УДК 539.3:519.6 **DOI: 10.20998/2078-9130.2024.2.316075**

Є.Ю. МІСЮРА, С. Ю. МІСЮРА, Н.В. СМЕТАНКІНА

ЕВРИСТИЧНИЙ ПІДХІД ДО ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ ЛІТТЛА НА ПРИКЛАДІ ЗАДАЧІ КОМІВОЯЖЕРА

У статті розглянуто задачу комівояжера (комівояжер — бродячий торговець; англ. Travelling Salesman Problem, TSP), яка є однією з найвідоміших та найважливіших оптимізаційних задач у теорії графів та прикладній математиці. Вона має широке практичне застосування, включаючи логістику, планування маршрутів та управління ресурсами. Суть задачі полягає у пошуку найвигіднішого маршруту, що проходить через задані міста лише один раз, а потім повертається до початкової точки. В умовах даної задачі застосовуються критерій вигідності маршруту (тобто найкоротший та найдешевший маршрут) і відповідні матриці відстаней (в кілометрах), тобто основна мета - мінімізувати загальну довжину маршруту або його вартість. Задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів. Для можливості застосування математичного апарату для розв'язання проблеми, її представлено у вигляді математичної моделі. Проблему комівояжера формулюють у вигляді моделі на графі, де міста представлені як вершини, а відстані між ними - як ребра. Авторами запропоновано застосування евристичного методу до розв’язання даної задачі. Для цього вдосконалено програмну реалізацію алгоритму Літтла, який вибирає для розбиття множини з мінімальною межею з усіх можливих гілок, а не з двох отриманих в результаті останнього розбиття. При цьому використовується евристичний підхід до вибору множини з межею не більше, ніж мінімальна. Продемонстровано роботу програми на прикладі проїзду автомобілем між містами України, заданими реальної матрицею відстаней (в кілометрах). У статті розглянуто модернізований метод Літтла для розв’язання задачі комівояжера, що демонструє значно вищу швидкість роботи порівняно з методом повного перебору. Основна ідея - використання евристичного підходу для скорочення простору пошуку та зниження витрат ресурсів. Тестування на прикладі міст України з використанням реальної матриці відстаней у кілометрах підтвердило ефективність алгоритму, який обирає оптимальні розв’язання, зберігаючи мінімальні межі витрат.

Ключові слова: математична модель, задача комівояжера, метод Літтла, евристичний підхід, матриця відстаней, задача оптимізації

The article deals with the Traveling Salesman Problem (TSP), which is one of the most famous and important optimization problems in graph theory and applied mathematics. It has a wide range of practical applications, including logistics, route planning and resource management. The essence of the problem is to find the most profitable route that passes through the given cities only once and then returns to the starting point. In the conditions of this problem, the route profitability criterion (that is, the shortest and cheapest one) and the corresponding distance matrices (in kilometers) are used, that is, the main goal is to minimize the total length of the route or its cost. It is given that the route should pass through each city only once, in this case the intersection is among the Hamiltonian cycles. For the possibility of applying the mathematical apparatus to solve the problem, it is presented in the form of a mathematical model. The Traveling Salesman Problem is formulated in the form of a model on a graph, where cities are represented as vertices, and the distances between them are represented as edges. The authors proposed the use of a heuristic method to solve this problem. For this, the software implementation of Little's algorithm has been improved, which selects for partitioning a set with the minimum limit from all possible branches, and not from the two obtained as a result of the last partition. At the same time, a heuristic approach is used to select a set with a limit no greater than the minimum. The work of the program is demonstrated on the example of driving between cities of Ukraine, given by the real matrix of distances (in kilometers). The article considers the modernized Little method for solving the Traveling Salesman Problem, which demonstrates a significantly higher speed of work compared to the method of complete search. The main idea is to use a heuristic approach to reduce the search space and reduce resource costs. Testing on the example of cities of Ukraine using a real matrix of distances in kilometers confirmed the effectiveness of the algorithm, which selects optimal solutions while saving minimum cost limits.

Keywords: mathematical model, traveling salesman problem, Little’s method, heuristic approach, distance matrix, optimization problem

Вступ. Однією з найбільш популярних задач оптимізації є задача комівояжера. Стаття, яка з’явилася в 1832 році як концептуальна ідея в книзі «комівояжера – як він повинен вести себе і що повинен робити для того, щоб доставляти товар і мати успіх у своїх справах – поради старого кур'єра» (нем. Der Handlungsreisende – wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur); яка розвинулася в XIX столітті як задача на графах (Вільям Гамільтон) і задача оптимізації (Карл Менгер); і остаточно сформувалася як задача мандрівного торговця (англ. Traveling Salesman Problem), яку запропонував Хасслер Уїтні з Прінстонського університету; є актуальною і до цього дня. Незважаючи на простоту визначення та формалізації задача, а так само відносну легкість теоретичного розв’язання (йдеться про малорозмірних одновимірних випадках, в яких задача може бути розв’язана за допомогою найпростіших методів повного перебору) задача комівояжера є досить складною задачею відшукання оптимального шляху, що вимагає наявність потужного математичного та інженерного апаратів. Тому, починаючи з другої половини XX століття, дослідження задачі комівояжера стало мати здебільшого теоретичний сенс, де сама задача виступає в якості моделі для розробки нових алгоритмів оптимізації, які продовжують удосконалюватися і зараз.

