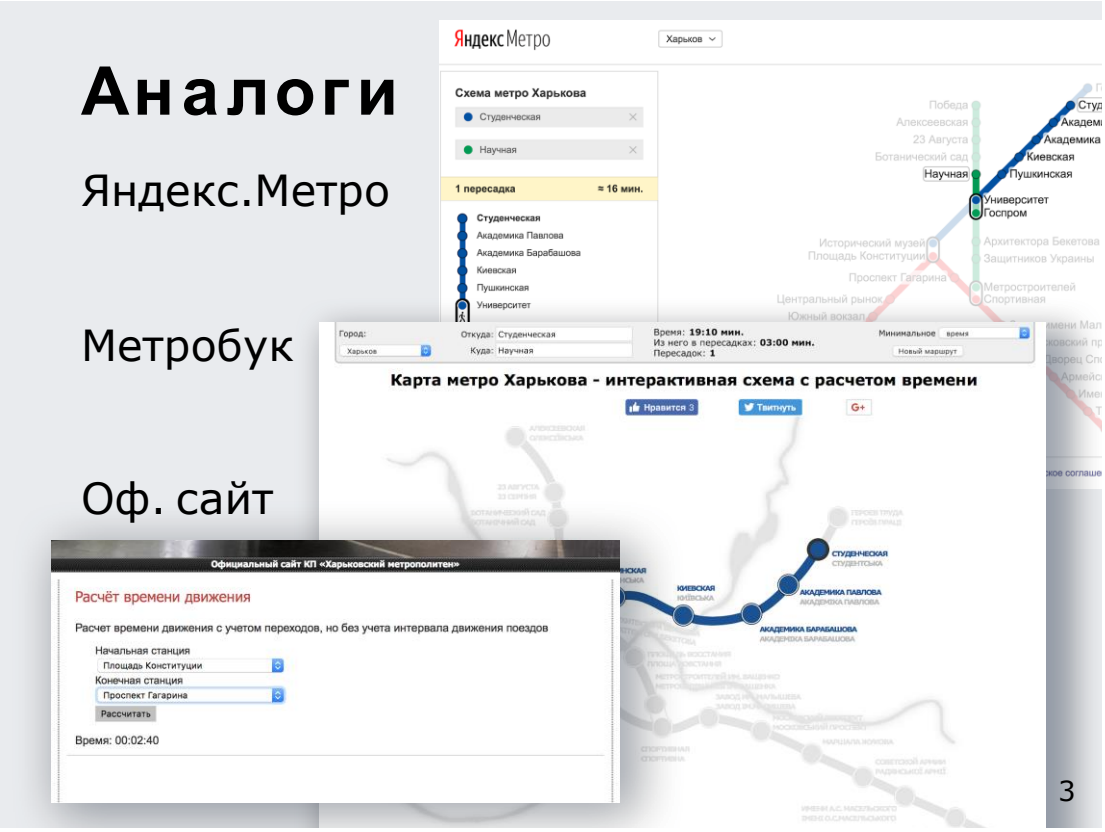


Рисунок А.3 – Слайд «Аналоги»

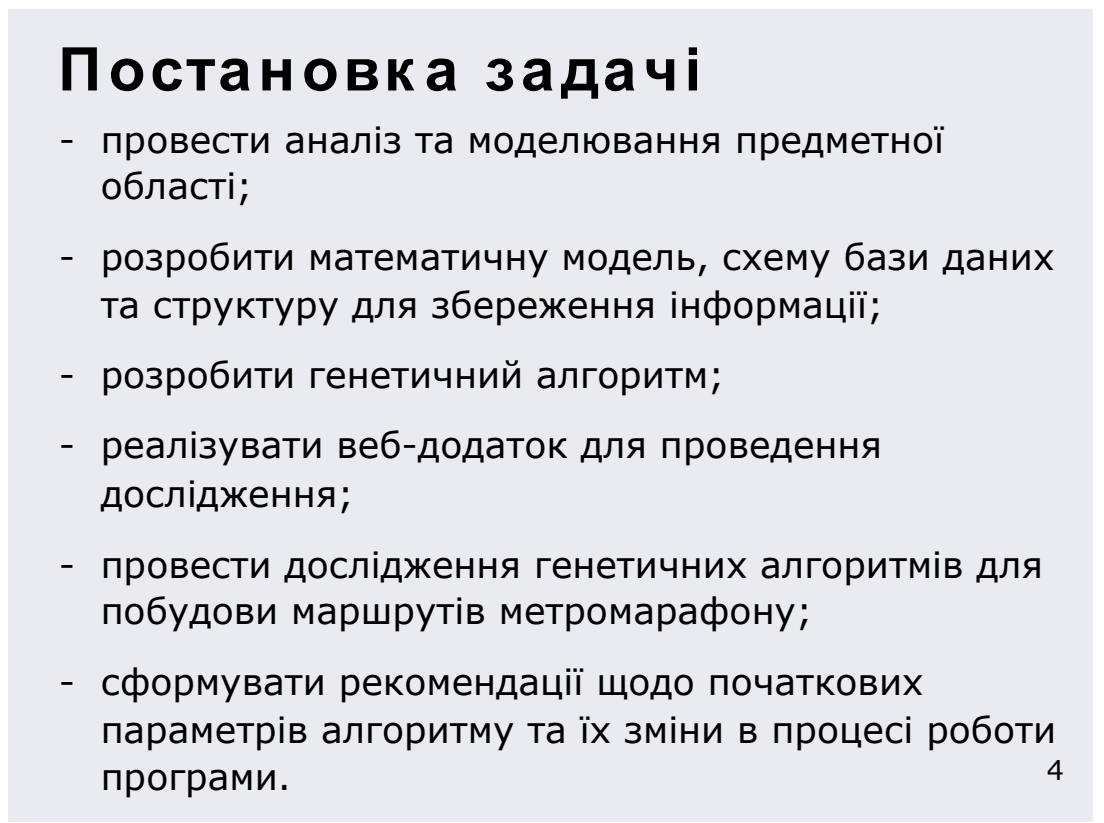
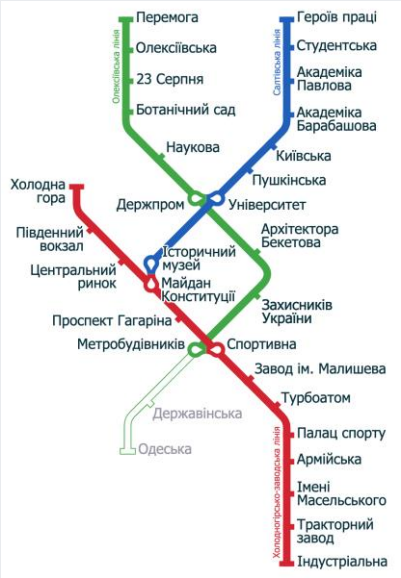
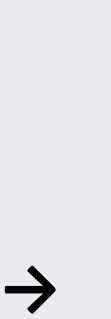
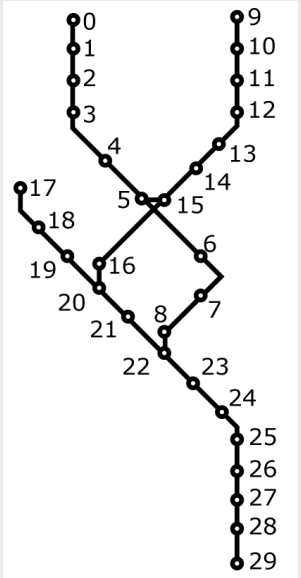


Рисунок А.4 – Слайд «Постановка задачі»

Аналіз предметної області







Харківський метрополітен
30 станцій, 3 переходи

Граф G = (V, A)
30 вершин, 29 ребер

5

Рисунок А.5 – Слайд «Аналіз предметної області»

Математична модель

G = (V, A) – граф метрополітену з вагами дуги t(a), де t(a) > 0 та t(a) Q. Проблема станцій полягає в тому, щоб знайти найдешевший маршрут T, де кожній станції s S (набір станцій) принаймні один вузол в V_S належить T.

$$\min \sum_{a \in A} t_a x_a$$

де *a* – певна дуга (*a* A)
t(a) – функція розрахунку ваги ребра a
x(a) – значення з множини {1,0}, (враховується чи належить ребро до оптимального маршруту)

6

Рисунок А.6 – Слайд «Математична модель»

Математична модель

Обмеження

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 \qquad \forall v \in V$$

Гарантує, що рішення складається тільки з набору підмаршрутів. Де δ+(v) – набір дуг без вершини v, а δ-(v) – набір дуг з вершиною v.

$$\sum_{a \in \delta^+(V_s)} x_a \geq 1 \qquad \forall s \in S$$

Кожна станція відвідується принаймні один раз в деякому підмаршруті

$$\sum_{a \in \delta^+(C)} x_a + \sum_{a \in \delta^-(C)} x_a \geq 2 \qquad \forall C \subset V, C$$

$$x_a \in \mathbb{N}_{\geq 0} \qquad \forall a \in A$$

Гарантує, що кожна станція з'єднана з усіма іншими станціями.