Аналіз літератури. Методи розв'язання задачі комівояжера поділяються на теоретичні і евристичні. Теоретичні методи ґрунтуються на пошуку оптимального розв’язку шляхом перебору всіх можливих варіантів. Їх успішні програмні реалізації представлені цілим рядом останніх публікацій: з використанням динамічного програмування [1], метод гілок і меж [2], множників Лагранжа [3], що відтинають площини [4], композитні [5], інші [6 – 8]. Незважаючи на очевидне гідність даних методів, а саме, точність отриманого розв’язку, всі вони вимагають великого обсягу обчислень. Це пов'язано з алгоритмічної складністю задачі, а саме, оптимізаційна постановка задачі комівояжера відноситься до класу NP-важких завдань (з ростом числа міст вона не може бути розв’язана теоретичними методами за час, менше кількох мільярдів років). Необхідно відзначити, що активне вдосконалення ЕОМ не рятує ситуацію, що породжує розвиток наближених (евристичних) методів і їх композицій.

Аналіз літератури підтверджує успішне застосування евристичних методів до розв'язання задачі комівояжера, до останніх значних робіт в цій області можна віднести наступні:

модифікований алгоритм найближчого сусіда [9], заснований на POPMUSIC (Partial Optimization Metaheuristic Under Special Intensilcation Conditions) шаблоні;

поліпшені версії алгоритму бджолиної колонії [10], такі як combinatorial artificial bee colony optimization (CABC) и quick artificial bee colony optimization (qCABC);

метод пошуку сходженням до вершини (Fast heuristic) та його модифікації (Fast-2 heuristic, Fast-3 heuristic) [11], використовуючи алгоритм Керніган-Ліна як опорного (початкового) розв'язку;

використання можливостей розпаралелювання інших генетичних алгоритмів [12] і його модифікаційних композицій [13 – 15];

метод оптимізації, заснований на фізіології дерев (Tree Physiology Optimization, TPO), який розработаний у 2013 году A. Hanif Halim і застосований до задачі комівояжера в роботі [16].

Крім очевидних недоліків самих евристичних методів, існує досить серйозна проблема, пов'язана з їх неуніверсальністю. Часто, покладаючись на NP-еквівалентність різних варіантів задачі комівояжера, дослідники ставлять задачу порівняльного аналізу можливостей пошуку глобальних оптимальних розв'язків, часу обчислень, статистичних характеристик і умов збіжності для кожного запропонованого ними евристичного алгоритму. При цьому тестування алгоритмів проходить на модельних задачах, що відрізняються від реальних, які мають складну економічну постановку. Останні, в свою чергу, вимагають перенастроювань поточних алгоритмів в кращому випадку або неможливість їх застосування взагалі, що є серйозною проблемою.

І якщо програмна реалізація евристичних методів вимагає глибоких знань в області різних програмно-апаратних платформ, то запрограмувати алгоритм Літтла (окремий випадок методу гілок і меж), відомий з 1963 року, здавалося б, не повинно скласти ніяких труднощів. Широкий вибір щодо простих середовищ програмування (Java, Python, Matlab, C/C++, …), представлений у вільному доступі, тільки сприятливо сприяє цьому. Однак, аналізуючи літературу, присвячену задачі комівояжера, наприклад, [17, 18], автори статті зіткнулися з однією достатньою серйозною помилкою в програмній реалізації даного методу, а саме, множина для розбиття вибиралася на поточній ітерації з двох поточних підмножин, отриманих в результаті останнього розбиття, здійсненого на попередній ітерації. Хоча, по суті, слід вибирати для розбиття множини з мінімальною межею з усіх можливих. З чого і витікає наступна постановка задачі даної статті.

Мета дослідження. Мета - підвищення ефективності програмної реалізацію алгоритму Літтла, яка вибирає для розбиття множини з мінімальною межею з усіх можливих гілок, а не з двох отриманих в результаті останнього розбиття. Метод Літтла застосовується до багатьох.

NP-повних задач донині. При цьому використовується евристичний підхід до вибору множини з межею, не більше, ніж мінімальна. Продемонструвати роботу програми на прикладі задачі комівояжера, а само на прикладі проїзду автомобілем між містами України, заданими реальною матрицею відстаней (в кілометрах).