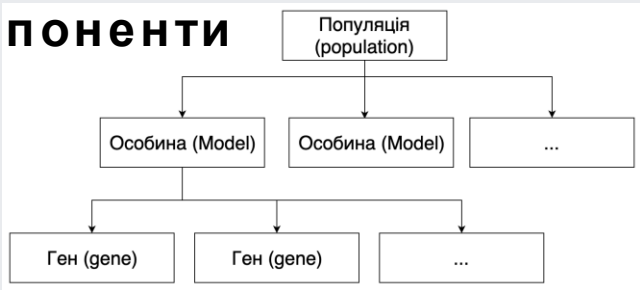
7

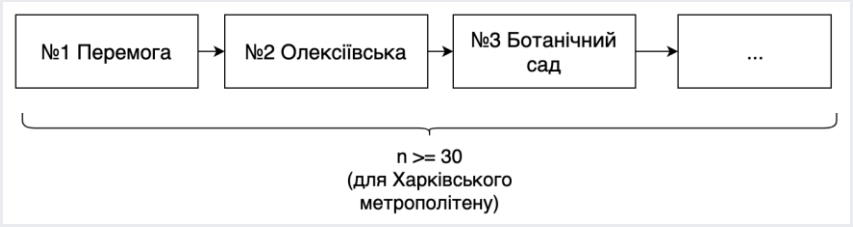
Рисунок А.3 – Слайд «Математична модель»

Рисунок А.3 – Слайд «Загальна схема генетичного алгоритму»

Необхідні компоненти

- популяція;
- особина;
- ген;





9

Рисунок А.9 – Слайд «Необхідні компоненти»

Функція пристосованості

$$f_i = f(G_i) < f_j = f(G_j)$$

Якщо значення функції пристосованості для хромосоми G_i буде меншим за значення функції для хромосоми G_j, то таку хромосому можна вважати більш пристосованою.

10

Рисунок А.10 – Слайд «Функція пристосованості»

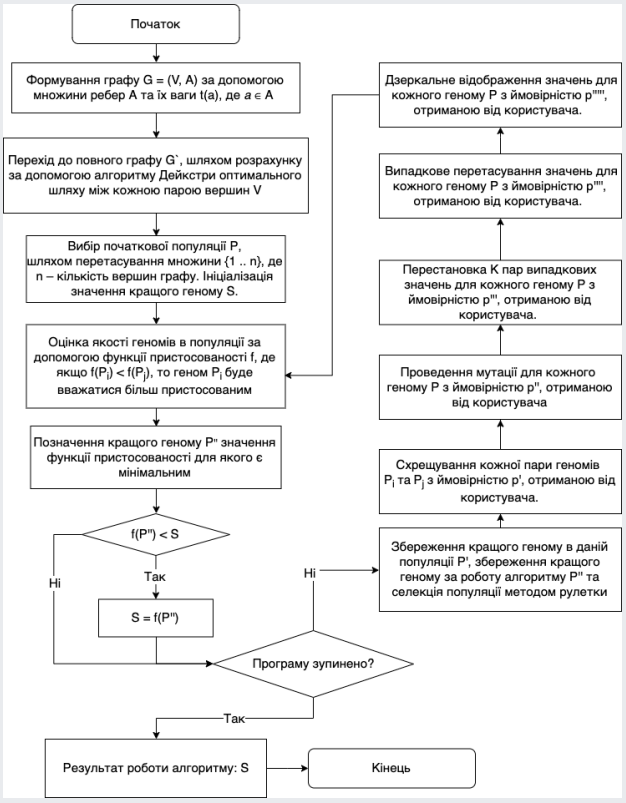
Функції мутації та схрещування

- випадкове перетасування випадкового інтервалу;
- дзеркальне відображення випадкового інтервалу;
- перестановка K пар випадкових значень місцями;
- схрещування або «crossover» двох окремих геномів.

Схема адаптації алгоритму для метромарафону 11

Рисунок А.11 – Слайд «Функції мутації та схрещування»

Схема розробленого алгоритму



12

Рисунок А.12 – Слайд «Схема розробленого алгоритму»

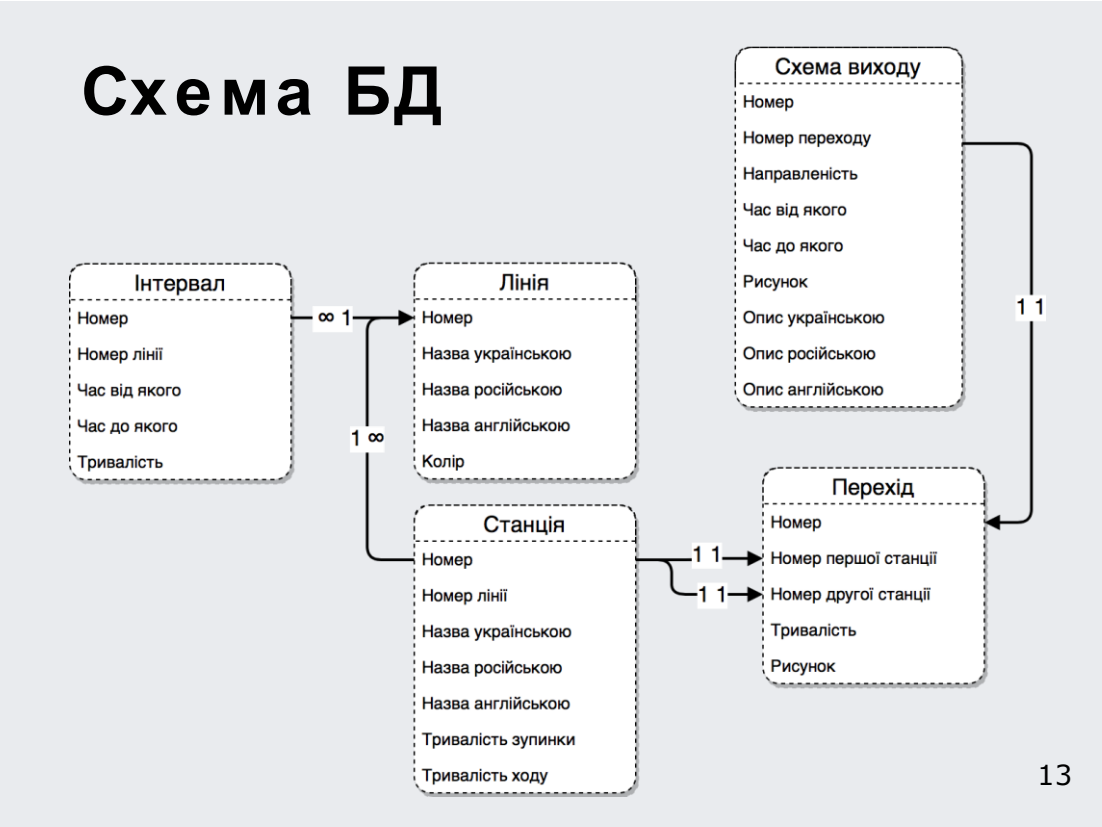


Рисунок А.13 – Слайд «Схема БД»

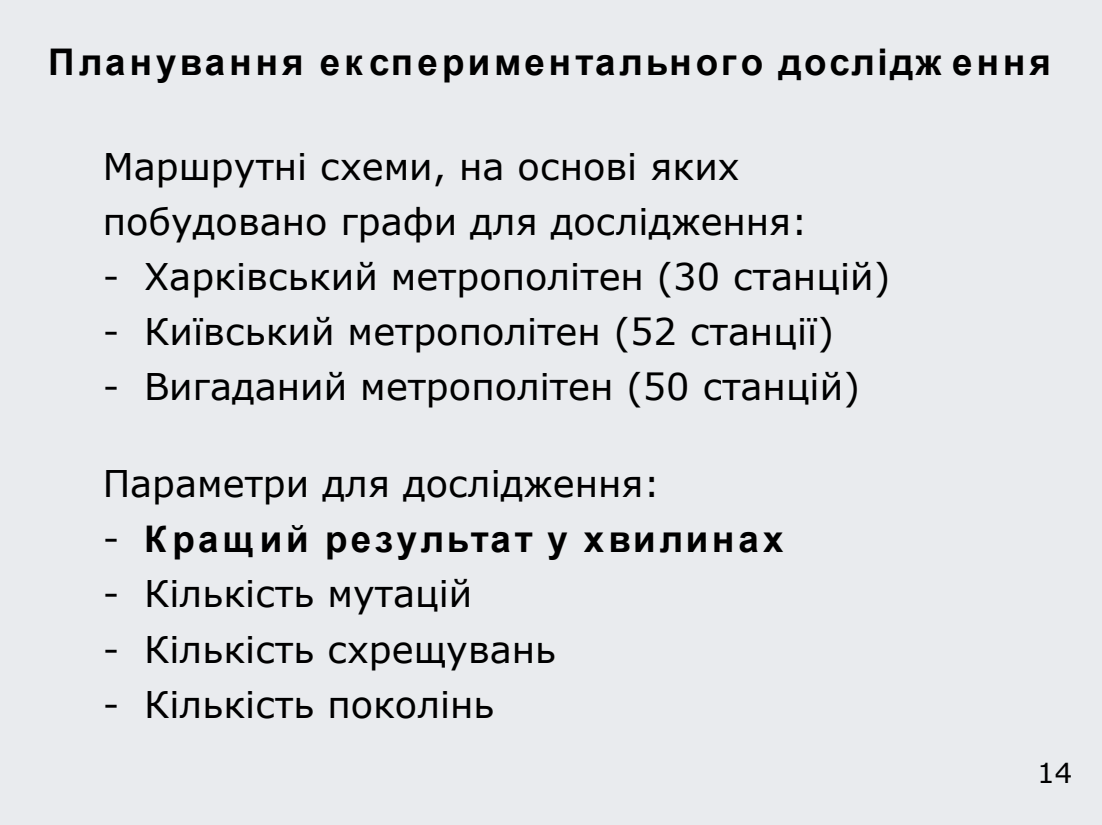


Рисунок А.14 – Слайд «Планування експериментального дослідження»

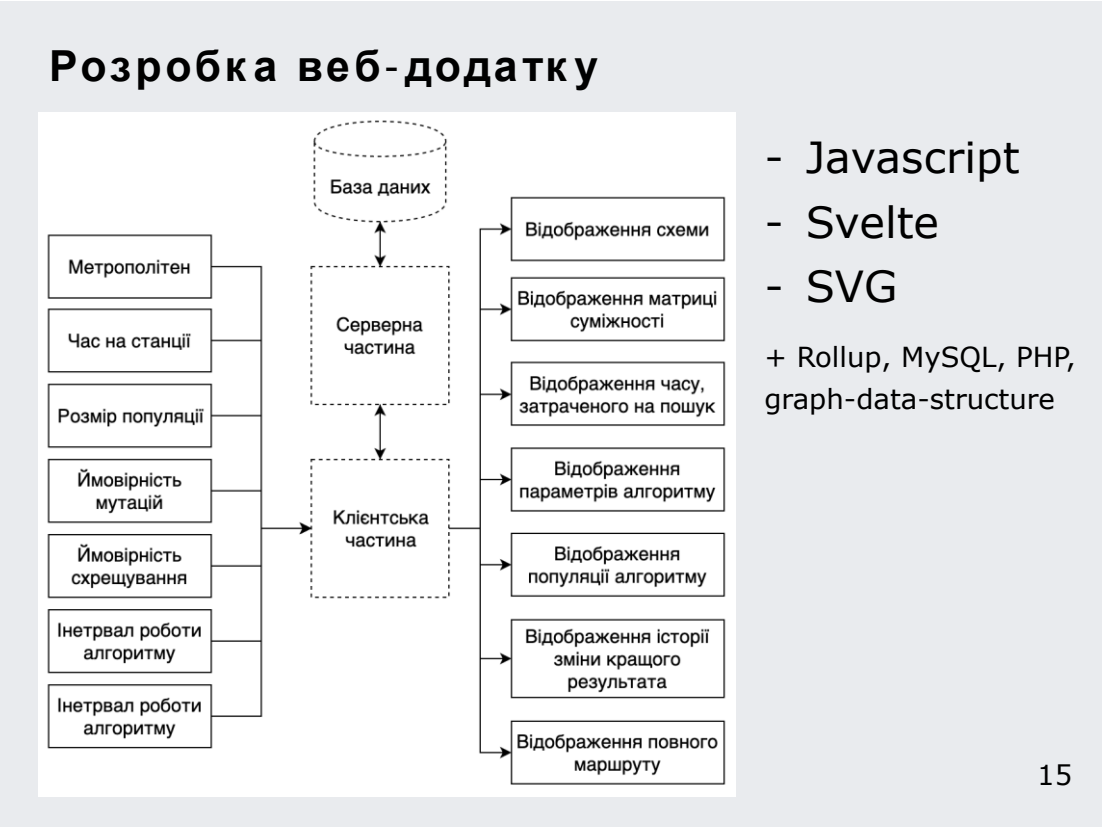


Рисунок А.15 – Слайд «Розробка веб-додатку»

Рисунок А.16 – Слайд «Стартовий екран веб-додатку»

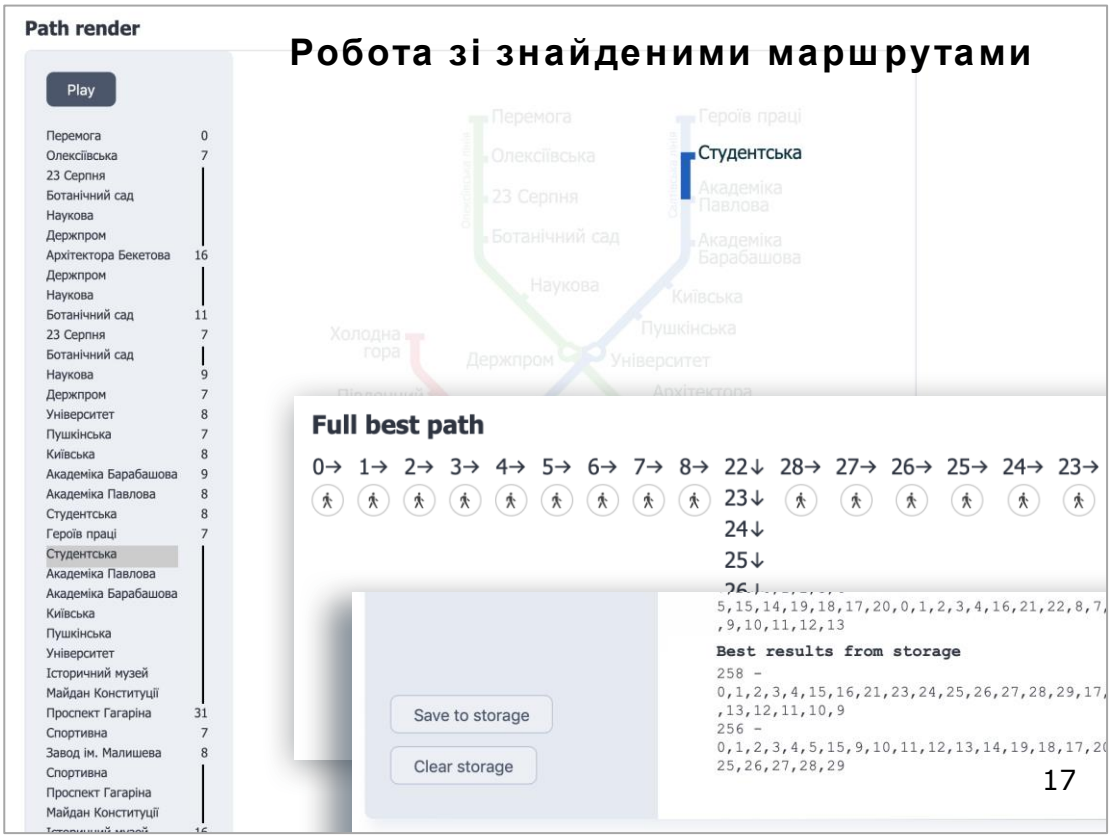


Рисунок А.17 – Слайд «Робота зі знайденими маршрутами»

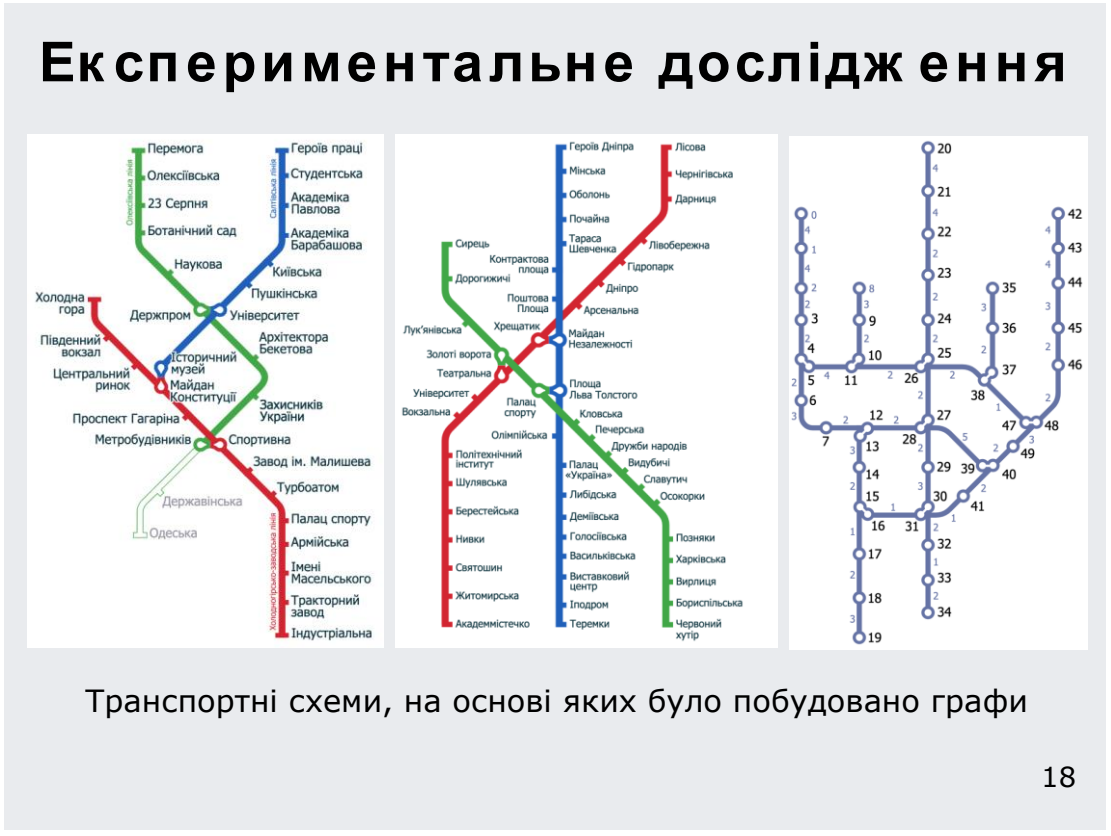


Рисунок А.18 – Слайд «Експериментальне дослідження»

Визначення часових параметрів

1 хв. на запуск

3 спроби

50 мс. ітерація

5 хв. на станції

Динаміка пошуку результату для Харківського метрополітену

19

Рисунок А.19 – Слайд «Визначення часових параметрів»

Визначення параметрів для дослідження

№	Ймовірності або їх зміна					К-сть особин в популяції, шт
	Мутації взагалі	Мутації «Перестановка пар»	Мутації випадкового перетасування	Мутації дзеркального відображення	Схрещування	
1	95%	10%	10%	5%	1%	20
2	5%	5%	5%	5%	98%	20
3	10%	10%	10%	5%	80%	50
4	10%	5% → 30%	30% → 5%	5%	10%→80%	20
5	10%	30% → 5%	5% → 30%	5%	80%→10%	20

Набори параметрів та їх зміни для тестування

20

Рисунок А.20 – Слайд «Визначення параметрів для дослідження»