Математична постановка задачі. Задача комівояжера може бути формалізована у вигляді моделі на графі G:=(V,E), де вершини графа V означають міста відправлення / призначення, а ребра між вершинами графа $\{i, j\}$ – це шляхи сполучення між містами. Тоді розв'язати задачу комівояжера означає знайти гамільтонів цикл з мінімальною вагою в повному зваженому графі G:

$$f = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} \to \min \quad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j = \overline{1,n}, \sum_{j=1}^{n} x_{ij} = 1, \forall i = \overline{1,n} \quad (2)$$

Тут c_{ij} – відстань між i-й і j-й вершиною в графі; $x_{ij} \in \{0,1\}$ та $x_{ij} = 1$, якщо шлях проходить з i-ї вершини в j-ту та $x_{ij} = 0$ в іншому випадку.

Така постановка задача наводить на думку про моделі дискретної оптимізації, де шуканий маршрут представляється у вигляді значень множини змінних приналежності, а умовою того, що значення множини змінних визначать маршрут, є описані далі обмеження (3) – (4). Тобто, кожна вершина такого графа повинна бути інцидентною решті вершин через два ребра (вхідні і вихідні ребро):

$$\forall i \in V, \sum_{j \in V/i} x_{ij} = 2 \quad (3)$$

Таким чином, обмеження (3) – це умова кратності, при якому кожна вершина повинна мати одне вхідне і одне вихідне ребро. Однак, тут виникає проблема, пов'язана з тим, що умова (3) поширюється і на значення змінних, відповідних окремим циклам, де кожна вершина належить лише одному циклу, що в свою чергу породжує обмеження (4), а саме, умова усунення підмаршрутів:

$$\sum_{i \in S, j \notin S} x_{ij} \geq 2 \quad (4)$$

де S – множина вершин, таких, що

$1 \leq |S| \leq |V| - 1$.

У класичній літературі існують і інші підходи як до побудови математичних моделей задачі комівояжера, так і альтернативні умови усунення підшляхів. Автори статті привели модель задачі, описану (1) – (4), яка послужила основою написання програмного коду, наведеного нижче.

Програмна реалізація алгоритму Літтла з використанням евристики. Необхідно відзначити, що в загальному випадку метод являє собою повний перебір всіх можливих варіантів з вибраковуванням

явно неоптимальні розв’язання. Програмна реалізація методу (алгоритму) включає два основних етапи – на першому етапі здійснюється приведення матриці витрат і розрахунок нижньої оцінки вартості маршруту; на другому, основному, етапі здійснюється програмна реалізація алгоритму Літтла. Евристика підключається на другому етапі в момент вибору множини з мінімальною оцінкою. Опис вдосконаленого алгоритму для пошуку оптимального маршруту наведено в табл. 1.

Наступним необхідним кроком є введення евристики в алгоритм (табл. 1, п. 5) в зв'язку з тим, що в алгоритмі гілок і меж фактично будується дерево, у вузлах якого приймається рішення вибрати ребро (h,k) або ні (см. табл. 1), і породжуються два розгалуження – Sw(h,k) та Sw/o(h,k). Далі, з введенням евристики, кращий варіант для наступної ітерації вибирається не тільки по оцінці, а й по глибині дерева, тому що чим глибше обраний елемент, тим ближче він до кінця підрахунку.

Таблиця 1 – Опис програмної реалізації алгоритму Літтла

Перший етап Приведення матриці витрат і обчислення нижньої оцінки вартості маршруту	Другий (основний) етап
1. Обчислюємо найменший елемент в кожному рядку	1. Обчислення штрафу за невикористання для кожного нульового елемента наведеної матриці витрат. Штраф за невикористання елемента з індексом (h, k) в матриці означає, що це ребро не включається в наш маршрут, а значить мінімальна вартість «невикористання» цього ребра дорівнює сумі мінімальних елементів в рядку h і стовпці k а) Шукаємо усі нульові елементи в наведеній матриці б) Для кожного з них вважаємо його штраф за невикористання в) Вибираємо елемент, якому відповідав би максимальний штраф (будь-який, якщо їх декілька)
2. Переходимо до нової матриці витрат, віднімаючи з кожного рядка змінну, отриману в п.1	2. Тепер нашу множину S розбиваємо на множини, що містять ребро з максимальним штрафом (Sw) і не містять це ребро (Sw / o)
3. Обчислюємо найменший елемент в кожному стовпці	3. Обчислення оцінок витрат для маршрутів, що входять в кожне з цих множин а) Для множини Sw/o оцінка витрат дорівнює сумі оцінки витрат множини S і штрафу за невикористання ребра (h, k) б) При обчисленні витрат для множини Sw візьмемо до уваги, що раз ребро (h, k) входить в маршрут, то значить ребро (k, h) в маршрут входити не може (див. умови (3) – (4)), тому в матриці витрат пишемо c (k, h) = infinity, а так як не одно ребро, що виходить з h, і неоднозначне ребро, що приходить в k, вже використовуватися не можуть, тому викреслюємо з матриці витрат рядок h і стовпець k в) Після цього наводимо матрицю, і тоді оцінка витрат для Sw дорівнює сумі оцінки витрат для S і r (h, k), де r (h, k) − сума констант приведення для зміненої матриці витрат
4. Переходимо до нової матриці витрат, віднімаючи з кожного стовпчика змінну, отриману в п.3. Як результат, маємо матрицю витрат, в якій в кожному рядку і в кожному стовпці є хоча б один нульовий елемент	4. З всіх нерозбитих множин вибирається та, яка має найменшу оцінку
5. Обчислюємо межу на даному етапі як суму змінних п.1 і 3 (дана межа буде вартістю, менше якої неможливо побудувати шуканий маршрут)	5. Підключаємо евристику