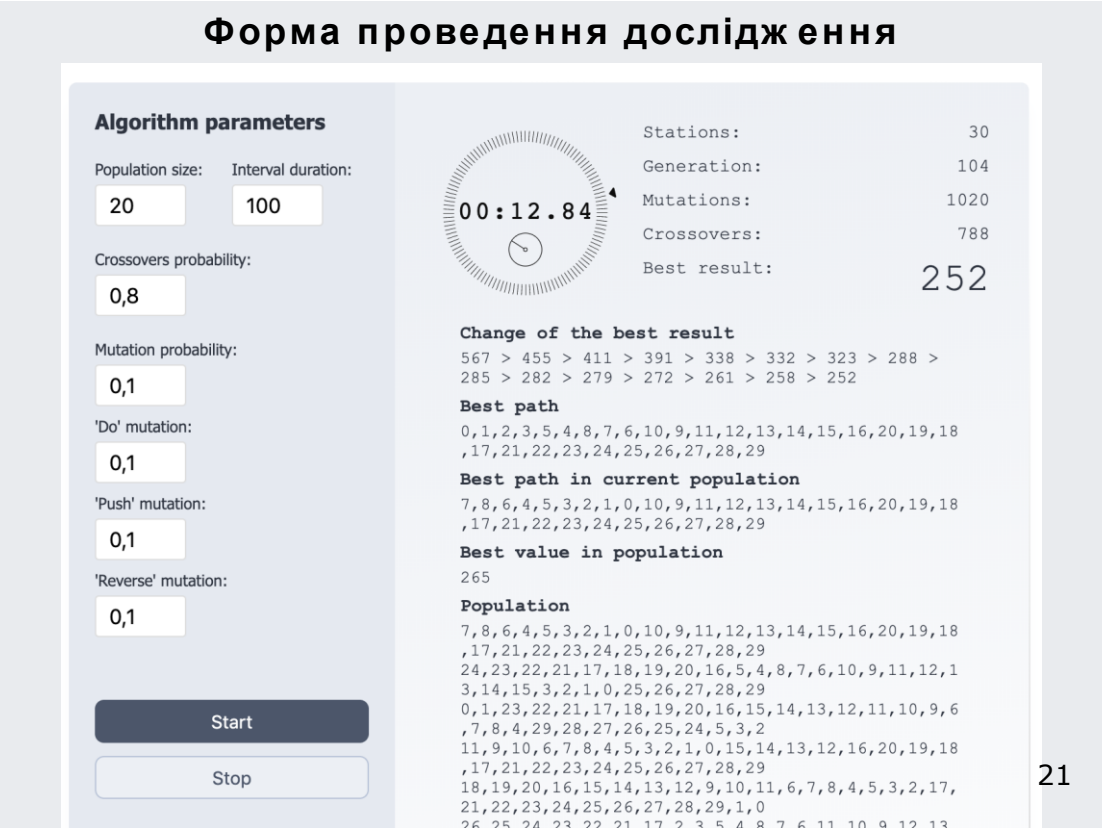


Рисунок А.21 – Слайд «Форма проведення дослідження»

Показник	Експеримент номер				
	1	2	3	4	5
Кращий результат, хв	314	252	262	242	258
К-сть мутацій	126314	1812	2545	2540	2766
К-сть схрещувань	254	2799	245	1747	1473
К-сть поколінь	305	299	308	301	310

Показник	Експеримент номер				
	1	2	3	4	5
Кращий результат, хв	657	430	445	414	433
К-сть мутацій	152980	1103	5544	1664	2781
К-сть схрещувань	80	3029	6176	2697	2340
К-сть поколінь	308	309	311	303	301

Показник	Експеримент номер				
	1	2	3	4	5
Кращий результат, хв	572	415	409	409	421
К-сть мутацій	15079	1271	5419	1245	1775
К-сть схрещувань	224	5513	5914	2854	2748
К-сть поколінь	310	306	301	307	313

Рисунок А.22 – Слайд «Результати експерименту»

Другий етап дослідження

Було використано набір параметрів №3

Ймовірності					К-сть особин в популяції, шт
Мутації взагалі	Мутації «Переста-новка пар»	Мутації випадкового перетасування	Мутації дзеркального відображення	Схрещу-вання	
10%	10%	10%	5%	80%	50

Серія №1 – без зміни параметрів

Серія №2 – з ручним керуванням параметрів

Після 30 сек. стагнації результату
змінити ймовірності:
дзеркального відображення — 5% → 30%
схрещування — 80% → 98%

23

Рисунок А.23 – Слайд «Другий етап дослідження»

Результати другого етапу експерименту

Показник	Результат
Кращий результат, хв	427
К-сть мутацій	20549
К-сть схрещувань	10673
К-сть поколінь	2259

Без зміни параметрів

Показник	Результат
Кращий результат, хв	410
К-сть мутацій	42367
К-сть схрещувань	20727
К-сть поколінь	2162

Зі зміною параметрів

Різниця кращого результату – 17 хвилин

24

Рисунок А.24 – Слайд «Результати другого етапу експерименту»

Рекомендації до алгоритму

- не задавайте ймовірність мутацій більшу за 10%;
- використовуйте мутацію з випадковим перетасуванням геному на початку роботи алгоритму;
- зменшуйте ймовірність мутації з випадковим перетасуванням, коли значення перестає швидко змінюватися, та збільшуйте ймовірність мутації з перестановкою пар;
- якщо значення довго перебуває в локальному оптимумі (30 сек та більше), збільшіть ймовірність мутації з дзеркальним відображенням або\та ймовірність схрещування.

25

Рисунок А.25 – Слайд «Рекомендації до алгоритму»

Висновки

- проведено аналіз та моделювання предметної області;
- розроблено математичну модель, схему бази даних та структуру для збереження інформації;
- розроблено генетичний алгоритм, адаптований до пошуку оптимальних шляхів для метромарафону;
- реалізовано веб-додаток для проведення досліджень;
- проведено дослідження генетичних алгоритмів на трьох графах, які моделюють транспортні схеми;
- сформовано рекомендації щодо початкових параметрів алгоритму та їх зміни в процесі роботи програми;
- зроблено доповідь на XXII міжнародному молодіжному форуму та написано статтю до науково-технічного журналу «Біоніка інтелекту».

26

Рисунок А.26 – Слайд «Висновки»

ПРОГРАММНАЯ СИСТЕМА НАВИГАЦИИ В ХАРЬКОВСКОМ МЕТРОПОЛИТЕНЕ.

Гордиенко Т.А.
Научный руководитель – к.т.н., доцент Мазурова О.А.
Харьковский национальный университет радиоэлектроники
(61166, Харьков, пр. Науки,14, каф. Программной инженерии,
тел. (057) 702-14-46)
e-mail: goldsteals@gmail.com; телефон (097) 339-46-29

Developed software system builds optimal routes between Kharkiv metro stations. There is an opportunity to share a route and find an alternative one. System computes time of route and shows at what time user needs to enter the subway to get to the desired/chosen/preset/determined station. Application will be useful for people who do not want to leave their house too early. The system is written in form of web-application using javascript and works completely on the client side.

Проблема ориентирования в большом городе — одна из проблем повседневной жизни, знакомая большинству горожан. В больших городах городской транспорт имеет свою развитую и достаточно сложную структуру. Особенно сложной является проблема ориентирования в метрополитене. Последнее десятилетие современные технологии частично решают эту проблему. Проблему ориентирования в метро по маршруту и времени частично решают существующие программные решения в виде навигаторов. Навигаторы автоматически определяют геопозицию пассажира, позволяют построить маршрут между двумя точками, и в режиме реального времени показывают передвижения по маршруту.

В больших городах вроде Нью-Йорка, Москвы или Лондона городской транспорт имеет свою мощную экосистему, в которую входит разработка и поддержка приложений для ориентирования. По статистике 2012 года, дневной пассажиропоток Харьковского метро превышает полмиллиона человек. Однако в Харькове городскому транспорту не уделяется должного внимания. Решение от официального сайта Харьковского метро лишь частично решает проблему ориентирования в метро по маршруту и времени. Информативность в приложении «Яндекс.Метро» у Харькова сильно урезана по сравнению с Москвой или Санкт-Петербургом.

Таким образом, разработка приложения для ориентирования в Харьковском метро является актуальной. Такое приложение должно позволить человеку не только построить маршрут, но и оценить время, которое нужно будет потратить на него. Приложение должно учитывать изменение интервалов поездов и показать пользователю оптимальное поведение на станциях и переходах.

Для моделирования предметной области передвижения метрополитеном были разработаны:

– реляционная база данных, которая содержит информацию о линиях метро, станциях, переходах, интервалах и выходах из вагонов;

– графовая модель схемы метрополитена и алгоритм отрисовки схемы на странице в браузере;

– оптимизационная модель выбора наилучшего пути передвижения с учетом интервалов поездов [2].

Разработанная графовая модель метрополитена моделирует структуру метрополитена с учетом времени, за которое поезд проезжает перегон. Алгоритм отрисовки схемы в браузере основан на многослойности и позволяет, не изменяя структуры самой схемы, добавить к SVG представлению схемы слой-подложку и слой с найденным маршрутом.

Для нахождения оптимального пути передвижения был реализован и модифицирован алгоритм Дейкстры [1], который учитывает:

– время, которое поезд движется по перегону;

– время, которое пассажир будет ожидать поезда;

– время, которое пассажир потратит на переход между станциями.

На основании разработанных моделей и алгоритмов разработана программная система, которая позволяет пользователю:

– получить оптимальный маршрут между станциями;

– получить альтернативные маршруты;

– узнать время поездки с учетом интервалов поездов,

– узнать расположение двери вагонов, в которые лучше зайти, чтобы выйти сразу на переход.