Наступним необхідним кроком є введення евристики в алгоритм (табл. 1, п. 5) в зв'язку з тим, що в алгоритмі гілок і меж фактично будується дерево, у вузлах якого приймається рішення вибрати ребро (h,k) або ні (см. табл. 1), і породжуються два розгалуження – Sw(h,k) та Sw/o(h,k). Далі, з введенням евристики, кращий варіант для наступної ітерації вибирається не тільки по оцінці, а й по глибині дерева, тому що чим глибше обраний елемент, тим ближче він до кінця підрахунку.

Програмний код, відповідний до описаного вище алгоритму гілок і меж з підключенням евристики, написаний на мові програмування php, представлений нижче.

В даний час немає усталеної точки зору на критерії порівняльної оцінки ефективності різних алгоритмів розв'язання задачі комівояжера [4, 5, 8, 17, 18]. Тому автори статті вирішили продемонструвати працездатність алгоритму на різних матрицях досить великих розмірів, які випадково генеруються, за допомогою порівняльного аналізу з алгоритмом

повного перебору. Результати порівняльного аналізу показані на рис. 1 та 2.

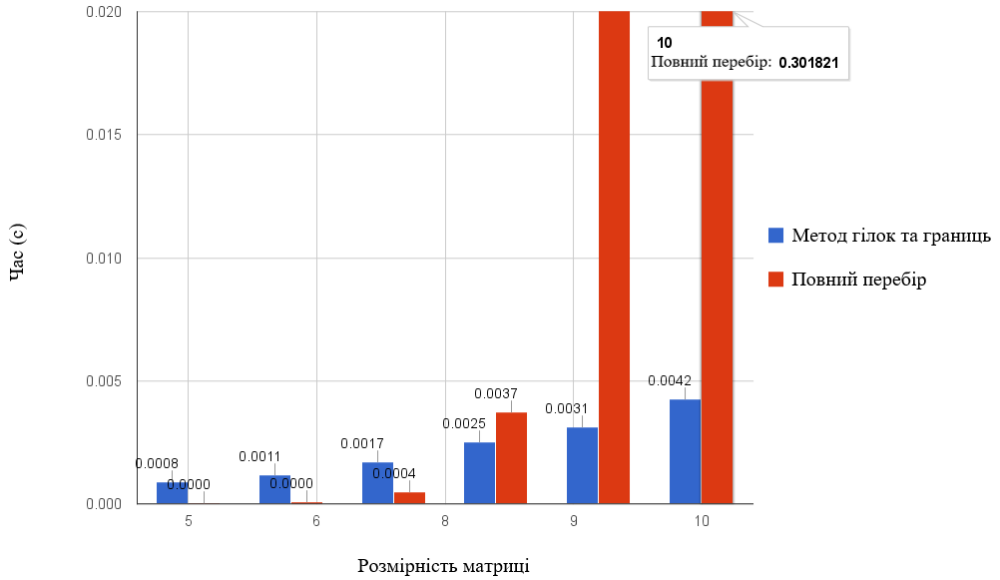

Рис. 1 - Порівняльний аналіз методу гілок і меж з підключенням евристики та методу повного перебору для випадкових матриць від 5x5 до 10x10 за часом

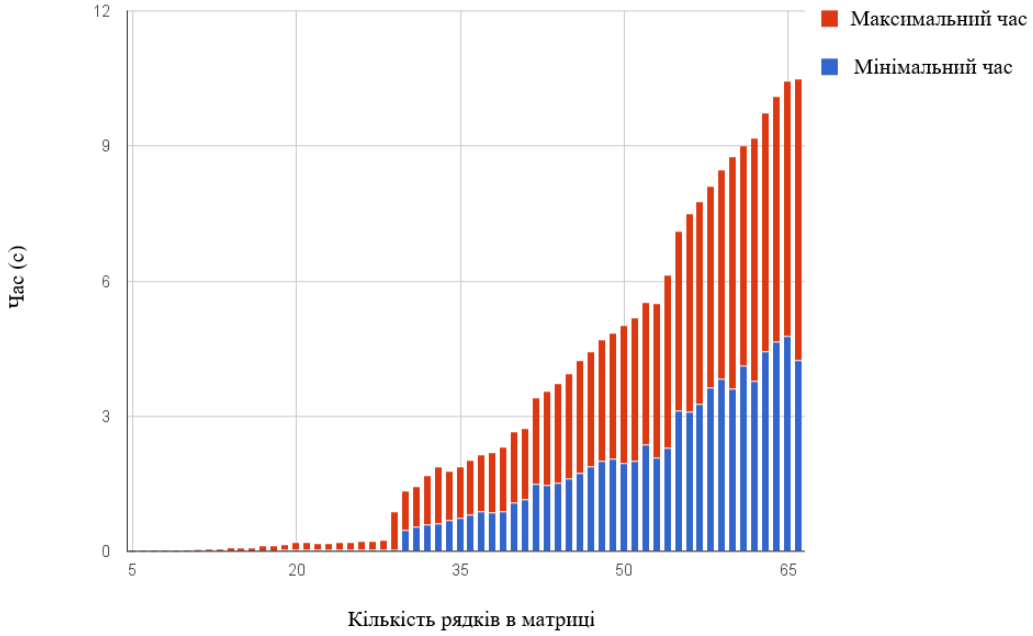

Рис. 2. - Максимальний і мінімальний витрачений час для матриць від 5x5 до 66x66

Застосування користувальницької програми до розв'язання задачі проїзду автомобілем між містами України, заданими реальної матрицею відстаней (у кілометрах). Розглядається наступна постановка задачі. Автомобіліст повинен відвідати певне число міст України (див. рис. 3) і повернутися в пункт відправлення. Визначити, в якому порядку він повинен об'їхати міста, щоб дорога зайняла найкоротший термін (всі пункти відвідуються один раз).

Як приклад роботи програми, нижче наведений перший ітераційний крок розбиття множини на дві підмножини.

	Київ	Сімферополь	Дніпро	Донецьк	Житомир	Ів.-Франківськ	Луганськ	Львів	Маріуполь	Харків	Одеса	Полтава	Рівне	Херсон	Хмельницький	Чернігів	Чернівці
Київ		826	476	729	140	585	810	541	740	478	475	340	328	567	324	142	511
Сімферополь			459	547	854	1144	707	1136	439	660	478	634	1042	266	895	963	1078
Дніпро				249	615	940	379	930	312	219	454	193	802	330	689	581	873
Донецьк					867	1188	146	1179	113	299	703	395	1035	529	939	837	1121
Житомир						444	948	402	896	617	503	478	189	594	184	283	371
Ів.-Франківськ							1138	133	1221	1062	793	923	276	884	253	727	138
Луганськ								1308	273	330	832	476	1151	689	1069	918	1250
Львів									1212	1018	786	880	212	877	237	685	297
Маріуполь										411	635	485	1082	421	972	850	1154
Харків											673	144	809	548	801	587	991
Одеса												583	692	218	544	613	728
Полтава													667	474	663	448	849
Рівне														783	193	472	325
Херсон															636	704	819
Хмельницький																467	187
Чернігів																	654
Чернівці																	

Рис. 3 - Вихідна матриця відстаней (у кілометрах)

Віднімання мінімумів по рядку подано у табл. 2.

Таблиця 2 – Віднімання мінімумів по рядку

	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	INF	287	589	0	445	510	401	600	338	335	200	179	427	184	0	368
2	560	144	281	588	878	281	870	173	394	212	368	767	INF	629	695	809
3	283	INF	56	422	747	587	737	119	26	261	0	600	137	496	494	493
4	616	87	INF	754	754	594	1066	0	186	590	282	913	416	826	722	1005
5	0	426	727	INF	304	648	262	756	477	363	338	40	454	44	141	228
6	452	758	734	311	INF	845	0	1088	929	660	790	134	751	120	118	117
7	537	618	594	675	865	INF	1035	0	57	559	203	869	416	796	643	974
8	408	748	1046	269	0	1015	INF	1079	885	653	747	70	744	104	550	161
9	627	150	0	783	1108	0	1099	INF	298	522	372	960	308	859	735	1038
0	334	26	155	473	918	26	874	267	INF	529	0	656	404	657	441	844
1	257	187	485	285	575	454	568	417	455	INF	365	465	0	326	393	507
2	196	0	251	334	779	172	736	341	0	439	INF	514	330	519	302	702
3	139	564	846	0	87	802	23	893	620	503	478	INF	594	4	281	133
5	140	456	755	0	69	725	53	788	617	360	479	0	452	INF	281	0
6	0	498	695	141	111	616	543	708	445	471	306	321	562	325	INF	509
7	324	453	934	184	66	903	110	967	804	541	662	129	632	0	465	INF

Знаходження мінімальних по рядках
Мінімальний по рядках: 0 144 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Майже нова хвилинна межа 3345
Результат віднімання мінімумів по рядку
Віднімання мінімумів за стовпцями
Знаходження мінімальних за стовпцями
Мінімальний за знаходження мінімальних по рядках и: 0 0 0 0 0 0 0 0 0 68 0 0 0 0 0 0 0
Нова хвилинна межа 3413
Результат віднімання мінімумів за стовпцями представлено у табл. 3.