Программная система реализована в виде веб-приложения и адаптирована для мобильных устройств. Так как вся информация кешируется и обрабатывается на клиентской стороне, приложением можно пользоваться, один раз загрузив его в браузере. База данных реализована с помощью СУБД MySQL.

Новизна разработанной системы заключается в расчете времени поездки с учетом интервалов поездов, алгоритме отрисовки схемы на странице браузера и информационной наполненности, которой не обладают аналоги. Таким образом, разработанная система позволит пассажирам узнать как и за сколько времени доехать до определенной станции быстрее и эффективнее.

Список литературы:

1. Томас Х. Кормен, Чарльз И. Лейзерсон. Алгоритмы: построение и анализ. — 2-е изд. — М.: Вильямс, 2006. — 1296 с.

2. Гребеннік І. В. Методи підтримки прийняття рішень : навч. посібник / І. В. Гребеннік, Т. Є. Романова, А. Д. Тевяшев, Г. М Яськов ; МОН України, Харк. нац. ун-т радіоелектроніки. — Харків : ХНУРЕ, 2010. — 127 с.

УДК 004.421.2:519.174.2

Т. О. Гордієнко, О.О. Мазурова[1]

[1]*Харьківский національний університет радіоелектроніки*

ДОСЛІДЖЕННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ ДЛЯ ПОШУКУ ОПТИМАЛЬНИХ ШЛЯХІВ В СИСТЕМІ ПРОВЕДЕННЯ МЕТРОМАРАФОНІВ

Робота присвячена дослідженню генетичних алгоритмів на прикладі пошуку оптимальних шляхів для підтримки проведення метромарафонів в системах типу «Transit Challenge» або «Subway Challenge». На основі правил проведення метромарафонів та задачі комівояжера сформульована проблема станцій — пошуку оптимального шляху, який би дозволив відвідати всі станції метрополітену за найкоротший час. На На базі теорії графів розроблена математична модель системи метромарофону. Для вирішення проблеми станцій розроблено генетичний алгоритм: обрано спосіб представлення геному, правила мутації популяцій і схрещування геномів. На основі експериментального дослідження генетичного алгоритму підібрано найбільш ефективні параметри та розроблено рекомендації стосовно вирішення проблеми станцій для метромарафонів різної розмірності.

Ключові слова: *генетичний алгоритм, граф, задача комівояжера, математична модель, пошук оптимального шляху, проблема станцій, схрещування геномів.*

Постановка проблеми

Метро – це досить популярний засіб пересування у великих містах. З недавніх пір метро стало визивати навіть певний спортивний інтерес. Так у багатьох країнах світу серед мандрівників та просто зацікавлених у такого роду змаганнях популярним став сценарій використання метро в рамках проведення метромарафонів.

Це змагання називають по-різному: «Transit Challenge» чи «Subway Challenge». В країнах СНГ воно більш відомо як метромарафон. Суть метромарафону проста: відвідати всі станції метрополітену за найкоротший час. Існує навіть сайт rapidtransitchallenge.com, де зафіксовані правила та рекорди цього змагання.

Перед тим як почати змагання учасники заздалегідь планують маршрут та час, який буде затрачено. Складність цього процесу в тому, що чим більший метрополітен, тим більше різних маршрутів існує. А врахування навіть таких незначних деталей, як інтервали між електропоїздами в різний час, час зупинки на станціях та завантаженість, може дати виграш у часі під час таких змагань.

В Україні є два великих метрополітени – Харківський та Київський, які мають 30 та 52 станції відповідно. Але в Україні поки що такі змагання не проводилися, відсутні також публікації стосовно результатів побудови відповідних маршрутів. Отже, існує практична задача створення системи «Transit Challenge» для проведення українських метромарафонів, та відповідно наукова проблема побудови найкращого маршруту для них.

Аналіз основних досліджень

З метою пошуку шляхів метро краще за все моделюють у вигляді графа, де вершини графу – станції, а перегони – ребра. Методів рішення такого роду задач в теорії графів існує багато. В випадку проведення метромарафону, де потрібно знайти маршрут, який включає всі станції, задача зводиться до задачі комівояжера [4]. Вона має досить високу складність обчислення, якщо знаходити її точний розв'язок. Тому часто використовують евристичні методи, які хоч і не дозволяють знайти оптимальний маршрут, але дають гарний компроміс між оптимальністтю маршруту та затраченим часом на його пошук.

Деякі з цих евристичних методів намагаються скоротити вже побудований маршрут, видаляючи з маршруту групи ребер та замінюючи їх іншими, що дозволяє отримати новий маршрут. Інші методи комбінують пошук глобальних та локальних розв'язків.

Такі методи як, мурашиний алгоритм, генетичні алгоритми та нейронні мережі використовують ідеї природних процесів та показують досить високу ефективність [2].

Постановка задачі

Отже, на основі теорії графів необхідно розробити математичну модель системи метромарафону, яка буде охоплювати всю предметну область разом з існуючими правилами, а також розробити генетичний алгоритм пошуку маршруту для участі в метромарафоні. Також необхідно спланувати та провести експериментальне дослідження алгоритму, виробити рекомендації стосовно параметрів алгоритму та його використання на моделях метромарафону різної розмірності.

Розробка математичної моделі системи метромарафону

При моделюванні систему метрополітену доцільно представити у вигляді графів, де вершина – це станція, а дуга – це перехід чи перегон між станціям. Середній час, необхідний для переходу з однієї вершини до іншої, моделюється як вага дуги. Оскільки метрополітен – це дуже налагоджена система, яка працює по чіткому графіку, то такий середній час наближається до фактичного часу здійснення кожного перегону метрополітену.

З пошуком оптимального шляху по всім станціям метрополітену тісно пов'язана проблема комівояжера, так звана (Traveling Salesman Problem, TSP), а також її узагальнена версія Generalized Traveling Salesman Problem (GTSP). В них враховується, що потрібно відвідати всі вершини графа. А в нашому випадку ще й не має ніякого значення, яка лінія та напрямок використовуються для того, щоб відвідати станцію [3].

Якщо представити транспортну мережу у вигляді певного графу, то отримаємо наступну структуру:

- скінченний набір станцій, що позначимо за S.
- скінченний набір ліній метрополітену L, де кожна лінія $l \in L$ – це впорядкований набір станцій.

Припустимо, що кожна лінія працює в обох напрямках і що кожна лінія складається щонайменше з двох станцій. Отже, тепер систему можна змоделювати як орієнтований граф G = (V, A), де V – множина вершин, які зв'язані між собою множиною ребер A. Цей граф має такі властивості:

- для кожної лінії $l \in L$ і всіх станцій s (s∈ L) завжди є початкова вершина $\vec{v}(s, L) \in V$;
- для кожної лінії $l \in L$ і всіх станцій s (s∈ L) завжди є кінцева вершина $\overleftarrow{v}(s, L) \in V$.

Визначимо множину всіх вершин s ∈ S, що належать станції, як:

$V(s) = US(L \in L, s \in L)\{\vec{v}(s, L), \overleftarrow{v}(s, L)\}$.

Позначимо дуги, що відповідають сегментам, які утворюють перегін між двома станціями, як $(\vec{v}(s, L), \overleftarrow{v}(s`, L)$, де s` наслідує s для всіх s ∈ L. Аналогічно позначимо дугу $(\vec{v}(s, L), \overleftarrow{v}(s``, L))$, де навпаки s наслідує s``. Позначимо як R набір дуг, що з'єднує вершини тільки однієї лінії та напрямку.

Для моделювання переходів додамо дуги (u, v) для кожної можливої пари вершин у кожному V(s). Такі дуги дозволять змінювати напрямок.

Кожна дуга a ∈ A має певну вагу t(a) > 0. Якщо a відноситься до R, то будемо вважати, що це час в дорозі між станціями, а якщо a R, то це відповідає часу, необхідному на перехід до іншої станції. Отриманий граф назвемо метрографом (графом метрополітену) від L та S.

На основі такої моделі мтеромарафону можемо сформулювати проблему станцій та проблему перегонів. В рамках предметної області більш цікава проблема станцій, яка полягає в тому, що потрібно відвідати всі вершини змодельованого графу, саме вона задовольняє умовам, при яких має значення відвідування кожної станції, а не проїзд по всім перегонам на лінії.

Користуючись моделлю метромарафону сформулюємо проблему станцій наступним чином: нехай G = (V, A) є графом метрополітену з вагами дуги t(a), де t(a)>0 та t(a) ∈ Q, тоді необхідно знайти найдешевший маршрут T, де кожній станції s ∈ S принаймні один вузол в V_S належить T.

Проблема станцій дуже тісно пов'язана з асиметричним випадком задачі комівояжера і є NP-повною. Математична модель такої задачі буде виглядати таким чином:

$$\min \sum_{a \in A} t_a x_a$$

де a – певна дуга, що a ∈ A, t(a) – функція розрахунку ваги ребра a, x(a) – значення з множини {1, 0}, при якому враховується чи належить дане ребро до оптимального маршруту чи ні.

Також за умовами проблеми станцій ми повинні додати до математичної моделі певні обмеження. Обмеження

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0 \qquad \forall v \in V$$

гарантує, що рішення складається тільки з набору підмаршрутів. Де δ+(*V*) – набір дуг без вершини v, а δ - (*V*) – набір дуг з вершиною v.

Завдяки другому обмеженню

$$\sum_{a \in \delta^+(V_s)} x_a \geq 1 \qquad \forall s \in S$$

кожна станція відвідується принаймні один раз в деякому під маршруті, де S – це множина станцій.