Таблиця 3 – Результат віднімання мінімумів за стовпцями

	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	INF	287	589	0	445	510	401	600	338	267	200	179	427	184	0	368
2	416	0	137	444	734	137	726	29	250	0	224	623	INF	485	551	665
3	283	INF	56	422	747	587	737	119	26	193	0	600	137	496	494	493
4	616	87	INF	754	754	594	1066	0	186	522	282	913	416	826	722	1005
5	0	426	727	INF	304	648	262	756	477	295	338	40	454	44	141	228
6	452	758	734	311	INF	845	0	1088	929	592	790	134	751	120	118	117
7	537	618	594	675	865	INF	1035	0	57	491	203	869	416	796	643	974
8	408	748	1046	269	0	1015	INF	1079	885	585	747	70	744	104	550	161
9	627	150	0	783	1108	0	1099	INF	298	454	372	960	308	859	735	1038
0	334	26	155	473	918	26	874	267	INF	461	0	656	404	657	441	844
1	257	187	485	285	575	454	568	417	455	INF	365	465	0	326	393	507
2	196	0	251	334	779	172	736	341	0	371	INF	514	330	519	302	702
3	139	564	846	0	87	802	23	893	620	435	478	INF	594	4	281	133
5	140	456	755	0	69	725	53	788	617	292	479	0	452	INF	281	0
6	0	498	695	141	111	616	543	708	445	403	306	321	562	325	INF	509
7	324	453	934	184	66	903	110	967	804	473	662	129	632	0	465	INF

Початок підрахунку штрафів у нулів
Підрахування ступеня у нулів:
Кінець підрахунку штрафів у нулів
Максимуми по рядках: 118 193 26 87 40 140 57 136 56 26 324 26 4 117 111 70 (табл. 4).
Максимальний ступінь 0 знаходиться на позиціях (11:14)
Межа, яка не містить ребро (14,2): 3413 та містить ребро 3413
Результат роботи програми наведено на рис. 4., а підрахування степенів у нулів подано у табл. 4.

Таблиця 4 – Підрахування степенів у нулів по позиціям

(1:5)=0;	(3:12)=26;	(7:9)=57;	(10:12)=26;	(13:5)=4;	(16:1)=111;
(1:16)=118;	(4:9)=87;	(8:6)=136;	(11:14)=324;	(15:5)=0;	(17:15)=70
(2:3)=0;	(5:1)=40;	(9:4)=56;	(12:3)=0;	(15:13)=40;	
(2:11)=193;	(6:8)=140;	(9:7)=26;	(12:10)=26;	(15:17)=117;	

Висновки. У статті розкрито ідею методу Літтла, показано можливості програмної реалізації даного методу і виявлено недоліки попередніх реалізацій, здебільшого пов'язані з відсутністю модернізованих евристичних підходів, що пояснюється наступним. Задача комівояжера (приклад проїзду автомобілем між містами України) полягає в пошуку самого вигідного маршруту. Тому необхідно зберігати попередні кроки (матрицю витрат), і при необхідності, перевіряти всі потенційні розв’язки на попередніх кроках.

Рис. 4 - Розв’язання задачі комівояжера модифікованим алгоритмом Літтла з використанням авторської програми

Однак, такий підхід має на увазі тільки застосування повного перебору, що тягне за собою колосальні витрати часу і машинної пам'яті. Тому автори віддали перевагу евристичному підходу розв’язання задачі, який показав істотну швидкість порівняно з методом повного перебору. Швидкість роботи наведеної авторами програмної реалізації модифікованого алгоритму Літтла набагато вище, що дозволяє застосовувати останній замість повного перебору.

Список літератури

1. Salii, Y. `Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization // European Journal of Operational Research, 2019. – 272(1), – pp. 32–42. https://doi.org/10.1016/j.ejor.2018.06.003
2. Inayatullah, S. *A* Note on Branch and Bound Algorithm for Integer Linear Programming / S. Inayatullah, W. Riaz, H. A. Jafree, T. A. Siddiqi, M. Imtiaz, S. Naz, S. A. Hassan // Current Journal of Applied Science and Technology, 2019. – 34(6). – pp. 1–6. https://doi: 10.9734/cjast/2019/v34i630155
3. Berger A. A time- and space-optimal algorithm for the many-visits TSP. Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms / A. Berger, L. Kozma, M. Mnich, R. Vincze // Society for Industrial and Applied Mathematics, 2019. – pp. 1770–1782. https://doi:10.1137/1.9781611975482.106
4. Kesen S. E. Integrated Production Scheduling and Distribution Planning with Time Windows / S. E. Kesen, T. Bektaş // Cham: Springer, Lean and Green Supply Chain Management, 2019. – pp. 231–252. https://doi.org/10.1007/978-3-319-97511-5_8
5. Wu J. Mixed steepest descent algorithm for the traveling salesman problem and application in air logistics / J. Wu, L. Zhou, Z. Du, Y. Lv // Transportation Research Part E: Logistics and Transportation Review, 2019. – 126. – pp. 87–102. https://doi.org/10.1016/j.tre.2019.04.004
6. O’Neil R. J. Decision diagrams for solving traveling salesman problems with pickup and delivery in real time / R. J. O’Neil, K. Hoffman // Operations Research Letters, 2019. – 47(3). – pp. 197–201https://doi.org/10.1016/j.orl.2019.03.008.
7. Tawhid M. A. Discrete sine-cosine algorithm (DSCA) with local search for solving traveling salesman problem / M. A. Tawhid, & Savsani, P. // Arabian Journal for Science and Engineering, 2019. – 44(4). – pp. 3669–3679. https://doi.org/10.1007/s13369-018-3617-0
8. Paul, J. A. Shared Capacity Routing Problem − An omni-channel retail study / Paul, J. A., Niels, S., Remy, K. & René, De // European Journal of Operational Research, 2019. − 273(2). − pp. 731–739. https://doi.org/10.1016/j.ejor.2018.08.027
9. Taillard, É. D. POPMUSIC for the travelling salesman problem / É. D. Taillard, K. Helsgaun // European Journal of Operational Research, 2019. – 272(2). – pp. 420-429. https://doi.org/10.1016/j.ejor.2018.06.039
10. Karaboga, D. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms / D. Karaboga, B. Gorkemli // International Journal on Artificial Intelligence Tools, 2019. – 28(01). – pp. 1950004:1–1950004:28. https://doi.org/10.1142/S0218213019500040
11. Dell'Amico, M. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem / Dell'Amico, M., Montemanni, R. & Novellani, S. // arXiv preprint arXiv:1906.02962, 2019. – pp. 1–18.
12. Juneja, S. S. Travelling Salesman Problem Optimization Using Genetic Algorithm / S. S. Juneja, P. Saraswat, S. K. Chowdhary // 2019 Amity International Conference on Artificial Intelligence (AICAI), IEEE, 2019. – pp. 264–268. https://doi:10.1109/aicai.2019.8701246
13. Varadarajan S. The massively parallel mixing genetic algorithm for the traveling salesman problem / S. Varadarajan, D. Whitley // Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2019. –pp. 872–879. https://doi.org/10.1145/3321707.3321772
14. Maity, S. A rough multi-objective genetic algorithm for uncertain constrained multi-objective solid travelling salesman problem /

S. Maity, A. Roy, M. Maiti // Granular Computing, 2019. – 4(1). – pp. 125–142. https://doi.org/10.1007/s41066-018-0094-5
15. Singhal S. Hybrid Genetic Algorithm: Traveling Salesman Problem / S. Singhal, H. Goyal, P. Singhal, J. Grover // International Conference on E-Business and Telecommunications, Springer, Cham, 2019. – pp. 376–384. https://doi.org/10.1007/978-3-030-24322-7_48
16. Halim A. H. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem / A. H. Halim, I. Ismail // Archives of Computational Methods in Engineering, 2019. – 26(2). – pp. 367–380. https://doi.org/10.1007/s11831-017-9247-y
17. Gilbert H. Optimizing a Generalized Gini Index in Stable Marriage Problems: NP-Hardness, Approximation and a Polynomial Time Special Case / H. Gilbert, O. Spanjaard // Algorithmica, 2019. – 81(7). – pp. 2653–2681. https://doi.org/10.1007/s00453-019-00550-3
18. Pelofske E. Solving large Maximum Clique problems on a quantum annealer / E. Pelofske, G. Hahn, H. Djidjev // Cham: Springer, International Workshop on Quantum Technology and Optimization Problems, 2019. – pp. 123–135. https://doi.org/10.1007/978-3-030-14082-3_11

References (transliterated)