Обмеження

$$\sum_{a \in \delta^+(C)} x_a + \sum_{a \in \delta^-(C)} x_a \geq 2 \qquad \forall C \subset V, C$$

$$x_a \in \mathbb{N}_{\geq 0} \qquad \forall a \in A$$

гарантує, що кожна станція з'єднана з усіма іншими станціями. Тобто кожен елемент множини станцій повинен мати хоча б по одному зв'язку в кожному з напрямків.

Розглянемо варіанти функції t, яка відповідає за розрахунок ваги кожної з дуг:

- для розрахунку часу на перегоні між двома станціями використаємо такий варіант:

t = K + F, K ∈ N≥0, F ∈ N≥0,

де K - час, який витрачає потяг на подолання перегону між станціями у хвилинах, F – інтервал руху поїздів в певний час доби (у хвилинах);

- для розрахунку на перехід між двома лініями:

t = J + F, J ∈ N ≥ 0,

де J – це час на перехід між станціями; слід зазначити, що він може відрізнятися в залежності від того, в якому напрямку необхідно перейти зі станції на станцію.

Розробка генетичного алгоритму для пошуку оптимального маршруту

Задача коммівояжера є транс обчислювальною та відноситься до класу NP-повних задач. Для Харківського метрополітену кількість можливих рішень складає 2.6525286×10^{32} при кількості станцій 30, а для Київського це число набагато більше – 8.0658175×10^{67} на 52 станціях. Якщо уявити, що комп'ютер здатний аналізувати по 1×10^{23} рішення в секунду, то на розрахунок оптимального шляху знадобиться щонайменше 84 роки.

Ідея генетичних алгоритмів вперше була представлена у 1960 році як техніка оптимізації, яка наближає результат до оптимального рішення [5]. Отже ідея використати генетичні алгоритми для рішення задачі комівояжера є цілком перспективною.

Для застосування генетичного алгоритму потрібно описати основні параметри та функції, які буде застосовано в процесі його виконання. До основних складових алгоритму належать: популяція, особина, геном та хромосома. Після визначення їх представлення потрібно сформулювати основні функції генетичного алгоритму: функцію відбору, пристосованості, схрещування та функції мутації.

Кожне рішення проблеми метромарофону можна представити як вектор P, що є набором всіх станцій в порядку їх слідування в маршруті. Тоді в алгоритмі вектор P буде грати роль геному або, інакше кажучи, набору хромосом $\{P_1 \ldots P_n\}$. Але таке рішення приховує декілька неприємних моментів. По-перше, при такому підході ми отримаємо вектори різної довжини, які буде дуже складно схрещувати між собою, а по-друге, багато векторів буде невалідними. В результаті кожен з геномів потрібно буде перевіряти на валідність, що збільшить складність алгоритму.

Кращим рішенням буде взяти на роль геному перестановку множини станцій метрополітену. Для Харківського метрополітену це перестановка 30. Але при такому підході поряд може опинитися пара станцій між якими немає прямого маршруту. Для вирішення цієї проблеми було вирішено перейти до повного графу. Для цього потрібно буде розрахувати найкоротші маршрути між кожною парою станцій. Складність такого розрахунку висока $O(n^3)$, але для Харківського метрополітену це всього лише 27000 варіантів.

Функцією пристосованості має бути цільова функція, яку можна застосувати в ролі порівняльного показника якості. В даній проблемі ним є час, який буде затрачено на подолання даного маршруту.

Нехай $Gi = \{g_{ik}\}$ $(k = \overline{1,n})$ - хромосомі і-й особини, де g_{ik} значення k-го гена на і-й особині, а n - кількість генів в хромосомі. Тоді при можемо сформулювати умову пристосованості хромосоми:

$$f_i = f(G_i) < f_j = f(G_j),$$

а саме: якщо значення функції пристосованості для хромосоми G_i буде меншим за значення функції для хромосоми G_j, то таку хромосому можна вважати більш пристосованою.

Для розробленого алгоритму було обрано наступні функції мутації, які правильно сходяться між собою та дають змогу вибиратися з локальних оптимумів:

- випадкове перетасування випадкового інтервалу генома; ця мутація є швидкою, але не дуже акуратною; вирішено використовувати її на ранніх етапах еволюції, оскільки в фіналі дуже низька ймовірність поліпшення рішення;
- дзеркальне відображення випадкового інтервалу генома; ця мутація буде використовуватися для виходу з локального оптимуму;
- перестановка K пар випадкових значень в геномі місцями; ця досить непогана, але повільна мутація буде ефективно використовуватися ближче до завершення еволюції;
- схрещування двох окремих геномів; в цій мутації будемо обирати точку на геномі, розрізатимемо по ній обидва геноми та мінятимемо їх місцями.

В результаті на кожній ітерації ми застосовуємо ті чи інші мутації і формуємо з їх нащадків нове покоління, додаючи до них декілька випадкових маршрутів (схему алгоритму див. на рис. 1).

Дослідження генетичного алгоритму

Для дослідження розробленого генетичного алгоритму використано наступні графи:

- граф на основі маршрутної схеми Харківського метрополітену (30 вершин);
- граф на основі маршрутної схеми Київського метрополітену (52 вершини);
- граф, який імітує транспортну схему з 10 переходами та 50 вершинами, що дозволив дослідити алгоритм в складніших умовах.

Під час дослідження вирішено змінювати наступні параметри алгоритму:

- кількість особин в популяції – g;
- ймовірність схрещування геномів – p';
- ймовірність проведення мутації – p'' ;
- ймовірність застосування мутації перестановки декількох пар (p''')? випадкового перетасування (p'''') та дзеркального відображення (p''''').

Алгоритм було протестовано на 5 наборах даних по 3 рази на кожному графі. Кожен тест тривав одну хвилину. В першому наборі даних ймовірність мутації встановлено на 95%. Цей набір має на меті перевірити залежність результату від кількості мутацій. Другий набір даних більш стандартний для такого роду задач – невелика ймовірність мутацій (5%) та велика схрещування (95%). Третій набір має на меті перевірити роботу з популяцією в 50 особин, тобто більше чим в 2 рази більшою, чим в інших наборах. Четвертий набір базується на зміні ймовірності мутацій і схрещування в процесі роботи програми. Збільшення ймовірності мутації з перестановкою пар та зменшення з випадковим перетасуванням, а також збільшення кількості схрещувань. У п'ятому наборі стратегія зміни параметрів повністю протилежна попередньому пункту.

Нижче наведено таблицю з набором параметрів для проведення експерименту.

Ймовірність/№	1	2	3	4	5
Мутації взагалі	95%	5%	10%	10%	10%
Мутації з перестан. пар	10%	5%	10%	5→30%	30→5%
Мутації з перетасуванням	10%	5%	10%	30→5%	5→30%
Мутації дзерк. відображення	5%	5%	5%	5%	5%
Схрещування	1%	98%	80%	10→80%	80→10%
К-сть особин в популяції, шт	20	20	50	20	20

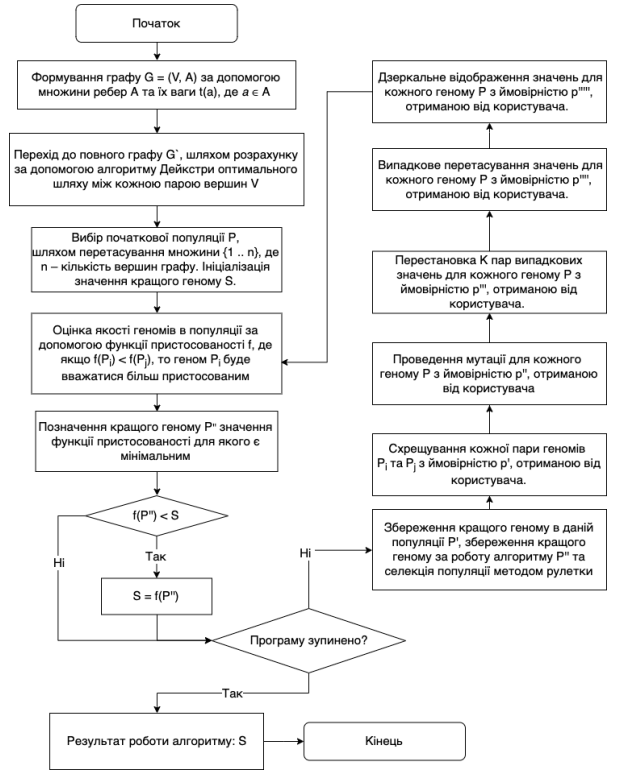


Рисунок 1 – Схема розробленого генетичного алгоритму

Кращі результати дослідження (виміряються в хвилина проходження оптимального маршруту) на всіх трьох графах наведено в таблиці нижче.

Граф	Експеримент номер				
	1	2	3	4	5
Харківський	314	252	262	242	258
Київський	572	415	409	409	421
Вигаданий	657	430	445	414	433

Аналізуючи на порівнюючи отримані дані, можна зробити висновок, що найоптимальнішою виявився набір параметрів номер 4, як і було прогнозовано.Набір даних №1, де ймовірність мутації була найвищою, показав найгірший результат. Це пояснюється тим, що особини дуже швидко мутували, не встигаючі обмінятися між собою генами.

Другою частиною експерименту є порівняння результату звичайної роботи алгоритму з початковими параметрами та роботи з ручним коректуванням параметрів. Очікувалось невелике зменшення фінального результату при ручному коректуванні параметрів. Тестування було проведено з третім графом, що має 50 вершин та 10 переходів.

Кращі результати другого етапу наведено в таблиці нижче.

Вид тестування	Експеримент номер	
	1	2
Без коректування	427	421
З коректуваннями	410	409

Отже, зміна параметрів хоч і не суттєво, але впливає на фінальний результат. В результаті в кожному експерименті був знайдений більш оптимальний маршрут.