1. Salii, Y. *Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization*. European Journal of Operational Research. 2019. 272(1). Pp. 32–42. https://doi.org/10.1016/j.ejor.2018.06.003
2. Inayatullah, S., Riaz, W., Jafree, H. A., Siddiqi, T. A., Imtiaz, M., Naz, S. & Hassan, S. A. *A Note on Branch and Bound Algorithm for Integer Linear Programming*. Current Journal of Applied Science and Technology. 2019. 34(6). Pp. 1–6. https://doi: 10.9734/cjast/2019/v34i630155
3. Berger, A., Kozma L., Mnich M. & Vincze R. *A time- and space-optimal algorithm for the many-visits TSP*. Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics. 2019. Pp. 1770–1782. https://doi: 10.1137/1.9781611975482.106
4. Kesen, S. E. & Bektaş T. *Integrated Production Scheduling and Distribution Planning with Time Windows*. Cham: Springer, Lean and Green Supply Chain Management. 2019. Pp. 231–252. https://doi.org/10.1007/978-3-319-97511-5_8
5. Wu, J., Zhou L., Du Z. & Lv Y. *Mixed steepest descent algorithm for the traveling salesman problem and application in air logistics*. Transportation Research Part E: Logistics and Transportation Review. 2019. 126. Pp. 87–102. https://doi.org/10.1016/j.tre.2019.04.004
6. O'Neil, R. J. & Hoffman, K. *Decision diagrams for solving traveling salesman problems with pickup and delivery in real time*. Operations Research Letters. 2019. 47(3). Pp. 197–201. https://doi.org/10.1016/j.orl.2019.03.008.
7. Tawhid, M. A.& Savsani, P. *Discrete sine-cosine algorithm (DSCA) with local search for solving traveling salesman problem*. Arabian Journal for Science and Engineering. 2019. 44(4). Pp. 3669–3679. https://doi.org/10.1007/s13369-018-3617-0
8. Paul, J. A., Niels, S., Remy, K. & René, De *Shared Capacity Routing Problem − An omni-channel retail study*. European Journal of Operational Research. 2019. 273(2). Pp. 731–739. https://doi.org/10.1016/j.ejor.2018.08.027
9. Taillard, É. D. & Helsgaun, K. *POPMUSIC for the travelling salesman problem*. European Journal of Operational Research. 2019. 272(2). Pp. 420–429. https://doi.org/10.1016/j.ejor.2018.06.039
10. Karaboga, D. & Gorkemli B. *Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms*. International Journal on Artificial Intelligence Tools. 2019. 28(01). Pp. 1950004:1–1950004:28. https://doi.org/10.1142/S0218213019500040
11. Dell'Amico, M., Montemanni, R. & Novellani, S. *Matheuristic algorithms for the parallel drone scheduling traveling salesman problem*. arXiv preprint arXiv:1906.02962. 2019. Pp. 1–18.
12. Juneja, S. S., Saraswat P. & Chowdhary S. K. *Travelling Salesman Problem Optimization Using Genetic Algorithm*. 2019 Amity International Conference on Artificial Intelligence (AICAI), IEEE. 2019. Pp. 264–268. https://doi:10.1109/aicai.2019.8701246
13. Varadarajan, S. & Whitley, D. *The massively parallel mixing genetic algorithm for the traveling salesman problem*. Proceedings of the Genetic and Evolutionary Computation Conference, ACM. 2019. Pp. 872–879. https://doi.org/10.1145/3321707.3321772
14. Maity, S., Roy, A. & Maiti M. *A rough multi-objective genetic algorithm for uncertain constrained multi-objective solid travelling salesman problem*. Granular Computing. 2019. 4(1). Pp. 125–142. https://doi.org/10.1007/s41066-018-0094-5
15. Singhal S., Goyal H., Singhal P. & Grover J. *Hybrid Genetic Algorithm: Traveling Salesman Problem*. International Conference on E-Business and Telecommunications, Springer, Cham. 2019. Pp. 376–384. https://doi.org/10.1007/978-3-030-24322-7_48
16. Halim A. H. & Ismail I. *Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem*. Archives of Computational Methods in Engineering. 2019. 26(2). Pp. 367–380. https://doi.org/10.1007/s11831-017-9247-y
17. Gilbert, H. & Spanjaard, O. *Optimizing a Generalized Gini Index in Stable Marriage Problems: NP-Hardness, Approximation and a Polynomial Time Special Case*. Algorithmica. 2019. 81(7). Pp. 2653–2681. https://doi.org/10.1007/s00453-019-00550-3
18. Pelofske E., Hahn G. & Djidjev H. *Solving large Maximum Clique problems on a quantum annealer*. Cham: Springer, International Workshop on Quantum Technology and Optimization Problems. 2019. Pp. 123–135. https://doi.org/10.1007/978-3-030-14082-3_11

Надійшла (received) 21.12.2024

Відомості про авторів / Сведения об авторах / About the Authors

Місюра Євгенія Юріївна (Misiura Ievgeniia Iuriivna) – кандидат технічних наук, доцент кафедри Економіко-математичного моделювання, Харківський національний економічний університет ім. С. Кузнеця, м. Харків; тел.: (050) 551-35-56; e-mail: misuraeu@gmail.com; ORCID: https://orcid.org/0000-0002-5208-0853.

Місюра Сергій Юрійович (Misura Serhii Yuriyuvych) – доцент кафедри Математичного моделювання та інтелектуальних обчислень в інженерії, НТУ «ХПІ», м. Харків; тел.: (050) 984-57-15; e-mail: misurasy@gmail.com. ORCID: https://orcid.org/0000-0002-5048-1610.

Сметанкіна Наталія Володимирівна (Smetankina Natalia Volodymyrivna) – завідувач відділу вібраційних і термоміцнісних досліджень, Інститут енергетичних машин і систем ім. А. М. Підгорного НАН України (ІЕМС НАН України), м. Харків; тел.: (098)369-43-23; e-mail: nsmetankina@ukr.net. ORCID: https://orcid.org/0000-0001-9528-3741.