Отже, під час вирішенні проблеми метромарафону за допомогою генетичних алгоритмів не рекомендовано задавати високу ймовірність мутацій або змінювати її за межі 10 відсотків. Мутація з випадковою перестановкою краще працює на початку роботи алгоритму. Після швидкої зміни значень варто спробувати зменшити її ймовірність та збільшити ймовірність мутації з перестановкою пар. Якщо значення довго перебуває в локальному оптимумі, варто спробувати збільшити вірогідність мутації з дзеркальним відображенням або ймовірність схрещування.

Висновки та перспективи

В роботі на основі теорії графів запропоновано математичну модель системи метромарафону та сформульовано проблему станцій як проблему пошуку оптимального маршруту для метромарафону, який би охопив всі стації метрополітену. Для зниження складності вирішення цієї NP-повною задачі було запропоновано пошук оптимального маршруту за допомогою генетичних алгоритмів.

В розробленому алгоритмі в якості геному було взято перестановку по кількості станцій метрополітену та перейдено до повного графу, сформульовано правила схрещування геномів між собою, що дозволяють вибиратися з локальних оптимумів.

Результати експериментального дослідження дозволили знайти оптимальні значення для параметрів алгоритму, що дозволяє знайти досить оптимальний маршрут в метрополітені за більш-менш реальний час, не застосовуючи повний перебір варіантів.

Список літератури

1. Baaj, M. H., and Mahmassani, H. S. An AI-based approach for transit route system design. Journal of Advanced Transportation. 25(2), 287 c. — 1990.
2. Chakroborty, P., and Dwivedi, T. Optimal route network design for transit system using genetic algorithms. 34(1), 200 c. — 2003.
3. Larrañaga P., Kuijpers C. M. H., Murga R. H. Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artificial Intelligence Review: SCOPUS, 1999 – vol. 13, no. 2, pp. 129–170.
4. Nagata Y. and Soler D. A new genetic algorithm for the asymmetric traveling salesman problem. Expert Systems with Applications: SCOPUS, 2012 – vol. 39, no. 10, pp. 847–853.
5. Ngamchai, S. and Lovell, D. Optimal time transfer in bus transit route network design using a genetic algorithm, Journal of Transportation Engineering. 129 (5), 510–521 c. — 2003.

Рецензент: д-р техн. наук, проф., Харьковский национальный университет радиоэлектроники, Харьков

Автор: ***ГОРДІЄНКО Тарас Олександрович***
Харківський національний університет радіоелектроніки, магістр, напрям Програмна інженерія.
Моб. тел. – 097-339-46-29, e-mail – trsgrdnk @gmail.com
Автор: ***МАЗУРОВА Оксана Олексіївна***
Харківський національний університет радіоелектроніки, кандидат технічних наук, доцент, доцент кафедри Програмної інженерії.
Раб. тел. – 70-21-446, моб. тел. – 097-615-95-21, E-mail – oksana.mazurova@nure.ua.

Дослідження генетичних алгоритмів для пошуку оптимальних шляхів в системі проведення метромарафонів

Робота присвячена дослідженню генетичних алгоритмів на прикладі пошуку оптимальних шляхів для підтримки проведення метромарафонів в системах типу «Transit Challenge» або «Subway Challenge». На основі правил проведення метромарафонів та задачі комівояжера сформульована проблема станцій — пошуку оптимального шляху, який би дозволив відвідати всі станції метрополітену за найкоротший час. На базі теорії графів розроблена математична модель системи метромарафону. Для вирішення проблеми станцій розроблено генетичний алгоритм: обрано спосіб представлення геному, правила мутації популяцій і схрещування геномів. На основі експериментального дослідження генетичного алгоритму підібрано найбільш ефективні параметри та розроблено рекомендації стосовно вирішення проблеми станцій для метромарафонів різної розмірності.

Ключові слова: *генетичний алгоритм, граф, задача комівояжера, математична модель, пошук оптимального шляху, проблема станцій, схрещування геномів.*

Investigation of genetic algorithms for the search of optimal ways in the Transit challenge system

The work is devoted to the study of genetic algorithms on the example of the search for the best ways to support "Transit challenge" or "Subway Challenge" system. Based on the rules of "Transit challenge" and the tasks of the salesman, the problem of stations was formulated - the search for the best way to visit all subway stations in the shortest possible time. Based on the graph theory, a mathematical model of the metrophone system was developed. To solve the station problem, a genetic algorithm has been developed: the method of genome representation, population mutation and genome crossing rules have been chosen. On the basis of the experimental study of the genetic algorithm the most effective parameters were selected and recommendations on the solution of the station problem for "Transit challenge" of different dimensions were developed.

Keywords: *genetic algorithm, graphs, traveling salesman problem, mathematical model, finding the optimal path, station problem, genome crossing.*

ДОДАТОК Г

Лістинг програмного коду

```
function GAStop() {
  clearInterval(mainInterval);
  dispatchBestResult();
}

function GAStart() {
  initData();
  GAInitialize();
  mainInterval = setInterval(render, intervalDuration);
}

function initData() {
  currentGeneration = 0;
  bestValue = undefined;
  best = [];
  bestValuesArray = [];
  currentBest = 0;
  population = [];
  values = new Array(populationSize);
  mutationCount.update(n => 0);
  crossoverCount.update(n => 0);
}

function render() {
  GANextGeneration();
}

function GAInitialize() {
  const stationsCount = graph.nodes().length;
  population = Array.apply(null, Array(populationSize)).map(item =>
    randomIndivial(stationsCount)
  );
  setBestValue();
}

function GANextGeneration() {
  currentGeneration++;
  population = selection(
    population,
    currentBest,
    best,
    values,
    populationSize
  );
  population = crossover(
    population,
```

```
    dis,
    populationSize,
    crossoverProbability
  );
  population = mutation(population, populationSize, mutationProps);
  setBestValue();
}

function setBestValue() {
  values = population.map(item => evaluate(item, dis));
  currentBest = getCurrentBest(values);

  if (bestValue === undefined || bestValue > currentBest.bestValue) {
    best = population[currentBest.bestPosition].clone();
    bestValue = currentBest.bestValue;
    bestValuesArray = [...bestValuesArray, bestValue];
  }
}


function onStationHover(index, rowIndex) {
  clearListInterval();
  showStation(index);
  showingRow = null;
}
function onPlay() {
  clearListInterval();
  initInterval();
}

function initInterval() {
  let index = 0;
  timerId = setInterval(() => {
    if (index < stationsPath.length) {
      showStation(stationsPath[index].id);
      showingRow = index;
      index++;
    } else {
      clearListInterval();
      disableStationHover();
    }
  }, ANIMATION_DURATION);
}
function clearListInterval() {
  showingRow = null;
  clearInterval(timerId);
}
function showStation(index) {
  isMapActive = false;
  showingStation = index;
}
function disableStationHover() {
  isMapActive = true;
```

```
    showingStation = null;
  }

<div class="svg-render">
    <img
      class="svg-render__back {isMapActive ? 'active' : ''}"
      alt="metro-image"
      src={metroImage} />

    {#if schemeSVGData}
      <svg
        font-family={schemeSVGData.font}
        viewBox={schemeSVGData.viewBox}
        xmlns="http://www.w3.org/2000/svg"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        class="map {isMapActive ? 'map-active' : ''}">

        {#if schemeSVGData.defs}
          <defs>
            {@html schemeSVGData.defs}
          </defs>
        {/if}

        <g fill={colors.text}>
          {#each stationsPath as station, index (station.id)}
            {#if station}
              <g class="station {showingStation == station.id ? 'fadein' : ''}">
                <g stroke={colors[station.color]}>
                  {@html station.path}
                </g>

                <g fill={colors[station.color]}>
                  {@html station.stop}
                </g>

                <text style={station.style}>
                  {@html station.text}
                </text>
              </g>
            {/if}
          {/each}
        </g>

      </svg>
    {/if}

  </div>

export const time = readable(0, function start(set) {
  const beginning = new Date();
  const beginningTime = beginning.getTime();

  const interval = setInterval(() => {
```

```
    const currentTime = new Date().getTime();
    set((currentTime - beginningTime) * 4);
  }, 10);

  return function stop() {
    set(0);
    clearInterval(interval);
  };
});

export function evaluate(indivial, dis) {
  return indivial.reduce(
    (sum, element, index, array) => (sum += dis[element][array[index - 1]])
  );
}

export function randomIndivial(length) {
  return Array.from(Array(length).keys()).shuffle();
}

export function getCurrentBest(values) {
  const min = values.min();

  return {
    bestValue: min,
    bestPosition: values.indexOf(min)
  };
}

function createRoulette(values = []) {
  const fitnessValues = values.map(item => 1.0 / item);
  const sum = fitnessValues.reduce((tempSum, el) => (tempSum += el));

  let tempSum;
  return fitnessValues
    .map(item => item / sum)
    .map(item => (tempSum = (tempSum || 0) + item));
}

function doCrossover(x, y, population, dis) {
  let newPopulation = population.clone();

  let child1 = getChild("next", x, y, newPopulation, dis);
  let child2 = getChild("previous", x, y, newPopulation, dis);
  newPopulation[x] = child1;
  newPopulation[y] = child2;

  return newPopulation;
}

function getChild(fun, x, y, population, dis) {
  let solution = new Array();
  let px = population[x].clone();
```

```
  let py = population[y].clone();
  let dx, dy;
  let c = px[randomNumber(px.length)];
  solution.push(c);
  while (px.length > 1) {
    dx = px[fun](px.indexOf(c));
    dy = py[fun](py.indexOf(c));
    px.deleteByValue(c);
    py.deleteByValue(c);
    c = dis[c][dx] < dis[c][dy] ? dx : dy;
    solution.push(c);
  }
  return solution;
}

function wheelOut(roulette = []) {
  const rand = Math.random();
  roulette.forEach((item, i) => {
    if (rand <= item) {
      return i;
    }
  });
  return 0;
}

export function randomNumber(boundary) {
  return parseInt(Math.random() * boundary);
}

export default function initArrayPrototypes() {
  Array.prototype.clone = function() {
    return this.slice(0);
  };
  Array.prototype.max = function() {
    return Math.max.apply(null, this);
  };
  Array.prototype.min = function() {
    return Math.min.apply(null, this);
  };
  Array.prototype.shuffle = function() {
    if(!this.length) return this;
    for (
      var j, x, i = this.length - 1;
      i;
      j = randomNumber(i), x = this[--i], this[i] = this[j], this[j] = x
    );
    return this;
  };
  Array.prototype.indexOf = function(value) {
    for (var i = 0; i < this.length; i++) {
      if (this[i] === value) {
        return i;
      }
```

```
    }
  };
  Array.prototype.deleteByValue = function(value) {
    var pos = this.indexOf(value);
    this.splice(pos, 1);
  };
  Array.prototype.swap = function(x, y) {
    if (x > this.length || y > this.length || x === y) {
      return;
    }
    var tem = this[x];
    this[x] = this[y];
    this[y] = tem;
  };
  Array.prototype.roll = function() {
    var rand = randomNumber(this.length);
    var tem = [];
    for (var i = rand; i < this.length; i++) {
      tem.push(this[i]);
    }
    for (var i = 0; i < rand; i++) {
      tem.push(this[i]);
    }
    return tem;
  };
}

function subsctractTimeOfStation(path, timeOnStation) {
  if (path.length > 2) {
    path.weight = path.weight - (path.length - 2) * timeOnStation;
  }

  return path;
}

function countDistances(graph) {
  const points = graph.nodes();
  const length = points.length;

  return Array.apply(null, Array(length)).map((item, index) =>
    Array.apply(null, Array(length)).map(
      (item, insideIndex) =>
        ~~graph.getEdgeWeight(points[index], points[insideIndex])
    )
  );
}
```

ДОДАТОК Д

Відгук керівника роботи

ВІДГУК

на атестаційну роботу магістра

Гордієнка Тараса Олександровича, гр. ПЗСм-18-1

(спеціальність - Інженерія програмного забезпечення,
освітньо-професійна програма - Програмне забезпечення систем).

Тема атестаційної роботи: «Дослідження генетичних алгоритмів для реалізації підтримки метромарафонів в системі "Transit challenge"»

Структура атестаційної роботи: пояснювальна записка 101 сторінки; 5 розділів; 4 додатки; графічна частина 25 слайдів.

Метро є досить популярним засобом пересування у великих містах. В Україні є два великих метрополітени – Харківський та Київський, які мають 30 та 52 станції відповідно. Досить популярним сценарієм використання метро є побудова найкоротшого маршруту між двома чи трьома станціями. Але є більш цікавий сценарій, коли потрібно відвідати всі станції метрополітену. І саме такий сценарій представляє спортивний інтерес у багатьох країнах світу серед мандрівників та просто зацікавлених людей.

Магістерське дослідження присвячено дослідженню генетичних алгоритмів та розробці моделі метромарафону, що дозволять отримати оптимальний маршрут для проведення метромарафону, та створенню програмної системи підтримки метромарафонів "Transit challenge".

Магістрант гр. ПЗСм-18-1 Гордієнко Т. О. проаналізував існуючи аналоги та проблеми, пов'язані з пошуком оптимальних шляхів в метрополітені, розробив математичну модель метромарафону, сформулював проблему станцій, на базі моделі побудував алгоритм пошуку оптимального шляху для проведення метромарафону та провів експеримент, в результаті якого сформулював поради щодо встановлення параметрів для алгоритму. Програмна система реалізована з використанням сучасних програмних засобів, клієнт-серверних та інших інформаційних технологій.

Магістрант провів досить детальний аналіз технічної та спеціальної літератури, використав отриману інформацію для обґрунтування прийнятих в роботі науково-дослідних рішень, спираючись на літературу сформулював набори параметрів для тестування. Записка написана грамотно, вимоги стандартів дотримані. Результати роботи досить повно відображені в пояснювальній записці та на слайдах презентації. За результатами атестаційної роботи зроблено доповідь на Молодіжному форумі та подано статтю до науково-технічного журналу «Біоніка інтелекту».

Вважаю, що магістрант гр. ПЗСм-18-1 Гордієнко Т. О. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121- «Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Керівник атестаційної роботи магістра
к.т.н., доц. каф. ПІ — Мазурова О.О.

«16» 12 2019 р.

ДОДАТОК Ж

Зовнішня рецензія

Рецензія

на атестаційну роботу магістра

студент групи **ПЗСм**-18-1 ***Гордієнку Тарасу Олександровичу***

(спеціальність **– 121- Інженерія програмного забезпечення**,

Освітня програма - **Програмне забезпечення систем**)

Тема роботи: «Дослідження генетичних алгоритмів для підтримки системи метромарафонів “Transit Challenge”»

Структура пояснювальної записки: 5 розділів; ____ сторінкок; 32 рисунка; 4 додатка.

Одним з рекордів, які фіксує Книга рекордів Гіннеса, є рекорд найменшого часу, за який вдалося відвідати всі станції Нью-Йоркського метрополітену. Такий же рекорд є і для Лондонського метрополітену. Більш того, такі змагання проводяться і в інших містах по всьому світі. При цьому Україна, яка має одні з найбільших метрополітенів в СНД, участі в таких змаганнях не приймала. Дане дослідження є невеликим, але все таки кроком в напрямку вирішення цієї проблеми.

Атестаційна робота спрямована на дослідження генетичних алгоритмів для вирішення задачі пошуку оптимальних шляхів для метромарафонів або “Transit Challenge”.

В першому розділі Гордієнко Т. О. провів аналіз проблемної області та аналогів, сформулювала постановку задачі. В другому розділі наведено вимоги до програмної системи обліку спортивних досягнень. В другому розділі наведено вимоги до програмної системи з пошуку оптимальних шляхів для метромарафону. В третьому розділі розроблено математичну модель метромарафону та сформульовано проблему станцій, проведено дослідження генетичних алгоритмів та адаптовано основні параметри алгоритму для проблеми станцій. Для розробки алгоритму опрацьовано літературу, що стосується теорії графів та використання генетичних алгоритмів для рішення транспортних задач. В четвертому розділі описано програмну реалізацію розробленої системи. В п’ятому розділі проведено експериментальне дослідження та сформовано рекомендації щодо встановлення параметрів алгоритму та їх зміни в процесі роботи програми.

Представлена магістерська атестаційна робота за змістом відповідає темі та виконана за поставленим завданням. В роботі наявний достатній обсяг цитування наведеної літератури. Пояснювальна записка магістра викладена у логічній послідовності та проілюстрована вичерпними графічними матеріалами. Пояснювальна записка написана грамотно, якість оформлення – висока, вимоги стандартів дотримані. Результати роботи

Вважаю, що магістрант гр. ПЗСм-18-1 Гордієнко Т. О. готовий до самостійної інженерної діяльності. Атестаційна робота відповідає вимогам до атестаційних робіт магістрів та заслуговує оцінки « відмінно (98)». Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121- «Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Рецензент

проф. каф. ІУС
к.т.н. Васильцова М.В.

ДОДАТОК И

Внутрішня рецензія

Рецензія
на атестаційну роботу магістранта групи ПЗСм-18-1
Гордієнка Тараса Олександровича
(спеціальність - Інженерія програмного забезпечення,
освітньо-професійна програма - Програмне забезпечення систем).
Тема: «Дослідження генетичних алгоритмів для реалізації підтримки метромарафонів в системі "Transit challenge"»

Структура атестаційної роботи: пояснювальна записка 101 сторінка; 5 розділів; 4 додатки; графічна частина 25 слайдів.

Тема метромарафону, тобто відвідування всіх станцій метро за якомога коротший час, представляє спортивний інтерес серед ентузіастів в багатьох країнах світу. Проводяться навіть змагання, які називають «Transit challenge», «Subway challenge» або просто метромарафон. Магістерська робота присвячена дослідженню наукової проблеми побудови оптимального шляху для метромарофону з використанням генетичних алгоритмів та практичній проблемі побудови відповідної програмної системи підтримки метромарафонів.

В першому розділі пояснювальної записки проведено аналіз проблемної області та проаналізовано існуючи аналоги та проблеми, пов'язані з пошуком оптимальних шляхів в метрополітені, сформульовано постановку задачі. В другому розділі наведено вимоги до програмної системи. В третьому розділі розроблено математичну модель метромарафону та сформульовано проблему станцій, проведено дослідження генетичних алгоритмів для її вирішення та розроблено адаптований генетичний алгоритм, спроектовано базу даних. В четвертому розділі описано програмну реалізацію розробленої системи. В п'ятому розділі проведено експериментальне дослідження та сформовано рекомендації щодо встановлення параметрів алгоритму та їх зміни в процесі роботи програми.

Результати роботи наочно і досить повно відображені в пояснювальній записці та на слайдах презентації. Проект є актуальним і має практичну спрямованість. Пояснювальна записка написана грамотно, якість оформлення - висока, вимоги стандартів дотримані.

До зауважень можна віднести те, що в пояснювальній записці не в повній мірі обґрунтовано вибір деяких початкових параметрів для проведення експериментального дослідження генетичного алгоритму.

Вважаю, що магістрант гр. ПЗСм-18-1 Гордієнко Т. О. готовий до самостійної інженерної діяльності. Атестаційна робота відповідає вимогам до атестаційних робіт магістрів та заслуговує оцінки «відмінно (98)». Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121- «Інженерія програмного забезпечення» (освітньо-професійна програма «Програмне забезпечення систем»).

Рецензент

_____ Шубін Т. Ю.

проф. кафедри ПІ